POLITECNICO DI TORINO

Master's Degree in MECHATRONIC ENGINEERING

Politecnico di Torino

EPFL

Master's Degree Thesis

Benchmarking a Multi-Robot System Composed of Flying Quadrotors for the Coverage of Known Assets

Supervisors

Prof. Marcello CHIABERGE

Prof. Alcherio MARTINOLI

Dott. Lucas WAELTI

Candidate

Francesco SCATIGNO

October 2025

This page intentionally left blank.

Contents

Chapter 1 Introduction

Today, Micro Aerial Vehicles (MAVs) are widely used for different tasks such as industrial inspection, surveillance, and emergency response due to their high maneuverability and utility. Currently, there are numerous techniques that aim to balance the time required for coverage with spatial coverage efficiency and to improve energy consumption.

The main component in this context is the path planner, which is responsible for computing feasible and optimal paths that follow a set of viewpoints chosen as the best locations to evaluate the purpose of the exploration. The improvements in perception and computation have enhanced the quality of MAV autonomy and opened research to new techniques based on the use of AI.

To improve computational speed, a single robot can be used, which works well in the case of small environments to scan. However, in the case of large scenarios, a single robot may not be sufficient due to its limited battery life and sensor faults. For these situations, the use of multi-robot systems can provide better performance in terms of robustness and coverage efficiency.

Coverage path planning approaches are split into two main areas: model-based and model-free approaches. The difference between these two methods lies in the prior knowledge of the environment that will be explored, which serves as a parameter to determine the trajectory and to plan the next viewpoint.

Model-based approaches utilize CAD models or point-cloud maps to plan efficient coverage paths, leveraging the available geometric information to avoid collisions with objects and coordinate between multiple robots.

In the literature, there are different techniques based on the model-free approach such as frontier-based and Next Best View (NBV), which represent the state of the art in this field. Frontier-based approaches identify boundaries between explored and unexplored regions and select one of these frontiers as the next target point to guide the robot during navigation.

NBV methods sample different viewpoints in the space and select the optimal one based on an information gain function that considers both the sensor orientation and the potential increase in environmental knowledge if the robot moves to that pose.

As a new solution, we uses a novel multi-robot coverage path planning

(CPP) [1] approach for observing objects with known geometry. Our technique is based on the computation of the online path provided by a ground station, in contrast to the classical methods where the paths are optimized offline and require extensive computational resources.

The feasibility of this method is enabled by the use of a PH-tree data structure, which accelerates the identification of unobserved regions without the need to track frontiers or calculate NBVs. This approach allows for real-time task assignment and path planning, making it suitable for resource-constrained MAV systems.

The purpose of this report is to present and compare the results obtained with our method and the state-of-the-art technique. To ensure a fair comparison of our method against different coverage strategies, we have selected and adapted two representative techniques from frontier-based and NBV approaches to work with known environmental geometry. This adaptation allows for direct performance comparison under equivalent conditions.

In this work a comprehensive benchmarking study is presented that evaluates performance across different asset sizes and complexity levels. The comparison employs multiple metrics based on energy consumption, distance traveled, and exploration time to provide a comprehensive assessment of each approach.

To validate the practical applicability of our findings, we have tested all three techniques in realistic simulation environments that closely replicate real-world conditions and constraints.

The experimental setup is based on the use of Webots as simulation environment and ROS Noetic as the middleware framework for inter-component communication. In simulated experiments, the communication between the ground control computer and MAVS is established through MAVROS that provides the standard protocol translation between ROS messages and MAVLink communication standard.

As a complementary contribution, we also present a model-free approach for single-drone operation using a voxel map data structure for environment representation and a TSDF-based path planner for path generation and execution. This paves the way towards potential implementations of multi-robot frameworks for exploration mission in unknown environmnents.

Chapter 2 Literature review

2.1 Coverage

Coverage path planning for multi-robot systems aims to define optimal trajectories that enable multiple robots to observe various objects in space while minimizing operational costs such as time and energy consumption.

The technique implemented by Waelti et al. [1] provides an innovative solution based on the use of a PH-Tree data structure, making multi-robot systems more efficient in terms of energy consumption and capable of performing several tasks online.

To ensure a fair comparison between the PH-Tree-based technique and other approaches present in the literature, we decided to select two different techniques belonging to distinct groups of exploration methods: sampling-based techniques and frontier-based techniques.

For the first group, we selected the NBV approach since it represents the main technique used in autonomous exploration research. As a technique belonging to the second group, we selected an approach based on the use of a collector of different frontiers that guarantees good performance for frontier selection.

The selected techniques are presented in this chapter and represent a comprehensive insight into the current state-of-the-art in multi-robot coverage systems.

2.1.1 Previous works

The work of Waelti et al. [1] represents the primary method analyzed in this benchmarking study, as it introduces a novel approach for multi-robot coverage path planning (CPP). The key distinguishing features of this method compared to traditional approaches lie in the utilization of the PH-tree data structure for spatial indexing and its fully online execution, which eliminates the need for frontier tracking.

The method employs a hybrid system architecture that ensures clear separation between task assignment and individual robot path planning responsibilities. The ground station is responsible for generating tasks for each robot and managing potential failures related to robot collisions or unsafe

path conditions. This centralized architecture enhances robustness against problems during exploration and accelerates the computational process for individual robot task execution.

Each robot computes its trajectory without requiring offline optimization, as is typically necessary in classical techniques. This online approach provides greater flexibility and adaptability compared to pre-computed methods, allowing the system to respond dynamically to changing conditions and robot configurations.

The experimental validation conducted by Waelti et al. [1] demonstrates promising results in terms of coverage efficiency and scalability across various scenarios. However, as discussed previously, the objective of this work is to provide a comprehensive comparison with state-of-the-art techniques in coverage path planning.

For this purpose, we have selected two dominant approaches in CPP: the frontier-based method developed by Caiza et al. [2] and the Next-Best-View technique proposed by Bircher et al. [3]. These methods have been adapted to work with known environments, allowing for direct performance comparison under equivalent conditions with the PH-tree-based approach.

2.1.2 PH-Tree

The PH-tree (PATRICIA-hypercube-tree) designed by ZÃ¤schke et al. [4] is the core component of our technique, as it guarantees better performance in identifying promising areas of exploration. It combines binary PATRICIA-tries with hypercube navigation to achieve efficient data access and storage.

With respect to tradition spatial indexing data structure such as KD-Trees, which split the space along one dimension per node, the PH-tree splits space in all dimension simultaneously at each node, thereby reducing the tree depth compared to binary alternatives.

One key advantage is its space efficiency achieved through a combination of prefix-sharing and bit-stream serialization. In this way, the PH-tree achieves storage requirements equal to or even lower than non-indexed structures.

The main aspect that guarantees better performance for path planning application is the query performance, which benefits from the hypercube structure, allowing efficient spatial range searches. Furthermore, the structure's ability to perform efficient nearest-neighbor queries makes it ideal for identifying unobserved regions in coverage applications.

Another significant advantage is the PH-tree's update efficiency. The structure requires modification of at most two nodes for any insert or delete operation, without the need for rebalancing. This characteristic makes it particularly suitable for online applications where frequent updates are required, such as real-time coverage path planning systems.

In the context of multi-robot coverage, the PH-tree enables efficient spatial queries to identify unobserved voxels without the computational overhead of frontier detection or Next-Best-View calculations. This capability allows our method to perform task assignment in real time while maintaining spatial awareness of the coverage progress.

2.1.3 NBV technique

Next-Best-View (NBV) technique is one of the most widely used approaches for autonomous exploration and coverage path planning. The main concept behind it is the selection of the most informative viewpoint from a set of candidates based on the maximum value of information gain associated with each sampled viewpoint, which considers the potential increase in environmental knowledge.

The Bircher et al. approach [3] represents the first version of NBV-based coverage planner and is widely used in autonomous exploration research. Their receding-horizon planner employs a tree-based sampling strategy where viewpoints are iteratively sampled and evaluated based on information gain criteria. Using an RRT* tree construction algorithm, a finite iteration random tree is computed, and based on a dedicated information gain function, the first edge of the best branch is chosen as the next viewpoint. In this approach, the environment is mapped using an occupancy map where each voxel is classified as occupied or free. In Figure 2.1, it is possible to see the exploration of a house using the approach proposed by Bircher [3], where the points represent the sampled points of the tree expansion when the system searches for a path to a prescribed target. The pink lines represent the different branches of the tree structure, while the green one indicates the optimal branch with the highest information gain.

The drawback of this implementation is the computational overhead associated with repeated tree construction and the tendency toward local optima due to its receding horizon nature. At each iteration, the entire tree is discarded except for the executed branch, resulting in significant computational waste and potential loss of valuable path information. Schmid et al. [5] introduced a novel NBV approach based on the main concept of the RRT* path planner.

Their informative path planning algorithm expands and maintains a single tree of candidate trajectories. Instead of discarding non-executed branches, this method uses the concept of rewiring to keep nodes alive and refine intermediate paths. This approach allows the algorithm to use a single objective function while maximizing path utility.

The method demonstrates particular strength in applications requiring accurate 3D reconstruction, introducing a novel TSDF-based gain formulation that accounts for measurement uncertainty and surface quality. Using TSDF data acquired during exploration, it is also possible to improve the

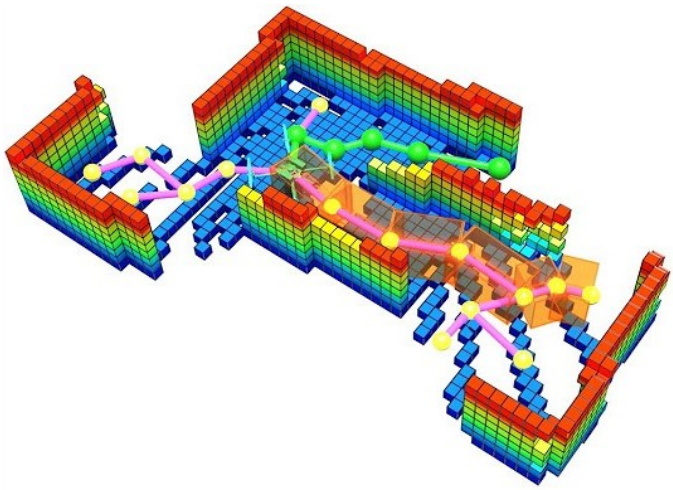

Figure 2.1: Bircher's NBV exploration in unknown space

path and evaluate the impact a viewpoint has on map reconstruction. The proposed gain function considers both sensing noise (which scales quadratically with depth) and state estimation uncertainty.

However, NBV approaches face several computational challenges due to information gain evaluation. In these techniques, gain computation requires 95% of the planner's runtime, especially when based on volumetric information derived from frequently updated occupancy maps. This computational burden becomes particularly significant in real-time applications with resource-constrained systems.

In the context of this benchmarking study, the NBV technique realized by Bircher et al. [3], adapted as described in section 4.2 for exploration of known assets, serves as a good representative approach for sampling-based coverage planning, allowing direct comparison between our technique and frontier-based methods in terms of coverage efficiency, computational performance, and path quality under the same environmental conditions.

2.1.4 Frontier-based technique

Frontier-based technique is one of the most widely used approaches in autonomous exploration. It is based on identifying frontiers, which are defined as the boundaries between known and unknown space, and selecting one as a navigation goal until complete exploration is achieved.

Rapid exploration with Multi-Rotors by Cieslewski et al. [6] introduces an innovative approach to frontier selection for high-speed scenarios. Instead of computing information gain for each frontier, it uses a reactive mode that selects one frontier within the field-of-view of the robot. The method addresses the challenge of exploration time versus coverage quality trade-offs,

demonstrating that while traditional frontier-based approaches can increase total path length, they significantly reduce exploration time when robots can maintain consistently higher speeds.

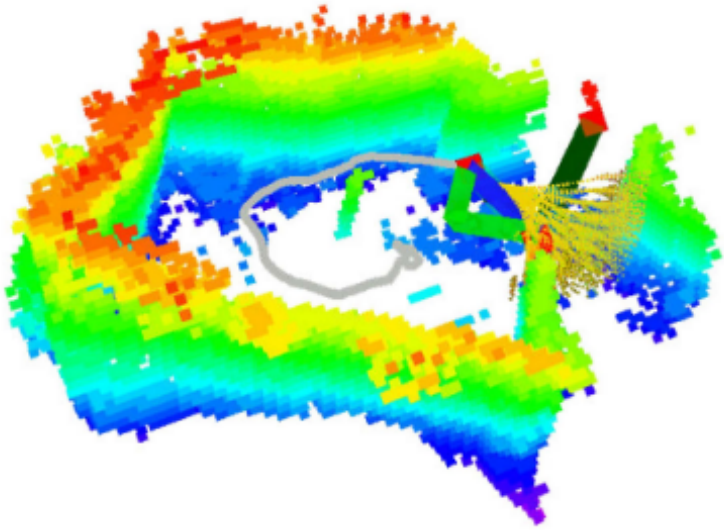

Figure 2.2: Frontier-based exploration in unknown space

The Efficient Frontier-based exploration strategy (EFP) developed by Zhang et al. [7] presents an innovative strategy to address the challenges of inefficient and incomplete map construction. Using UFOMap, it improves 3D environment representation and implements a hierarchical frontier structure. This approach selects the best frontier using Fast Euclidean Clustering (FEC), which reduces the computational time required for obtaining relevant viewpoints. The algorithm maintains both local and global frontier lists, enabling consideration of only local update ranges during frontier detection without traversing the entire map.

The Caiza et al. [2] approach uses a novel exploration method that combines traditional frontier-based exploration with a collector strategy, which stores information regarding frontiers and updates the environmental map. In this way , this approach provides an efficient way to enhance the exploration and to have a map that considers the uncertainty of measurements provided by the robot's sensors.

The collector strategy is defined by two different phases: filtering and selection of best frontier. In each iteration, the system stores and validates frontiers detected and computes an information gain associated with each frontier based on the unknown area that would be seen if that frontier was chosen. At the end, the collector chooses the best frontier taking into account the information gain and the distance to be traveled from the robot's

position to the frontier.

For our purpose, Frontier-based techniques represent a good choice as a comparative approach since they are widely used in real-world and industrial application. By implementing a version based on Caiza's approach [2] , we have realized a version described in section 5.1 for exploration of known assets that can guarantee a fair comparison with other techniques.

2.2 Exploration

As a second part of this work, we chose to move forward in the design of a framework for exploration of unknown assets. In recent years, several techniques have defined methods for simultaneously mapping and navigating unknown environments. The continuous update of the map using data gathered by sensors present on the robot needs to be managed well in order to obtain sufficient information about the environment and to guarantee collision-free navigation of the robot.

Our technique uses data obtained by the ToF sensor and defines frontiers as boundaries between known and unknown areas. In the following section, we present the work of Zhou et al. [8] that guarantees smooth exploration through the use of a frontier manager that stores all obtained data.

2.2.1 FUEL

FUEL (Fast UAV Exploration) is a hierarchical framework developed by Zhou et al. [8] and represents an innovative method for frontier-based exploration in unknown environments. This approach provides a novel solution to the fundamental challenges of traditional frontier-based approaches through the use of a FIS (Frontier Information Structure) and hierarchical path planning.

The aspect that guarantees improvement in terms of time required for exploration is the incremental update of FIS. Traditional frontier-based methods require full map analysis at each iteration, which can suffer from high computational overhead in case of frequent environmental changes. Through the use of FIS, during exploration the system updates only frontiers that belong to portions of the environment that have changed since the last iteration. When new frontiers appear in the map, the system clusters them with already present ones.

After this step, it saves the cells occupied by the frontiers and associates an axis-aligned bounding box to each cluster. During exploration, the dimensions of each cluster are considered and updated, and especially when cells of existing frontiers are no longer present, it removes them from FIS

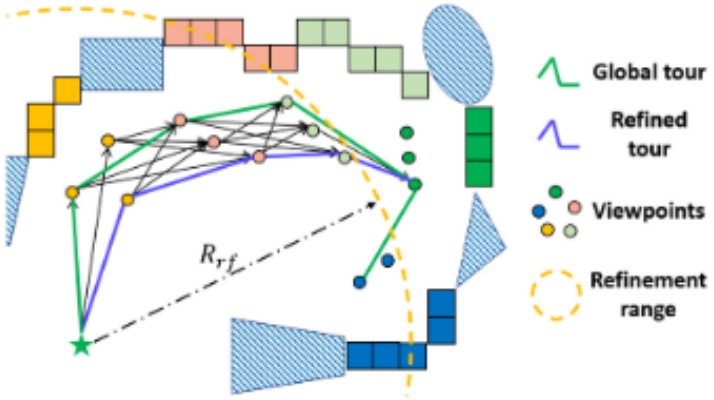

Figure 2.3: Hierarchical path planner used in FUEL

The other main innovation is the use of a hierarchical planner defined by three steps that guarantees separation between global path planning and local trajectory optimization. In Figure 2.3, it is possible to see an example of hierarchical planner where the final path differs from the starting one, guaranteeing a more efficient exploration. As the first step, the planner finds a global tour to cover the existing frontier clusters by solving a standard Asymmetric Traveling Salesman Problem (TSP). In this way, the system can solve the problem quickly using existing algorithms.

After that, a local viewpoint refinement is applied to obtain an improvement in the exploration rate. This refinement process optimizes the positioning of viewpoints within each cluster to maximize information gain while considering vehicle dynamics and sensor constraints.

As the last step, it generates a B-spline trajectory between the new viewpoints to obtain a trajectory that enables the drone to use all its dynamic capabilities. The B-spline representation ensures smooth trajectories that respect acceleration and velocity constraints, allowing for efficient and safe navigation through complex environments.

One of the key advantages of FUEL is its scalability to large-scale environments. The incremental frontier maintenance approach ensures that computational requirements do not grow exponentially with environment size, For these reasons, we have decided to use the FIS as part of our exploration techniques comparison, as it guarantees fair performance in complex environments such as buildings. FUEL serves as a representative example of modern frontier-based techniques that successfully balance exploration efficiency with computational tractability.

Chapter 3 Our technique

3.1 System architecture

The PH-tree-based technique implemented in [1], which is benchmarked in this work, is based on the use of a hybrid system architecture that combines the advantages of centralized control with the flexibility of distributed planning. As shown in Figure 3.1, the system is divided into two main levels: a centralized level composed of the ground station that is responsible for task assignment and coverage state monitoring, and a distributed level composed of robots that generate their own paths and coordinate with each other for the execution of planned trajectories.

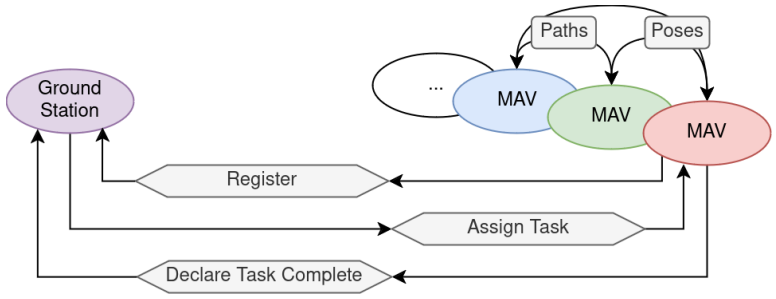

Figure 3.1: System architecture of our approach

The ground station acts as a coordination center that manages different roles during the exploration process. First, it maintains a global observation map that incorporates knowledge regarding voxels that have been observed and those that need to be covered.

Through the use of a PH-tree, the ground station identifies unobserved regions and generates specific tasks for each robot. Another task performed by the centralized component is managing robot registration and unregistration, allowing flexible configuration of robots during exploration based on the number of voxels remaining to be observed.

All communications between the ground station and robots follow a request-acknowledgment protocol that ensures the synchronization necessary for effective dynamic task planning and allocation.

Individual robots possess local autonomy that allows them to perform several tasks during exploration. When the ground station assigns a task to a robot, each drone is capable of determining whether that task is feasible based on its local position and capabilities. Using RRT* path planning, they evaluate a path taking into account the viewpoints provided by the ground station and collision constraints with other robots. Inter-robot coordination is based on the use of open channels where each robot transmits its position and planned trajectory to avoid conflicts and optimize the utilization of operational space. The use of two different levels of communication in the system, where communications between the ground station and robots require acknowledgment from the receiving party to ensure synchronization, while inter-robot communication is performed on open channels without synchronization, minimizes bandwidth requirements while maintaining coordination effectiveness.

3.2 Environment representation

The environment representation is defined by the use of a voxel discretization of the 3D space. Each voxel has the same size L and corresponds to a discrete volume element of the known asset. By changing the L parameter, we improve the accuracy of the coverage tracking while increasing the computational requirements of the system. Starting from a 3D mesh or a point cloud, the system incorporates this information into the voxel grid structure by associating each point p with a voxel. The indexing is based on an intrinsic relationship where the index i=$[i_x, i_y, i_z]$ is obtained by the floor operator i= $\frac{p}{L}$. The system uses three distinct types of maps, each serving a specific purpose in the coverage framework. The **Observation map** shown in Figure 3.4 acts as a tracking map for the coverage progress throughout the mission. This map presents information about voxels status and the face status for each of voxel's six face. The face status can be one of four states:

- **UNKNOWN** if uninitialized
- **UNOBSERVED** if observable but not yet observed
- **OBSERVED** if successfully observed
- **OBSTRUCTED** if blocked by other voxel

Based on the faces status, a voxel can be:

- **UNOBSERVED** when has at least one face unobserved
- **OBSERVED** when has at least one face observed
- **UNREACHABLE** when all faces arr obstructed

.

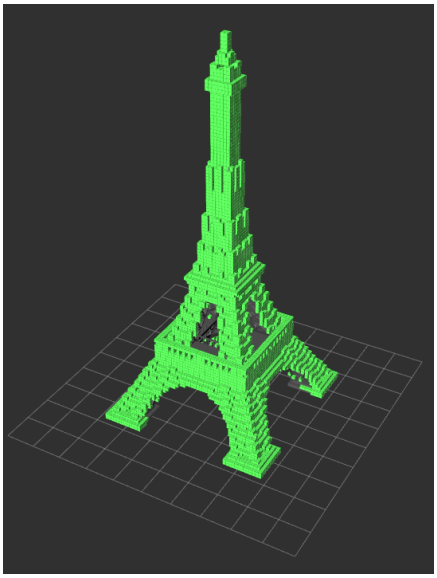

Figure 3.2: Region of interest

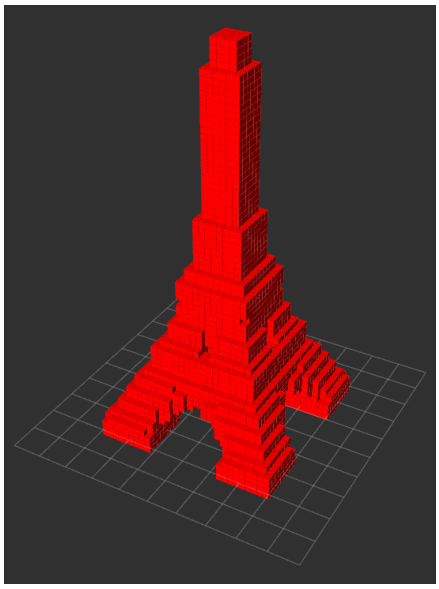

Figure 3.3: Avoidance map

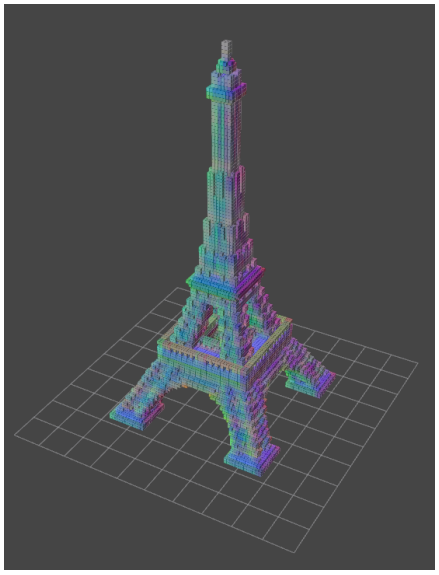

Figure 3.4: Observation map

Figure 3.5: The three different types of maps used by our method

At the beginning of exploration, all voxels are set as UNOBSERVED or OBSTRUCTED based on neighboring voxels and respectively the voxels' face status. When a robot completes a task, the ground station performs ray-tracing to identify the observed voxels and changes their status based on that evaluation.

The **Region of interest map** shown in Figure 3.2 is a crucial subset of observation map that is used for the definition of the target area for coverage. Using a PH-Tree data structure, the system stores all voxels within the ROI to enhance performance regarding spatial queries for task generation.

One of the main aspects computed on the ROI is the observability check for all voxels within the ROI. In some cases, voxels might be buried under other voxels or located in unreachable hollow volumes. Through the use of a sampling technique of viewpoints on the bounding box of the ROI, the system checks if the voxels are observable by tracing rays from the closest viewpoint to the voxel and checking if there is line-of-sight.

In this way, the number of voxels N_{PH} is less than the number of voxels present on the observation map N_{map}, and the ROI map results lighter than the other one.

The last map used is the **Avoidance map** shown in Figure 3.3, and it is used for the design of collision-free path planning. Due to this purpose, the avoidance map is an independent voxel grid containing N_{avd} voxels that presents a coarser resolution with respect to the observation map. This resolution difference can considerably reduce memory footprint while maintaining sufficient detail for safe navigation. The typical configuration uses $L_o = 0.2$m for the avoidance map compared to L = 0.1m for the observation map, resulting in an 8:1 reduction in the number of voxels.

The construction process begins by including all voxels from the observation map in the avoidance map, then iteratively adding additional voxels by considering all twenty-six possible neighbors to each already present voxel. This expansion operation is repeated until a desired "thickness" is achieved between the surface of the observation map and the avoidance map, creating a safety buffer that accounts for robot dimensions and positioning uncertainties.

3.3 Task Assignment Strategy

The task assignment strategy has been designed to work in multi-robot coverage missions and to operate entirely online, adapting to changing conditions and robot availability during the mission.

When the ground station identifies K_{PH} nearest unobserved voxels to each robot's current position using PH-Tree data structure, the task assignment process begins.

Using the spatial query defined by the PH-tree, the assigned tasks are

feasible for the robot and contribute to homogeneous coverage progress.

Each task is defined and includes several key components that guarantee sufficient information for individual robots to plan their paths. A task consists of:

- a target voxel index
- one of voxel's face that requires observation
- a set of viewpoints that provide a LOS to that face

The presence of multiple viewpoints instead of just one enhances flexibility in path planning and enables the robot to choose the most suitable one with respect to its current state and local constraints. The technique used to obtain different viewpoints is based on using a ray-tracing process from the voxel face to be observed.

As a first step, the system traces rays from the target voxel in different directions and evaluates potential viewpoints based on multiple criteria: the viewpoint must be obstacle-free, there must be line-of-sight from the voxel to the viewpoint, and the distance from the voxel center to the viewpoint must be within the sensor's operational range.

The ground station stores the tasks assigned to each robot in a buffer with a limited capacity of 20 to obtain several advantages such as avoiding redundant evaluations when a potential task is already in the robot's buffer, rejecting tasks that have already been discarded, and enabling efficient task sorting.

The parameter used for task sorting is a cost function c(w,r) that evaluates the suitability of a viewpoint w for robot r. It takes into account factors such as distance, heading alignment, and accessibility.

The function used is the following :

$$c(w, r) = 0.6(0.1\Delta p_x + 0.1\Delta p_y + 0.8\Delta z) + 0.4\frac{|\Delta\theta|}{\pi}$$

where:

- Δp is the difference between viewpoint position and robot position
- $\Delta\theta$ is the difference between viewpoitn's heading and robot's heading

The task assignment strategy guarantees robustness against robot failures and handles dynamic conditions thanks to several mechanisms applied at each iteration. If a task is completed, the ground station updates the observation map and assigns new tasks to keep the system working.

If a robot reports a task as aborted, the ground station assigns another one or modifies task parameters to improve feasibility.

For a better comparison with the other two techniques, we have decided to use the actual framework while changing the task manager strategy, guaranteeing that all techniques work with the same path planner and the same failure manager.

3.4 Individual Robot Path Planning

The path planning component used by the robots is a distributed solution that enables each robot to find a path to assigned viewpoints while coordinating with other robots to avoid collisions and improve system performance.

Moving the planning process to the robots instead of the ground station reduces computational effort and provides robots with the flexibility to adapt quickly to local conditions and dynamic environments.

The algorithm used to find the path is RRT*, which has been adapted for coverage purposes. The algorithm starts with the creation of a random tree that grows until a viewpoint is reached. Each time a new node is added to the tree, the robot checks if there is line-of-sight between it and the target viewpoints. In this way, the system is capable of identifying the most accessible viewpoint quickly.

Once a path is obtained, the system improves path quality by performing iterative pruning to remove redundant waypoints when direct line-of-sight exists between non-consecutive waypoints. Collision avoidance is implemented through multiple layers of checking. As a first step, the algorithm verifies if there are collisions with the avoidance map derived from the 3D model, ensuring that there are no intersections of the path with obstacles.

Subsequently, the system considers the current pose and planned paths of other robots and estimates if there are any conflicts during trajectory execution.

If during movement the robot encounters an unpredictable object or intersects the path of another robot, the system is able to stop the robot and request a new task. In this way, the coverage process does not need to stop and can continue efficiently.

Path execution is managed through trajectory generation that considers the vehicle's dynamic constraints and capabilities. Using parameters such as maximum velocity and acceleration limits, it generates a trajectory that guarantees effective observation of the assets to be explored.

Chapter 4 NBV technique

Starting from the implementation developed by [3], we have readapted it to work with known assets in contrast to unknown ones. This technique is part of the vast area of sampling-based methods that rely on sampling new points in the configuration space. The state-of-the-art technique proposes the use of a receding-horizon strategy which samples points randomly and selects the best one by considering the associated gain regarding the new knowledge acquired.

The main aspect of this algorithm is to generate an exploration path composed of viewpoints and to choose the first viewpoint of the best path as the most advantageous. The use of continuous feedback guarantees robustness against errors and enables rapid assessment of the remaining area to explore. This method is based on the use of an occupancy map in which the environment is depicted through small volumes, denoted 'voxels', each volume having an associated observation state. Typically, we have three types of voxels:

- V_{free}: free volumes which can be used to generate the path
- V_{occ}: occupied volumes representing objects
- V_{res}: residual volumes that cannot be associated with any of the previously mentioned classes

When proceeding with exploration, we start from all volumes as V_{unm} (unmapped volumes) and set an observation state for each voxel. The exploration ends when $V_{free} \cup V_{occ} = V \setminus V_{res}$ [3] and the drone returns to the starting station.

4.1 SOTA Gain Computation

The gain is determined by counting the amount of unmapped space that can be explored if the robot reaches the sampled point. In the literature, there are many formulas used to exploit this gain, which differ based on which aspects are prioritized. The method analyzed as a starting point uses the following formula to compute the gain for the sampled point n_k :

$$Gain(n_k) = Visible(M, \xi_k)e^{-\lambda c(\sigma_{k-1}^k)}$$

where:

- $Visible(M, \xi_k)$ is the set of unmapped and visible voxels from configuration ξ_k defined as pose and heading of the robot and that belong to the occupancy map M
- $c(\sigma_{k-1}^k) \in \mathbb{R}_+$ is the cost of following the path σ_{k-1}^k from the configuration ξ_{k-1} to the configuration ξ_k. It is defined as the Euclidean distance between the coordinates of the starting and end configurations.
- $\lambda \in \mathbb{R}_+$ is a coefficient which penalizes high path costs [3]

4.2 Implementation Approach

Our implementation maintains the main paradigm of the state-of-the-art approach while introducing modifications to adapt it to explore known structures. As described previously, we chose to modify the task generation step to enable a fair comparison of the three techniques described.

For this purpose, we designed a class that defines the main functions of our receding-horizon NBV technique and a main function that serves as a wrapper for task generation.

The concept behind the nbv implementation is illustrated in **Figure 4.2**. The system is defined by a pipeline of interconnected modules, each responsible for exploiting specific aspect of the optimization process while maintaining bidirectional communication to achieve optimal performance. The architecture follows a top-down hierarchical approach where each block functions as a consumer of the previous block's output and provides feedback. The three main blocks are:

- **Voxel Candidate Selection** block focuses on high-level decisions about where to sample and which candidates to evaluate. Thanks to the use of adaptive sampling strategies that use performance metrics to enhance the sampling approach in real-time, we reduce the required sampling points from 50-100 to 15-30 points with respect to the original approach.
- **Gain Computation** block concentrates on the evaluation of viewpoint quality through a multi-factor evaluation mechanism using a `ThreadPool`[1] for parallel processing.
- **Spatial Caching System** block operates as the system's memory component, storing information about successful sampling regions to influence future decisions.

[1] `ThreadPool` is a design pattern that manages a pool of worker threads to execute tasks concurrently.

The **Voxel Candidate Selection** block serves as the coordinator of the entire system, managing the flow of information between the lower-level computational components and the higher-level decision-making processes. The core innovation in the system is the parallel processing capabilities, enabled by the `ThreadPool` architecture, which guarantees concurrent execution across all three blocks.

This approach allows the Voxel Candidate Selection to the execution of different sampling approaches simultaneously while the **Gain Computation** is performed in parallel. Another aspect that improves execution is the use of a **Spatial Caching System**, which operates as a continuous background process, updating and maintaining cached gain computations and successful sampling regions without blocking the primary computation pipeline (Figure 4.1).

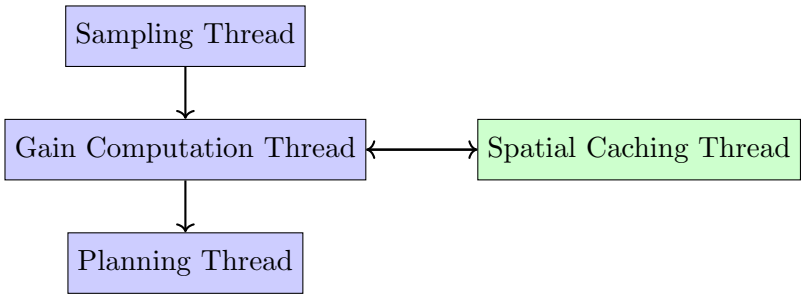

Figure 4.1: Thread execution pipeline showing parallel processing architecture

The bidirectional arrow in the diagram emphasize the communication protocols defined between the blocks. The Voxel Candidate Selection receives information and performance metrics from the lower blocks and adjusts its sampling strategy and selection criteria based on real-time performance. The Gain Computation block communicates with the upper and lower blocks, exchanging specific data types:

- **From Voxel Candidate Selection**: received sampled viewpoint candidates defined by position coordinates and orientation
- **From Spatial Caching System** : received previously computed gain values usign viewpoint hash keys
- **To Spatial Caching System**: sends computed information gain values

Due to this structure, we generate a feedback loop where frequently accessed computations are automatically cached, and the cache contents influence future sampling decisions.

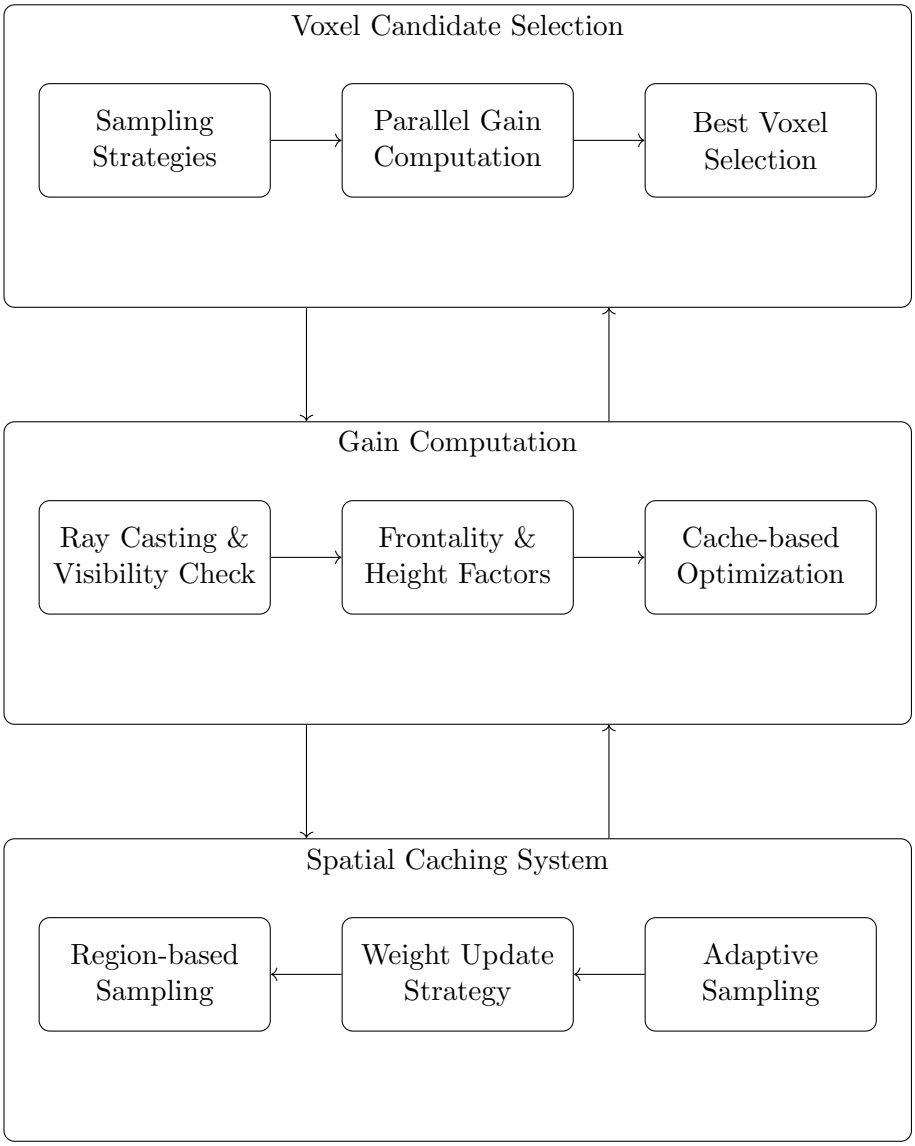

Figure 4.2: Architecture of improved NBV system

This modular design ensures that each block can be optimized or replaced without affecting the others, provided the communication interfaces are maintained. It enables the system to scale dynamically based on available computational resources,with each block adjusting its internal parallelization level according to the system's current capacity. In the following subsection, the sampling strategy used to sample new viewpoints will be presented.

4.3 Sampling strategy

Regarding the sampling technique, the function implements an adaptive sampling strategy to search promising viewpoints in the 3D space. Instead of using a uniform sampling of the environment around the asset, this method focuses its search on areas that yield high information gain. To maintain in memory the most valuable regions, we design a sampling-cache data structure that stores a collection of sampling regions. Each region is defined by a structure composed of:

- A center point
- A radius defining the region extension
- A weight indicating the sampling priority

For point sampling, we used a probabilistic weight-based selection define by: 70% of sampling using a cache-base weighted approach that selects promising spatial regions from the sampling cache and generates new viewpoint inside these high-success regions,while 30% of sampling using the **Algorithm 1** for broader exploration. This algorithm sample a new point selecting between four strategies based on a random probability value: δ-perturbation around the previous best viewpoint, circular sampling around the object, side-based sampling from cardinal directions, or distant sampling for wide-area coverage.

The weighted approach chooses a region based on the regions weight through weighted random selection. The region with the higher weight is chosen, and a point is sampled within it. We save the polar coordinates of the point in the x-y plane and the height is determined by evaluating the optimal value to achieve the highest probability of observing the most unobserved voxels. In this way, we create a cylindrical sampling volume around each region center and concentrate our research in promising regions while still allowing for broader exploration. As a first step, this cache is initialized by creating different regions defined as:

- **Continuity Region** with center at the robot's current position, weight of 0.5 (lower sampling probability in the weighted random selection) and radius of 2 meters
- **Object-centric Regions** strategically positioned around the object's bounding box at a distance calculated as $d_{optimal} = \alpha \cdot \max(w_{obj}, h_{obj}, l_{obj})$ where $\alpha = 1.5$ is a scaling factor and w_{obj}, h_{obj}, l_{obj} are the object's width, height, and length respectively. These regions are placed at 45° intervals around the object's perimeter when the object geometry permits circular navigation, otherwise positioned at the corners and midpoints of the bounding box faces. Each regions has a weight of 1.0 and a sampling radius of $0.5 \cdot d_{optimal}$.

Each region is selected randomly, where a weight of 0.5 is assigned to the Continuity Region and 1 to each Object-centric Region.

In this way, we ensure the provision of good viewpoints of the unexplored object while maintaining some sampling points near the current drone's position.

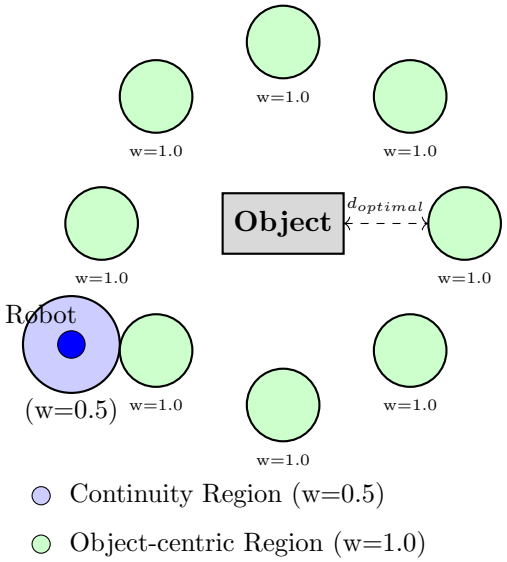

Figure 4.3: Spatial sampling cache regions showing continuity region around robot position and object-centric regions distributed around the target object

Based on the exploration result, the Sampling Cache is modified through the following steps, improving the performance of the next sample step. The algorithm adds a new region to the cache when points are sampled from the 30% alternative sampling strategy (Algorithm 1) and yield higher information gain than a threshold. The system sets the weight of the new region proportional to the gain value and updates the total weight to maintain probabilistic selection. The weight of existing regions are updated based on how many high-gain viewpoint are sampled into their boundaries. If the updated weight is lower than a threshold, it means that region has been successfully explored and it is removed from the cache. Subsequently, high-gain viewpoints that have generated new sampling regions serve as starting points for selecting new viewpoints, which will be located in close proximity to these points and will also reduce the path length required to reach them.

When the adaptive sampling is not applied and the cache is empty, the system falls back to using the **Algorithm 1**, which implements several sampling strategies chosen by random probability.

Algorithm 1 Sample New Point

$Box \leftarrow$ GetBoundingBox
$C_{object} \leftarrow (Box_{first} + Box_{second})/2$ ▷ Object center
p $\leftarrow$ GetRandomProbability
if $n_{previous}$ available AND $p < 0.2$ **then** ▷ Strategy 1: δ sampling
    return $n_{previous} + \delta$
else if $0.2 \leq p$ **and** $p < 0.8$ **then** ▷ Strategy 2: Circular sampling
    $n_{sample} \leftarrow$ SamplePointAroundObject
    ComputeOptimalHeight(n_{sample})
    return n_{sample}
else if $0.8 \leq p$**and** $p < 0.94$ **then** ▷ Strategy 3: Sampling on object sides
    $n_{sample} \leftarrow$ SamplePointOnRandomSide
    ComputeOptimalHeight(n_{sample})
    return n_{sample}
else ▷ Fallback Strategy: More distant sampling
    $n_{sample} \leftarrow$ SamplePointDistant
    ComputeOptimalHeight(n_{sample})
    return n_{sample}
end if

Based on the value of the uniform probability $p \in [0, 1]$, one of the following technique is applied:

- **Best Viewpoint Perturbation** ($p < 0.2$, 20% chance): samples new viewpoints by adding small random perturbations to the current best viewpoint to achieve fine-grained exploration around the already successful areas.
- **Circular Sampling Around Object** ($0.2 \leq p < 0.8$, 60% chance): samples points in a circular manner along the x-y plane around the object using a beta distribution approximation to favor optimal viewing distances.
- **Side-Based Sampling** ($0.8 \leq p < 0.94$, 14% chance): chooses one of the four sides of the object (North,East,South,West) and samples different viewpoint to provide sufficient views ensuring comprehensive coverage
- **Fallback Wide-Area Sampling** ($p \geq 0.94$, 6% chance): uses a wider circular pattern to sample more distant viewpoints and guarantees that the system does not get stuck in local optima during the exploration.

In **Figure 4.4**, it is possible to see one visual representation of the sampling stage of our NBV implementation during the inspection of Eiffel

Tower. The green points represent the viewpoints that surround the entire object with higher concentration in regions with the highest probability of sampling points with good information gain within them. In this case, the areas around the back of the Eiffel Tower present the regions characterized by the highest values of the probability weight and consequently the highest number of sampled points.

4.4 Gain Computation

The other main function used in this technique is the computation of the Information Gain associated with each of the sampled viewpoints. This function is crucial for making informed decisions about where the robot should move to maximize the efficiency. As shown is the **Algorithm 2**, this is a multi-factor evaluation mechanism designed to works with the Cache system. As a first step, the algorithm uses a cache mechanism to avoid redundant calculation and improve computational efficiency. We use mutex protection to ensure threads safety in multi-thread environments allowing concurrent access, and implement a size limit with automatic cleanup that removes older entries when the cache exceeds a fixed-limit of items to prevent unbounded memory growth. After the definition of the caching structure, the function needs to simulate the position of the drone and define the orientation to evaluate the ray-cast. For this reason, it evaluates the orientation needed to visualize the object by considering the nearest unobserved voxel position with respect to the sampled point and returns the angle ψ on the x-y plane. The value of ψ is obtained by the following formula:

$$\psi = \arctan 2(y_{voxel} - y_{drone}, x_{voxel} - x_{drone})$$

where (x_{voxel}, y_{voxel}) is the nearest unobserved voxel coordinates and (x_{drone}, y_{drone}) is the drone position in the x-y plane. Using this angle, the ray casting process is initialized and the main part of the function starts measuring important information. The ray casting process simulates the robot's depth camera by casting multiple rays according to the camera's field of view (FOV) and resolution. It determines which voxels would be hit by each ray when the drone is in that pose and records the face that would be observed. The function uses the set of observed voxels to define the $Visible(n_k)$ as in the article [3]. The use of the voxel grid's hash function to generate unique identifiers for each voxel coordinate enables us to eliminate duplicates in that set, ensuring that each voxel is counted once and preventing double-counting of information. Each voxel is classified based on its observation status, which can be:

- **UNOBSERVED**: voxels that haven't been seen before
- **OBSERVED**: voxels that have already been seen

To evaluate the information gain IG, the function uses is the following:

$$IG(n_k) = N_{unobserved} \cdot F_{frontality} \cdot F_{height} \cdot F_{distance}$$

The number of unobserved voxels $N_{unobserved}$ is one parameter of the final formula and is multiplied by specific factors.

One of these factors is the frontality factor $F_{frontality}$, which ensures the robot faces the object directly for optimal observation quality. It is defined as:

$$F_{frontality} = \max(0, \cos(\theta_{camera-object})) = \max\left(0, \frac{\vec{d}_{camera} \cdot \vec{d}_{object}}{|\vec{d}_{camera}| \cdot |\vec{d}_{object}|}\right)$$

where $\vec{d}_{camera}$ is the camera's forward direction and $\vec{d}_{object}$ is the direction from camera to the object. Its value is between 0 and 1 and depends on the cosine value between the camera's forward direction and the object direction, In this way, the drone is force to see the voxels frontally.

The height factor F_{height}:

$$F_{height} = e^{-\alpha \cdot |z_{sample} - z_{optimal}|}$$

that addresses energy consumption. For our comparison, we consider vertical movements as the most energy-consuming action, generally consuming 3-5 times more energy than horizontal movements. This factor considers the position along the z-axis of the sampled point and an optimal height $z_{optimal}$ to see the object, considering only the x and y coordinates of the sampled point. This value is calculated by considering the lateral distance $d_{lateral}$ from the sampled point and the object's center on the x-y plane:

$$z_{optimal} = z_{obj} + \beta \cdot d_{lateral}$$

with $d_{lateral} = \sqrt{(x_{sample} - x_{obj})^2 + (y_{sample} - y_{obj})^2}$ and β the height scaling coefficient that indicates how much the viewing height increases with respect to the lateral distance. Starting from the object center's z coordinate, it adds a height proportional to the lateral distance, creating a cone-like optimal viewing surface. This shape has been chosen to address two main aspects: **Energy optimization** and **Camera angle optimization**. The cone keeps the drone at lower altitudes when the drone is close to the object, avoiding vertical movements that consume more energy, and guarantees adequate viewing angles to the object's surface.

As a final factor, we consider the distance factor $F_{distance}$:

$$F_{distance} = e^{-\gamma \cdot \max(0, d_{lateral} - d_{optimal})}$$

where γ is the distance penalty coefficient and $d_{optimal}$ is the optimal viewing distance, penalizes viewpoints that are too distant from the object being explored. When all these factors are evaluated, the function associates the computed value with the sample point and saves everything in the cache memory.

Algorithm 2 Compute Gain

$Cache_{key} \leftarrow$ CreateCacheKey(position, yaw)
if $Gain_{cache}$ contains $Cache_{key}$ **then return** $Gain_{cache}[Cache_{key}]$
end if
$Box \leftarrow$ GetBoundingBox
$C_{object} \leftarrow$ GetObjectCenter(Box)
$v_{center} \leftarrow C_{object} -$ position
$\psi \leftarrow \text{atan2}(v_{center}(y), v_{center}(x))$
$q \leftarrow$ Quaternion(ψ, z)
SetPoseDepthCamera(q, position)
$hit_{indices}, hit_{faces} \leftarrow$ DepthCameraCastRays
if $hit_{indices}$ is empty **then**
    $Gain_{cache}[Cache_{key}] \leftarrow 0.0$
    return 0
end if
$n_{unmapped} \leftarrow 0, n_{observed} \leftarrow 0$
for index in $hit_{indices}$ **do**
    hash $\leftarrow$ HashIndex(index)
    if $V_{observed}$ contains hash **then continue**
    end if
    add(hash) to $V_{observed}$
    if ContainVoxel(index) **then**
        status $\leftarrow$ GetStatus(index)
        if status = UNOBSERVED **then** $n_{unmapped}++$
        else$n_{observed}++$
        end if
    end if
end for
$k_f \leftarrow 1.0$ ▷ Frontality factor
if $v_{center} > 0$ **then**
    $v_{forward} \leftarrow q * x$
    $k \leftarrow v_{forward} \cdot v_{center}$ ▷ Cosine of the angle between the two vectors
    $k_f \leftarrow \max(0.1, k)^2$
end if
$l_{dist} \leftarrow$ position $- C_{object}$
$h_{optimal} \leftarrow$ ComputeOptimalHeight($l_{lateral}$)
$k_h \leftarrow$ ComputeHeightFactor(position, $h_{optimal}$, l_{dist})
$k_d \leftarrow$ ComputeDistanceFactor($position$)
gain $\leftarrow k_{unmapped} \cdot n_{unmapped} \cdot k_f \cdot k_h \cdot k_d$
$Gain_{cache}[Cache_{key}] \leftarrow$ gain
return gain

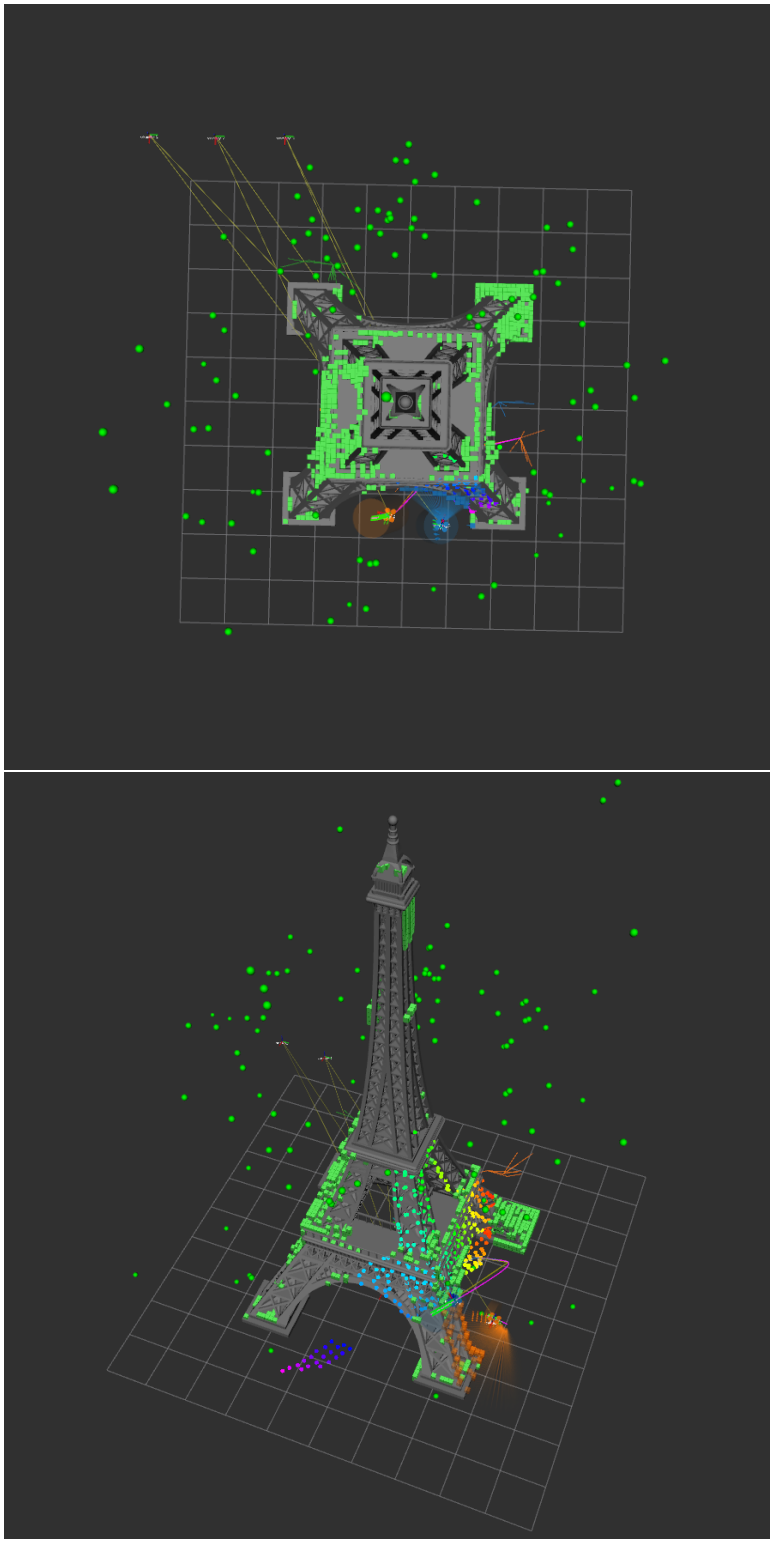

Figure 4.4: Sampling visualization during Eiffel tower's exploration. Green points show the sampled viewpoints generated by the adaptive sampling

4.5 Comparison

In this subsection, we will analyze the main differences between our implementation and the state-of-the-art approach [3]. The comparison will highlight the improvements and their positive impacts.

Regarding the sampling strategy used in both implementations, they present significant differences that enhance the system by preventing wasted time in finding good viewpoints. In the state-of-the-art technique, they used a simple sampling strategy that considers a uniform distribution around the exploration environment. In this way, the sampling does not adapt based on previously sampled viewpoints and cannot guarantee finding a good one at the end of one iteration.

However, our strategy used obtain information from previous successful viewpoints and improves future sampling by generating new regions that could be optimal for sampling. This knowledge is saved in memory and updated throughout the mission. Another aspect is the use of performance metrics to adapt the sampling strategy in real-time.

From a computational perspective, we can say that our technique prevents getting stuck in local optima and reduces the computation time needed per iteration since it requires fewer sampling points than the original to find one that is good enough.

The following table lists the performance metrics for both techniques:

Metric	**Original Approach**	**Our Approach**
Samples to find sufficient viewpoints	50–100	15–30
Success rate per sample (%)	15–25	45–65
Computation time (s)	0.5–2.0	0.15–0.5

Table 4.1: NBV Performance Comparison Metrics across mockups asset coverage using single robot

Regarding gain computation, the original version used only two metrics to define the value of information gain: $Visible$ and the cost c to reach the sampled point from the current drone position.

In this way, it considers only seeing as many unobserved voxels as possible and reducing the distance that must be traveled, without considering energy consumption reduction. Our implementation instead adds two other factors: energy consumption and frontality of viewing.

In this way, the ground station chooses a viewpoint that requires less energy consumption to reach and ensures stable visualization of the final viewpoint, discarding those with unstable positions during path travel. Another differing aspect regards the processing architecture used.

The original version uses sequential processing where it evaluates one viewpoint at a time, and performance scales linearly with problem size. Our

parallel processing architecture solution enables the system to use all available processing power and evaluate multiple candidates simultaneously, distributing computation across all available cores. The following table shows a metric comparison of the techniques:

Metric	**Original Approach**	**Our Approach**
100 viewpoints evaluation (s)	15-25	2-4
Peak Memory usage (GB)	1.6-3.2	2.4-3.2
CPU utilization	12-25%	85-95%

Table 4.2: NBV Performance Comparison Metrics

Other advantages derive from the use of a caching system that reduces average computation time and eliminates redundant calculations. In the original version, since there is no caching storage, the system recomputes the gain for viewpoints already analyzed, evaluating them multiple times and losing computational time that could improve the algorithm's performance.

Using caching, we scale our computational power to what is available on the computer where the code is run, and it ensures higher quality viewpoint selection since it enables us to save important information for sampling strategy adaptation.

Using storage limitation and dynamic management, we prevent unbounded memory growth that can block the entire simulation, and we automatically manage cache content by eliminating unnecessary information.

These improvements collectively transform the NBV approach from a general exploration tool into a specialized, efficient, and robust system for known asset inspection and mapping, making it particularly suitable for inspection tasks.

Chapter 5 Frontier-based technique

For the second comparison technique, we consider as a starting point the frontier-based technique developed by [2]. This approach belongs to the frontier-based technique in which the next viewpoint is chosen by finding the optimal boundary between unknown and known space.

This method presents an innovative collector strategy capable of achieving global exploration and map creation. The state-of-the-art technique stores and validates the frontiers detected during exploration.

In this way, they balance the exploitation of the known map with the exploration of unknown areas [2].

As with the previously discussed NBV technique, this technique starts by considering the entire environment as unknown and, during the exploration, associates to each voxel a status that can be either free or occupied.

The general task terminates when the total amount of the occupied or free voxels corresponds to the entire set of starting voxels, with the exception of some voxels considered as residual that remain unexplored.

The main goal of the technique is to find the frontier with the highest information gain by identifying the frontiers and clustering them to reduce the amount of data to be stored.

Through the use of continuous frontier validation, it guarantees robustness against map updates and enables efficient assessment of the remaining area to explore.

The basic concept used regards the definition of frontier, defined as the free voxels with at least one unknown neighbor. We collect all of the found frontier in the F set defined as follows:

$$F = \{v_f \in V_{free} : \exists \text{neighbor}(v_f) \in V_{un}\}$$

where v_f depicts a frontier voxel, V_{free} is the set of free voxels, V_{un} is the set of unknown voxels, and neighbor(v_f) represents the neighboring voxels of v_f in the 3D grid. Subsequently, the entire set is checked by considering the previous version of the set F and computing the intersection of the two sets, defined as F_l.

On this set, a mean shift clustering algorithm is applied to reduce computational complexity and remove redundant exploration targets. The two

parameters that characterize this algorithm are the kernel and the bandwidth. The Kernel is a selected function used for instantiate how neighboring points affect the clustering process. In our case,we choose a Gaussian kernel. On the other hand, bandwidth is a parameter that defines the size of the neighborhood that is considered during clustering. The choice of the bandwidth is relevant since it balances the computation time and the desired outcome. The kernel regulates the dimension of the "window" over which the mean is calculated. The frontiers are saved in the collector within the set F_{GC}.

In order to choose the best frontier, the algorithm associates to each clustered frontier an information gain as a measure of the unexplored region would be visible from a viewpoint positioned at 1 meter in the same direction of the unobserved face of the best frontier center voxel. It is defined as:

$$IG(f) = \frac{N_{unknown}}{N_{total}}$$

where $N_{unknown}$ and N_{total} are the number of unknown voxels and the number of voxels inside a cubic sampling region centered in the frontier center. This region is define by:

- The center located in the centroid of the frontier cluster
- The edge length equals to 2 time the camera sensor range

The collector filters the frontiers by discarding those that have a lower information gain value compared to a certain threshold.

It also updates the information gain considering the distance between the drone and the frontier, and removes the frontiers already visited.

The frontier with the highest gain is chosen and used by the path planner as the next viewpoint.

5.1 Implementation Approach

Our implementation follows the main paradigm of the state-of-the-art approach with significant modifications to enable exploration of known assets and improve computational efficiency.

As done for the NBV technique, we chose to modify only the task generation step of [1] to ensure a fair comparison between the three techniques. The designed class outputs a task structure containing the voxel to be considered for next viewpoint selection.

Figure 5.1 illustrates the structure of the frontier-based implementation, defined by a pipeline of interconnected modules responsible for exploiting specific aspects of the process. The modules are:

- **Frontier Detection & Clustering** module focuses on identifying and clustering frontier voxels

- **Information Gain Computation** module concentrates on the mathematical evaluation of frontier quality
- **Filtering & Validation** module operates as memory and validation component of the system
- **Selection & Navigation** module choose the final frontier

The Frontier Detection and Clustering module begins by processing each voxel from the PH-tree structure, determining whether it represents a valid frontier point with proper face status or neglecting it.

After frontier creation, it employs an optimized mean-shift algorithm that reduces the computational complexity of subsequent operations by allowing the system to consider groups of frontiers instead of individual ones. In this way, the number of elements to process moves from $|F|$ individual frontiers to $|C|$ frontier clusters, where $|C| \ll |F|$, reducing the computational effort required by information gain evaluation.

The Spatial Grid optimization block enhances clustering performance using a spatial hash grid that accelerates neighbor search, reduces memory access patterns, and enables efficient batch processing of frontier candidates.

Subsequently, the Information Gain Computation module evaluates the information gain value of each frontier.

The use of adaptive sampling enables the system to concentrate computational resources in areas likely to yield sufficient information to establish the potential of the considered frontier.

The system performs visibility analysis to determine line-of-sight between the robot position and frontier location. Finally, energy and distance factors are evaluated to ensure the algorithm considers energy optimization during exploration. The Filtering and Validation module ensures that only feasible and valuable frontiers proceed to the selection phase. It applies various analyses, including obstruction-free verification and confirmation that information gain exceeds a fixed threshold. The final module handles the decision-making phase by evaluating a final utility factor and sorting all available frontiers based on this parameter. The chosen frontier is converted into a task structure and sent, like the other two techniques, to one of the available robots.

5.2 Frontiers Detection and Clustering

The main step in our implementation is the frontier detection that presents fundamental changes from traditional approaches, implementing a two-phase strategy defined as: individual frontier creation and advanced clustering algorithm.

The first phase focuses on checking each voxel present in the PH-tree structure and creating individual frontiers from them. This approach differentiates from Caiza's [2] method since it does not consider the boundaries between free and unknown space but between seen and unseen space within the reconstructed asset.

As shown in **Algorithm 3** (lines 1-3), the first step is to check if the `VoxelGrid` contains voxels related to the reconstructed asset. Another important step is related to the initialization of the face status if not already defined (lines 4-8). In this way, the system associates to that voxel a status based on an analysis of the neighborhood voxels. If a neighboring voxel exists in the grid, the face is marked as OBSTRUCTED; otherwise, it is marked as UNSEEN, indicating a potential observation target. After this step, the system creates an individual Frontier object for each voxel using the **CreateNewFrontier 4** method. A frontier object is defined by the following characteristics: position, information gain, boolean value that stores the validity of the frontier, and a list of the voxels inside the frontier (lines 9-12). The initial individual frontier creation acts as an intermediate representation that store spatial information before the clustering algorithms groups adjacent frontiers into clusters.

After the initialization of all individual voxels, the system proceeds with the use of an advanced mean-shift clustering algorithm designed for real-time robotic applications.

To manage the computational complexity for larger numbers of frontiers, the algorithm applies a pre-filter stage that removes spatially redundant frontiers (those within 0.3 meters of each other) ensuring that the overall performance remains sufficiently good while maintaining representative coverage of the frontier space. The 0.3-meter threshold was selected based on empirical analysis across our test assets. This value provides optimal performance characteristics:

- **Computational efficiency**: Reduces frontier processing load by 45-65
- **Coverage preservation**: Maintains >96% of theoretical maximum information gain
- **Scale adaptability**: Works effectively across asset sizes from 2m to 300m
- **Voxel relationship**: Equals approximately three times the voxel size (0.1m), ensuring geometric consistency

Algorithm 3 Find Frontiers

```
1: F ← ∅                                          ▷ Initialize empty frontiers list
2: V_indices ← GetVoxelIndicesPHTree
3: for v_i ∈ V_indices do          ▷ V_indices: set of voxel indices from PH-tree
4:     if VoxelGridContains(v_i) then
5:         voxel ← GetVoxel(v_i)
6:         if HasUnknownFaceStatus(voxel) then
7:             for face = 0 to 5 do
8:                 v_neighbor ← GetNeighbor(v_i, face)
9:                 if VoxelGridContains(v_neighbor) then
10:                    SetFaceStatus(voxel, face, OBSTRUCTED)
11:                else
12:                    SetFaceStatus(voxel, face, UNSEEN)
13:                end if
14:            end for
15:            UpdateStatus(voxel)
16:        end if
17:        f_new ← CreateNewFrontier(voxel)
18:        F ← F ∪ {f_new}
19:    end if
20: end for
21: return F
```

Algorithm 4 Create New Frontier

```
1: frontier.index ← voxel.index
2: frontier.position ← GetCenter(voxel.index)
3: frontier.information_gain ← 0.0                 ▷ Initially set to zero
4: frontier.valid ← true
5: frontier.last_update_time ← GetCurrentTime
6: frontier.voxels ← {voxel.index}                 ▷ Initialize with single voxel
7: return frontier
```

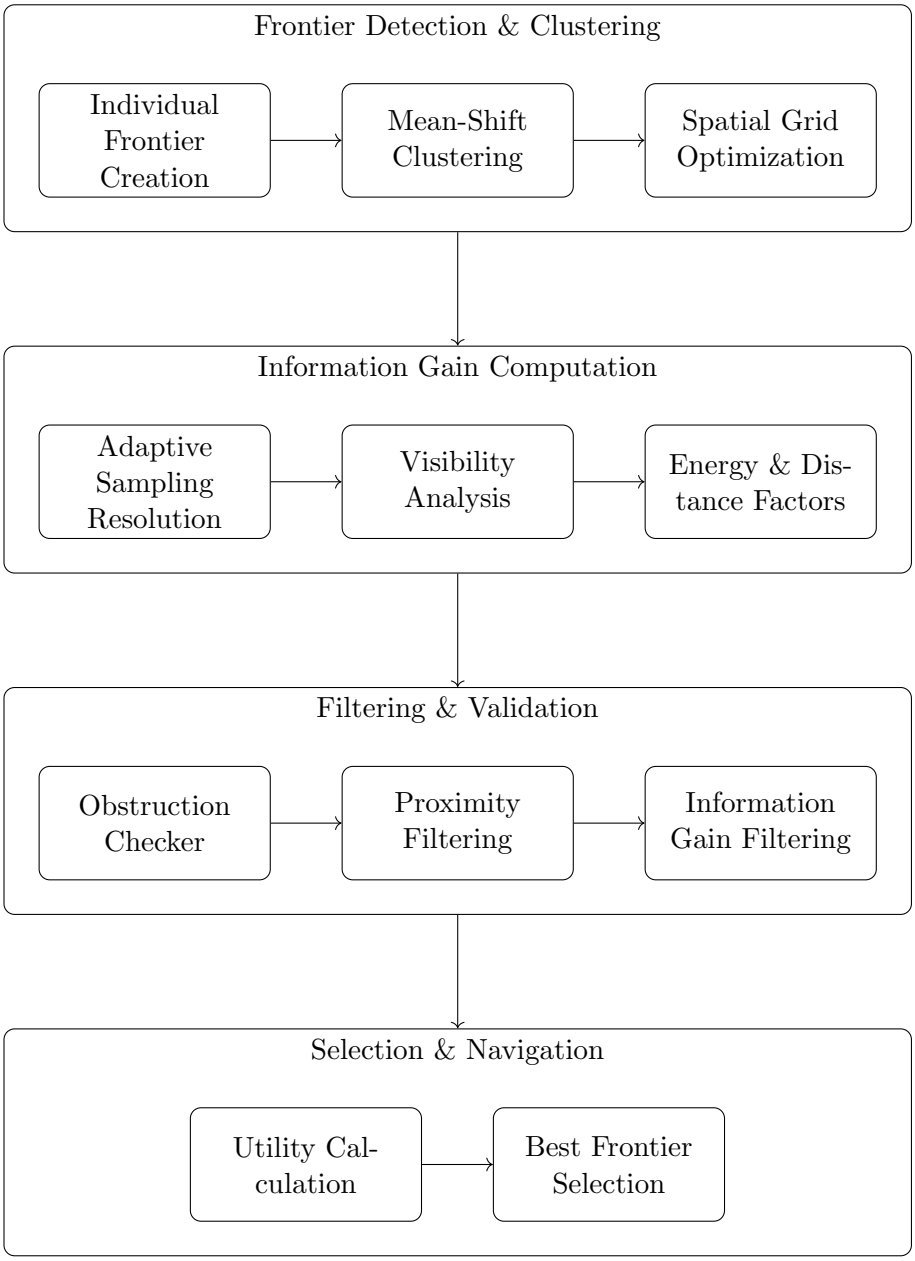

Figure 5.1: Architecture of improved Frontier-Based system

The first innovation is the use of a spatial grid system that guarantees a reduction of computational complexity for neighbor identification. This spatial grid is a hash-based data structures where each cell stores references to frontier points within its boundaries. Using a cell size 1.5 times the bandwidth, it ensures a good balance between grid resolution and search efficiency.

In this way, the neighbor search moves from a complexity of $O(n^2)$ brute-force comparison to a complexity of $O(n \cdot k)$ for grid-based lookup. The complexity reduction is achieved as follows:

- **Grid-based approach ($O(n \cdot k)$)**:
  1. Hash each frontier to its grid cell: $O(1)$ per point, $O(n)$ total
  2. For neighbor search, only check the $3 \times 3 \times 3 - 1 = 26$ surrounding cells
  3. Expected number of points per cell : $N_p = \frac{n}{|cells|}$
  4. Total complexity: $O(n \cdot 27 \cdot N_p) = O(n \cdot k)$

In a mean-shift iteration, the spatial grid accelerates neighbor identification and ensures that all relevant neighbors within the bandwidth are considered. The 3×3×3 grid search pattern ensures comprehensive neighbor coverage while maintaining computational efficiency.

Another update with respect to the original Caiza's [2] version is the kernel function. Their method uses a uniform kernel that considers all neighbors into the bandwidth in the same way, neglecting their relative distance. Our implementation uses a Gaussian kernel with adaptive weighting that optimizes convergence speed and cluster quality by giving more importance to closer neighbors.

The Gaussian kernel uses a cutoff at 2.5 times the bandwidth following the standard practice in mean-shift clustering to balance computational efficiency with clustering accuracy. The weight threshold of 0.01 orresponds to the 1% significance level commonly used in statistical analysis and eliminates negligible contributions that would not significantly affect the mean-shift computation but consume computational resources.

After convergence, the algorithm performs cluster assignment that prevents redundant cluster creation.

The final step is the construction of the frontier object from the clustering results with a proper validation.

The information gain associated to it corresponds to the sum of each frontier clustered and the center point is chosen as the average of the voxels position if it coincides with the position of a voxel or the nearest one.

An example of the clustering visualization is shown in **Figure 5.2**. In the picture are shown the top five frontier clusters ranked by information gain. Colors indicate frontier priority: Red (highest information gain), Orange (2nd highest), Yellow (3rd), Purple (4th), and Pink (5th).

During the exploration , frontiers are updated at each iteration following three phases:

- Validation phase: the system checks if in each existing frontier there are voxels remain UNOBSERVED. Frontiers containing voxels that have been explored are removed from the global frontier set.

- Integration phase: newly detected frontiers from the current iteration are pre-filtered and after integrated into the global set.
- The system applies Information gain recomputation for all valid frontiers based on current robot position and exploration state.

In this way, the system ensures that the frontier set remains current and computationally manageable throughout the exploration mission, adapting to the changing environment state as areas are progressively observed and mapped.

5.3 Gain Computation

The other main components of our implementation is the information gain computation that is characterized by different evaluation since it is based on factors specific for energy-efficient asset inspection. The formula used to evaluate the gain of a frontier is the following one:

$$IG(f) = k_g \cdot \frac{N_{unobserved}}{N_{total}} \cdot E_{factor} \cdot V_{factor}$$

where:

- k_g: knowledge gain coefficient
- $N_{unobserved}/N_{total}$: ratio of unobserved to total voxels in the frontier region
- Δz: vertical distance with respect to the robot position
- E_{factor}: energy factor considering altitude differences

  $$E_{factor} = e^{-\beta \cdot |\Delta z|}$$

  where β is the energy cost factor
- V_{factor}: visibility factor based on line-of-sight analysis

$$V_{factor} = \begin{cases} 1.0 & \text{if line-of-sight exists} \\ 0.5 & \text{if line-of-sight is obstructed} \end{cases}$$

The innovation in our approach is the consideration of the vertical movement cost. As in the NBV case, the major quantity of energy consumed by the drone is attributed to the vertical movements and for that reason we want to reduce the displacement along the z-axis and improve the movement within the plane parallel to the floor.

This value is used to compute the energy factor since it is defined as $e^{-\beta \cdot \Delta z}$ where beta is a constant factor. In this way, the gain will be lower than the expected one if is relative position is too far from the current robot position along the vertical axis. The other factor used is the Visibility factor that is a penalty factor if there is no line of sight between the robot position and the frontier center.

By using a ray casting simulation, the algorithm analyzes if there is a line of sight between the position of the drone and the frontier center.

As shown it the **Algorithm 5**, the algorithm defines a cubic sampling region with an extension equal to the parameter α where it samples point to define how many voxel are unseen inside. implements probabilistic culling of sample points that are distant from the frontier center.

Points beyond 70% of the maximum sampling radius are skipped with 66% probability, effectively concentrating computational resources on the most relevant sampling regions while maintaining statistical coverage. These threshold values have been empirically obtained trying to balance computational efficiency and sampling accuracy.

5.4 Collector filtering and frontier selection

After the computation of the information gains for each frontier, the collector saves all the data in a cache and checks all of them by a three-stage filtering. The first one is related to the obstruction of the frontier.

During the exploration some frontiers can become unreachable and for that reason it will delete all the frontiers that presents unreachable voxels within it. The second one is the proximity filtering that prevents the selection of frontiers too close to recently explored area. The last step is the check if the gain value is higher than a fixed threshold and remove from the memory the ones that don't respect this constraint. Finished the filtering step the collect select the best frontier using a utility factor that balances information gain with the travel length. Defined as:

$$U(f) = \frac{IG(f)}{d(r, f)}$$

where:

- $IG(f)$ is the information gain
- $d(r, f)$ is the Euclidean distance from the robot position r to frontier f The utility function uses total Euclidean distance $d(r, f)$ for travel cost estimation, while the information gain computation $IG(f)$ separately considers vertical displacement Δz for energy-specific penalties. This dual approach accounts for both path planning costs (total distance) and energy consumption patterns (vertical movement penalty),

providing a comprehensive evaluation that optimizes both travel efficiency and energy expenditure.

In this way, the final selection depends on the information gain but also to the distance since the desired behavior of this approach is the exploration of the drone along the object preferring exploring one area before without moving to another one and come back again since not fully explored.

After the selection the algorithm converts the frontier into a task structure as defined in [1] and sends it to the ground station node in order to continue the exploration of the asset.

Algorithm 5 Frontier Information Gain Computation

$voxelLookup \leftarrow$ CreateSpatialIndex($\mathcal{V}_{current}$)
for $f_i \in \mathcal{F}$ **do**
  $d \leftarrow \|f_i.position - p_{robot}\|_2$
  $\Delta z \leftarrow \|f_i.z - p_{robot}.z\|_2$
  $\gamma_{energy} \leftarrow \exp(-\beta \cdot \Delta z)$
  $p_{min} \leftarrow f_i.position - \alpha$
  $p_{max} \leftarrow f_i.position + \alpha$
  $N_{unknown} \leftarrow 0, N_{total} \leftarrow 0$
  for $(x, y, z) = p_{min}$ **to** p_{max} **step** $r_{sampling}$ **do**
    $p_{sample} \leftarrow (x, y, z)$
    $d_{center} \leftarrow \|p_{sample} - f_i.position\|_2$
    if $d_{center} > 0.7\alpha$ **then**
      continue
    end if
    $v_{idx} \leftarrow$ GetVoxelIndex(p_{sample})
    $key \leftarrow$ Hash(v_{idx})
    if $key \in voxelLookup$ **then**
      $N_{total}++$
      if GetVoxelStatus(v_{idx}) = UNOBSERVED **then**
        $N_{unknown}++$
      end if
    end if
  end for
  $\rho \leftarrow N_{unknown} / \max(N_{total}, 1)$
  $f_i.gain \leftarrow k_g \cdot \rho \cdot \exp(-\lambda \cdot \Delta z) \cdot \gamma_{energy}$
  $\vec{d} \leftarrow (f_i.position - p_{robot}) / \|f_i.position - p_{robot}\|_2$
  $hasLOS \leftarrow$ CheckLineOfSight(p_{robot}, $\vec{d}$, $f_i.index$)
  if not $hasLOS$ **then**
    $f_i.information\_gain \leftarrow f_i.information\_gain \times 0.5$
  end if
end for

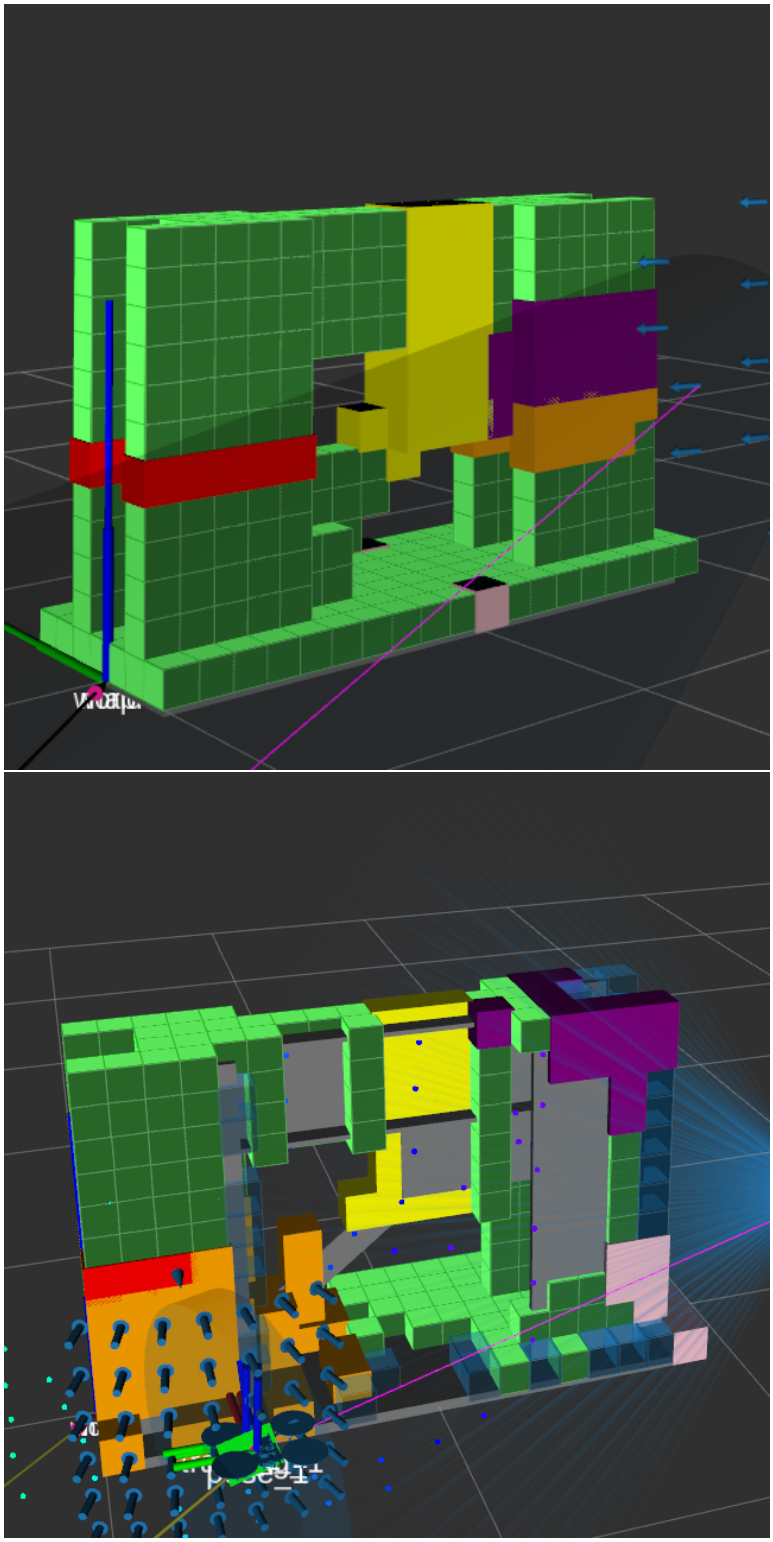

Figure 5.2: Frontier visualization of two view taken at different times during exploration showing the top five frontier cluster ranked by information gain

5.5 Comparison

In this subsection, we will analyze the main differences between our implementation and the state-of-the-art frontier-based approach described in [2]. The comparison will highlight the architectural modifications and their impact on exploration efficiency for asset inspection scenarios.

Regarding the frontier detection strategy, the two implementations present fundamental conceptual differences that significantly impact exploration behavior. The original approach employs a classical frontier detection method that identifies boundaries between free and unknown space in completely unknown environments. This strategy works well for general exploration but requires extensive computation to process large frontier sets and may generate redundant exploration targets. In contrast, our implementation leverages a-priori knowledge of asset geometry by treating each voxel in the reconstructed model as an individual frontier candidate. This paradigm shift enables more targeted exploration since the system focuses exclusively on relevant geometric features rather than arbitrary space boundaries. Additionally, our approach incorporates face status initialization that considers neighboring voxel relationships, ensuring more accurate frontier validation. From a computational perspective, our technique implements several optimizations absent in the original work. The spatial grid acceleration system reduces neighbor search complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, while adaptive sampling concentrates resources on promising frontier regions rather than uniformly sampling the entire frontier space. The following table compares the performance metrics of both techniques:

Metric	**Original Approach**	**Our Approach**
Frontiers processed per iteration	200–500	50–150
Clustering computation time (s)	1.5–3.0	0.3–0.8
Memory usage efficiency (%)	60–75	80–95
Frontier validation accuracy (%)	70–85	85–95

Table 5.1: Frontier-Based Performance Comparison Metrics across mockup asset coverage using single robot

The memory usage efficiency is the ratio of the memory use for frontier data and the total allocated memory for the entire program. The metric Frontier validation accuracy is defined as the percentage of detected frontiers that presents a information gain higher than a fixed threshold of 0.4 . It evaluates how the algorithm identifies frontiers that contribute meaningful information gain when visited. Regarding information gain computation, the original version uses a simplified approach based primarily on the ratio of unknown voxels within a cubic region around each frontier, as opposed to our approach where... While effective for general exploration, this method

does not account for energy consumption patterns specific to UAV operations. Our implementation extends the gain formula with energy-aware factors including vertical movement penalties $e^{-\beta\Delta z}$ and visibility analysis through line-of-sight verification. The energy factor addresses the critical aspect of UAV power consumption, where vertical movements require significantly more energy than horizontal displacement. The visibility factor ensures that selected frontiers maintain clear observation paths, reducing the likelihood of mission failures due to obstructed viewpoints. Another significant difference lies in the filtering and validation architecture. The original approach applies basic obstruction checking and distance-based filtering with limited memory management capabilities. Performance degrades with duration of exploration as the frontier sets grow without pruning mechanisms. Our collector strategy implements a three-stage filtering system with caching mechanisms that maintain exploration quality while managing computational resources. The following table illustrates the architectural differences:

Aspect	**Original Approach**	**Our Approach**
Information gain factors	1	3
Filtering stages	2	3
Energy consideration	Not implemented	Exponential decay

Table 5.2: Frontier-Based Architectural Comparison

The integration within our task management framework represents another fundamental distinction. Rather than operating as an independent exploration module, our frontier-based technique generates tasks compatible with multi-robot coordination protocols. This integration enables seamless comparison with NBV and standard approaches under identical operational conditions while supporting distributed exploration scenarios. However, these specializations introduce certain trade-offs. The original approach maintains generality across diverse exploration scenarios, while our implementation is optimized specifically for asset inspection tasks where geometric priors are available. The increased computational complexity of multi-factor gain evaluation may impact real-time performance on resource-constrained platforms, though spatial optimizations and caching mechanisms largely compensate for this overhead. These improvements collectively transform the frontier-based approach from a general-purpose exploration tool into a specialized, energy-efficient system for known asset inspection, making it particularly suitable for infrastructure monitoring and industrial inspection applications where geometric models are available and energy consumption optimization is critical.

Chapter 6 Experimental Setup

The experimental setup for the simulation of all techniques exposed is based on the use of Webots as simulation environment and Ros Noetic as middleware framework for inter-component communication. The computer used for the simulation presents an Intel Core i7-7700 processor running at 3.6GHz with 8 cores, 64-bit architecture, and 16GB of RAM. For the real experemts we use the Starling platform produced by ModalAI Inc., as shown in Fig. 6.1 a). It spans 190 diagonally and weighs slightly more than 300 . The MAV runs the PX4 autopilot and is interfaced over ROS using MAVROS. State estimation is performed by monocular VIO, using a 45 downward looking global shutter wide-angle camera, running Qualcomm's proprietary VIO solution mvVISLAM. The MAV is equipped with two forward-facing sensors: a 4K camera with an overall FOV of 100, and a ToF camera with horizontal and vertical Field of View (FoV) of, respectively, 85.1 and 106.5, able to measure distances in the range $[0.5, 4]$. In order to localize the start location of the robots in the arena 10.1 we use a Motion Capture System (MCS) with millimeter accuracy from Motion Analysis Inc. The robots use their onboard VIO to compute their motion from their start location while the MCS allows to compensate for drift in odometry throughout the flight. For evaluating the performance of our method with respect to the other two techniques, we chose several assets that differ in shape complexity and dimension.

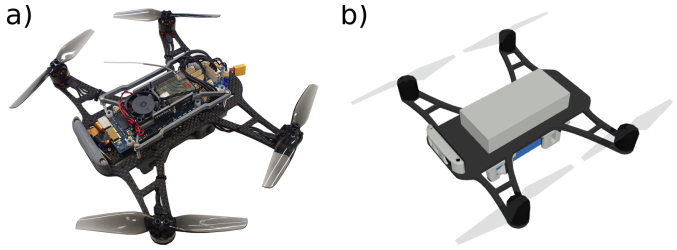

Figure 6.1: **a)** Real and **b)** modeled Starling MAV by ModalAI Inc.

For the virtual simulation we have used the following assets:

- **Structure composed by two columns and one diagonal**
- **Ferrari 296 GTB**
- **Tank car**
- **Armadillo**
- **Eiffel Tower**

. In the Appendix, a dedicated section contains all the meshes used (Section 10.3). The experiment are realized with different numbers of robots,ranging from a single one to a maximum of 5 robots simultaneously. To analyze the different techniques, we decided to use several parameters based on distance traveled, time elapsed during the exploration, and energy consumed by MAVs.

The Coverage Efficiency parameter measures the effectiveness of the exploration by the ratio of voxels explored to the starting number of voxels in the PH-tree data structure.

Coverage speed is expressed in cm^3/s and indicates how fast the system is able to obtain information about the environment. It is defined as follows:

$$CS = \frac{V_{explored}}{t_{total}} = \frac{voxel_{explored} \cdot l^3}{t_{total}}$$

where $V_{explored}$ is the volume explored, defined as the products of voxels explored times the voxel's dimension l, and t_{total} the total time spent for the exploration.

Energy Efficiency is particularly important for real drone applications due to battery constraints that can affect the overall performance. It is defined as the ratio between voxels explored and the total amount of energy consumption.

Regarding the energy consumption calculation, we have used the parameters obtained by Zhou [9]. The implemented energy model considers two main components:

- **Aerodynamic drag energy**: $E_{drag} = \tau \cdot D * \vec{v} \cdot \Delta\vec{s}$
- **Gravitational potential energy**: $E_{pot} = m \cdot g \cdot \Delta z$

where τ is the drag coefficient, D is the diagonal aerodynamic drag matrix, m is the drone mass and g the gravitational acceleration. These parameters work for both single and multi-robot cases and enable us to compare the three techniques.

Chapter 7 Results

7.1 Simulation results

Starting from single MAV exploration, we used the data obtained during the execution of each technique and realized a graph that illustrates the decline in remaining voxels over time. Figure 7.3 shows the exploration behavior of a single drone using our PH-Tree-based technique during mochup coverage. The orange line represent a mathematical interpolation of the data point, providing a smoothed trend line.

As can be seen from Table 7.1, the parameters obtained from the Ferrari's coverage demonstrate the efficiency of our technique with respect to the other two. Our technique presents the highest values in coverage speed and energy efficiency, demonstrating that our solution satisfies the actual challenges of MAVs that cannot guarantee a huge battery. The box plot diagrams shown in Figure 7.4 and 7.5 emphasize the better performance of our technique since it presents the lowest median values in terms of distance traveled and energy consumed by MAVs. The tight dimensions of the box guarantee the possibility to use our approach in real experiments and to have high enough confidence in the results. The statistical analysis reveals several advantages regarding using our implementation:

- The narrow box plots for our technique demonstrate highly consistent result across different experimental conditions
- The small standard deviation demonstrates that our technique has stable performance independently of asset complexity
- The tight statistical distributions provide confidence that performance benefits will scale consistently to larger or more complex exploration tasks
- The absence of outliers reduces the risk of unexpected poor performance

Parameter	**Our Technique**	**NBV**	**Frontier-based**
Coverage efficiency	0.99	0.98	0.99
Coverage speed (cm^3/s)	3862	2024	3007
Energy efficiency (J^{-1})	168.29	110.28	101.16
Time (s)	190	210	175
distance (m)	59.3	63.5	57.65
Energy (J)	4.7	7.1	7.8

Table 7.1: Parameters for Ferrari exploration by single MAV

These results also show that the use of PH-TREE as a data structure to obtain areas to explore guarantees faster coverage of the assets and less waste of time in points already explored. The NBV has the poorest parameters, and this situation is due to its probabilistic approach.

Sometimes the system gets stuck in an area characterized by partially visible voxels, and the probabilistic sampling doesn't find a valid viewpoint as fast as the other two techniques. In the case of simple objects, the results obtained by the use of the Frontier-based technique are more or less comparable to the ones obtained by our system.

This situation is due to the small number of voxels to explore. In Histogram 7.1, the comparison between the three techniques with respect to the distance traveled and the energy consumption is shown. From that graph, it is easy to understand that the NBV suffers in the exploration of known assets more than the frontier-based technique.

In the case of big objects, the system requires more time and more iterations to elaborate the frontiers and to cluster them. As can be seen in Tables 7.2 and 7.3, the results obtained by the NBV and frontier-based techniques are lower than the previous case.

Our approach presents the best values for all the parameters, and the difference between the same parameter of the other techniques is significantly larger.

The use of multiple MAVs emphasizes the improvements of our method with respect to the other two, especially in energy-related parameters. Our method, in the Eiffel coverage with 3 or 5 MAVs, presents energy consumption less than one-third of both the other approaches and with distance traveled less than half. The frontier-based technique is characterized by the highest worsening of performance in the case of big and complex assets. The increase in time needed by the computation and a not totally efficient frontier choice makes the system not suited for inspection of environments such as houses or towers. By conducting different tests, we have obtained Figure 7.2 that shows the mean value along time and the standard deviation. Thanks to this figure, we can see how the system works in the same way during different iterations and presents the lowest standard deviation at every timestamp.

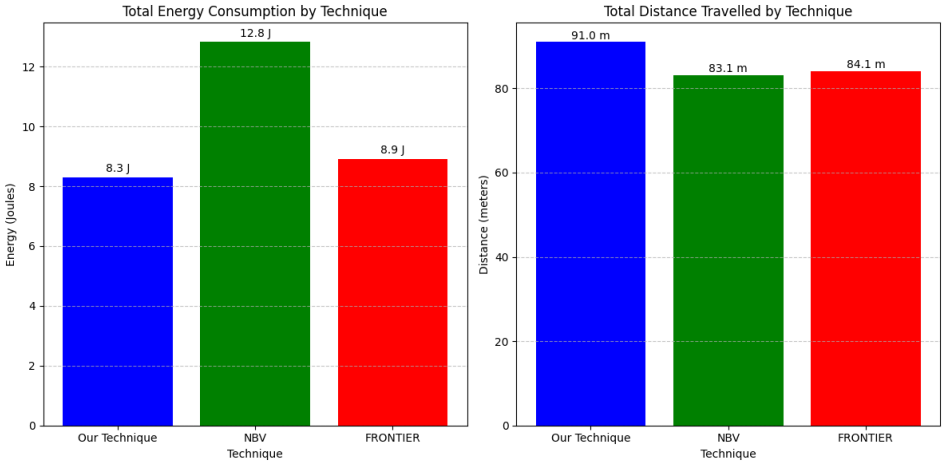

Figure 7.1: Histogram of distance traveled and energy consumption during exploration of Ferrari asset using 3 robots

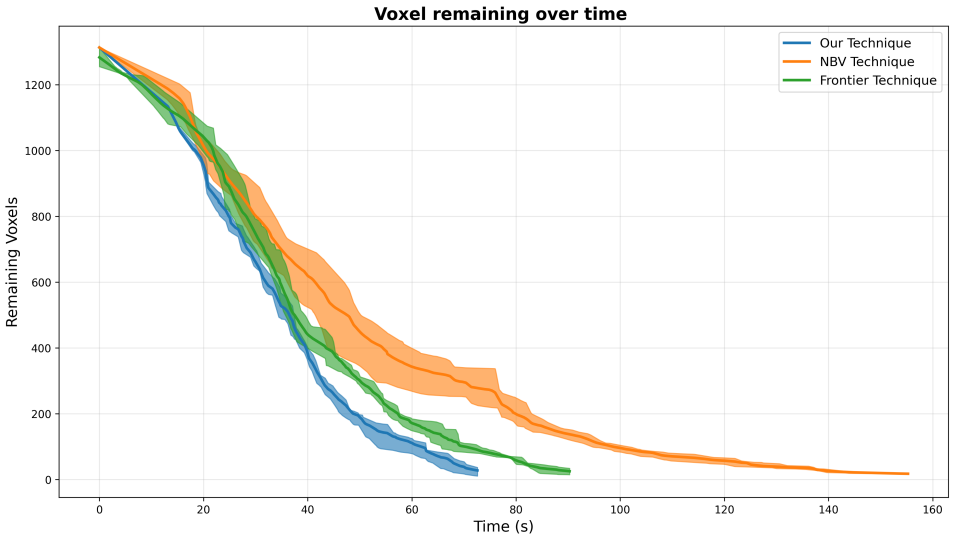

Figure 7.2: Exploration comparison of the three techniques using 3 robots for Armadillo coverage

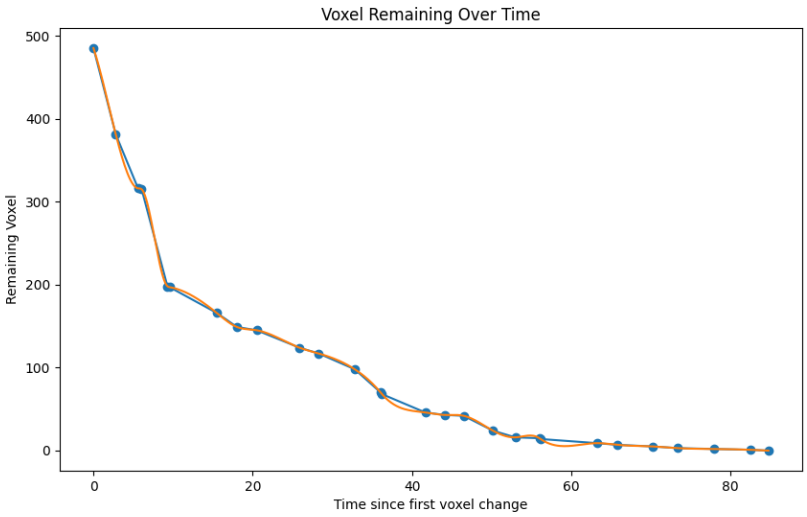

Figure 7.3: Exploration behavior of single drone during mockup's coverage

Parameter	**Our Technique**	**NBV**	**Frontier-based**
Coverage efficiency	0.99	0.98	0.99
Coverage speed (cm^3/s)	25235	8456	13194
Energy efficiency (J^{-1})	147.16	33.08	56.88
Time (s)	257	755	427
distance (m)	266.53	848.84	411.57
Energy (J)	41	193	114

Table 7.2: Parameters for exploration of Eiffel Tower by 3 MAVs

The Frontier-based technique presents a higher standard deviation, especially during the first 12 tasks, and this is due to the necessity of higher computational time regarding frontier choice. The NBV presents the highest standard deviation in the middle of exploration, and this affects the reliability of this technique in the presence of assets with complex shapes.

Parameter	**Our Technique**	**NBV**	**Frontier-based**
Coverage efficiency	0.99	0.98	0.99
Coverage speed (cm^3/s)	34112	13311	17741
Energy efficiency (J^{-1})	102.11	32.74	52.03
Time (s)	189	479	355
distance (m)	296.89	856.94	511.26
Energy (J)	63	195	124

Table 7.3: Parameters for exploration of Eiffel Tower by 5 MAVs

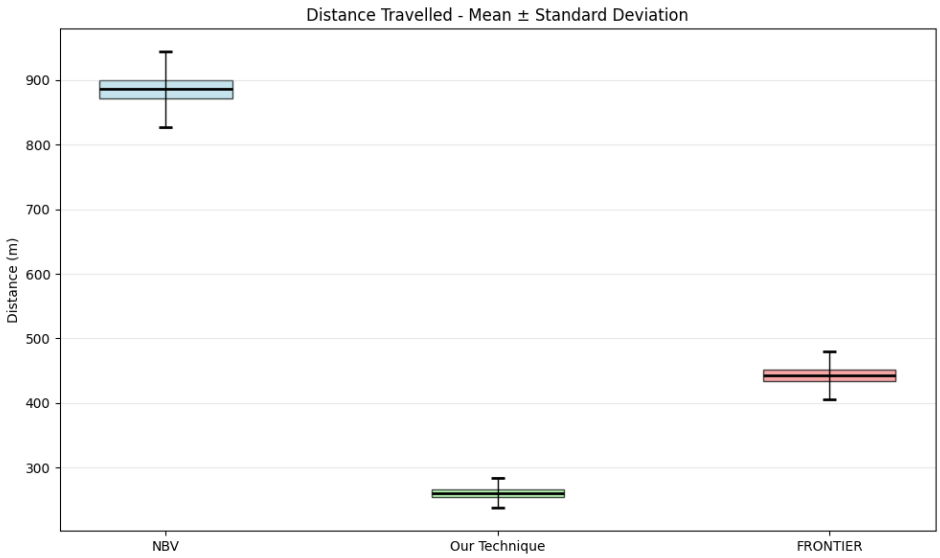

Figure 7.4: Box plot of distance traveled by 3 robots during Eiffel's coverage

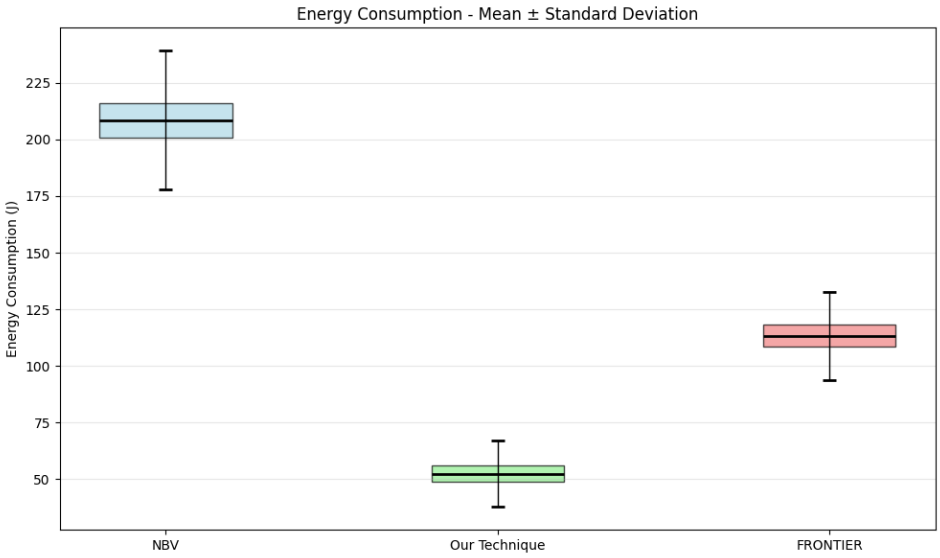

Figure 7.5: Box plot of energy consumption by 3 robots during Eiffel's coverage

Chapter 8 Unknown exploration technique

Techniques for coverage of known assets guarantee good performance for the exploration of a priori-known assets by using energy-efficient path planning and real-time task assignment, but are limited since they cannot adapt when environmental modifications occur or when prior geometric knowledge is unavailable. For this reason, increasing research efforts focus on finding innovative solutions for the exploration of environments about which we have no prior knowledge. However, unknown environment exploration presents unique challenges that require different approaches for frontier detection, map construction, and navigation in uncertain conditions. In this chapter, we will present our work regarding a novel exploration technique for unknown environments. Starting from the state-of-the-art techniques, we have designed a new framework capable of exploring a space and obtaining a detailed map simultaneously.

With respect to navigation tasks, in the case of exploration of unknown environments, the system requires that the robot makes decisions based on incomplete information.

Our method aims to find a reliable solution to this challenge through the use of three innovations: an asynchronous pipeline architecture, intelligent frontier selection, and a path planner based on TSDF-informed sampling.

Our implementation is composed of two primary components that work together:

- **VoxMap system** (provided and adapted) that processes sensor data obtained from the ToF sensor and updates a 3D representation of the environment. With a probabilistic voxel-based approach, it enables the system to define occupied space and frontiers.
- **Robot Exploration system** that uses an asynchronous pipeline to manage several processes such as receiving data from VoxMap, choosing the best frontier, and planning a trajectory to reach it.

The integration between these components enables continuous exploration where mapping and navigation decisions are made concurrently, max-

imizing exploration efficiency while ensuring safe operation in unknown environments.

In the following subsections, each important part of our framework will be discussed in detail, focusing on the main innovations.

8.1 SOTA techniques

In literature, there are different techniques used for exploration that belong to the class of frontier-based approaches. Frontiers, defined as the areas that define a boundary between the known and unknown space, are used as elements for defining the next viewpoint during the exploration, guaranteeing good performance in terms of time spent during the exploration and distance traveled. In our project, we decided to use the concept exposed in the work of Zhou et al. [8] regarding the use of a sophisticated frontier manager. While their frontier manager is used for trajectory optimization and planning efficiency, our version focuses on frontier selection scalability, providing 10-100x performance improvement over individual frontier evaluation in high-density scenario. In the work of Oleynikova et al. [10] explains how to use the TSDF-data obtained from the sensor for mapping and planning in order to improve the performance of path finding and achieve safer trajectories. Since our work uses a TSDF map of the environment, we chose to incorporate these values into the RRT* algorithm. While Oleynikova focuses on direct TSDF-to-ESDF conversion for collision checking in MAV navigation, we introduce a novel TSDF-informed sampling strategy specifically designed for exploration scenarios.

8.2 Asynchronous pipeline

The core architecture of our exploration framework is composed of an asynchronous pipeline that coordinates the different subsystems (mapping, planning and control) to achieve safe and efficient exploration of unknown environments.

This solution has been chosen to guarantee optimal performance in real-time exploration scenarios.

Each phase of the exploration is separated by the pipeline into three main threads that operate in parallel:

- **Mapping thread** that manages sensor data acquisition and processing
- **Planning thread** that executes path planning through selected frontiers
- **Control thread** that manages trajectory execution and robot control

The mapping thread processes data acquired by the VoxMap and updates the map used by the path planner. It also stores all the frontiers using the same approach of [8] and saves them into a thread-safe data structure that can be accessed by other pipeline components without blocking the mapping process.

The planning thread manages the frontier choice and path generation. When valid mapping data is available and the system is in searching state, the thread executes planning operations and generates trajectories using the path planner integrated with TSDF-informed sampling.

The control thread manages trajectory execution and uses the updated map information for collision avoidance. The thread continuously monitors the robot's state and provides feedback about execution progress. A collision is defined as situation where the distance between the robot and the occupied voxels is less than the robot's safety footprint (robot radius + safety margin of 0.2 meter). If the sensor detects a distance inferior to the safety footprint, the control thread blocks the entire trajectory execution and asks for a new path or a new target from the planning thread.

To ensure robust communication between threads, we decided to use thread-safe message queues. In this way, the data are processed separately and do not affect the overall performance of the exploration framework.

The pipeline integration includes a state machine that coordinates exploration state transitions. The system manages transitions between searching for new frontiers, moving toward targets, and handling mission completion. Data consistency is maintained through timestamp-based validation and version-based synchronization mechanisms that ensure logical consistency between mapping, planning, and control data.

As a last aspect covered by this implementation, it uses a graceful shutdown procedure that ensures orderly termination of all threads with appropriate timeouts and cleanup operations.

8.3 VoxMap

The main component used in this framework as map builder during the exploration is the **VoxMap**. It was not implemented here and was provided directly by the project supervisor. Thanks to a probabilistic representation of the three-dimensional space through voxels, it enables to keep information regarding the occupied space and to identify dynamically unexplored areas. This particular map is composed of several components that are:

- **Probability voxel grid** that contains information about the occupied space
- **TSDF voxel grid** that, based on the Truncated Signed Distance Field, enables the system to have a geometric representation of surfaces
- **Frontier manager** that controls and updates the frontiers based on the new values of the probability voxel grid

As discussed in the chapter about our technique, each voxel is defined by a status and its faces status. In addition, each voxel stores a flag that enables the system to consider it as a frontier or not and a log-odds value about the occupation status.

When the VoxMap obtains data from the ToF sensor attached to the robot, it manages all new information acquired in two steps. First, by using a callback function that saves all the data obtained, and second, an update function that handles the integration into the occupancy map as shown in Algorithm 19.

Every time the map is updated, the system updates the log-odds of voxels that are affected by the changes in the environment. Using ray-tracing from the sensor position, they set the value of voxel's log-odds seen by the depth camera by adding a particular value based on the ray status.

If the log-odds value is higher than a threshold of 0.5, the voxel will be considered as occupied. These values are:

- $P_{occ} = 0.88$ if a ray hits a voxel surface
- $P_{free} = 0.45$ if the space is free along the ray's path

Another step computed during the map's update is the identification of frontiers. A frontier is defined as a non-occupied voxel near an occupied one that represents the transition to unexplored space. The system identifies them by checking the faces status of each voxel.

The frontier identification starts by considering the newly inserted occupied voxels. It finds the potential frontier location by examining all six adjacent faces. If that face is UNSEEN, it represents a potential frontier location. As a last step, it chooses the best position by considering the position that best aligns with the sensor's observation pose.

In this way, the obtained frontiers provide good target areas for the exploration of unknown areas with higher compatibility with the previous robot movements.

As a final step before sending the frontiers to the robot, the system uses a filter that considers frontiers with a straightforward line of sight.

Algorithm 6 VoxMap Update Function

```
1: success ← false
2: if queue is empty then                                  ▷ Preliminary control
3:     return success
4: end if
5: while queue is not empty do                             ▷ Merging step
6:     measurement ← queue(0)
7:     pose, grid ← GetDatafromMeasurement(measurement)
8:     InsertscanIntoOccupancyGrid(pose, gird)
9:     PopMeasurement(queue)
10:    success ← true
11: end while
12: if success then                                        ▷ Frontiers update
13:    frontiers_LOS ← UpdateLosCache                      ▷ Update Los frontiers
14:    frontiers ← FilterFrontiersByLineOfSight(pose)
15:    frontiers_updated ← true
16:    last_frontiers_update ← CurrentTime()
17:    PublishSynchronizedFrontiers()
18: end if
19: return success
```

All frontiers that satisfy the conditions imposed by the filter are saved into a vector in order to have a cache of all possible frontiers available.

During the exploration, the system will be able to move also to areas not visible from the actual position. The cache is updated at each iteration by a double check: the first is a verification that saved frontiers are still frontiers, and the second one checks the TSDF value of each frontier and if it is higher than a threshold, it will be kept in the cache memory.

By the use of a publisher, the robot receives filtered frontiers and uses them as the basis for target choice.

In Figure 8.1 is possible to see the defferences between voxels,frontiers and filtered frontiers. The gray voxel are occupied voxel, the violet voxels are frontiers defined by the voxmap that don't satisfied the filter condition and the red voxels are filter frontiers that are updated each time new measurement is available.

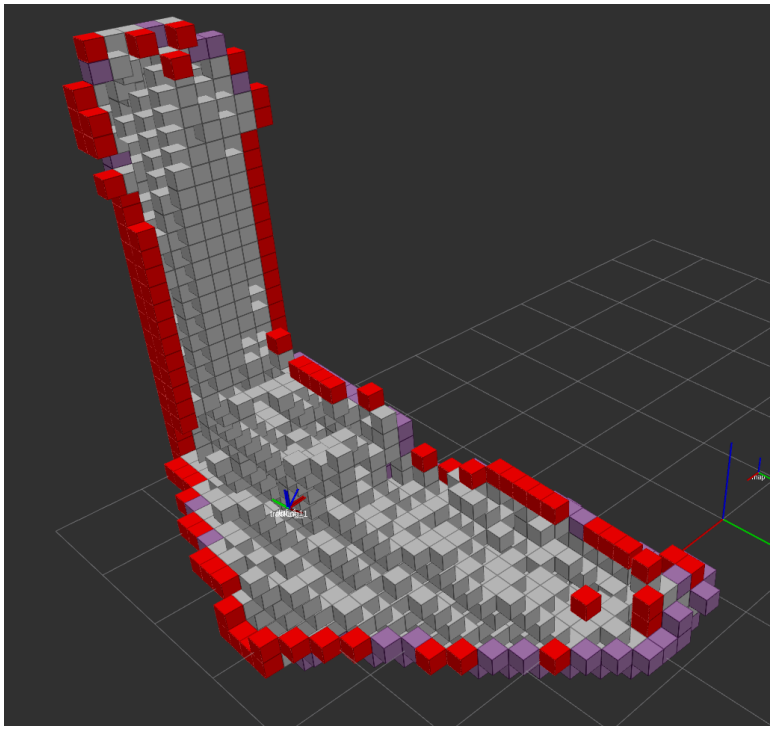

Figure 8.1: Voxmap's map representation

8.4 Best Frontier selection

The choice of the best frontier is crucial since it affects the efficiency and completeness of the exploration process. When the system obtains the frontiers from the Voxmap, it computes a comprehensive utility score $U(f_i)$ that combines several factors: distance factor and information gain potential.

The **Distance factor** considers the travel cost to reach the frontiers, preferring closer target that guarantee energy efficiency and reduced execution time. It is computed in the following way:

$$D(f_i) = e^{-d_{xy} \cdot w_{xy}}$$

where d_{xy} is the 2D Euclidean distance between frontier centroid and robot position, and $w_{xy} = 2.0$ is the distance weight.

The **Information gain** $G(f_i)$ evaluates the amount of new information about the environment that would be discovered by visiting that frontier. It is defined as the ratio of unexplored voxels inside a cube centered at the frontier center with an edge equal to the sensor range.

The Size factor $S(f_i)$ considers the numbers of frontiers within a distance of 0.8 meters from the frontier center. in this way, the system will

chose frontiers in a area with an high frontier density, which typically correspond to complex geometric structures or transition zones between different environmental regions

$$S(f_i) = log(1 + |F_i|)$$

where F_i is the number of frontiers with the region. The final selection combines these factors through a weighted utility function:

$$U(f_i) = w_d \cdot D(f_i) + w_g \cdot G(f_i) + w_a \cdot S(f_i)$$

where w_d, w_g, w_a are weighting factors.

The frontier with the highest utility score is selected as next target and the system moves to find a pose able to see the selected frontier.

8.5 Viewpoint selection

After frontier selection, the robot samples multiple viewpoints within a spherical region centered on the selected frontier. The sampling process generates candidate positions within a variable radius ranging from 0.5 to 2 meters around the frontier center.

The system uses the Voxmap to obtain information about the sampled viewpoint and to apply different validation checks:

- **Collision-Free Verification** that checks if the sampled position is inside or too-close to an obstacle. The system uses the safety radius to guarantee a sufficient margin with respect to occupied voxels.
- **Line-of-sight Validation** that verifies if there is at least one line-of-sight between the viewpoint and the target frontier using ray-tracing technique.

The remaining viewpoints are evaluated by using a scoring mechanism based on different criteria that take into account position and orientation.

8.5.1 Viewpoint Score

The viewpoint score is defined by the following formula:

$$S_{total}(v) = w_d \cdot S_{distance}(v) + w_h \cdot S_{height}(v) + w_o \cdot S_{orientation}(v)$$

where w_d, w_h, w_o, w_f are weighting factor, $S_{distance}$ is a distance score, s_{height} is a height preference score and $S_{orientation}$ is a orientation alignment score. The distance score considers the Euclidean distance between the viewpoint

and the robot position:

$$S_{distance}(v) = e^{-||v-p_{robot}||\cdot w_{distance}}$$

where v is the viewpoint, p_{robot} is the robot position and $w_{distance}$ is a distance weighting factor.

The Height Preference score favors viewpoint that maintains the same altitude of the current robot position in order to minimize the energy consumption:

$$S_{height}(v) = e^{-|z_v-z_{robot}|\cdot w_{height}}$$

where z_v and z_{robot} are respectively the heights of the viewpoint and robot, and w_{height} is a factor that controls the height preference strength.

The Orientation Alignment score considers if the viewpoint heading aligns with the robot's current orientation in order to minimize rotational movement:

$$S_{orientation}(v) = e^{-|\psi_{required}-\psi_{current}|\cdot w_{orientation}}$$

where $\psi_{required}$ is the heading needed to face the frontier from the viewpoint and $\psi_{current}$ is the robot's current heading.

In Figure 8.2, it is possible to see the viewpoints sampled around the selected best frontier. The blue cube represents the best frontier and the blue viewpoint represents the selected one as the next target. There are other viewpoints of different colors that summarize information about their feasibility.

The green viewpoints are sampled points that have passed the two checks described before, the orange ones are the sampled points that have passed the validation checks but have a score lower than an imposed threshold of 0.5, and the red ones are the sampled points that do not respect the validity conditions of the sample check.

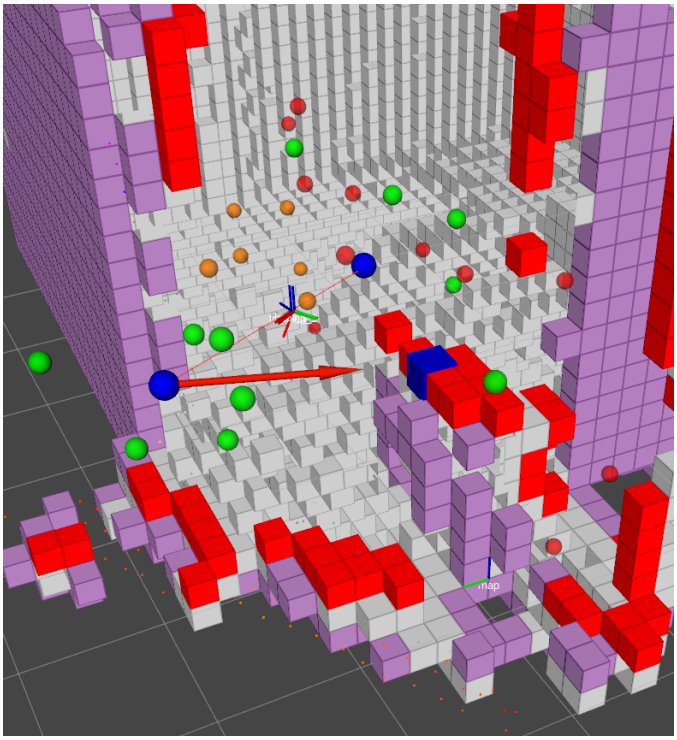

Figure 8.2: Viewpoint sampling and next target selection during exploration of unknown asset

The candidate viewpoint with the highest total score is selected as the immediate navigation target, while the original frontier remains the ultimate exploration objective.

This two-stage approach ensures that the robot can safely and efficiently position itself to gather maximum information about the unknown environment while maintaining robust navigation capabilities.

8.6 Path Planner

When the target position is chosen, the path is found by a novel path planner that takes inspiration from our implementation. Starting from an RRT* algorithm, we add new optimizations to be integrated in the unknown coverage exploration. The planner acts as a bridge between the VoxMap and the robot, ensuring safe navigation towards unexplored areas.

Since the system works in dynamic environments characterized by elevated frontier changes and incomplete environmental knowledge, we need a system able to generate trajectories efficiently and to compute real-time collision detection in the evolving map.

One of the changes applied in the RRT* is the TSDF-informed sampling.

We moved from uniform random sampling to intelligent sample generation that is more likely to yield valuable exploration paths.

The reason behind this choice is due to the fact that TSDF gives us information that enhances path finding since it enables the system to find corridors in regions far from obstacles.

If we consider an environment composed of complex structures, the TSDF map can guide the system to find efficient exploration where free space areas can offer safer paths and areas near surfaces can provide navigation corridors reliable for the robot's constraints.

The TSDF-informed sampling employs a probabilistic approach where the sampling probability at any point is influenced by the local TSDF values:

$$P(x) = P_{uniform}(x) \cdot W_{TSDF}(x)$$

where:

- $P_{uniform}(x)$ is the uniform sampling probability
- $W_{TSDF}(X)$ is the TSDF-based weight funtion define as:

  $$W_{TSDF}(x) = \begin{cases} 0 & \text{if } d_{TSDF}(x) \leq \tau_s \\ \min\left(1 + \alpha \cdot (d_{TSDF}(x) - \tau_s), W_{max}\right) & \text{if } d_{TSDF}(x) > \tau_s \end{cases}$$

  where $d_{TSDF}(x)$ is the signed distance field value at position x,$\tau_s = 0.5$ is the safety threshold distance, α is a scaling factor that controls the weight increase rate $W_{max} = 3.0$ is the maximum weight multiplier

In this way, the system samples points that are near surfaces or in open areas using a higher weight and avoiding obstacles using zero weight.

The other update regards the collision detection that uses a continuously updated internal representation of the environment, ensuring that planning reflects the most current understanding of the environment.

Traditional techniques of exploration suffer when they deal with large and complex environments to check collision-free movement during navigation.

In our case, we are using a **Spatial Hash Manager** that addresses this limitation and significantly reduces the computational complexity of collision queries.

The spatial hashing approach divides the 3D environment into uniform buckets containing a precise number of voxels within a specific spatial region. Each bucket contains the timestamp for cache validation and all voxel indices within the bucket's spatial region. In this way, collision detection moves from a linear search through all n voxels O(n) to a localized search within relevant k buckets O(k) with k much less than n.

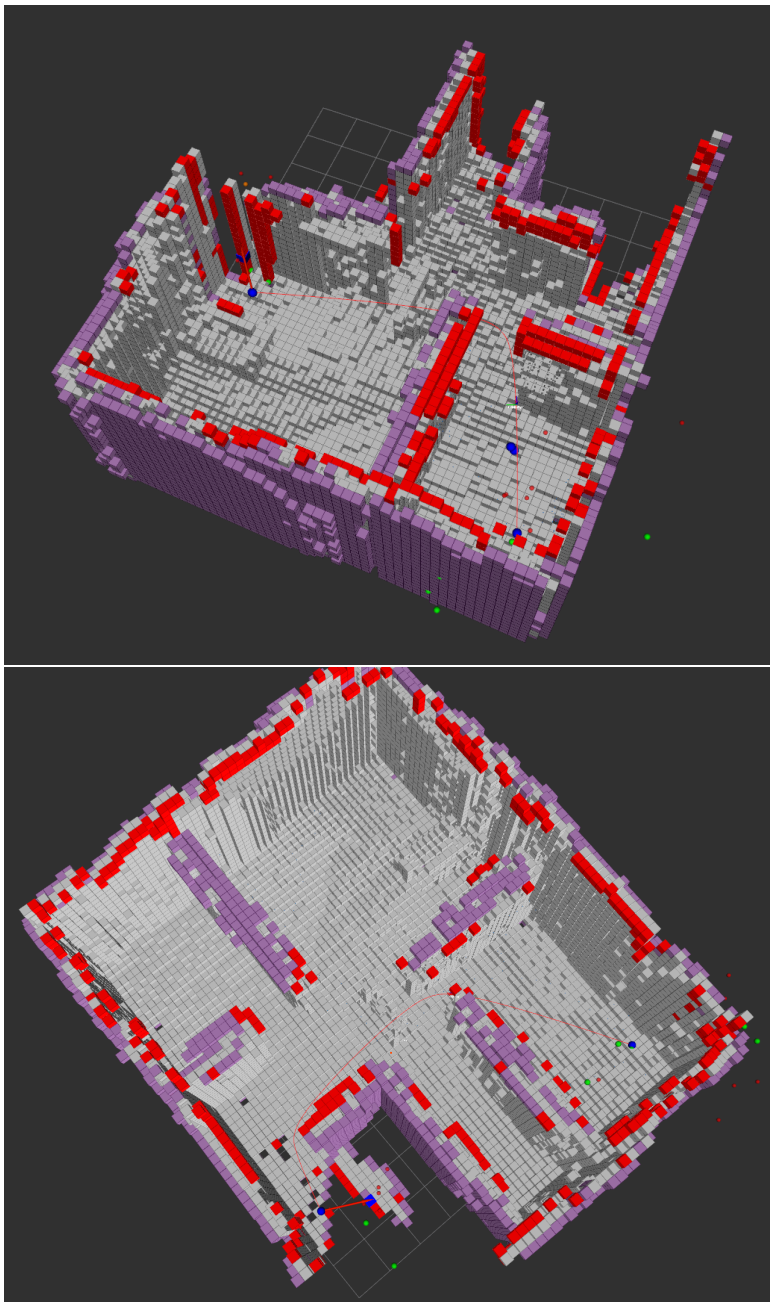

Figure 8.3: Trajectory generation into a complex environment

The other advantage of this implementation is the efficient memory management that guarantees a reduction of allocation overhead during frequent updates.

As shown in Figure 8.3, we can see the system's ability to generate a feasible trajectory inside the explored environment.

In order to integrate the path planner with the VoxMap, we adopt a multi-threaded environment that ensures data sharing while maintaining thread safety.

The system architecture separates data acquisition, processing, and planning into distinct threads that communicate through carefully designed interfaces.

Since the planner needs the current environmental information without blocking the mapping process or introducing race conditions, the system presents a double-buffering approach where the VoxMap updates a shared data structure and the planner maintains a local copy for planning operations.

Another interface designed to avoid circular dependencies between planner and VoxMap is a provider-based interface for TSDF data access that enables the path planner to access TSDF information without direct coupling to the VoxMap implementation.

This multi-threaded architecture ensures that the path planner can operate efficiently with real-time data updates from the VoxMap system while maintaining thread safety and data consistency throughout the exploration process.

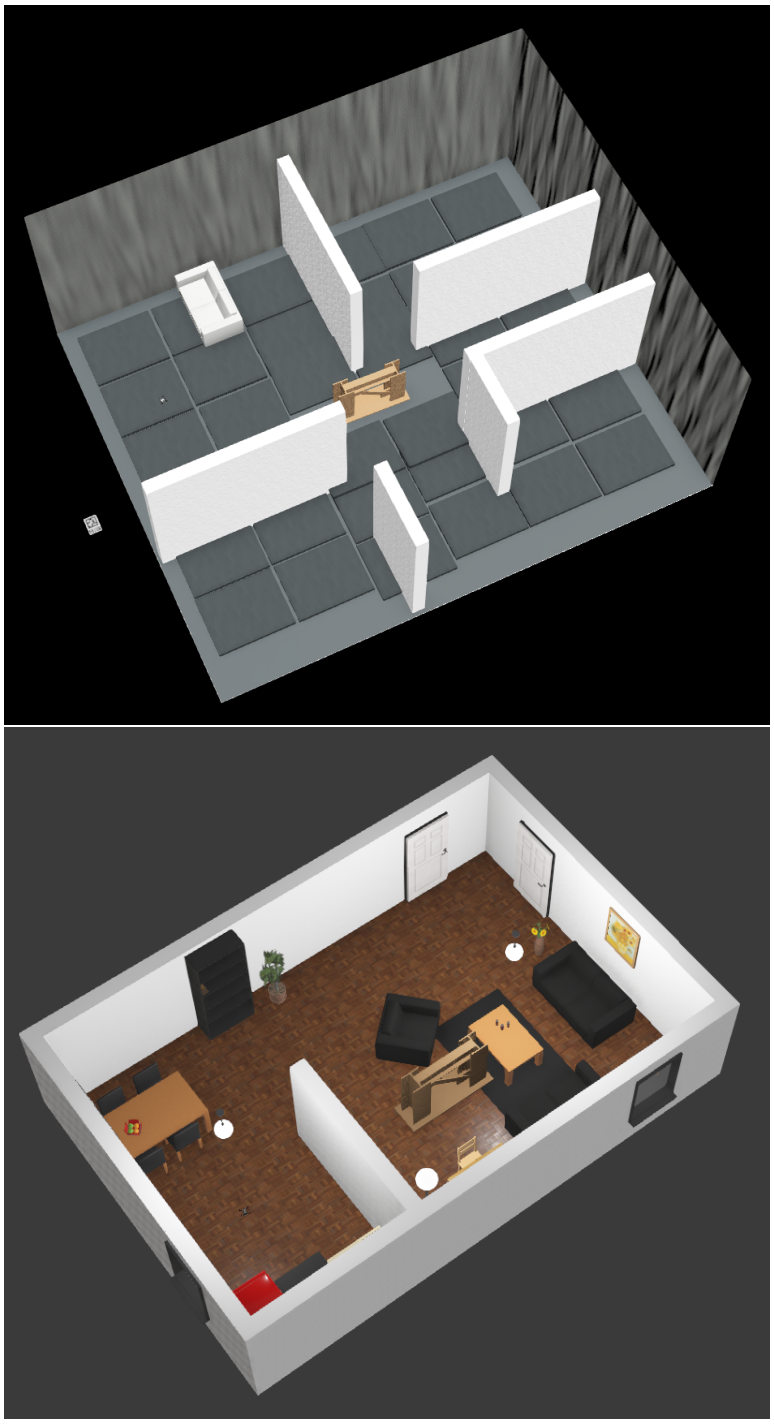

Figure 8.4: Webots words 1 and 2 used in the simulation

8.7 Simulation results

To validate the functionality implemented, we have designed Webots worlds as shown in Figure 8.4 that present narrow corridors and a complex shape.

The simulation demonstrates the effectiveness of our framework across both simulation environments.

The first one, characterized by complex geometry and narrow passages, has been completely covered in 350s, with each iteration requiring 0.5-0.8s, making this approach feasible for real-world testing.

Metric	**Environment 1**	**Environment 2**
Exploration Time (s)	350	270
Total Trajectory Distance (m)	80	65
Energy Consumption (J)	24	20
Energy Efficiency (J/m)	0.30	0.31

Table 8.1: Performance comparison between the two test environments

The energy consumption guarantees the use of MAVs since they are characterized by low energy storage, and the implementation of a safety return-to-base mechanism enables the system to operate optimally in case of unexpected battery discharge.

As shown in Figure 8.5, the last updated map in the simulation appears equal to the virtual representation and presents several details that are important for an efficient and safe movement of the robot.

In the second environment, characterized by a simpler layout, the robot explores it in 270s and with a iteration's time of 0.3-0.5s. These values are lower tha those of the previous environment due to the simpler complexity and corridors wide enough that not need a sophisticated computational checks.

The performance metrics highlight the adaptability of the system to different environments and the improvement in complex case as narrow corridors.

The use of asynchronous pipeline architecture and the intelligent frontier selection guarantees an high computation efficiency while maintaining a good energy efficiency and exploration completeness.

Table 8.1 summarizes the performance metrics obtained from both test environments.

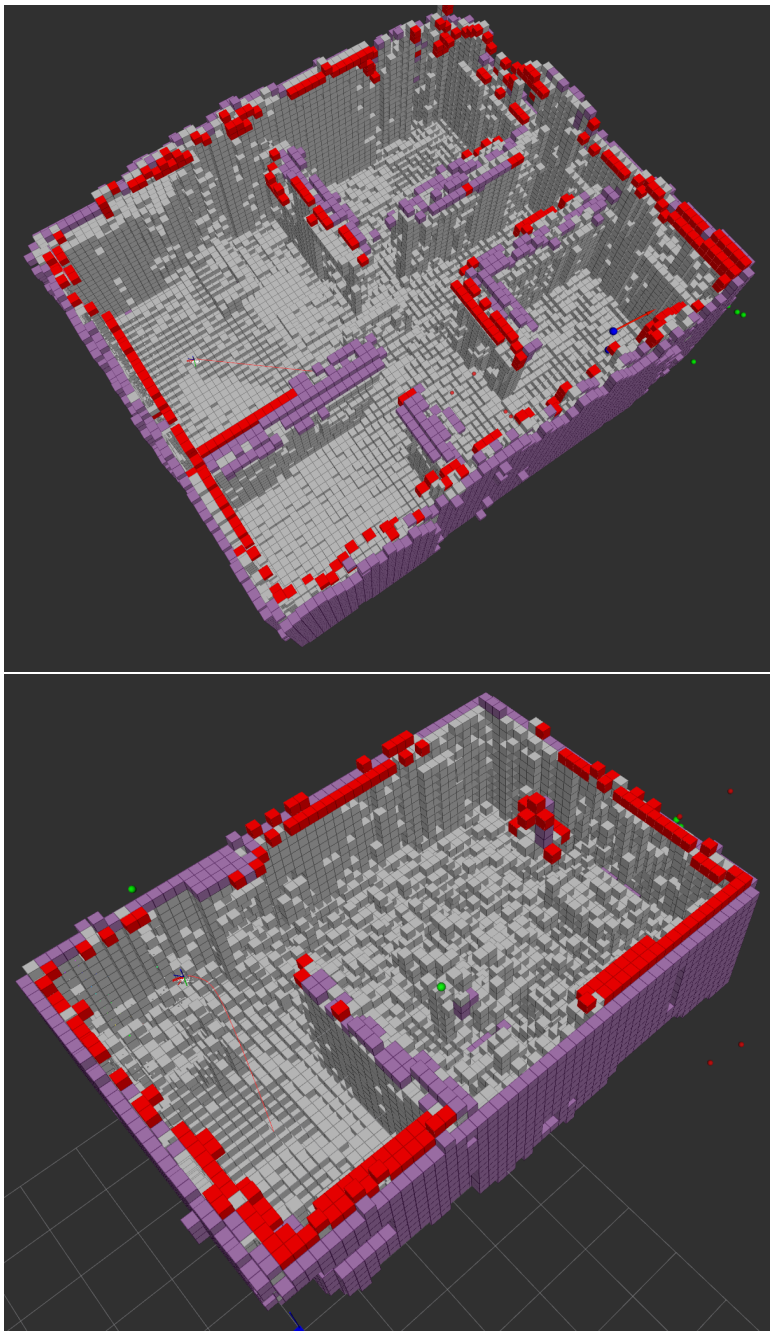

Figure 8.5: Environments reconstruction by the exploration

Chapter 9 Conclusion

9.1 Exploration techniques benchmark

The first aspect described in this report is a comprehensive benchmarking of multi-robot coverage path planning techniques for known asset exploration. As discussed throughout this report, we have analyzed the results obtained by comparing three different techniques: our technique, NBV, and frontier-based approaches.

Starting from the state-of-the-art versions of the chosen techniques for benchmarking, we have adapted them to work with the exploration of known assets.

The results obtained demonstrate clear advantages of our approach with respect to the other two in terms of performance and adaptability to the selected assets.

Our technique achieves the highest energy efficiency independently of environmental complexity and requires total energy consumption up to three times less than competing methods. The energy-aware task assignment strategy reduces vertical movements, which are the most energy-intensive operations, and privileges lateral displacements.

The use of the PH-tree data structure to define unobserved regions guarantees coverage speeds higher than NBV and frontier-based techniques, especially in multi-robot scenarios. Another advantage of using the PH-tree is the reduction of total path lengths, demonstrating an efficient exploration strategy for next target choice and reduced computational time required to evaluate viewpoints.

With increasing dimension and complexity of assets, the performance obtained by our approach remains optimal and indicates good scalability properties for several applications. The other approaches present high performance degradation, especially for complex assets in the case of NBV and large shapes for frontier-based techniques.

The real experiments validate the results obtained by simulations and demonstrate the possibility of using our system in practical work such as building coverage.

While our approach presents better results than the other two approaches, there are areas to investigate as next steps. The system is based on reliable

communication between robots and the ground station, but in some cases this situation cannot be satisfied.

Investigation of communication-constrained scenarios could enhance the practical applicability of our technique.

9.2 Unknown exploration technique

As the second part of this work, we have developed an unknown exploration technique that presents several technical innovations contributing to improved performance.

By separating mapping, planning, and control processes, the system becomes more robust and reactive to environmental changes. The three-thread architecture enables concurrent operation of sensor data processing, path planning, and trajectory execution without blocking dependencies.

The use of a Spatial Hash Manager reduces collision detection complexity, guaranteeing real-time path planning in complex environments.

The bucket-based approach with efficient memory management enables rapid collision queries essential for dynamic exploration scenarios.

TSDF-informed sampling is the main innovation of the RRT* algorithm used, since the sampling process is based on data obtained by sensors and enables the choice of reliable points with respect to the geometric surface information acquired by the system.

The probabilistic sampling approach guided by TSDF values significantly improves path quality in complex environments.

The simulation results demonstrate the effectiveness of our unknown exploration framework. The energy consumption metrics (24J and 20J respectively) validate the system's suitability for battery-constrained MAVs, while the implementation of safety return-to-base mechanisms ensures robust operation during unexpected situations.

As next steps, the system can be redesigned for multi-robot scenarios, providing an improved version that can manage a large number of robots.

In this way, we can obtain the scalability limits of our implementation. Although the simulations provide good results, real-world validation across different environments is necessary for practical deployment confidence.

This research demonstrates the advancement in multi-robot exploration capabilities of our techniques with respect to state-of-the-art techniques.

The comprehensive approach addressing both known and unknown environment scenarios provides a complete framework for autonomous exploration systems. The performance obtained guarantees energy efficiency with high coverage speed and system robustness against collisions between robots. The work represents a foundation for future research in energy-efficient autonomous exploration in known and unknown spaces that can be applicable for real-world inspection and monitoring scenarios.

Bibliography

[1] L. Walti, I. K. Erunsal, and A. Martinoli, "Multi-robot online coverage with a team of resource-constrained micro aerial vehicles," *IEE*, 2024.

[2] I. D. Changoluisa Caiza, A. Milas, M. A. Monte Grova, F. J. Pearez-Grau, and T. Petrovic, "Autonomous exploration of unknown 3d environments using a frontier-based collector strategy," *IEEE Internal Conference on Robotics and Automation (ICRA)*, 2024.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleyniova, and R. Siegwart, "Receding horizon next-best-view planner for 3D exploration," *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[4] T. ZÃ shke, C. Zimmerli, and M. C. Norrie, "The ph-tree-a space-efficient storage structure and multi-dimensional index," *Proceedings of the 2014 ACM Sigmod International Conference on Management of Data*, 2014.

[5] A. Munir and R. Parasuraman, "Exploration exploitation tradeoff in the adaptive information sampling of unknown spatial fields with mobile robots," *Sensors*, 2023.

[6] T. Cieslewsky, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," *IEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[7] H. Zhang, S. Wang, Y. Liu, P. Ji, R. Yu, and T. Chao, "Efp: Efficient frontier-based autonomous uav exploration strategy for unknown environments," *IEEE Robotics and Automation Letters*, 2024.

[8] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, 2021.

[9] Z. Zhengming, "Geometry-informed path planning for steel structure inspection," *Semester Project EPFL*, 2025.

[10] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," *RSS Workshop on Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*, 2016.

[11] B. Plancher, "Parallel optimization for robotics: An undergraduate introduction to gpu parallel programming," *IEEE IPDPSW*, 2024.

[12] R. Monica and J. Aleotti, "Contour-based next-best view planning from point cloud segmentation of unknown objects," *Autonomous Robots*, 2018.

[13] J. I. Vasquez and L. E. Suca, "Next-best-view planning for 3d object reconstruction under positioning," *Mexican International Conference on Artificial Intelligence*, 2011.

[14] M. Maboudi, M. Homaei, S. Song, S. Malihi, M. Saadatseresht, and M. Gerke, "A review on viewpoinrs and path planning for uav-based 3-d reconstruction," *IEEE Journal of selected topics in applied Earth observation and remote sensing*, 2023.

[15] D. Morilla-Cabello, L. Bartolomei, L. Teixeira, E. Montijano, and M. Chli, "Sweep-your-map: Efficient coverage planning for aerial teams in large-scale environments," *IEEE Robotics and Automation Letters*, 2022.

Chapter 10 Appendix

10.1 Installing/configuring instructions

10.1.1 Third-parties library

The library are located in ./libraries/ directory and are:

- **GeometricTools**
- **PH-Tree C++**
- **manif**
- **TSL**

10.1.2 Configuration

The parameters used in the NBV and frontier-based technique is in the path ./config/base.yaml with description.

General

- **trajectory_directory**: folder for trajectory data
- **filename**: mesh name of object to scan
- **task_generation_technique**: type of task generation used (Our method,NBV,Frontier-based)

NBV parameters

- **nbv/gain/free**: gain coefficient for free cells
- **nbv/gain/occupied**: gain coefficient for occupied cells
- **nbv/gain/unmapped**: gain coefficient for unmapped cells
- **nbv/gain/degressive_coeff**: degressive coefficient for gain
- **nbv/gain/zero**: threshold to consider zero gain
- **nbv/gain/limit_x_low**: high limit of sampling on x direction

- **nbv/gain/limit_x_high**: low limit of sampling on x direction
- **nbv/gain/limit_y_low**: high limit of sampling on y direction
- **nbv/gain/limit_y_high**: low limit of sampling on y direction
- **nbv/gain/limit_z_low**: high limit of sampling on z direction
- **nbv/gain/limit_z_high**: low limit of sampling on z direction

Frontier-based parameters

- **frontier/similarity_threshold**: similarity threshold for merging frontiers
- **frontier/max_frontiers**: maximum number of frontiers to keep
- **frontier/min_frontier_distance**: minimum distance between frontiers
- **frontier/max_frontier_distance**: maximum distance between frontiers
- **frontier/min_frontier_gain**: minimum gain for a frontier to be considered
- **frontier/box_info_gain_size**: size of the box used to compute the gain
- **frontier/knowledge_gain**: gain for the knowledge of the environment
- **frontier/resolution**: resolution of the map used to compute the frontiers
- **frontier/lambda**: lambda parameter for the frontier gain
- **frontier/distance_weight**: weight for the distance in the frontier gain
- **frontier/bandwidth**: Bandwidth for frontier gain
- **frontier/energy_cost_factor**: energy cost factor for the frontier gain

10.1.3 Running instructions

If you wnat to run the asset coverage you need to start in one terminal the following node:

```
roslaunch exploration ground_station.launch sim:=<false|true>
rviz:=<false|true> technique:=<0|1|2>
```

The sim parameter enables the system to work in virtual simulation or in a real experiment, the rviz parameter enable the visualization of the process with sever ros messages plotted and the technique parameter enables the choice of the technique you want to use. To choose the correct technique you can follow this Table:

Number	Technique
0	Our technique
1	NBV technique
2	Frontier-based Technique

In another terminal you can start a node for each robot:

```
roslaunch exploration robot.launch robot:=<"robot_name">
debug:=<true|false> arm:=<false|true> sim:=<false|true>
```

or several robots can be launched by the same terminal:

```
roslaunch exploration multi.launch arm:=<false|true>
sim:=<false|true>
```

If you want to run the exploration framework for unkonw environment, you need to start the following node in a terminal:

```
roslaunch exploration exploration.launch arm:=<false|true>
sim:=<false|true> rviz:=true
```

This framework works only for single robot. All experiment's data will be saved into the following folder: ./logs/trajectories. For the asset coverage technique, the data will be saved into a subfolder based on the sim parameter. If true, the data are saved in the `Simulation` subfolder, otherwise in the `Real` subfolder. Each experiment will have its own folder that follows the following nomenclature:

{`date_and_time`}_`technique`_{`technique_number`}_`object`_{`object_name`}

If you want to obtain the different graphs present in the results chapter, you can find several jupyter notebook in ./logs folder that generate all the graphs and save them into the ./logs/comparison. If not already present, the system will create a folder with a name that follows this nomenclature:

`{object_name}_num_robots_{number_of_robots_used}`

For the exploration of unknown environments, the data are saved within the Unknown subfolder. Each experiment has its own folder that follows this nomenclature:

`{date_and_time}_environment_{environment_number}`

10.2 Arena real experiment

Figure 10.1: Arena used for real experiment of coverage techniques

10.3 Meshes used

Figure 10.2: Armadillo's asset mesh

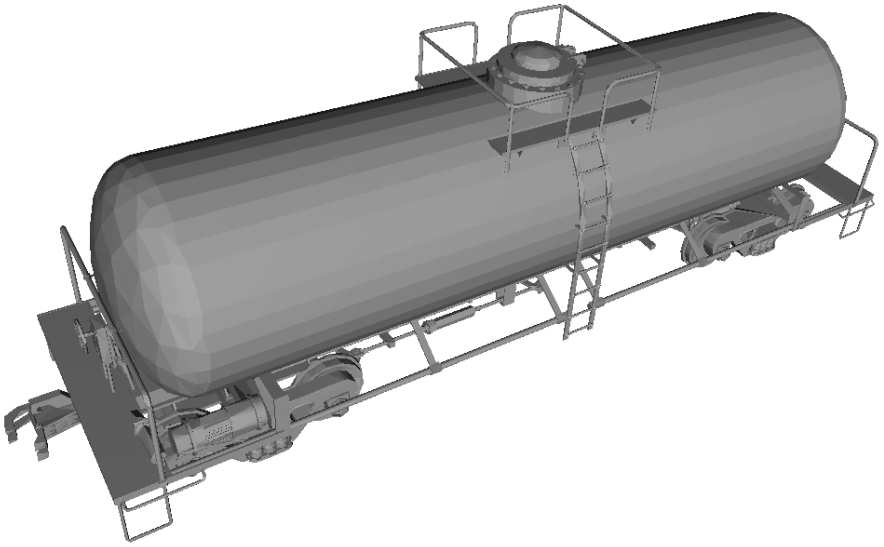

Figure 10.3: Tank car's asset mesh

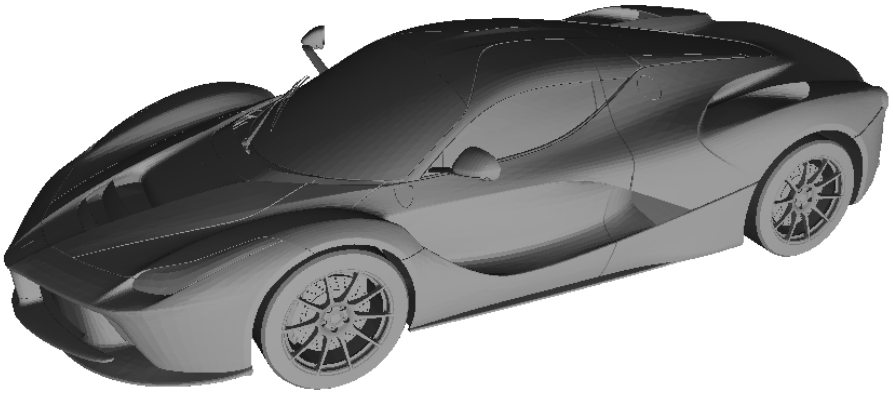

Figure 10.4: Ferrari's asset mesh

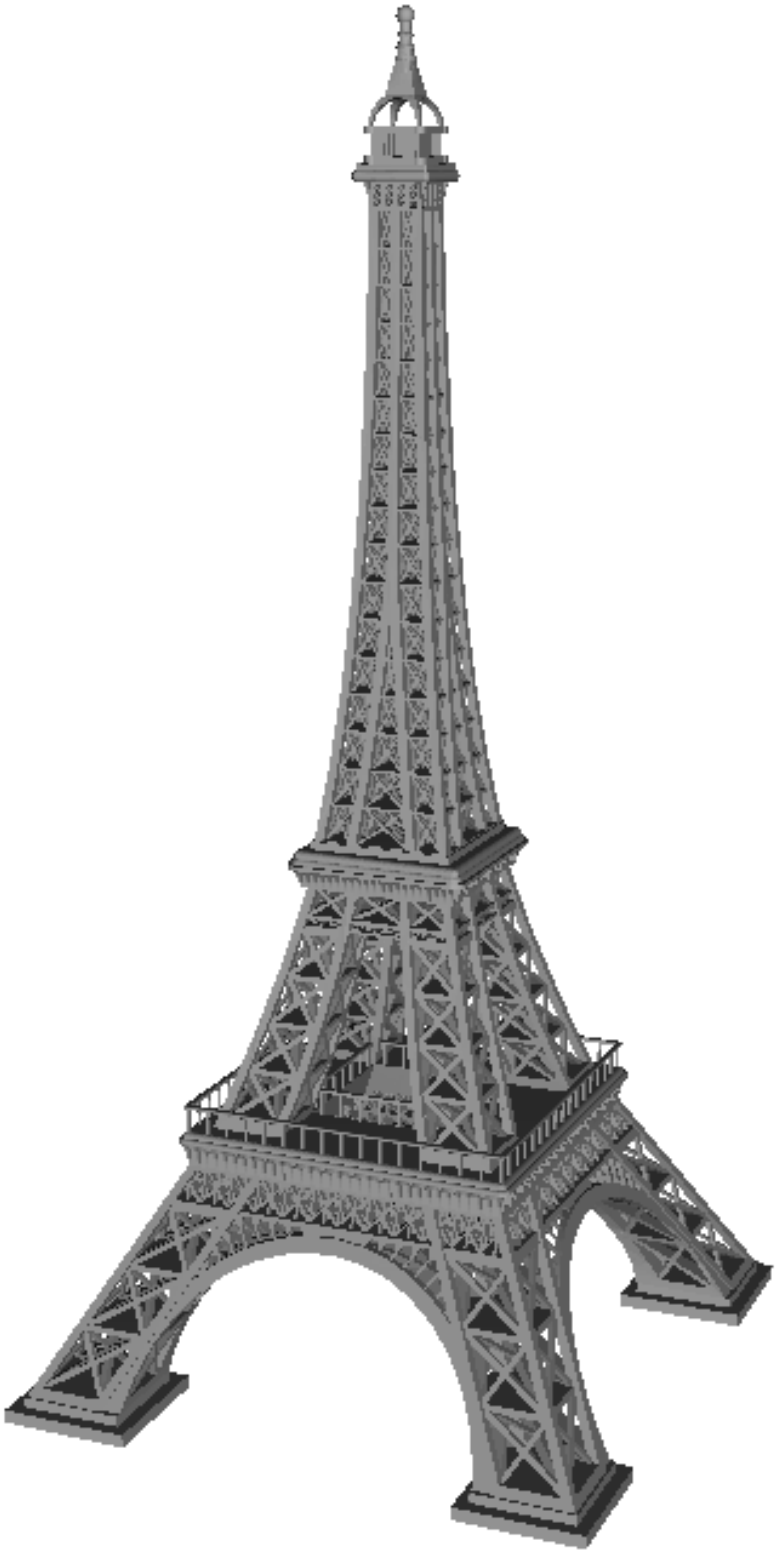

Figure 10.5: Eiffel tower's asset mesh

10.4 Experiments Results

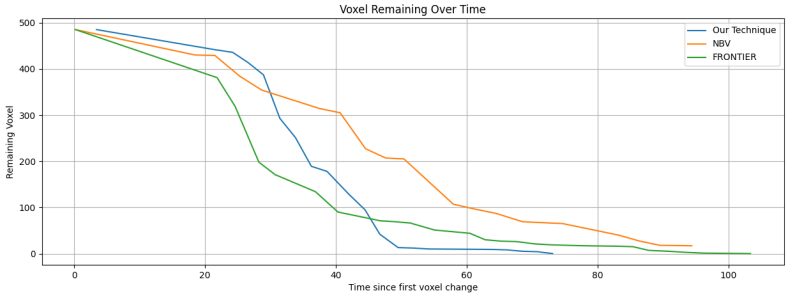

Figure 10.6: Exploration comparison of the three techniques using single robot for mockup coverage

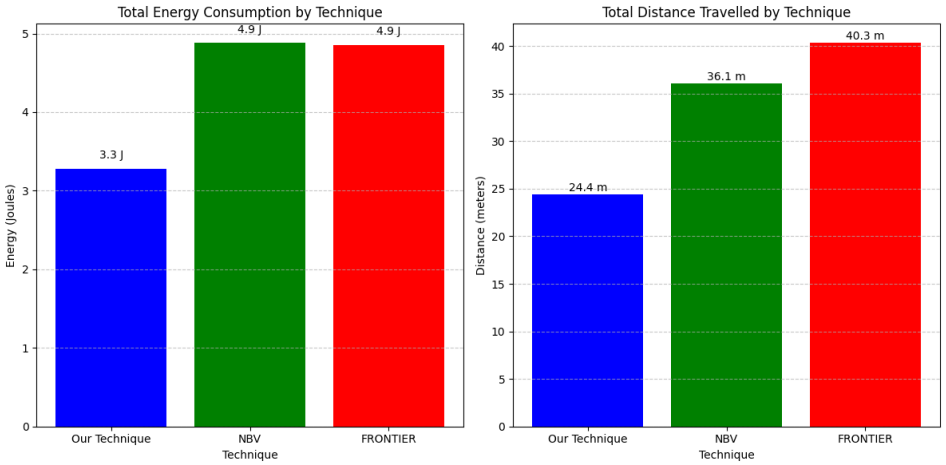

Figure 10.7: Histogram of distance traveled and energy consumption during exploration of mockup asset using single robot

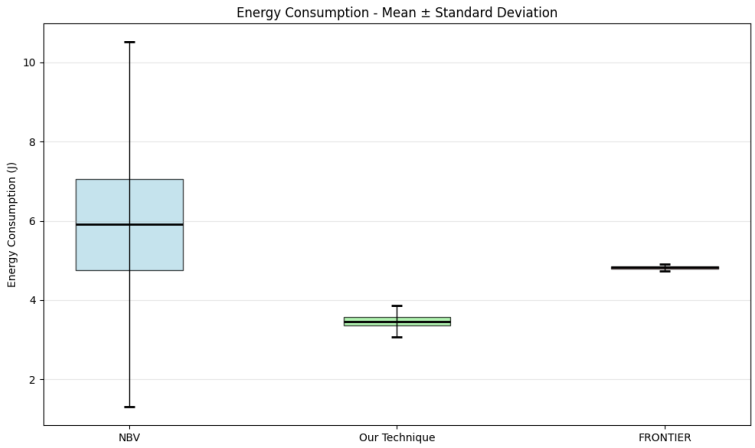

Figure 10.9: Box plot of energy consumption by single robot during mockup coverage

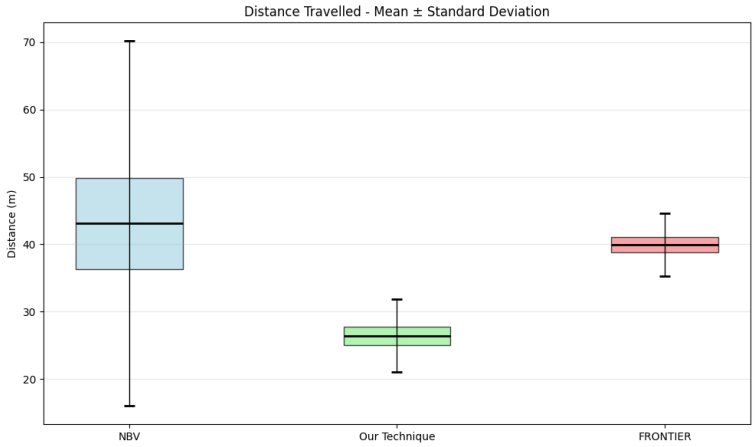

Figure 10.8: Box plot of distance traveled by single robot during mockup coverage

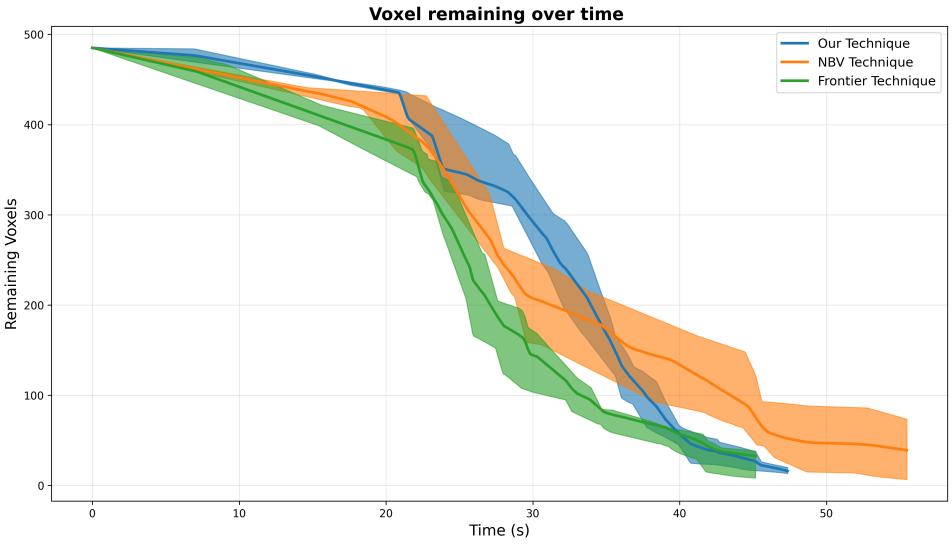

Figure 10.10: Exploration comparison of the three techniques using 3 robots for mockup coverage

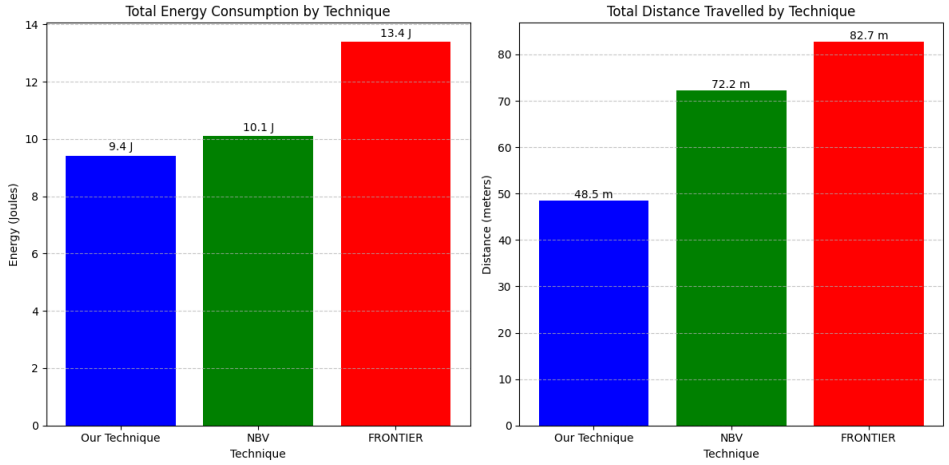

Figure 10.11: Histogram of distance traveled and energy consumption during exploration of mockup asset using 3 robots

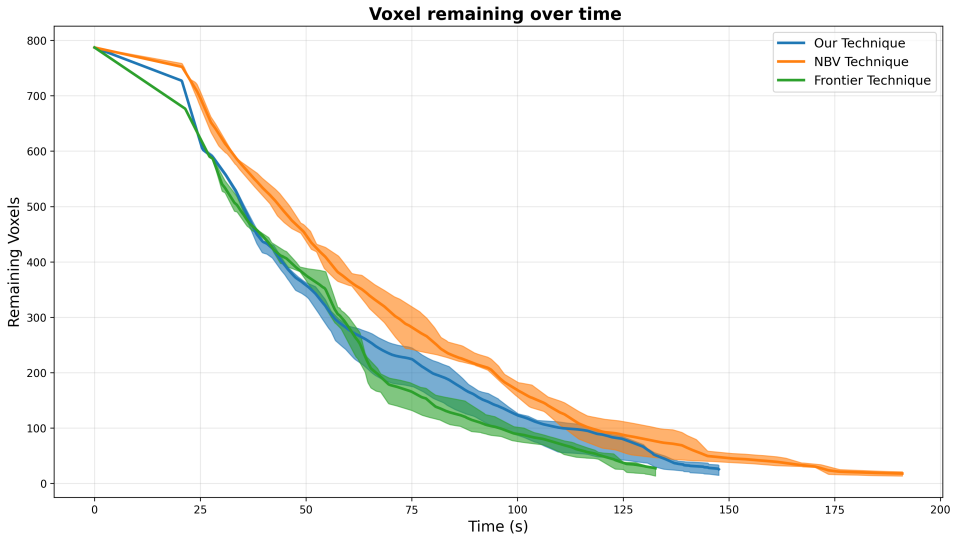

Figure 10.12: Exploration comparison of the three techniques using single robot for Ferrari coverage

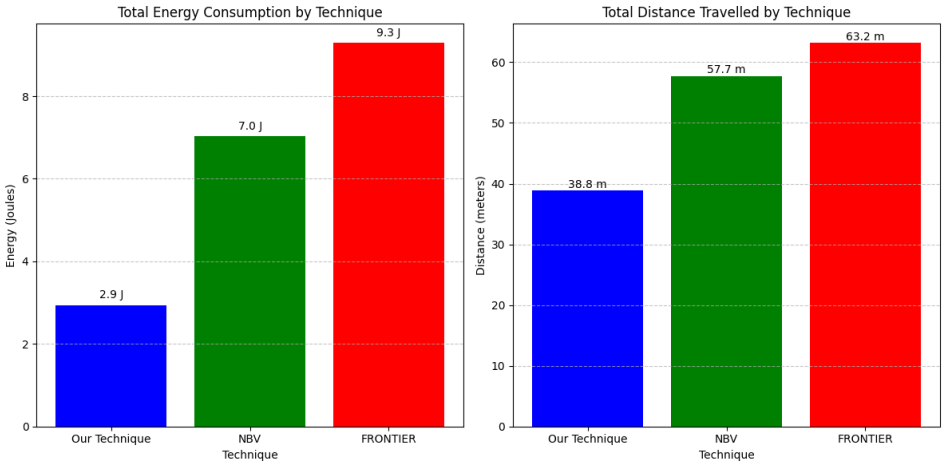

Figure 10.13: Histogram of distance traveled and energy consumption during exploration of Ferrari asset using single robot

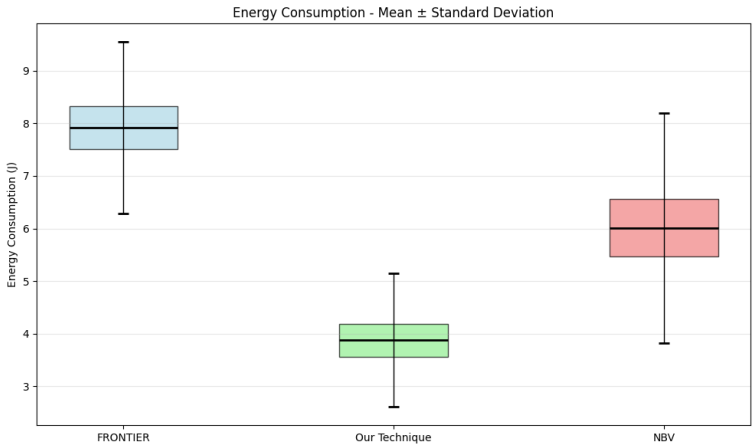

Figure 10.15: Box plot of energy consumption by single robot during Ferrari coverage

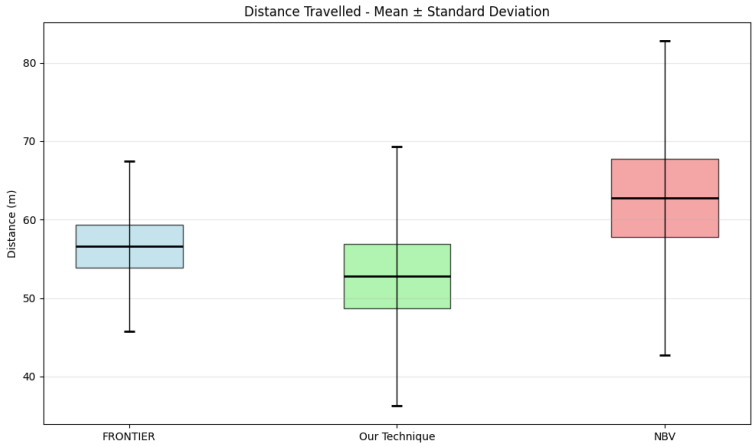

Figure 10.14: Box plot of distance traveled by single robot during Ferrari coverage

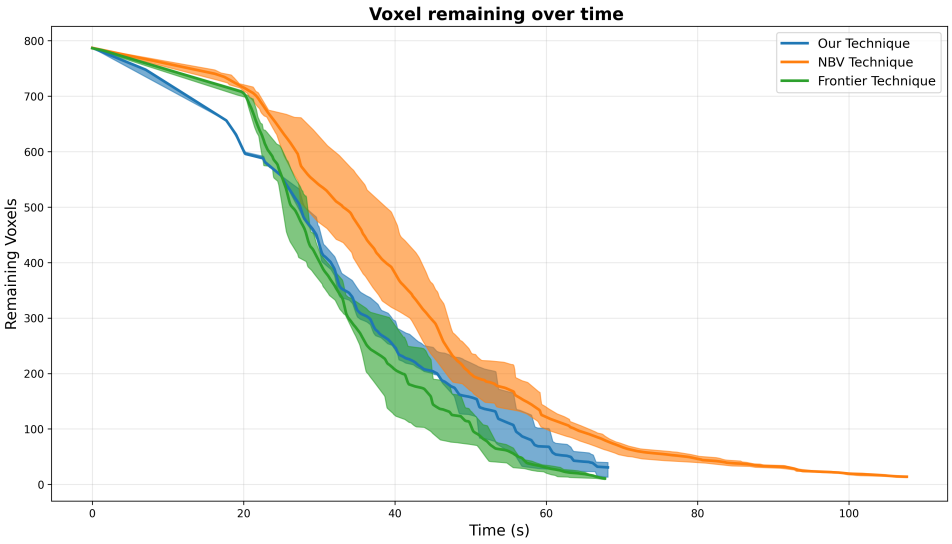

Figure 10.16: Exploration comparison of the three techniques using 3 robots for Ferrari coverage

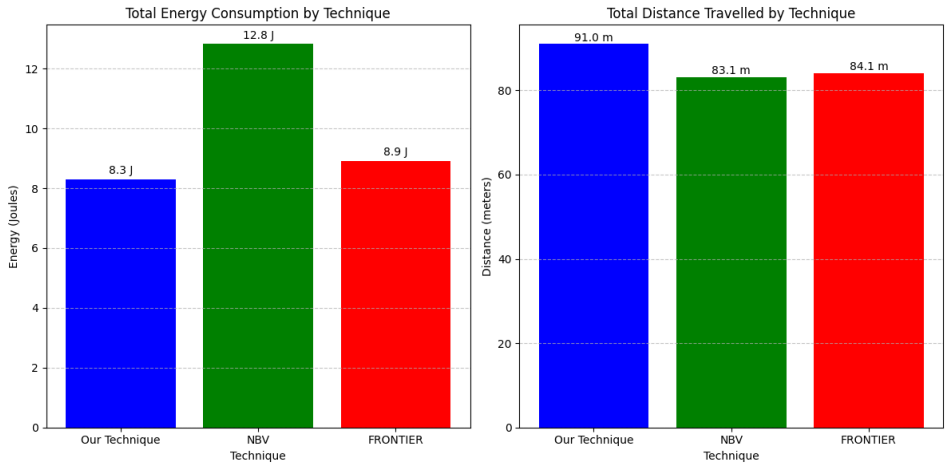

Figure 10.17: Histogram of distance traveled and energy consumption during exploration of Ferrari asset using 3 robots

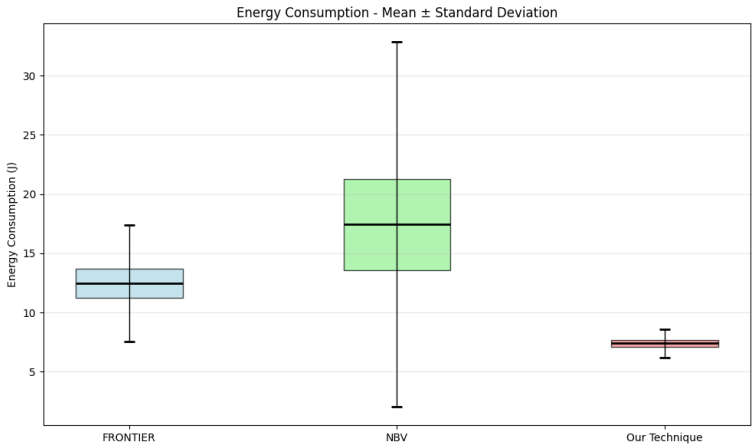

Figure 10.19: Box plot of energy consumption by 3 robots during Ferrari coverage

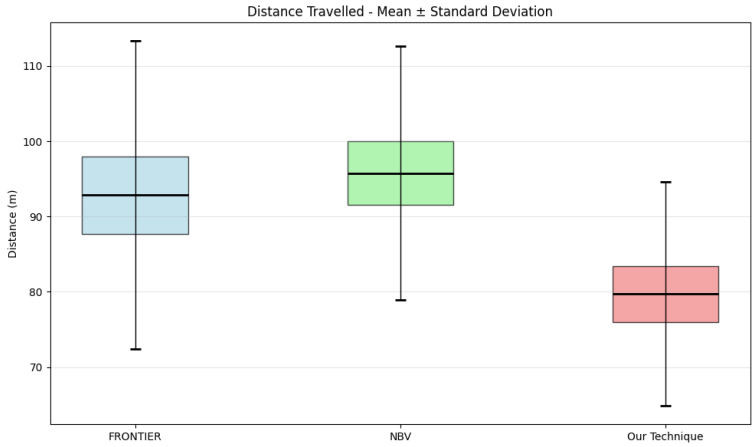

Figure 10.18: Box plot of distance traveled by 3 robots during Ferrari coverage

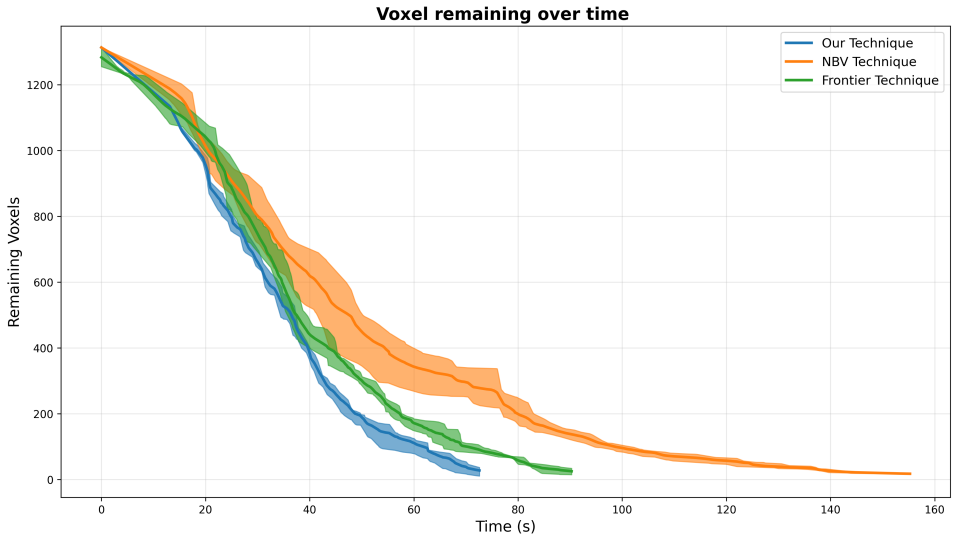

Figure 10.20: Exploration comparison of the three techniques using 3 robots for Armadillo coverage

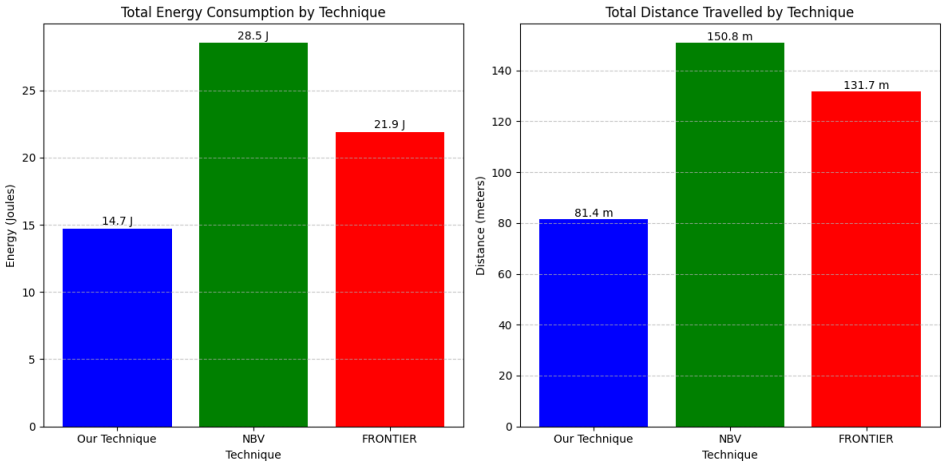

Figure 10.21: Histogram of distance traveled and energy consumption during exploration of Armadillo asset using 3 robots

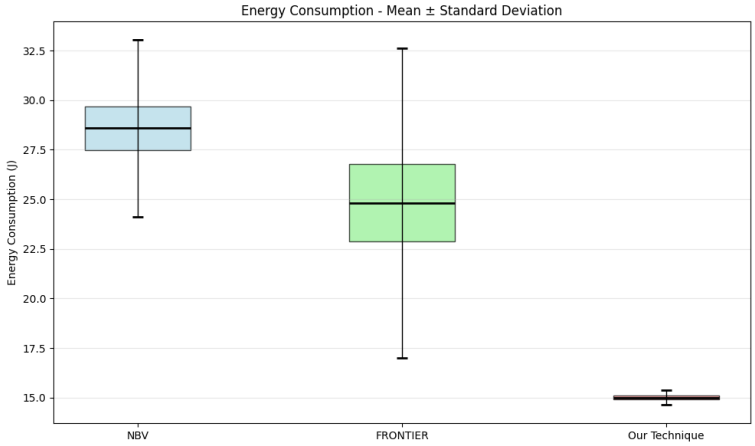

Figure 10.23: Box plot of energy consumption by 3 robots during Armadillo coverage

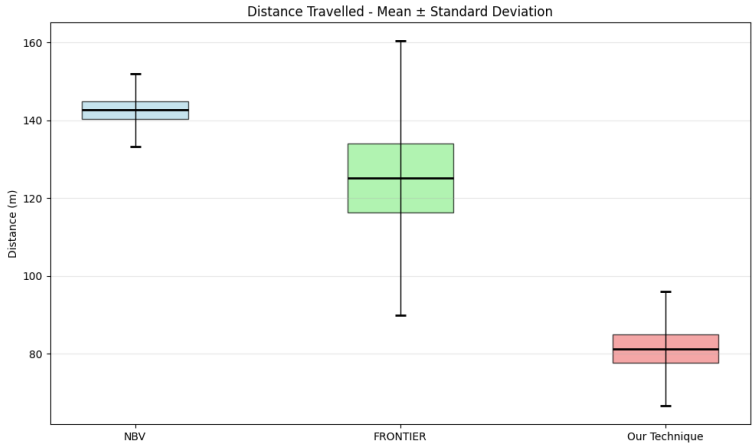

Figure 10.22: Box plot of distance traveled by 3 robots during Armadillo coverage

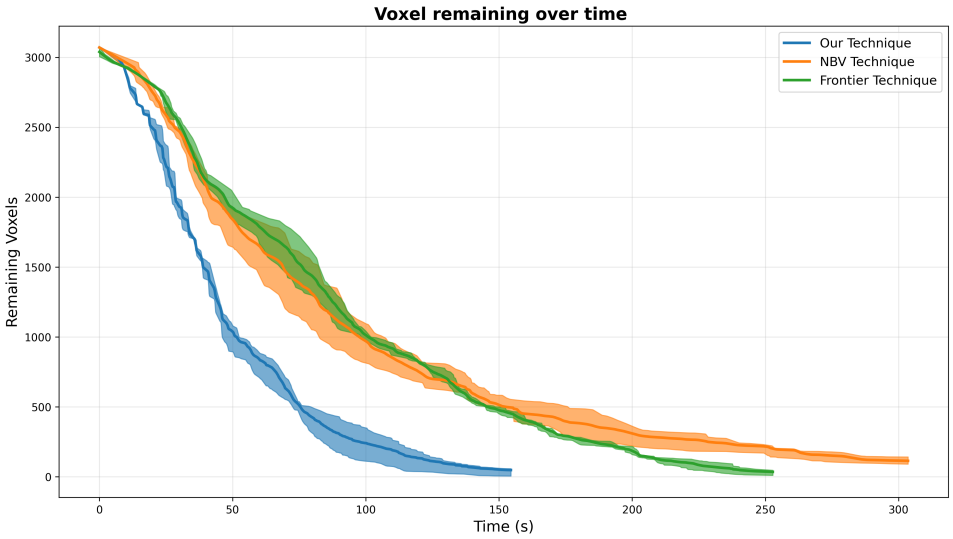

Figure 10.24: Exploration comparison of the three techniques using 3 robots for tank coverage

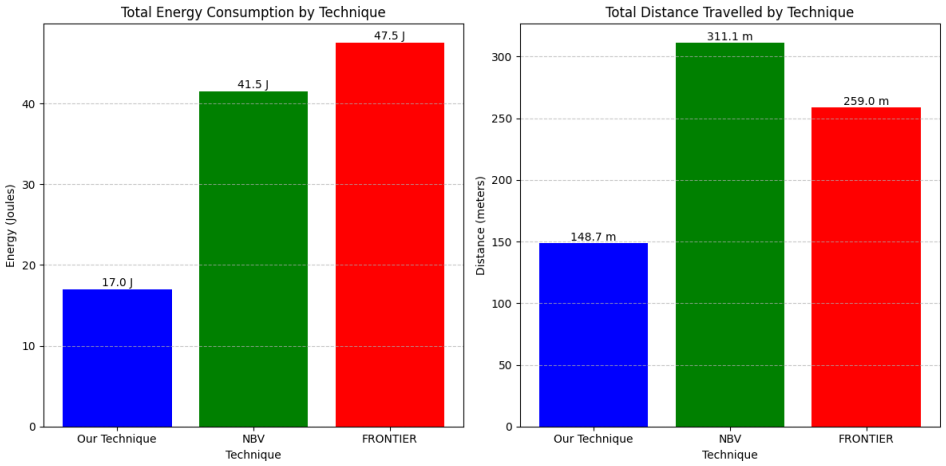

Figure 10.25: Histogram of distance traveled and energy consumption during exploration of tank asset using 3 robots

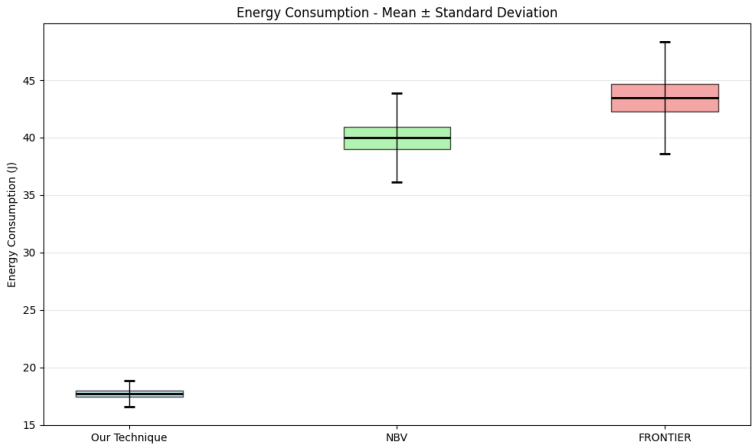

Figure 10.27: Box plot of energy consumption by 3 robots during tank coverage

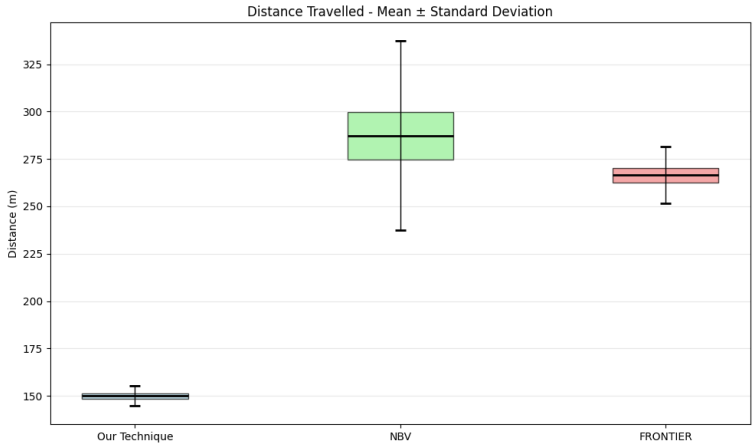

Figure 10.26: Box plot of distance traveled by 3 robots during tank coverage

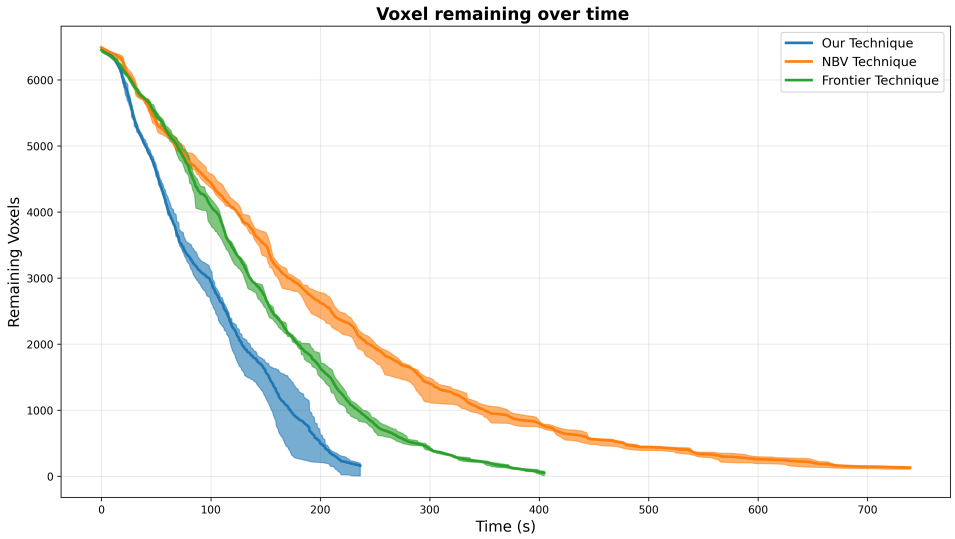

Figure 10.28: Exploration comparison of the three techniques using 3 robots for Eiffel coverage

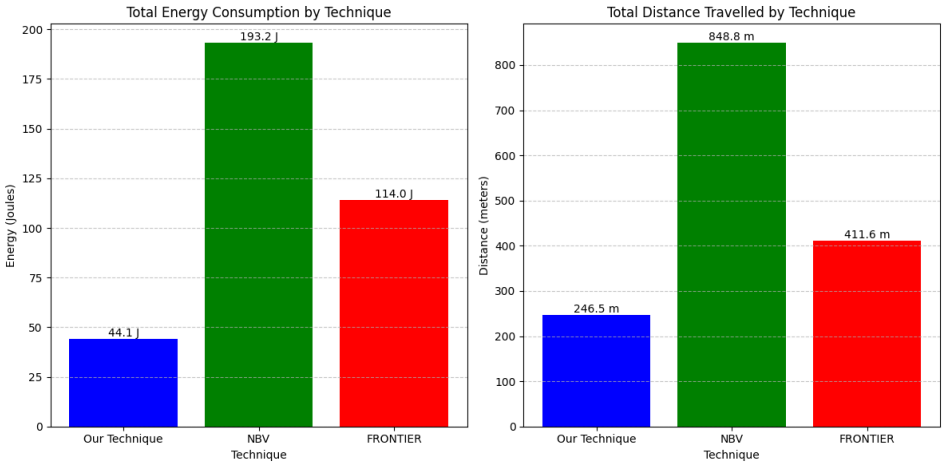

Figure 10.29: Histogram of distance traveled and energy consumption during exploration of Eiffel asset using 3 robots

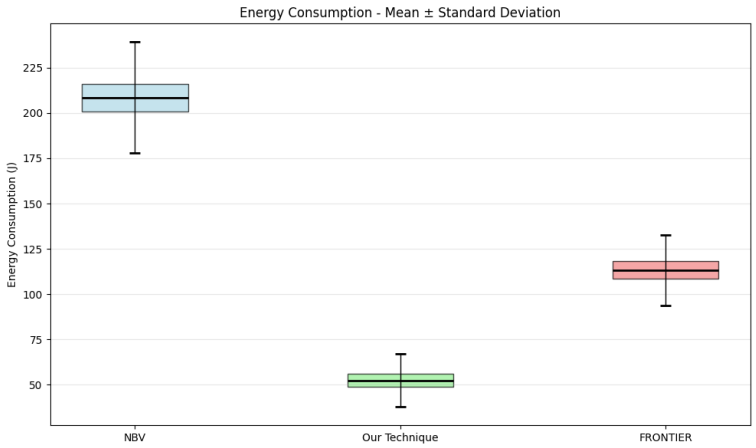

Figure 10.31: Box plot of energy consumption by 3 robots during Eiffel coverage

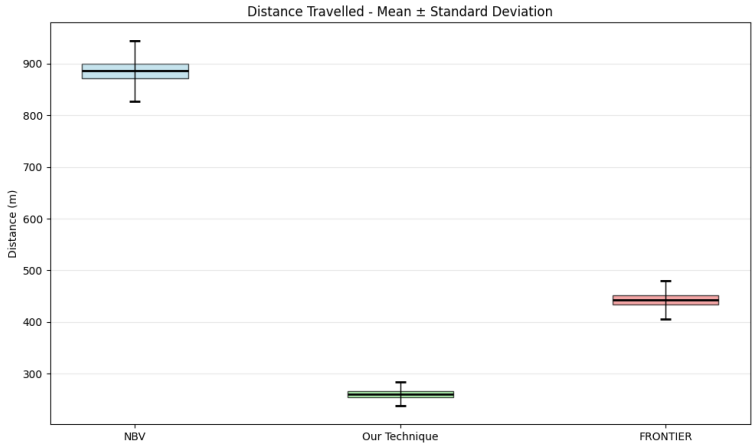

Figure 10.30: Box plot of distance traveled by 3 robots during Eiffel coverage

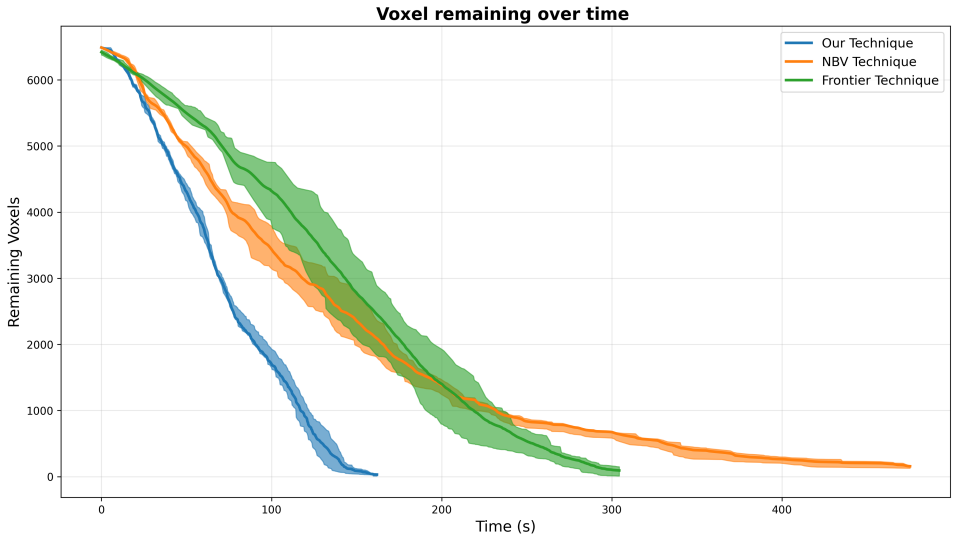

Figure 10.32: Exploration comparison of the three techniques using 5 robots for Eiffel coverage

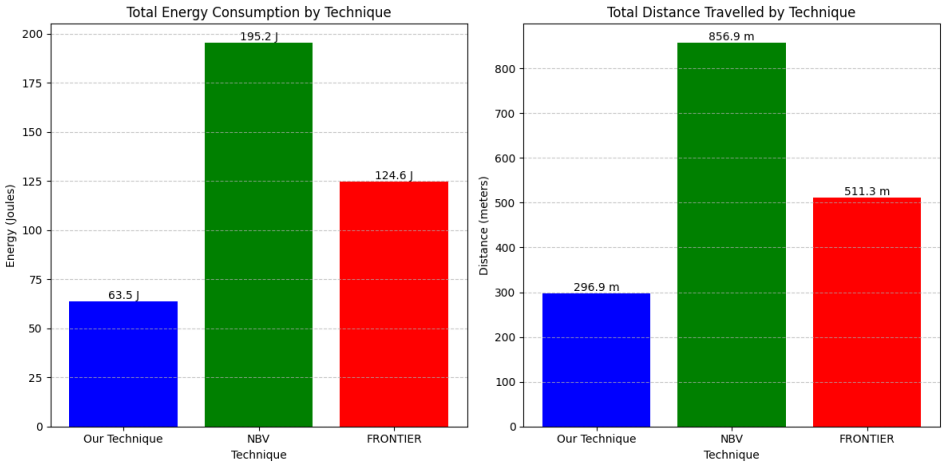

Figure 10.33: Histogram of distance traveled and energy consumption during exploration of Eiffel asset using 5 robots

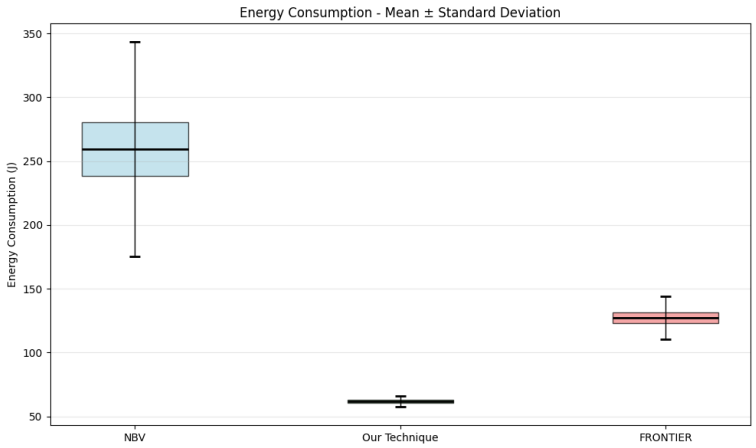

Figure 10.35: Box plot of energy consumption by 3 robots during Eiffel coverage

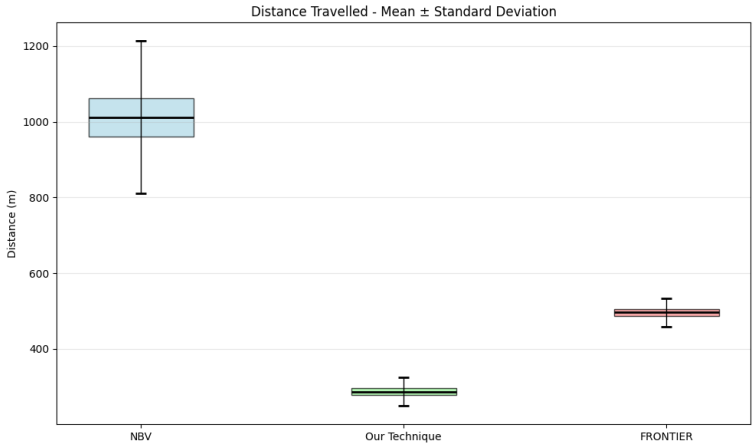

Figure 10.34: Box plot of distance traveled by 3 robots during Eiffel coverage

Acknowledgments

Con la stesura di questa tesi termina una fase importante della mia vita che sancisce l'inizio di una nuova avventura nel mondo dei grandi.

Per questo motivo, vorrei ringraziare le persone che mi sono state accanto negli ultimi due anni e che hanno reso in parte possibile questo traguardo. In primis, vorrei ringraziare i professori Marcello Chiaberge, Alcherio Martinoli e Lucas Waelti per avermi seguito durante la realizzazione del progetto di tesi e di essere sempre stati disponibili ad un chiarimento o ad una discussione sullo step successivo da realizzare.

Un ringraziamento particolare e dal profondo del cuore va ai miei genitori e a mio fratello, che mi hanno fatto sentire la loro vicinanza costante e incondizionata nei momenti più difficili e delicati di questo lungo percorso. Senza il vostro amore, il vostro sostegno e la vostra fiducia, tutto questo non sarebbe stato possibile.

A mia madre, la donna più forte che conosca, che mi ha insegnato l'arte della perseveranza e del sacrificio anche quando tutto sembra impossibile. Mi è sempre stata accanto nelle decisioni più difficili e importanti della mia vita, mettendoci sempre tutto il cuore e la determinazione nell'aiutarmi a risolvere ogni problema che abbiamo dovuto affrontare insieme durante questo percorso universitario. Le tue parole di incoraggiamento nei momenti di sconforto, la tua capacità di trovare sempre una soluzione anche nelle situazioni più complicate, e il tuo amore incondizionato sono stati la mia ancora di salvezza. Oggi, se sono qui a festeggiare la fine dei miei studi e l'inizio di una nuova fase della mia vita, è soprattutto grazie a te, ai tuoi insegnamenti e ai valori che mi hai trasmesso con l'esempio quotidiano.

A mio padre, l'uomo che mi ha insegnato l'importanza dell'onestà, dell' integrità e di seguire sempre le proprie passioni nella vita. È sempre stato d'accordo con le mie scelte, anche quelle più audaci e rischiose, senza mai esitare un istante e mostrandomi sempre piena fiducia nelle mie capacità. La tua disponibilità ad aiutarmi, sia fisicamente nei momenti in cui avevo bisogno di una mano concreta, sia economicamente, è stata fondamentale per permettermi di concentrarmi sui miei studi senza preoccupazioni eccessive. Mi hai insegnato che il lavoro duro e la dedizione ripagano sempre, e che è importante rimanere sempre fedeli ai propri principi.

A mio fratello, che con la sua naturale bontà d'animo mi ha insegnato l'importanza dell'aiutare gli altri e dell'essere sempre disponibili per chi ne ha bisogno. A modo tuo, con quella discrezione e delicatezza che ti contraddistinguono, mi sei sempre stato vicino, assecondando le mie decisioni anche quando non le comprendevi completamente e mostrando una pazienza infinita nei miei momenti di maggiore stress e nervosismo. Grazie per non esserti mai arrabbiato e per avermi ricordato l'importanza di non perdere il sorriso e la capacità di vedere il lato positivo delle cose.

Una dedica importante va alla persona che dal primo giorno del mio percorso mi è stata vicina: Valentina. La distanza che ci divideva avrebbe potuto creare problemi a qualsiasi coppia, ma tu hai compreso i miei sogni e sei sempre stata felice delle scelte che ho preso, anche quando significavano sacrifici e rinunce per entrambi. Le interminabili videochiamate insieme per poterci raccontare le nostre giornate e i nostri stati d'animo mi hanno fatto andare avanti con la consapevolezza di avere una persona che crede in me incondizionatamente.

Hai ascoltato con pazienza tutti i miei progetti, dalle idee più ambiziose ai dubbi più piccoli, e hai sempre apprezzato l'impegno che mettevo nel realizzarli. Quando le difficoltà sembravano insormontabili, la tua voce al telefono era la certezza che tutto sarebbe andato bene. La tua consulenza ha avuto un valore inestimabile per me: i tuoi consigli saggi e la tua capacità di vedere le cose da prospettive diverse mi hanno permesso di crescere non solo come studente, ma come persona. Grazie per aver trasformato ogni mia piccola vittoria in una festa condivisa e per aver saputo trovare le parole giuste nei momenti più difficili. La tua fiducia in me non ha mai vacillato, nemmeno quando io stesso dubitavo delle mie capacità. Tutto questo mi ha permesso di essere oggi qui a festeggiare la fine di questo percorso con la consapevolezza di aver sempre avuto al mio fianco la persona giusta. Nei tuoi occhi vedo l'orgoglio nei miei confronti e quanto tu non avessi mai dubbi delle mie vittorie, anche quando la strada sembrava in salita. Ti auguro di portare a termine tutti i tuoi obiettivi con la stessa determinazione che hai sempre dimostrato nel sostenermi, e stai certa che sarò sempre lìad essere il tuo consigliere, come tu sei stata per me la mia guida più preziosa.

Un ringraziamento speciale e profondo va a tutti i miei familiari: nonne, zii e cugine per essermi stati vicini con affetto costante in tutti i momenti più delicati e significativi di questo percorso. La vostra presenza, anche quando fisica non poteva esserci, si è sempre fatta sentire attraverso una telefonata nel momento giusto, un messaggio di incoraggiamento o semplicemente la certezza di sapere che c'eravate. Mi avete ascoltato con pazienza nelle mie incertezze e mi avete consigliato con saggezza nelle scelte più importanti, offrendomi sempre punti di vista diversi che mi hanno aiutato a vedere le situazioni da prospettive nuove. Avete sempre creduto in me, anche quando io stesso faticavo a farlo, vedendo sempre il meglio delle mie qualità e sostenendomi nei momenti di sconforto. Con il vostro esempio di vita mi avete

dimostrato che con l'impegno, la costanza e la determinazione si può raggiungere qualsiasi vetta, anche quella che inizialmente sembra impossibile da scalare. I vostri racconti di sacrifici e vittorie mi hanno insegnato il valore del duro lavoro e l'importanza di non arrendersi mai di fronte alle difficoltà. Per alcuni di voi Torino ha un significato particolare e profondo, legato a ricordi, esperienze e momenti che hanno segnato la vostra vita. Spero che questo giorno cosìimportante vi rievochi tutti i momenti belli e significativi passati in questa città meravigliosa, e che possiate sentire quanto questa laurea sia anche un po' vostra, frutto dei valori che mi avete trasmesso e del supporto che non mi avete mai fatto mancare.

Agli amici di sempre, quelli con cui tutto è iniziato: Vito, Vito, Cosimo, Antonello, Antonello, Emilio, Alberto, Antonio, Marco, Mario, Paolo e Nicola. La vostra amicizia è stata una delle costanti più preziose di questi anni universitari. Avete reso questo percorso più leggero e sopportabile con la vostra capacità unica di sdrammatizzare tutto, scherzando su quello che studiavo e persino sui presunti soldi guadagnati durante l'esperienza in Svizzera. Le vostre battute e le vostre prese in giro affettuose sono state spesso la medicina migliore contro lo stress e l'ansia da prestazione. La vostra amicizia autentica non è mai andata scemando nonostante la distanza fisica che ci ha separati in questi anni, e questo è la dimostrazione più grande del legame sincero che ci unisce. Avete continuato a considerarmi parte del gruppo, a includermi nei vostri progetti e a farmi sentire che il mio posto tra voi era sempre lìad aspettarmi, ogni volta che tornavo.

Durante il primo anno ho conosciuto molte persone e con alcune di esse ho avuto la fortuna di stringere un'amicizia sincera e duratura. Voglio ringraziare tutto il gruppo Electrolux: Federico, Gregorio, Marco, Michele, Ilario, Adrian, Isacco e Federico per avermi accettato per quello che sono, senza mai giudicarmi o cercare di cambiarmi. Mi avete fatto sentire valorizzato e importante, dandomi la sensazione di appartenere davvero a qualcosa di speciale. Con voi ho condiviso risate, sfide accademiche e momenti di pura spensieratezza. Rimarranno per sempre impressi nella mia memoria tutti quei momenti indelebili a base di Tamango, le serate infinite passate a discutere di tutto e di niente, e quella complicità che si crea solo tra veri amici.

L'esperienza in Svizzera mi ha aperto un mondo completamente nuovo. Ho conosciuto gente proveniente da ogni angolo del pianeta, ognuno con la propria storia, le proprie tradizioni e i propri sogni. Tuttavia, il richiamo alla madre patria è sempre stato forte, e trovare altri italiani in terra straniera è stato come ritrovare un pezzo di casa lontano da casa. Vorrei ringraziare Francesco, Edoardo, Arianna, Federica,Camilla,Francesco, Lorenzo, Andrea e Gregorio per essere stati i miei compagni in questa avventura straordinaria e per aver arricchito questa esperienza con la vostra presenza costante e il vostro supporto. Vi voglio ringraziare per tutti quei pranzi e cene improvvisati insieme, dove con ingredienti semplici e pochi franchi svizzeri rius-

civamo a creare momenti di vera felicità. La vostra capacità di trasformare anche la giornata più difficile in un'occasione di condivisione, una semplice risata al momento giusto o un punto di vista diverso che mi apriva nuove prospettive, mi ha dato la forza di continuare anche quando tutto sembrava più complicato del previsto.

Un ringraziamento speciale e sentito va ai miei coinquilini, che hanno condiviso con me non solo gli spazi fisici di casa ma anche le gioie, le preoccupazioni e le fatiche di questo intenso percorso universitario. Vivere insieme significa conoscersi davvero, nei momenti belli e in quelli più difficili.

A Piero e Sebastiano, per aver reso il primo anno da fuorisede molto meno pesante di quanto avrei mai immaginato. Avete sopportato con pazienza le mie videochiamate ad alto volume e le mie scelte culinarie. Mi avete assecondato in tutte le mie decisioni più malsane, come quella di continuare a cucinare con quel forno preistorico degli anni '80 che sembrava uscito da un museo.

A Thomas, Yacine ed Emiliane, che hanno saputo accogliermi con calore in una realtà completamente nuova, facendomi sentire parte integrante del gruppo sin dal primo giorno. Avete reso i momenti di convivialità unici e indimenticabili attraverso lo scambio reciproco dei prodotti tipici delle nostre nazioni: dalle specialità francesi di Emiliane ai piatti tradizionali di Yacine, fino alle birre che Thomas portava dal Belgio. Ogni cena diventava un viaggio culinario e culturale che mi ha arricchito profondamente, insegnandomi che le differenze sono una ricchezza quando condivise con il cuore aperto.

Per ultimo, ma non meno importante, vorrei ringraziare me stesso per essere stato sempre determinato a continuare, anche quando la strada si faceva in salita. Nei momenti di vera difficoltà, quando tutto sembrava crollare e i dubbi iniziavano a farsi strada, ho trovato la forza di lottare per i miei sogni senza mai arrendermi o buttarmi giù. Ho imparato ad essere il mio primo alleato, a credere nelle mie capacità anche quando gli altri dubitavano, e a rialzarmi ogni volta che cadevo. Non ho idea di cosa mi riserverà il futuro, ma sono certo che non avrò mai rimpianti per quello che ho realizzato e ottenuto con le mie sole forze, con determinazione, sacrificio e una buona dose di coraggio.