Visual Exploration

Solomon Chibuzo Nwafor
Dept. of Engineering
University of Girona
Girona, Spain
u1999124@campus.udg.edu

Muhammad Faran Akram
Dept. of Engineering
University of Girona
Girona, Spain
u1999088@campus.udg.edu

Enis Hidri
Dept. of Engineering
University of Girona
Girona, Spain
u1999254@campus.udg.edu

***Abstract*—Autonomous exploration of unknown environments remains a central challenge in mobile robotics. This work presents a complete framework for 3D exploration and mapping of indoor environments using a mobile robot equipped with a depth camera. The robot incrementally constructs a 3D model of its surroundings by navigating to strategic viewpoints that aim to maximize information gain. At each iteration, a new target location is selected based on the current state of the map and an exploration strategy that prioritizes coverage. A global planner generates a safe and feasible path to the selected viewpoint, which is then executed by the robot using a tracking controller to ensure smooth and accurate navigation. The system integrates perception, decision-making, and control modules in a unified architecture, enabling fully autonomous operation with minimal prior knowledge of the environment. The proposed approach has been validated in simulation, demonstrating the ability to produce detailed and consistent 3D reconstructions, and is designed for deployment on a real robotic platform.**

Keywords—D Exploration, Next Best View, RRT, Dubins, OctoMap, Pure Pursuit, TurtleBot, RGB-D Sensor, AMR

I. Introduction

Autonomous exploration of unknown environments is a fundamental and widely studied capability in mobile robotics. It is particularly critical in scenarios where no prior knowledge of the environment is available, such as disaster response, search and rescue missions, industrial inspection, and planetary exploration. In such cases, the robot must independently sense, interpret, and map its surroundings while simultaneously deciding where to move next in order to maximize information gain. These tasks must be performed in real-time and in potentially dynamic or partially structured environments, which makes the problem highly complex and multi-faceted.

This work presents a complete autonomous exploration framework built around the Next Best View (NBV) paradigm. NBV approaches aim to iteratively select the most informative viewpoints, allowing the robot to gradually improve its understanding of the environment [1]. The key idea is to move toward locations that maximize the expected information gain, based on what the robot already knows. At each selected viewpoint, the robot remains stationary and performs a full 360-degree rotation, capturing depth images of the environment. This scanning process allows the system to build both a volumetric 3D map, which provides a detailed spatial representation, and a 2D occupancy grid, which is essential for collision avoidance and path planning during navigation.

The robotic platform used in this project is a TurtleBot, a widely adopted differential-drive mobile robot in academia and research. It is equipped with an Intel RealSense D435i depth camera, which provides dense depth data in real-time. Notably, only the depth stream was used in our implementation to reconstruct the environment, simplifying the sensor processing pipeline while maintaining sufficient spatial resolution for autonomous decision-making. The captured depth images are processed and converted into 3D point clouds, which are integrated into a global map using the OctoMap framework [4]., a popular tool for volumetric mapping in robotics.

To determine feasible paths from the robot's current location to a target viewpoint, a global path planning module was developed using the Rapidly-exploring Random Tree (RRT) algorithm [2]. RRT is a sampling-based planning method particularly well-suited to high-dimensional and complex configuration spaces. In a second phase, the planner was enhanced to incorporate Dubins paths, which respect the kinematic constraints of differential-drive robots by limiting the curvature of planned trajectories.

To execute the planned paths, we implemented two motion controllers: a standard Proportional–Integral–Derivative (PID) controller and a Pure Pursuit controller. While the PID controller is widely used in control applications for its simplicity and robustness, the Pure Pursuit algorithm is specifically designed for path tracking and offers smooth, curvature-adaptive steering behavior, making it particularly suitable for real-world deployments with non-holonomic mobile platforms.

High-level task coordination and decision-making are managed using a behavior tree framework [**?**]. Figure 1 illustrates the structure of the implemented behavior tree. The main control loop begins with a finite number of iterations. At each step, the robot performs a 360-degree scan of the environment from its current position. Then, it enters a retry loop in which it attempts to compute the Next Best View (NBV) and generate a valid path to reach it. If the path planning is successful, the robot proceeds to follow the computed path. This structure ensures robustness by allowing recovery from planning failures and maintains modularity through the separation of scanning, decision-making, and motion control tasks.

The complete exploration system was first developed and tested in simulation, allowing for iterative refinement and validation under controlled conditions. After extensive testing in simulated environments, the framework was deployed on

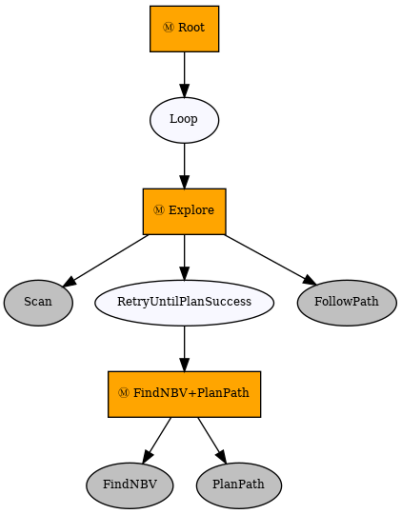

Fig. 1. Behavior tree used to coordinate exploration. The tree consists of scanning, NBV computation, path planning, and path execution in a modular loop.

a real TurtleBot platform. The system demonstrated reliable performance in physical indoor environments, confirming its ability to autonomously explore, build maps, and plan paths in real-time with minimal human intervention.

Overall, this work contributes a practical and modular solution for autonomous exploration, integrating perception, planning, and control in a cohesive framework. The approach is generalizable and can be adapted to other mobile robotic platforms and sensing configurations with minimal modification.

II. Methodology

We address the problem of exploring an unknown environment using an Autonomous Mobile Robot (AMR) equipped with RGBD camera, by employing an iterative exploration framework, as illustrated in Fig. 2. The proposed methodology requires the specification of several input parameters, including the expected size of the environment to be explored (referred to as the map domain), and the characteristics of the robot itself, such as its dimensions, sensor capabilities, and motion constraints

A. Sensing and Environment Representation

In contrast to probabilistic mapping approaches, our system relies on a deterministic volumetric representation using the OctoMap framework to build a 3D occupancy map of the environment. A fixed resolution of 0.03 meters is used for the occupancy grid throughout the exploration process.

To acquire sensor data, the robot performs a 360-degree rotation in place at its current location. During this rotation, depth images are continuously captured from the Intel RealSense D435i depth camera onboard. These images are converted into 3D point clouds using the depth image proc ROS package, which combines the raw depth image and

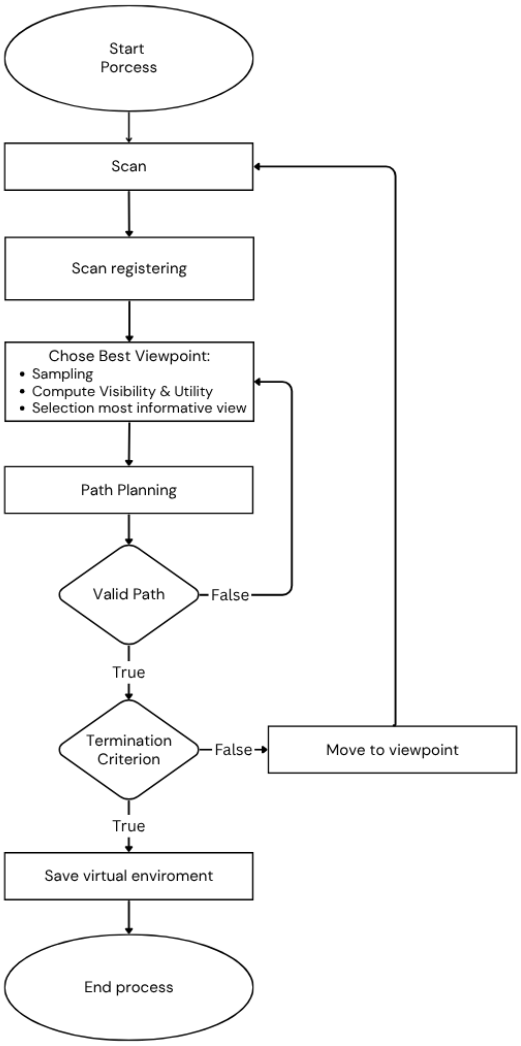

Fig. 2. Overview of exploration framework

the corresponding camera intrinsic parameters provided via camera info.

Assuming the exploration takes place in a flat indoor environment (i.e., without ramps or significant elevation changes), the raw point cloud is passed through a two-stage filtering process. First, a depth filter retains only points within the optimal operating range of the depth camera (0.5 to 2 meters). Second, a height filter removes points located above the robot's functional workspace, eliminating data from ceilings or other irrelevant upper structures and removing the ground to consider to avid considering it as an obstacle.

The resulting filtered point cloud is then forwarded to the OctoMap server, which integrates it into the global 3D occupancy grid using a fixed reference frame. This step is crucial because the camera is moving during the scan; aligning all data to a fixed frame mitigates distortion and noise caused by slight motion during data acquisition.

Importantly, map generation is performed on an external computer, not onboard the robot. Therefore, we had to account

for communication latency between the robot and the processing system to avoid inconsistencies in the mapping process.

Regarding robot localization, this project does not implement any explicit SLAM or pose correction algorithms. Instead, we use the built-in odometry of the robot as its pose estimate. This choice simplifies the system and allows us to focus primarily on planning and exploration. However, since odometry is based on dead reckoning, the approach is prone to cumulative drift in real-world deployments.

B. Sampling and Visibility Calculation

The exploration strategy is based on a sampling-based approach to determine candidate viewpoints for the robot. A predefined number of sample points is randomly generated within a fixed radius around the robot's current position. These points are selected within the known free space of the current occupancy map to ensure that all candidates are reachable and are not located within obstacles or unknown regions.

For each sampled point, we compute its visibility, that is, how much of the currently unknown environment can be observed from that location. To estimate this, we implemented a custom ray tracing routine. From each candidate viewpoint, rays are cast every 3 degrees, resulting in a maximum of 120 rays per viewpoint to simulate a full 360° field of view.

Each ray is projected from the candidate location into the environment and then discretized onto the 2D occupancy grid using the Bresenham algorithm. This algorithm allows us to efficiently convert continuous-ray trajectories into a sequence of grid cells. A ray terminates as soon as it encounters a known obstacle cell, simulating line-of-sight occlusion.

After all rays for a given sample point have been evaluated, we count the number of free cells traversed by the rays. This count represents the visible area from that specific viewpoint and is stored as a visibility score. The visibility scores for all sampled points are later used to select the Next Best Viewpoint that maximizes information gain during exploration.

C. Utility Evaluation

After computing the visibility for each sampled viewpoint, the system evaluates their utility to determine the most informative location for the next exploration step. The utility function balances three key aspects: the amount of visible free space, the presence of contour elements (i.e., known obstacle surfaces), and the travel cost in terms of distance. This trade-off ensures that the robot prioritizes viewpoints that offer high information gain while avoiding excessive or unnecessary movement.

The utility u_i of each candidate viewpoint i is computed as:

$$u_i = \left(\frac{1}{1+d_i} \cdot w_d + \frac{f_i}{\max(f)} \cdot w_f + \frac{ct_i}{\max(ct)} \cdot w_{ct} \right) \cdot ol_i \tag{1}$$

where:

- d_i is the Euclidean distance between the robot's current position and the candidate viewpoint,
- $\frac{1}{1+d_i}$ is the normalized inverse distance term, favoring closer viewpoints,
- f_i is the number of visible free cells from viewpoint i,
- $\max(f)$ is the maximum number of visible free cells among all candidates,
- ct_i is the number of visible contour cells (i.e., obstacle surfaces),
- $\max(ct)$ is the maximum number of contour cells among all candidates,
- w_d, w_f, w_{ct} are weighting coefficients for distance, visibility, and contour information, respectively, with $w_d + w_f + w_{ct} = 1$,
- ol_i is an overlap coefficient, used to enforce a minimum overlap threshold if scan-to-scan registration is considered; in our case, it is set to 1.

Increasing w_d gives preference to nearby viewpoints, while higher values of w_f and w_{ct} prioritize candidates with greater visibility of free space and obstacle surfaces, respectively.

In our implementation, the visibility component f_i is typically assigned a higher weight to promote the the rapid exploration of unknown space. Once the utility has been computed for all candidates, the viewpoint with the highest score is selected as the *Next Best View* (NBV) and a path towards it is planned using the global planning module.

D. Path Planners and Controllers

Once the Next Best View (NBV) is selected based on utility evaluation, a path must be generated to guide the robot safely and efficiently to the target viewpoint. To address this, we implemented two planning strategies: a standard sampling-based planner and a version adapted for non-holonomic constraints.

Initially, the global planner is based on the Rapidly-exploring Random Tree (RRT) algorithm. RRT is a sampling-based motion planning method well suited for high-dimensional configuration spaces. It incrementally builds a tree of feasible paths by randomly sampling the space and extending branches toward those samples, while ensuring collision-free connections using the current occupancy map.

To account for the non-holonomic nature of the TurtleBot, which cannot move sideways and must follow curvature-constrained trajectories, the planner was later extended using Dubins curves. The Dubins-based RRT planner generates paths that respect a minimum turning radius, ensuring that robot motion remains feasible given its differential drive kinematics [3].

For path execution, two controllers were implemented and evaluated: a standard Proportional–Integral–Derivative (PID) controller and a Pure Pursuit controller. The PID controller adjusts the robot's linear and angular velocities based on the positional error with respect to the goal, using feedback loops for proportional, integral, and derivative corrections.

The Pure Pursuit controller, on the other hand, operates by continuously computing a look-ahead point [5] along the path and adjusting the robot's steering to minimize the angular deviation from that point. This approach is particularly effective for smooth trajectory tracking and has been widely adopted in mobile robot navigation.

Figure 3 shows an example of a planned path executed by the robot using the Pure Pursuit controller. The green line represents the computed trajectory, while the robot follows it to reach the target viewpoint. The branching blue lines correspond to the RRT tree, and the lidar scan data are visible as radial rays. This highlights the controller's ability to accurately follow curved paths generated by the Dubins-based planner, even in the presence of obstacles.

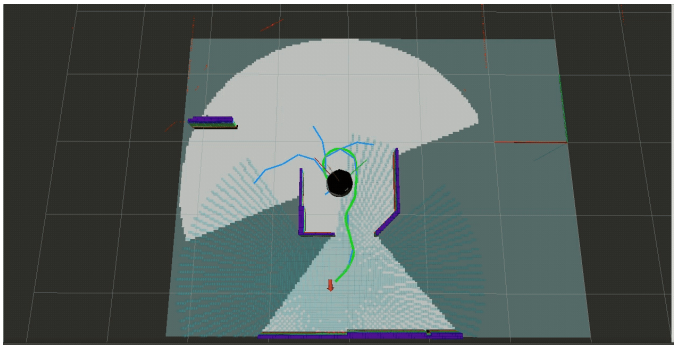

Fig. 3. Planned path execution using the PID. The green line is the followed trajectory; blue lines show the sampled RRT tree.

Both controllers were tested in simulation and on the real robot to assess performance. The Pure Pursuit controller generally provided smoother and more stable tracking, especially for curved paths generated by the Dubins-based planner.

III. Results

To evaluate the influence of the utility function weights on exploration behavior, we conducted a series of experiments using the simulated environment. Each configuration involved running the system for five consecutive Next Best View (NBV) iterations while varying the weight distribution among the utility components: distance, free space visibility (frontier), and contour (obstacle surfaces).

In the first test, we prioritized the distance factor by assigning it a dominant weight. As a result, the robot selected nearby viewpoints and exhibited minimal movement away from its initial position. While this approach minimized travel cost, it failed to significantly expand the explored area within the given number of iterations.

In contrast, when the frontier visibility weight was increased, the robot actively sought out unexplored areas. This led to a wider and more informative map being generated in the same number of NBV steps. The robot covered more ground, and the resulting 3D reconstruction provided a broader representation of the environment.

In the third experiment, emphasis was placed on the contour component. However, due to the robot's initial placement near several walls, the system favored viewpoints with high contour visibility in the immediate vicinity. Consequently, the robot tended to remain in the same area, focusing on observing already detected obstacle surfaces rather than expanding into unknown regions.

From these experiments, we concluded that the most effective weight configuration for exploration in our scenario is:

$$w_d = 0.05, \quad w_f = 0.6, \quad w_{ct} = 0.35$$

This combination ensures that the robot prefers informative viewpoints that reveal new free space, while also modestly favoring regions near already detected structures. The inclusion of a moderate contour weight helps guide the robot toward boundaries without overly restricting movement to known areas.

In the real-world deployment, the system successfully performed autonomous exploration, correctly navigating and covering the environment as intended. However, due to limited testing time and restricted access to the robot, the resulting 3D occupancy map exhibited a significant amount of noise. As a result, while the exploration behavior was consistent with simulation outcomes, the quality of the 3D reconstruction requires further improvement through additional tuning and more extensive testing.

IV. Conclusion

This work presented a complete autonomous exploration framework for indoor environments using a mobile robot equipped with a depth sensor. The system integrates viewpoint selection, path planning, and motion control into a modular architecture capable of incrementally building a volumetric map of unknown environments. Through extensive simulations, the proposed approach demonstrated consistent and effective performance across various weight configurations, allowing us to identify an optimal balance that maximizes coverage while minimizing unnecessary movement.

The simulation results confirm the reliability of the framework in terms of exploration behavior and mapping performance. However, while the system was successfully deployed on a real robot and was able to perform autonomous exploration, further work is needed to improve the quality of 3D reconstructions during real-world interactions. Limited testing time and hardware availability introduced noise and inconsistencies in the generated occupancy map, indicating that additional tuning and experimentation are required to enhance robustness and mapping accuracy under real conditions.

Future work will focus on refining the point cloud filtering process, improving integration timing between sensing and mapping modules, and increasing testing in real-world scenarios to achieve results comparable to those observed in simulation.

References

[1] Narcís Palomeras, Natalia Hurtós, Eduard Vidal, and Marc Carreras, "Autonomous Exploration of Complex Underwater Environments Using a Probabilistic Next-Best-View Planner," *OCEANS 2016 MTS/IEEE Monterey*, 2016.

[2] Steven M. LaValle, "Rapidly-Exploring Random Tree: A New Tool For Path Planning," *Technical Report*, Dept. of Computer Science, Iowa State University, 1998.

[3] F. Clérot, "RRT-Dubins," GitHub repository, https://github.com/FelicienC/RRT-Dubins, accessed May 2025.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[5] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," Carnegie Mellon University, Technical Report, 1992.