



POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es



Jorge Sanz Marrero

05 TRABAJO FIN DE MASTER

INDUSTRIALES

TRABAJO FIN DE MASTER

AUTONOMOUS 3D OBJECT SEARCHING PLANNER

SEPTIEMBRE 2025

Jorge Sanz Marrero

DIRECTOR DEL TRABAJO FIN DE MASTER:

Dr. Paloma de la Puente Yusty

Dr. Hannes Gamper



POLITÉCNICA



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Industriales



Master Degree in Automation and Robotics

AUTONOMOUS 3D OBJECT SEARCHING PLANNER

MASTER THESIS

Submitted for the degree of Master by:

Jorge Sanz Marrero

Under the supervision of:
Dr. Paloma de la Puente Yusty
Dr. Hannes Gamper

Madrid, 2025

I would like to express my deepest gratitude to the incredible people at CERN, particularly to my supervisor, Hannes, for his invaluable guidance, support, and encouragement throughout this project. I am equally grateful to Professor Paloma from the UPM, whose insightful advice and mentorship provided me with the clarity and confidence to navigate this research.

To my family and my friends—your unwavering support, patience, and encouragement have been invaluable throughout this journey. Whether it was through advice, understanding, or simply being there when it mattered most, you all played a crucial role in making this work possible.

Thank you for always standing by me.

Acknowledgement

This thesis has been partially supported by CERN, specifically the BE-CEM-MRO section, through a grant that enabled the research and development of this work. I am deeply grateful for the material and personal support provided throughout this project, as well as the invaluable guidance and resources that contributed to its completion.

Abstract

Efficient object search in unknown, cluttered environments remains a critical challenge in robotics, particularly under occlusions and partial observability. This thesis presents HOPE-OS (Heuristic Occlusion-aware Planning for Efficient Object Search), a novel heuristic-based Next Best View (NBV) planning algorithm that combines gradient-based Region of Interest (ROI) maps with occlusion-aware Field of View (FoV) simulation. The method guides the robot to informative viewpoints that maximize visibility while minimizing unnecessary motion.

HOPE-OS operates without prior environmental knowledge, supports multiple ROIs and target objects, is robot-agnostic, and achieves improved computational efficiency over existing approaches. Extensive evaluations were conducted in 2D and 3D simulated environments, including a realistic mock-up of CERN’s Large Hadron Collider (LHC), as well as real-world tests using a 9-degree-of-freedom UR10 robot. Notably, we introduce a novel benchmarking protocol comparing robot decisions to human strategies, demonstrating that HOPE-OS achieves human-level search efficiency.

The contributions of this work include (1) a real-time, heuristic NBV planning algorithm aware of occlusions, (2) a gradient-based ROI clustering approach for viewpoint sampling, and (3) a human-robot evaluation framework. This research advances the state of the art in robotic object searching, providing a versatile and efficient framework with potential applications ranging from industrial maintenance to search-and-rescue operations.

Table of Contents

Acknowledgement	iii
Abstract	v
List of Figures	viii
List of Tables	x
Abbreviations and acronyms	xii
Abbreviations and acronyms	xiii
1 Introduction	1
1.1 Motivation	1
1.2 System Requirements and Assumptions	1
1.3 Use cases	2
2 State of the art	5
2.1 Background Review	5
2.1.1 Definition of Object Searching Problem	5
2.1.2 Main Challenges	6
2.1.3 Different Methodologies for Object Searching	7
2.1.4 Limitations of Previous Work in Object Search	16
2.2 Heuristic-Based Search and Contributions	19
3 Materials and methods	23
3.1 Software	23
3.2 Hardware	25
3.3 2D Simulation	27
3.3.1 2D Simulated Environment	27
3.3.2 Metrics for Evaluation	29
3.3.3 Basic Search Algorithms	31
3.3.4 Gradient Representation	36
Integration into the Cost Function	41
Algorithm Workflow	41
Advantages of the Gradient-Based Approach	41
3.3.5 Approach with Field of View Simulation - HOPE-OS 2D	42
Clustering for Computational Efficiency	43
Incorporating Field of View Simulation	48
Enhanced orientation Selection	49

	Algorithm Workflow	51
	Summary	52
	Advantages of the Enhanced Approach	54
3.3.6	Human Decision Process	54
3.4	HOPE-OS 3D Simulation	56
3.4.1	3D Simulated Environment	56
3.4.2	Algorithm Workflow	57
	1. ROI and Obstacles Gradient Map Generation	57
	2. Evaluating Positions with FoV Simulation and Cost Function	60
	3. 3D Orientation selection	62
3.4.3	Summary of the HOPE-OS 3D Exploration Workflow	65
3.5	Real World Implementation	66
3.5.1	Frame Transformation	66
3.5.2	Octree Generation	66
3.5.3	CRF Implementation	67
4	Results and Discussion	69
4.1	2D Simulations	69
4.1.1	Basic Algorithms	69
	2D Basic Algorithms Behavior	72
	2D Basic Algorithms Comparison	73
4.1.2	FoV Simulation (HOPE-OS 2D) vs Gradient Algorithm Test	76
4.1.3	FoV Simulation (HOPE-OS 2D) Vs Human Decision Process	79
4.2	3D Simulations	83
4.3	Real World Tests	87
5	Conclusion and Future Work	93
5.1	Conclusions	93
5.1.1	Summary of Contributions	94
5.1.2	Strengths and Limitations	94
5.2	Future Work	94
Annexes		101
	Annex I - Sustainable Development Goals (SDGs)	101
	Annex II - Impact Evaluation and Ethical-Legal Considerations	103
	Annex III - Temporal Planning and Execution Budget	105

List of Figures

1.1	BLM sensor attached to the LHC pipe.	3
1.2	BLMs possible positions in the LHC	4
1.3	TIM robot.	4
3.1	Intel RealSense Depth Camera D435i	25
3.2	Kinova Gen3	26
3.3	TIM Robot	26
3.4	2D 50x50 Occupancy Gridmap	27
3.5	Sensor's FoV using ray tracing	28
3.6	Orientation selection in basic algorithms	31
3.7	Closest frontier example	32
3.8	Furthest frontier example	33
3.9	Information Gain Algorithm example	34
3.10	ROI Algoritghm expamle	36
3.11	Average ROI Position Error	36
3.12	ROI Gradient Map	39
3.13	Obstacles Gradient Map	39
3.14	Unexplored Gradient Map	40
3.15	Combined Gradient Map	40
3.16	Gradient Approach NBV	42
3.17	Desired NBV	43
3.18	Illustration of k-means clustering applied to the free space of the map. The centroids (marked as crosses) represent candidate positions for evaluation. . .	44
3.19	Clustering process	47
3.20	Reclustering process	47
3.21	Candidate Position 360° FoV Simulation	48
3.22	Sensor's FoV simulation for orienation selection	50
3.23	Algorithm Workflow Scheme	53
3.24	Human Decision Game Interface and Logo	55
3.25	3D Simulated Environment	56
3.26	3D ROI Gradient function	58
3.27	ROI Gradient 3D visual representation	58
3.28	3D Obstacles Gradient function	59
3.29	Obstacles Gradient 3D visual representation	59

3.30	Clustering and reclustering in 3D	60
3.31	Spherical FoV 3D simulation	61
3.32	Displacement Cost Function	62
3.33	3D Orientations Sampling	63
3.34	Sensor's Frustum Simulation	64
3.35	Sensor's Frustum Orientations Resampling	64
3.36	Octree visual representation	66
3.37	Octree generation example	67
3.38	CRF Object searching implementation scheme	68
3.39	CNN detection of the BLM	68
4.1	2D 50x50 Gridmap	70
4.2	Iterations required to solve the maps: (Top) positions only, (Bottom) positions and orientations.	71
4.3	Behavior plots of 2D algorithms	72
4.4	Convergency Test	74
4.5	Comparison of 2D basic algorithms	75
4.6	Grid maps for 2D testing	77
4.7	Algorithm behavior when α and β variations are applied	78
4.8	Algorithm results when α and β variations are applied. HOPE-OS 2D 1 (1), HOPE-OS 2D 2 (2), HOPE-OS 2D 3 (3)	79
4.9	Behavior plot: HOPE-OS 2D Algorithm vs Human Decision Process	81
4.10	Results plot: HOPE-OS 2D Algorithm (2) vs Human Decision Process (1)	81
4.11	Environments for 3D testing	84
4.12	3D Algorithm behavior	85
4.13	3D Algorithm results	86
4.14	Real world scenario: UR10 in the mock-up of the LHC	88
4.15	Box-plots of performance metrics from real-world tests	89
4.16	Real world test. Decision making example.	90
1	SDGs goals	101
2	Execution Budget	107

List of Tables

2.1 Characteristics of Different Object Search Methods 16

Abbreviations and acronyms

UPM Universidad Politécnica de Madrid

BLM Beam Loss Monitor

CERN European Organization for Nuclear Research

LHC Large Hadron Collider

BLM Beam Loss Monitor

CRF CERN Robotic Framework

FoV Field of View

LHC Large Hadron Collider

NBV Next Best View

NBP Next Best Position

NBO Next Best Orientation

ROI Region of Interest

SLAM Simultaneous Localization and Mapping

TIM Train Inspection Monorail

RGB-D Red, Green, Blue + Depth

2D Two-Dimensional

3D Three-Dimensional

UNESCO Codes

1203.02 Algorithmic Languages

1203.15 Heuristic

1203.26 Simulation

1203.14 Environment Control Systems

1203.17 Computational Methods

3311.01 Automation Technology

3311.02 Control Engineering

3399 Robotics

Glossary

Object Search The decision-making process of identifying and localizing a target object within an environment using active perception, planning, and sensor input.

Exploration The autonomous navigation of unknown environments with the goal of mapping or understanding them.

Field of View (FoV) The observable area a sensor can capture at a given time.

Frontier The boundary between known and unknown areas in a map used in exploration planning.

Heuristic A rule or method used to guide search algorithms toward optimal or near-optimal solutions.

Information Gain A metric used to evaluate how much uncertainty is reduced by a specific observation or action.

Next Best View (NBV) The optimal viewpoint that maximizes expected information gain during exploration.

Occupancy Grid Map A representation of the environment where each cell indicates the probability of being occupied.

Octomap A 3D mapping framework based on octrees that efficiently represents occupancy in large environments.

Pose The position and orientation of a robot or sensor in space.

Region of Interest (ROI) A selected portion of the environment likely to contain a target or useful information.

Volumetric Mapping Mapping technique that represents space using 3D volumes or voxels.

Voxel A volumetric pixel used in 3D grids to represent parts of space, typically in occupancy maps.

Units

All physical quantities in this thesis follow the International System of Units (SI), unless otherwise specified. The following units are commonly used throughout the document:

Length meter (m)

Time second (s)

Speed / Velocity meters per second (m/s)

Mass kilogram (kg)

Angle degree ($^{\circ}$), radian (rad)

Angular Velocity degrees per second ($^{\circ}/s$), radians per second (rad/s)

Volume cubic meter (m^3)

Chapter 1

Introduction

1.1 Motivation

In recent years, the field of robotics has witnessed significant advancements, leading to a surge in the variety and scope of its applications. As robots transition from research laboratories to real-world environments, solving complex challenges such as autonomous object searching in three-dimensional (3D) spaces has become increasingly critical. The capability of efficiently locating objects is essential in many high-impact applications—such as industrial inspection, warehouse automation, search-and-rescue missions, or domestic assistance. It is particularly valuable in scenarios where humans face challenges related to safety, time constraints, or cognitive overload.

Searching for objects mirrors a fundamental cognitive process in humans, where efficiently scanning environments and prioritizing attention is crucial for survival and decision-making. Translating this ability to robots involves both perception and strategic planning, especially in unstructured 3D environments.

From a broader perspective, autonomous object searching is not merely a technical challenge, it represents a fundamental step toward creating intelligent systems capable of understanding and interacting with their human-centered environments. This synergy between human ingenuity and robotic precision is poised to transform industries, enhance safety, and improve quality of life on a global scale.

1.2 System Requirements and Assumptions

The searching algorithm must ensure precise and effective autonomous exploration of three-dimensional environments while maintaining adaptability to various tasks and robot configurations. The exploration process should be thorough, ensuring comprehensive coverage of the necessary space to locate targets in an optimal manner, regardless of environmental complexity or obstacles.

The system is required to locate predefined objects with known characteristics, such as

geometry, color, or texture. This capability leverages prior knowledge, including user-defined preferences, object correlations, or historical data about probable object locations. Such information enhances the efficiency of the search process by guiding the exploration towards areas with higher likelihoods of success.

An essential assumption is that the robot’s position on the map is known at all times, enabling effective navigation and exploration. Accurate localization ensures that the system can plan efficient trajectories and systematically cover the search area. To enhance practicality and feasibility in real-world applications, the algorithm and associated processes are designed to run on a CPU within a reasonable time frame, minimizing dependency on extensive computational resources.

The exploration and mapping processes will employ octree-based maps, a spatial data structure that efficiently represents three-dimensional space. Octrees allow for hierarchical partitioning of the environment, enabling effective storage, retrieval, and updates of environmental information [1]. This approach ensures that the system can dynamically manage complex environments with varying levels of detail.

For exploration and detection, the system relies on a single RGB-D RealSense D435i camera, mounted on the robot’s end-effector. This camera provides high-resolution color and depth data, allowing the system to perceive and interpret its surroundings accurately. The placement of the camera on the end-effector offers flexibility in positioning and orientation, enabling the robot to inspect areas from various angles and adapt to different viewing requirements. The RealSense D435i is particularly advantageous due to its integrated depth-sensing capabilities, which facilitate detailed mapping and obstacle avoidance.

By integrating this sensor setup with robust algorithms and efficient data structures, the system should achieve effective object searching and mapping while remaining adaptable to a variety of operational scenarios, from structured industrial settings to unstructured exploratory tasks.

1.3 Use cases

The European Organization for Nuclear Research (CERN) is one of the world’s largest and most respected centers for scientific research. It is home to the Large Hadron Collider (LHC), the world’s most powerful particle accelerator, which spans a circular underground tunnel of 27 kilometers near Geneva, Switzerland. The LHC is critical for advancing our understanding of fundamental physics, enabling experiments that collide particles at nearly the speed of light.

In the context of the LHC at CERN, precise and efficient maintenance is critical for the continued operation and safety of the facility. One key component in this endeavor is the detection and localization of Beam Loss Monitors (BLMs) installed throughout the LHC tunnel, Figure 1.1. These sensors are crucial for monitoring beam stability and preventing potential damage to the collider. Notwithstanding, they are not always easy to find, as shown in Figure 1.2.

The primary practical objective of this thesis is to develop an intelligent robotic system capable of autonomously locating BLM sensors within the LHC tunnel. The system will employ a robotic arm mounted on a monorail that runs along the ceiling of the tunnel, called TIM (Train Inspection Monorail) and shown in Figure 1.3. The arm will be equipped with an RGB-D camera at its end effector to facilitate 3D exploration and detection of the BLM sensors. The monorail setup allows the robot to efficiently cover large sections of the tunnel while capturing detailed visual and depth information. This ensures accurate detection and localization of the sensors, even in complex or cluttered environments.

While the primary focus is on locating BLMs, the system’s design emphasizes adaptability, making it suitable for a wide range of robots and applications. For instance, it can be repurposed for detecting water leaks in industrial environments, where timely identification and localization of leaks are critical for preventing downtime and reducing maintenance costs.

In scenarios involving hazardous environments, such as nuclear facilities or chemical plants, the system could be adapted for integration with quadruped robots, such as a dog robot, to safely approach and inspect areas that are too dangerous for human intervention. Beyond these applications, the system could be expanded to tasks like structural integrity inspections, detecting gas leaks, or monitoring environmental conditions in tunnels, pipelines, or confined spaces.

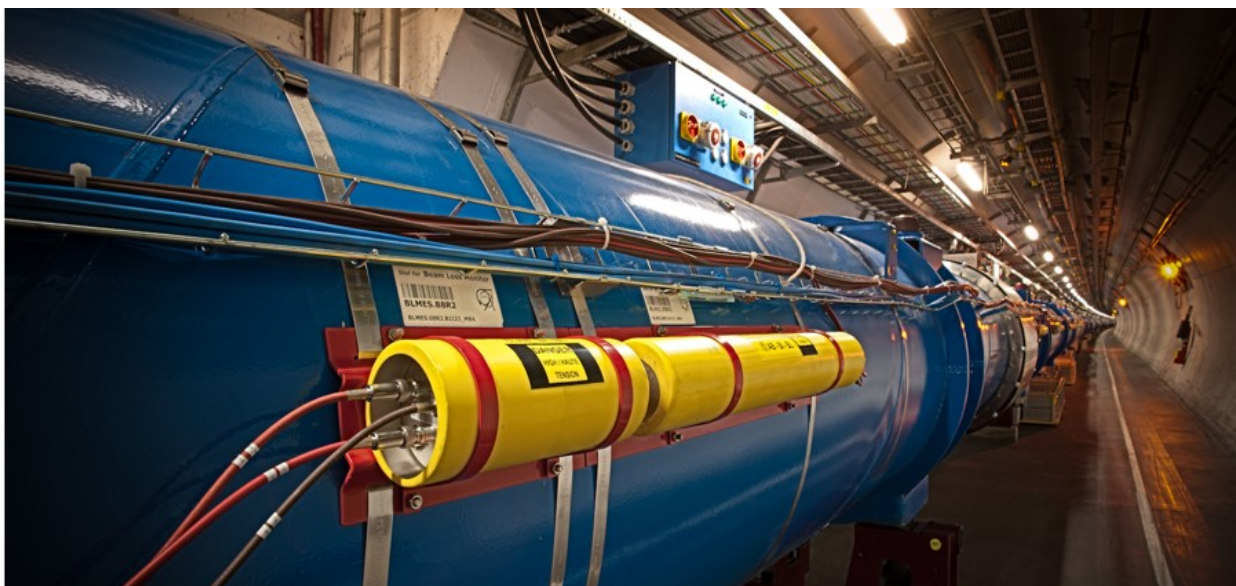


Figure 1.1: BLM sensor attached to the LHC pipe.

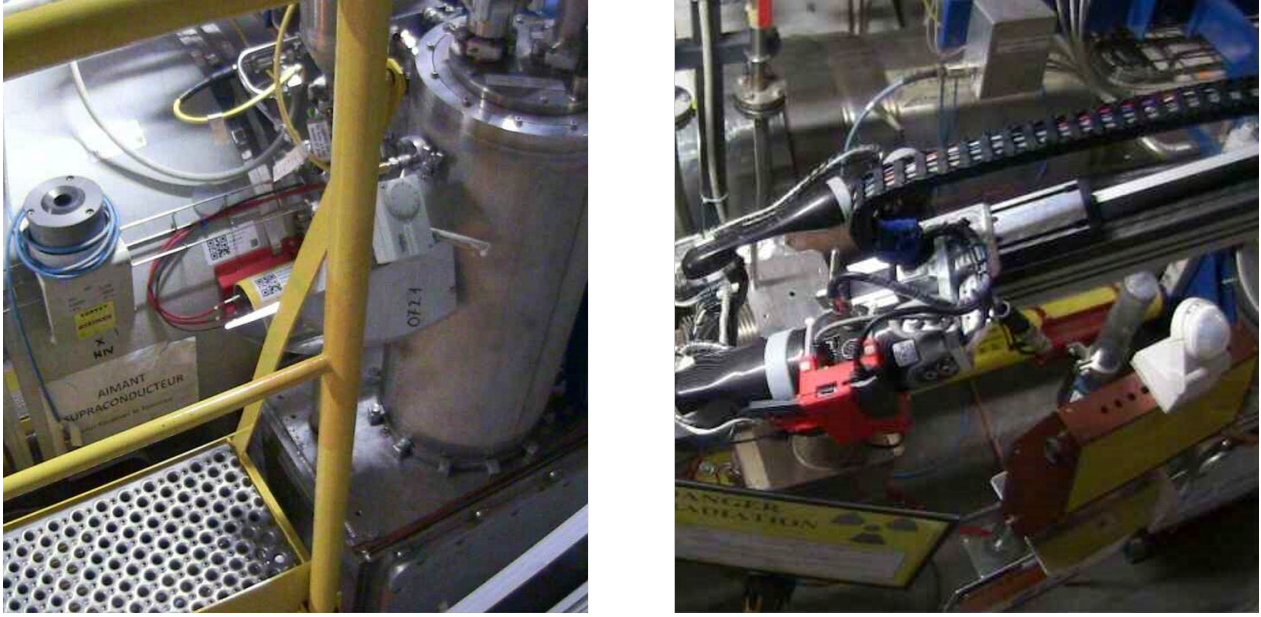


Figure 1.2: BLMs possible positions in the LHC

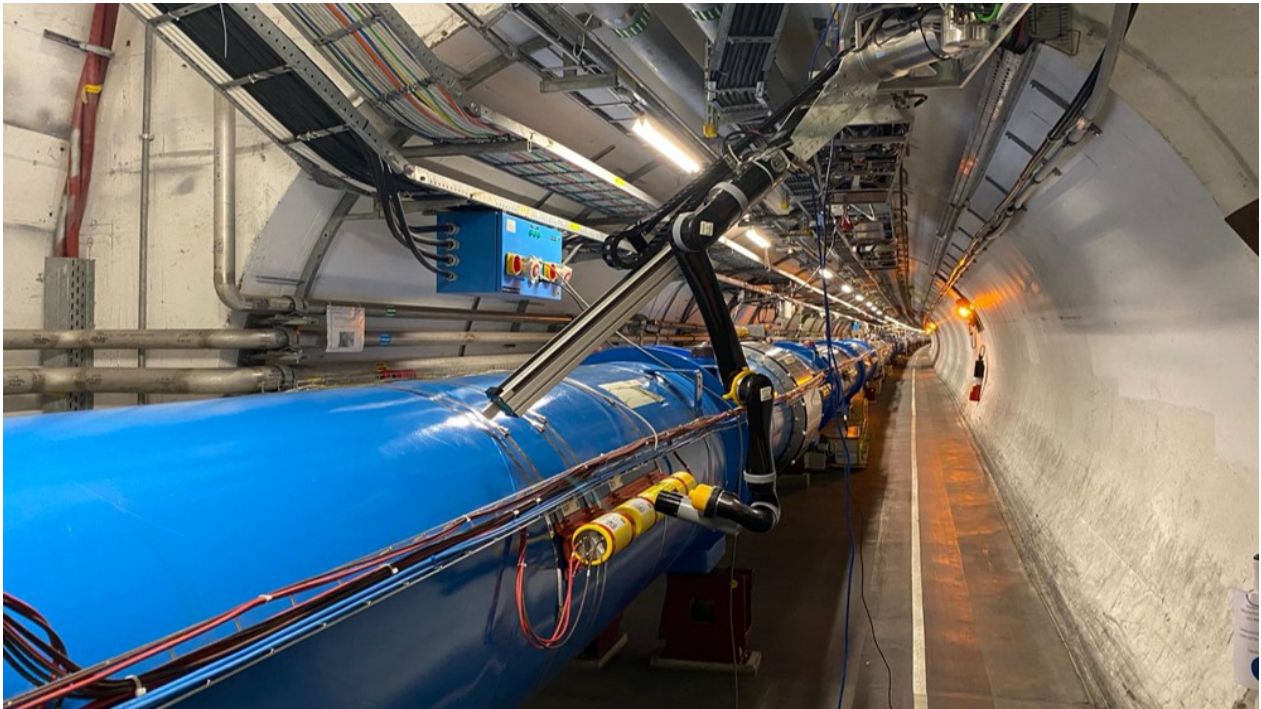


Figure 1.3: TIM robot.

Chapter 2

State of the art

2.1 Background Review

Object search in robotics encompasses various approaches aimed at efficiently locating specific objects within an environment. This problem has been studied since the 70s, leading to a diverse array of methodologies. In this section, we review the state-of-the-art of existing methods, since upon broader examination, the literature on this topic appears highly diverse and fragmented. Therefore, conducting a comprehensive survey of past work is highly valuable.

2.1.1 Definition of Object Searching Problem

Object searching in robotics involves the autonomous identification and localization of target objects within an environment using robotic systems. This task encompasses not only detecting the presence of objects but also exploring and navigating through the environment until the goal is achieved.

It is often mistaken with other similar problems such as object detection, object pose estimation or object tracking. The main characteristic to differentiate them is that object searching is a decision-making problem, whereas the others are perception problems.

Variants like active object detection, which involves controlling the sensor to optimize detection performance, do incorporate decision-making elements and use techniques akin to object search. However, these approaches generally assume the object is initially within the sensor's field of view, unlike in object search [2].

There are other problems which have the same practical goal but a completely different background approach, such as semantic visual navigation. Broadly defined, semantic visual navigation involves an agent navigating an unknown environment towards a specified target, such as a "kitchen" or a "chair," using millions of frames from behavioral datasets for training. Therefore, it involves object searching and identification in the process.

Despite the similarities, object search encompasses a wider range of scenarios, not necessarily involving vision or navigation. Researchers in object search often adopt planning-based

approaches with pre-defined environment models and real robot trials, while those in semantic visual navigation primarily use end-to-end deep learning for training agents in unpredictable settings. Integrating planning-based strategies with deep learning techniques could enhance the effectiveness and versatility of robotic systems in object search tasks. Some researches mix both techniques to improve the amount of space explored by a manipulator robot [3].

2.1.2 Main Challenges

The process of object search presents a multitude of challenges that must be addressed to develop effective robotic systems. These challenges are multifaceted and stem from various factors including sensory limitations, environmental complexities, and the need for adaptability. Understanding these challenges is essential for advancing the field of robotics, since object search is essential for a huge amount of tasks, specially in those applications where collaboration with humans or other robots is needed [2].

- **Sensory Limitations and Perceptual Uncertainty:** Unlike human vision, robotic perception systems often struggle with sensory noise and uncertainty. Robots rely on cameras and sensors that can be affected by environmental factors such as lighting variations, occlusions, and the inherent limitations of their field of view. For instance, a robot's camera might have difficulty distinguishing between similar-looking objects or detecting objects that are partially obscured. Thus, false positives or negatives are almost impossible to eliminate.
- **Partial Observability:** In real-world scenarios, robots frequently encounter partial observability, where they do not have a complete view of the environment or the target object. This limitation is exemplified in situations where a robot must search for an object in a cluttered or dynamic setting. For example, a robot tasked with finding a bottle in a room may only have a limited view through its camera, and must infer the location of the bottle based on partial information and movement. This requires sophisticated reasoning and decision-making processes to navigate and search effectively despite incomplete data.
- **Unstructured and Complex Environments:** Robotic systems often operate in environments that are highly complex and unstructured. Unlike controlled lab settings, real-world environments are filled with numerous objects that are not systematically arranged. This complexity poses significant challenges for object search, as robots must differentiate the target object from a multitude of potential distractions and handle various spatial arrangements and clutter.
- **Human Interaction and Language Processing:** In environments shared with humans, robots can benefit from natural language cues or verbal hints. However, interpreting and acting on these cues introduces additional layers of complexity. The ability of a robot to understand and utilize human-provided information effectively is crucial for enhancing search efficiency. For example, a robot might receive a verbal hint such as "the bottle is on the shelf behind the plants", which requires it to process the language and translate it into actionable steps.

- **Generalization Across Different Contexts:** Achieving effective object search requires the ability to generalize across different environments, objects, and robots. A system designed for a specific scenario might not perform well in a new or different setting. Therefore, creating robust and adaptable object search algorithms that can generalize across various contexts is a significant challenge. This involves developing models that are flexible and capable of learning from diverse experiences to handle new situations effectively.

Addressing these challenges is crucial for advancing robotic capabilities in object search. Ongoing research aims to develop solutions that tackle these issues by improving perception systems, enhancing reasoning algorithms, and integrating effective human-robot interaction methods. The ultimate goal is to create robotic systems that can perform object search with high efficiency and adaptability, paving the way for more autonomous and intelligent robots in practical applications.

2.1.3 Different Methodologies for Object Searching

Object search in robotics is fundamentally a sequential decision-making problem, where the robot must continuously make informed choices to locate a target object efficiently. Given this nature, various methodologies from the broader domain of decision-making problems can be applied. These methodologies can be categorized based on the type of decision-making algorithms they employ to formulate and execute the search strategy [2].

1. Heuristics-based.

They are strategies inspired by biological systems or everyday observations to guide the search process. For example, consider a situation where you need to find a product in an unfamiliar supermarket. Instead of wandering aimlessly or systematically checking every aisle, you might rely on heuristics based on your understanding of typical supermarket layouts—such as looking for yogurt near the cooling shelves. This approach has been tested in structured environments, as shown in the study by Joho et al. [4], where robots learned and applied heuristics to search for objects in office settings. By leveraging structured knowledge about the environment, such as prioritizing areas near desks or cabinets when looking for office supplies, the robots were able to significantly improve search efficiency. The results demonstrated the practical value of heuristic-based methods in reducing search time and optimizing resource usage, particularly in well-organized, man-made environments. This intuitive, experience-based approach is at the heart of heuristics-based methods.

Heuristics-based object search methods can be particularly effective due to their simplicity and ease of implementation. These methods often employ predefined rules or simple algorithms to guide the search strategy, making them less computationally intensive compared to other approaches. For instance, robots might be programmed with specific behaviors such as risk-taking or conservative searching, depending on the context and the type of objects being sought. Some studies [5], introduced a model for collective search using alpha-beta coordination in multi-robot systems. In this approach, robots are divided into two social groups: alpha and beta, each exhibiting distinct risk-taking

behaviors. Alpha robots adopt a more exploratory and risk-taking strategy, focusing on discovering new areas in the environment, while beta robots follow a more conservative approach, remaining in previously explored or confirmed regions.

Although this method is not considered as effective as some more advanced techniques, it has shown notable success in real-world applications due to its simplicity and adaptability.

Moreover, they can be adapted to different environments without requiring extensive modifications. They can utilize the spatial dependencies and structured nature of man-made environments to make informed decisions about where to search next. For example, in an office setting, a heuristic might dictate that searching near desks or cabinets is more likely to yield results when looking for office supplies. These methods capitalize on domain-specific knowledge, which can significantly reduce the search time and increase efficiency.

They provide robust solutions in scenarios where computational resources are limited or real-time decision-making is crucial. Additionally, these methods can serve as a foundation for more complex strategies, providing initial guidance that can be refined through other techniques.

Some of the most popular heuristic strategies which are usually combined are:

(a) Information Gain-based Exploration

This approach is based on the principle that the key to exploring an unknown environment is to maximize the amount of information gained about the state of the environment. Mathematically, this involves measuring the reduction in entropy, which represents the uncertainty about the state of the environment, after performing an action and obtaining an observation.

In practice, Information Gain is calculated by comparing the expected entropy before and after executing an action. The entropy of a probability distribution $H_p(x)$ quantifies the uncertainty or disorder within the distribution. In the context of exploration, the goal is to minimize the expected entropy of the belief state after performing an action [6].

This method has been successfully applied in robotic exploration tasks, such as in [7]. In this approach, a novel algorithm reformulates the traditional information gain function into a differentiable objective, allowing for gradient-based path optimization. Experiments conducted with the Turtlebot3 Burger robot in 2D indoor environments demonstrated the effectiveness of this method. The robot achieved a 173% improvement in area coverage and a 37% reduction in path length compared to baseline approaches. These results highlight the potential of Information Gain-based Exploration to enhance the efficiency and quality of robotic exploration in real-world applications.

(b) Frontier-based Exploration

The frontier-based exploration approach focuses on identifying and exploring the regions that lie on the boundary between known and unknown spaces. In a grid map,

each cell is classified as free, occupied, or unknown, depending on its probability of being occupied. Frontier cells are those free cells that have adjacent unknown cells. These frontier cells are grouped into frontier regions, which represent significant unexplored areas.

The exploration process advances by iteratively selecting a frontier as the target and moving towards it. This selection is usually performed using a utility function that evaluates and prioritizes the frontiers based on certain criteria, such as proximity or potential for discovering new areas. Once a frontier is reached, the map is updated with the new information obtained, and the process repeats until there are no more unexplored frontiers [8].

Usually this strategy is mixed with other methodologies such as entropy-based exploration [8]. They use a combination of frontier-based and entropy-based methods to enhance the process of mapping unknown environments. The frontier-based approach is initially utilized to narrow down the possible locations of targets by minimizing the explored region and avoiding inefficient zig-zag movements. The primary focus, however, is on reducing the uncertainty in classifying observable cells as either free or occupied, thereby maximizing the information gain through entropy reduction.

While frontier-based exploration is more closely related to general exploration than specifically to object searching, the two are inherently connected. When there is no prior information about the object's position, the goal is to explore the map as efficiently as possible to maximize the chances of locating the target. For example, the Wavefront Frontier Detector (WFD) algorithm [8] has been successfully implemented both in simulated environments and on physical robots such as the Kobuki TurtleBot, demonstrating its practical applicability in real-world scenarios by enabling systematic coverage of the space to facilitate object detection. Similarly, Holz et al. [9] combined frontier-based and entropy-based strategies to optimize exploration and map accuracy. In the context of object search, frontier-based exploration allows robots to systematically reduce the unexplored areas of an environment, thereby increasing the likelihood of detecting target objects. By continuously moving towards frontiers, the robot efficiently covers the environment, ensuring that all potential locations of the object are examined. This approach is particularly useful in structured environments where the spatial layout can be leveraged to predict the most promising areas to search [10].

These studies highlight how frontier-based exploration, though primarily designed for navigation and mapping, provides a foundation for object search when positional information is unavailable, ensuring systematic coverage of the environment.

(c) **Sampling-based methods**

Sampling-based methods are widely used in robotic exploration due to their ability to handle complex and high-dimensional spaces. These methods rely on single or multi-query planners to systematically search through an environment by expanding

a tree-like data structure. The exploration process iteratively progresses by moving towards the Next-Best-View (NBV) configuration that maximizes utility, thereby efficiently mapping the unknown spaces.

One prominent approach within sampling-based methods is the Receding Horizon Next-Best-View Planner (RH-NBVP) introduced by Bircher et al. [11]. This method involves generating viewpoints by expanding Rapidly-exploring Random Trees (RRTs) and executing the first node of the most promising branch in a receding horizon manner. This approach helps in mapping the environment incrementally, thus avoiding premature convergence to local minima.

The effectiveness of sampling-based exploration methods often hinges on the utility functions used to evaluate potential viewpoints. Popular utility functions include both exponential and linear formulations, which guide the selection of the most informative viewpoints. However, Schmid et al. [12] identified a limitation in these utility functions, noting that strictly increasing formulations may lead to the selection of sub-optimal nodes within longer sub-trees. They proposed using efficiency as a gain formulation to achieve more accurate Truncated Signed Distance Function (TSDF) reconstructions, expanding a single RRT* tree for better mapping accuracy.

In conclusion, while sampling-based methods offer robust solutions for autonomous exploration, particularly in terms of viewpoint generation and local planning, their effectiveness can be limited by factors such as localization uncertainty and environmental complexity. However, these exploration techniques are particularly valuable in the context of object searching, as efficient exploration of the map is a critical prerequisite, specially when there is no prior information about the object's location.

(d) **Predefined Regions of Interest**

Establishing Predefined Regions of Interest (ROIs) aims to optimize data collection in specific areas deemed important due to certain criteria. These methods seek to generate optimized trajectories for exploring these areas efficiently, rather than conducting a general search across the entire environment.

For instance, in the study by M. Naazare et al. [13], a planning approach is introduced for exploration and inspection using a mobile manipulator robot equipped with a depth camera. The goal is to create a volumetric map of a ROI within an unknown environment. The significance of an area is determined based on dummy sensor readings that provide scalar values, such as contamination levels. These measurements are integrated into a voxel map, which the algorithm uses to derive utility values for camera positions and orientations.

Overall, the ROI-based planning methods are particularly advantageous for scenarios requiring detailed and targeted exploration, making them well-suited for applications where a priori information is available and which demand extensive data collection in specific areas of interest.

2. Mathematical analysis.

One of the earliest influential models was proposed by Pollock [14], where the search problem involves a target moving between two cells according to a Markovian process. The primary challenge in Pollock's model is to allocate the search effort effectively across these cells to maximize the likelihood of finding the target. This problem is solved using dynamic programming, a method well-suited for optimizing decisions over time. By dividing the search effort strategically, Pollock's model helps to establish an efficient search pattern that can be adapted to various scenarios where the target exhibits Markovian movement. In this context, Markovian movement means that the target's future position depends only on its current position and not on its past positions.

Building on Pollock's work, Stone [15] generalized the moving object search problem, providing to the searcher with a probability distribution of the target's initial location and a detection function that determines the likelihood of detecting the target based on its location and the amount of effort expended.

Mangel [16] expanded the scope of the search problem by considering targets that move according to a diffusion process. His model tackles the "descriptive problem," focusing on the probability density function that describes the likelihood of an unsuccessful search up to a specific time, given the target's position and the search trajectory.

In their 2009 study, Kollar and Roy [17] developed a probabilistic model that leverages object-object and object-scene contextual relationships to enhance robotic object search strategies. By analyzing co-occurrence data from labeled images, the model predicts likely locations of target objects based on the presence of related items within a given environment. This approach enables robots to prioritize search areas more effectively, reducing the time and effort required to locate specific objects. Experimental results demonstrated that robots utilizing this model could successfully identify target objects in an office setting by focusing on areas with higher predicted likelihoods, thus, improving search efficiency compared to unguided exploration methods. Although this method combines probabilistic reasoning and semantic modeling, its foundation lies in mathematical analysis, as it uses a probabilistic framework to model relationships and optimize decision-making. Similar to Pollock and Stone's approaches, it employs probability distributions to predict object locations and guide search efforts. While less abstract than Pollock's or Stone's models, this work exemplifies the integration of mathematical theory with practical object search tasks.

Although these analytical methods were predominant in the early stages of object search research, they have seldom been directly applied to real robotic systems. The outcomes from this research are often general and abstract, offering significant theoretical insights but missing the practical implementation details required for robotic applications. Nonetheless, they establish a vital groundwork for the development of more sophisticated and practical search strategies that merge mathematical precision with real-world applicability.

3. Greedy, next-best view (NBV)

Greedy techniques in robotic object search prioritize the immediate maximization of information gain while minimizing costs. This approach is characterized by its simplicity and practicality, often making it a preferred method for real-world applications despite some inherent limitations.

Most exploration methods utilize this greedy approach, as it is optimal when considering the immediate next step (horizon 1). This is because the exploration problem involves an enormous branching factor, rendering multi-step planning infeasible. The primary objective of exploration is to gather new information, which requires continuous policy adjustments with each new belief state, making the measurements inherently unpredictable.

Ye and Tsotsos [18] formulated the 3D object search problem as a sequence of sensing actions constrained by a cost budget. In practice, a notable example is the vision-based system developed by Andreopoulos et al. [19] for the Honda ASIMO robot. This system utilizes a probability map over a $4 \times 4 \times 4$ grid and employs a one-step look-ahead greedy algorithm. The robot hypothesizes various candidate sensor states, comprising different body and gaze poses, and selects the state that maximizes the probability of localizing the target object.

A notable work by Dogar et al. [20] introduced a formulation of object searching by manipulation, where visibility and accessibility relations between objects were explicitly considered. They proposed a greedy algorithm to optimize search actions, demonstrating its effectiveness under certain conditions. Their research highlighted how robots could manipulate their environment to uncover occluded objects, a critical capability in cluttered or unstructured scenarios. This study laid the groundwork for practical implementations of greedy methods in object search. Building upon this foundation, the project "Object Search Manipulation Robot" implemented a greedy search algorithm to enable a robot manipulator to locate and retrieve objects efficiently.

Other works such as Wang et al. [3] studies various volumetric information metrics for selecting the NBV, often utilizing ray tracing techniques. One common method involves counting the number of unknown voxels. This can be done by either counting all the unknown voxels encountered or by focusing only on the frontier voxels, which are on the boundary between known free space and unexplored space [2]. In addition to counting-based metrics, probabilistic occupancy estimation metrics consider measurement uncertainty. Probabilistic information metrics are typically derived from entropy-based formulations. As each sensing ray travels through the environment, the total information gain is computed by aggregating the contribution of all voxels it intersects. This aggregation can be done by summing all voxel contributions—emphasizing rays that reach deeper into the scene—or by averaging them, which gives equal importance to all visible voxels regardless of depth. Nonetheless, estimating the likelihood of future measurements along these rays can lead to inaccuracies if occlusions are not properly accounted for.

To address this, some methods utilize a Hidden Markov Model to estimate the likelihood of an unknown voxel being visible from any viewpoint [2]. Other approaches weight the

information gain by the product of the emptiness probability of all preceding voxels to account for occlusion.

However, the simplicity of greedy methods comes at a cost. These algorithms do not consider the sequential nature of search actions, which can lead to suboptimal search paths when actions carry different costs or when searching for multiple objects. Sequential planning, which considers the order of actions, can often provide more efficient search strategies by accounting for these factors.

4. Graph search.

Graph search algorithms are not commonly employed for object search, however, some of them have found application primarily in scenarios where the target object is occluded by clutter. A* is a well-known graph search algorithm, it was adapted for this purpose [20]. In this context, the object search problem is redefined as a process of revealing occluded spaces by sequentially removing visible obstacles until the target object becomes visible. Each removal action carries a different cost, adding a layer of complexity to the search process. In addition, it does not consider perception uncertainty [2].

5. POMDP planning.

Partially Observable Markov Decision Process (POMDP) exploration focuses on how an agent can efficiently and effectively explore its environment when it has limited and imperfect perception of the world’s state.

A POMDP is defined by a set of states S , actions A , observations O , a state transition model T , an observation model Z , and a reward function R . The agent does not have direct access to the true state of the environment. Instead, it receives observations that provide partial information about the state. The goal is to find a policy that maximizes the expected cumulative reward over time, considering the uncertainties in both state transitions and observations [2].

Recent advances in online POMDP planning algorithms have further boosted interest in applying these methods to object search tasks. For instance, the DESPOT algorithm [21] has been utilized by J. K. Li et al. [22] to enhance the decision-making capabilities of autonomous systems in dynamic and uncertain settings.

Despite these advancements, POMDP planning has faced challenges in multi-robot search tasks due to its computational intensity. Hollinger et al. [23] found POMDP planning computationally prohibitive for multi-robot scenarios. However, as both the fields of robotics and POMDP planning advance, there is renewed interest in these approaches [24]. The key appeal of POMDP-based approaches lies in their generalization capability, versatility, and the ability to reason over sequences of actions, despite the computational challenges they pose. Consequently, previous work on object search has often sacrificed realism for computational efficiency, for example, by reducing the problem from three-dimensional to two-dimensional spaces.

Zheng et al. [2] present significant advances in mitigating traditional computational challenges associated with POMDP. Key innovations include the development of a

scalable and efficient multi-resolution planning algorithm using octrees for 3D multi-object search. Due to the problem with the big branching factor of this problem, previous research often propose limiting the search horizon. Thereby, optimizing the information-cost trade-off in finite horizons. In scenarios where both measurement and state spaces are confined, it is possible to generalize the principle of information gathering to any finite horizon [6]. In addition, they introduced COS-POMDP which expands the observation space of traditional object search POMDPs by incorporating observations about objects other than the primary target. For example, if certain objects are often found near each other, observations of one can inform the search strategy for finding another. This spatial correlation model allows the robot to make more informed decisions, reducing search times and improving overall performance.

6. Reinforcement learning.

It has emerged as a powerful approach for object searching and robotic exploration, leveraging the ability of agents to learn optimal policies through interactions with their environment. In the context of exploration, RL focuses on training agents to maximize their knowledge acquisition and efficiency in navigating unknown spaces.

An agent interacts with an environment through a series of actions, learning to achieve a set of goals by receiving feedback in the form of rewards. This feedback loop enables the agent to improve its decision-making policy over time. The key components include the state, representing the current situation of the agent within the environment; the action, which is the set of possible moves the agent can make; the reward, the feedback from the environment that evaluates the action taken; the policy, the strategy that the agent employs to determine actions based on states; and the value function, a prediction of future rewards guiding the agent towards long-term success.

A significant challenge in RL is the high sample complexity, meaning the large number of interactions required for the agent to learn effective policies. Additionally, ensuring that the learned policies generalize well from simulation to real-world environments remains a hurdle. Recent advancements have focused on improving sample efficiency and enhancing generalization capabilities. For instance, Shah et al. [25] achieved notable progress in outdoor visual navigation by integrating pre-trained models.

An example of RL applied to object searching is the study "Active Object Searching on Mobile Robots Using Reinforcement Learning" [26]. This work presents a method where an agent learns actions based on detected objects within its viewpoint. The agent is capable of searching for objects across different rooms and indoor environments, utilizing only prior knowledge of potential objects in a room. This approach demonstrates the potential of RL to enable efficient object searching by combining learned policies with environmental context, highlighting its applicability in structured and unstructured scenarios alike.

Method	Description	Advantages	Disadvantages
Heuristics-based	Strategies inspired by biological systems or everyday experiences, using predefined rules to guide the search process.	Simplicity, ease of implementation, and lower computational load. Adaptable to different environments and effective in real-world applications.	Less effective than more advanced approaches, may not be optimal in complex situations.
Information Gain-based	Aims to maximize the information gained from each action, by measuring the reduction in entropy.	Efficient in reducing uncertainty, ideal for environments where information acquisition is key.	Requires complex entropy calculations, may be resource-intensive in large or dynamic environments.
Frontier-based	Focuses on exploring the boundary regions between known and unknown spaces, advancing towards frontiers to expand the explored area.	Optimizes area coverage, ideal for quick and efficient mapping of unknown spaces.	May lead to inefficient movements if not combined with other strategies like entropy reduction.
Sampling-based	Uses sampling methods to explore the environment by expanding search trees like RRT, maximizing the utility of each next-best view.	Robust in viewpoint generation and local planning, effective in complex environments.	Effectiveness depends on the utility function, and it may suffer from premature convergence or suboptimal performance in complex environments.
Predefined Regions of Interest	Focuses exploration on specific predefined areas deemed important, generating optimized trajectories to maximize data collection in these areas.	Highly efficient for data collection in critical areas, useful in applications where prior information about areas of interest is available.	Less suitable for general exploration or environments where important areas are not known a priori.
Mathematical analysis	Mathematical models like Pollock's, which optimize search effort through dynamic programming based on the target's movement following a Markovian process.	Provides a solid theoretical foundation for designing efficient search strategies.	Generally abstract and difficult to apply directly to real robotic systems.
Greedy NBV)	Prioritizes immediate maximization of information gain with a short-term focus.	Simple and practical for real-world applications, optimal for short-term horizons. Commonly used approach.	Does not consider the sequential nature of actions, which can lead to suboptimal search paths when planning for long-term.
Graph search	Adaptation of graph search algorithm for object search, revealing occluded spaces by sequentially removing visible obstacles until the target object becomes visible.	Effective for search problems where the object is occluded, minimizes the costs of removal actions.	Not common in this field. Does not consider perception uncertainty, and may be computationally intensive depending on the number of obstacles and associated removal costs.

POMDP	Uses a Partially Observable Markov Decision Process approach to explore environments with limited and imperfect perception, maximizing the expected cumulative reward.	High generalization capability and versatility, allows planning of action sequences considering uncertainty. The use of octrees reduces the impact of the high branching factor. COS-POMDP improves efficiency.	Computationally intensive, especially in multi-robot search tasks, often sacrifices realism for computational efficiency.
Reinforcement learning	Focuses on learning optimal policies through interaction with the environment, maximizing knowledge acquisition and efficiency in navigating unknown spaces.	Autonomous learning capability, continuous improvement through feedback, and potential to generalize in real-world applications.	High sample complexity, requires a large number of interactions to learn effective policies, difficulty in generalizing from simulation to real-world environments.

Table 2.1: Characteristics of Different Object Search Methods

2.1.4 Limitations of Previous Work in Object Search

The task of object search in robotics can be divided into two primary subproblems: selecting which regions to search and determining how to search within those regions [2]. We focus exclusively on the second, assuming that ROIs are predefined manually. Despite significant advancements, previous research has faced several limitations that hinder the effectiveness and generalization of object search systems.

Numerous works have shown that object search tasks benefit significantly from the integration of prior knowledge or spatial priors about the environment. For example, early works such as those by Wixson and Ballard [27, 17], focused on using semantic knowledge to infer probable locations of objects. For instance, associating specific objects with typical locations (e.g., forks in kitchens) helps in selecting search regions. However, these approaches often relied on static and predefined knowledge, which may not generalize well to dynamic or unfamiliar environments. Additionally, these methods sometimes struggled with scalability, especially in complex environments where numerous potential search regions exist.

Once a search region is identified, a new challenge emerges: determining how to efficiently select a sequence viewpoints within that region to detect the object [27]. This problem is particularly difficult due to the need for efficient view planning and handling partial observability. Research has shown that planning views in a 3D space under a time constraint is NP-Complete [18]. Consequently, many previous works resorted to ad-hoc and greedy strategies that often lack robustness and efficiency [17] [28] [29]. Some approaches simplified the problem by reducing it to 2D searches or predefining sets of viewpoints [30], which limits their applicability in real-world, 3D environments.

Recent solutions are often categorized into frontier-based methods, sampling-based methods, or hybrid strategies combining both [13]. In frontier-based exploration, the robot navigates towards frontiers, the boundaries between known and unknown space, guided by utility functions that estimate exploration gain [8]. In contrast, sampling-based methods generate viewpoint configurations by growing tree-like structures such as Rapidly-Exploring Random

Trees (RRTs) [11]. Exploration progresses by selecting the NBV configuration with the highest expected utility, estimating the volume of unmapped space that could be mapped from a candidate pose.

A representative sampling-based method for active view planning is the Weighted-Gain Next-Best-View Planner (WG-NBVP) [13]. This approach builds on a classical RRT based next-best-view planner, augmenting it with a weighted information utility designed for focused inspections. In practice, WG-NBVP incorporates prior knowledge of ROI to guide the sampling of candidate viewpoints. The utility function used for selecting a new viewpoint is a weighted sum that balances exploring unknown space versus closely inspecting high-interest regions. This means the planner can favor viewpoints that cover critical ROI areas without completely neglecting unexplored regions.

Unlike more exhaustive planners, WG-NBVP does not perform lookahead search beyond the immediate next view. It grows an RRT of candidate view poses from the robot’s current position at each iteration, evaluates the information gain of each sampled node, and greedily selects the best node based on its current utility value.

Candidate positions in the RRT are assessed by casting rays to estimate the expected visible volume and ROI content from that point. Notably, the orientation of the camera at each candidate position is not freely optimized, instead, it is set using either ROI gradient or maximum expected volume gain.

The chosen viewpoint is then executed directly, without explicitly planning a long sequence of future viewpoints. Any intermediate RRT nodes on the path to that best view are visited only as a means to reach the goal, not for their own informational value. This one-step greedy strategy keeps planning tractable, but it foregoes reasoning about the sequential benefit of taking a series of views. In terms of computational complexity, building the RRT of n nodes can be on the order of $O(n \log n)$. However, the number of RRT nodes is limited to a predefined threshold in order to manage computational complexity. This cap (e.g. a few hundred nodes) bounds the planning time but must be tuned according to the map size, larger environments typically require more samples, otherwise the RRT may not adequately explore the space, affecting scalability.

Furthermore, WG-NBVP mitigates local minima by caching previously expanded RRT nodes and re-evaluating their information gain in subsequent iterations. Although this allows partial recovery from local traps, no formal guarantee of completeness or global optimality is provided.

To address the limitations of purely greedy strategies like WG-NBVP, more principled frameworks model the search problem as a Partially Observable Markov Decision Process (POMDP). A POMDP formulation explicitly handles uncertainty in the robot’s state and observations, and it enables reasoning over sequential actions to maximize long-term information gain.

In an object search context, the state includes the possible locations of the target object(s), and the robot maintains a belief (probability distribution) over those states. At each step the robot can take sensing actions (e.g. move the camera or navigate to a new viewpoint) and update its belief based on what it observes or fails to observe. This framework, in theory, yields an optimal policy for where to look next, considering future observations and rewards,

rather than just a myopic one-step view [2]. Early POMDP-based works often restrict the problem to 2D spaces or assume simplified observation models [31], such as fixed surface level placement or full FoV coverage [30], limiting their applicability in realistic 3D scenarios. Even a more advanced formulation by Wandzel et al. [28], which introduced an Object-Oriented POMDP (OO-POMDP) for multi-object search, was demonstrated in relatively simple settings (effectively constraining the search space or robot mobility to 2D).

A more recent and sophisticated solution is GenMOS (Generalized Multi-Object Search) [2], which extends OO-POMDP framework to handle realistic 3D scenarios involving multiple target objects. GenMOS leverages online planning using the Partially Observable Upper Confidence Bounds for Trees (POUCT) algorithm, a Monte Carlo Tree Search variant designed for POMDPs. This enables the robot to plan sequences of actions while accounting for both occlusions and the distribution of multiple target objects within the environment. To estimate the expected outcomes of candidate viewpoints, GenMOS employs ray tracing over an occupancy-based octree, generating occlusion-aware volumetric observations used for belief updates. The initial belief distribution is either manually defined or derived from real-world sensor input, such as a 3D point cloud. This strategy helps avoid allocating belief to physically implausible locations—such as free space with no supporting surfaces. However, since the belief is defined over a fixed cubic grid, and actual environments are often irregular in shape, part of the probability mass may extend into inaccessible or non-relevant areas. This misalignment can degrade the effectiveness of the planner [2].

Prior to GenMOS, Zheng et al. [32] introduced a planning framework known as MR-POUCT (Multi-Resolution POUCT), which incorporated hierarchical abstraction into the POMDP solver. This method exploited the inherent structure of octree-based belief representations to perform planning across multiple resolution levels simultaneously. By doing so, MR-POUCT improved scalability and accelerated convergence, particularly in large-scale 3D environments where uniform resolution planning can be inefficient. Despite these advantages, the added implementation complexity of multi-resolution planning meant that MR-POUCT was not included in the GenMOS framework. Instead, GenMOS relies on a single-resolution version of POUCT, prioritizing practical deployability and lower system complexity [33].

While the use of POUCT in GenMOS enables online, sequential planning under partial observability, it also introduces notable computational demands. As an anytime algorithm, POUCT incrementally builds a belief tree and returns a feasible—though not necessarily optimal—policy within a user-defined simulation budget. It focuses its search on the most promising branches, which allows for flexible trade-offs between performance and computation time. However, in the worst-case scenario, the size of the belief tree still scales exponentially as $O((|\mathcal{A}| \cdot |\mathcal{O}|)^d)$, where $\mathcal{A}$ is set of actions, $\mathcal{O}$ the set of observations, and d the planning depth.

In practical terms, this means that as the environment becomes larger or more complex, particularly when uncertainty is high, maintaining high-quality planning performance demands significantly more simulations and deeper trees. This exponential growth in computational requirements presents a major challenge for real-time operation or for deployment on robots with limited processing capabilities. The impact of this limitation is evident in the evaluation results of GenMOS: even in their most efficient, non-ground-truth-based configuration, the

average planning time per Next-Best-View decision was reported to be approximately 7.4 seconds. Such delays may be acceptable in controlled experiments but pose clear limitations for time-sensitive or reactive tasks in dynamic environments.

Finally, deep learning has led to growing interest in learning-based approaches for robotic object search. These methods typically adopt an end-to-end framework, where action policies are learned directly from raw visual input through large-scale training [34, 35]. Most of these systems are trained entirely in simulation environments. A critical challenge in this area remains the generalization of learned models: performance often degrades when deployed in new environments, with different object categories, lighting conditions, or robotic platforms.

To address this, ongoing research has focused on bridging the gap between simulation and reality through techniques such as domain adaptation and sim-to-real transfer [36, 37, 38]. While progress has been made, achieving robust generalization across diverse scenarios still requires extensive data collection, careful environment modeling, and often considerable computational resources, making deployment in practical settings nontrivial.

In response to the limitations identified in recent planning frameworks and learning-based approaches, we propose HOPE-OS (Heuristic Occlusion-aware Planning for Efficient Object Search). Unlike deep learning methods that depend on extensive training data and struggle with generalization, HOPE-OS makes no assumptions about prior knowledge or domain-specific learning, enabling straightforward deployment across diverse robotic platforms. By focusing on the how-to-search problem under minimal prior assumptions, HOPE-OS offers a scalable and adaptable alternative for efficient object search in complex, real-world scenarios.

In contrast to previous methods, our proposed HOPE-OS algorithm achieves significantly lower time complexity, which enables greater scalability and real-time applicability. Unlike WG-NBVP and GenMOS, it allows for the evaluation and selection of multiple camera orientations per candidate position, increasing the likelihood of informative observations without requiring costly re-planning. The use of a gradient map to represent regions of interest provides a smooth and guided exploration strategy, progressively steering the search toward relevant areas.

Compared to deep learning-based approaches, HOPE-OS requires no pretraining, annotated datasets, or simulation-to-reality adaptation. It operates entirely based on online sensor input and general heuristics, which makes it more adaptable across different platforms and environments. This combination of efficiency, simplicity, and robustness positions HOPE-OS as a practical solution for object search in unknown and cluttered 3D spaces.

2.2 Heuristic-Based Search and Contributions

The choice of adopting a heuristic-based approach for object searching in this thesis stems from its simplicity, adaptability, and efficiency in real-time decision-making scenarios. Heuristic methods are particularly well-suited for environments where computational resources are limited, as they leverage domain-specific knowledge to guide the search process. This makes them an excellent choice for addressing object searching in unknown and dynamic environments.

In this work, a novel heuristic-based algorithm (HOPE-OS) is proposed to tackle the challenges of object searching in completely unknown environments. Unlike many existing approaches that rely on prior maps or detailed knowledge of the environment, the developed algorithm operates with minimal prior information. It assumes the existence of predefined Regions of Interest (ROIs), which are known to the agent or robot. This allows the search process to focus on the "how to search" aspect of the problem, as the ROIs already define "where to search." By narrowing the scope to how the search is conducted, the algorithm efficiently and precisely navigates through the search space.

A key motivation behind the design of HOPE-OS is to develop a planning framework that is not only adaptable and environment-agnostic, but also computationally scalable. In contrast to tree-based methods with higher time complexity, such as RRT-based planners or POMDP solvers, the algorithm is designed to operate with a complexity that grows linearly. This is achieved through a combination of heuristic scoring, focused viewpoint sampling, and iterative clustering, all of which contribute to a lightweight, real-time decision-making process.

The concept of Regions of Interest (ROIs) used in this algorithm is inspired by the work of Naazare et al. [13], where targeted exploration focuses on predefined regions to enhance the efficiency of mapping and inspection tasks. Similarly, in this algorithm, ROIs provide the foundational guidance for search, ensuring that the robot concentrates its efforts on the most promising areas of the environment.

The algorithm is designed to work with a sensor capable of acquiring depth information or point cloud data, such as an RGB-D camera, a LiDAR, or a stereo camera. These sensors are essential for performing raytracing operations, which are used to determine the occupancy status of the environment. The depth data allows the algorithm to dynamically build and maintain a voxel-based map of the environment as the robot explores. The map classifies voxels into four categories

1. **Unexplored voxels:** voxels which have not been explored.
2. **ROI voxels:** indicating regions likely to contain the target object.
3. **Free voxels:** areas determined to be unoccupied through raytracing.
4. **Occupied voxels:** regions identified as obstacles.

Using this map, the algorithm evaluates potential Next-Best-View (NBV) positions by combining relevant information into a unified cost function designed to maximize information gain and information quality (ROI) while minimizing displacement cost. The process is divided into two sequential steps: first, determining the best next position, and second, selecting the optimal orientations for that position. This separation ensures that the algorithm systematically optimizes both spatial positioning and viewpoint selection, enabling efficient and precise exploration in unknown environments. For each candidate position, the algorithm simulates potential spherical field of view (FoV), assessing the visibility and accessibility of the surrounding free voxels. These simulations take into account occlusions caused by detected obstacles and prioritize positions offering the greatest overall benefit.

Once the algorithm identifies which voxels could potentially be visible from a given candidate

position, it evaluates the importance of those voxels. This evaluation considers whether the voxels are near an ROI, belong to an ROI, or are far from an ROI. The process assesses three main factors:

1. **Distance** to the candidate position, minimizing the effort required to move to the NBV.
2. **Quality** of information, giving higher priority to views that include ROI voxels.
3. **Quantity** of information, favoring positions that reveal the highest number of unexplored voxels.

After determining the optimal position, the algorithm proceeds to identify the best orientation at that location. This is accomplished by sampling multiple potential orientations and simulating the camera's Field of View (FoV) for each. For every sampled orientation, the algorithm evaluates the quantity and quality of visible voxels, considering factors such as occlusions and the relevance of the observed voxels to the search objectives. By analyzing these metrics, the algorithm selects the orientation that maximizes expected information gain while ensuring the highest quality of acquired data. The robot then moves to this position, updates the map based on new observations, and repeats the process until the target object is located. This iterative approach ensures that each step of the search process is both precise and efficient. The proposed algorithm introduces several significant contributions. First, it incorporates dynamic FoV simulation, which optimizes decision-making in real time by considering the impact of obstacles on visibility. Second, the multi-factor cost function ensures a balanced strategy that avoids common pitfalls such as local minima. Third, the requirement for sensors capable of depth perception or point cloud generation ensures precise raytracing and accurate voxel classification, even in environments with complex obstacles. Lastly, the algorithm could be redesigned to be scalable, by using octree map representation instead of voxel maps, functioning effectively in both small and large environments with minimal computational requirements.

By focusing on the "how to search" problem, this heuristic-driven approach demonstrates its effectiveness in structured but unknown environments. The predefined ROIs act as an anchor for the search, allowing the algorithm to maximize its efficiency and precision. Additional details on the implementation and experimental validation of this approach are provided in Chapter 3 and 4, where its performance is evaluated in simulated and real-world scenarios.

Chapter 3

Materials and methods

3.1 Software

The development of this project involved various software tools tailored to address different phases of design, simulation, and implementation.

- **Simulation** For the initial phase of development and understanding the problem, MATLAB was chosen. This decision was motivated by MATLAB's powerful capabilities for real-time visualization of variables and results, facilitating a deeper understanding of the complex algorithms involved. Its interactive interface allowed for rapid exploration of different approaches and validation of fundamental concepts before moving to more advanced implementations.



- **Implementation in the CERN Robotic Framework (CRF):** Once the theoretical foundations were established and initial models validated, the system implementation was carried out in C++, leveraging the CERN Robotic Framework (CRF). Developed by CERN's BE-CEM-MRO team, the CRF is a modular architecture designed for controlling and interacting with robots in complex and semi-structured environments. This framework provides a robust infrastructure for integrating various robotic modules, enabling efficient communication and control of the implemented systems.



This in-house framework provided a modular structure for integrating the robotic arm, sensors, and algorithms. It supported efficient communication and execution of tasks, making it indispensable for the real-world deployment of the system. However, The implementation of the system in C++ was required several key external libraries, each chosen for its efficiency and reliability in specific tasks:

- Eigen: This versatile library was fundamental for linear algebra operations, particularly for transformations between frames and matrix manipulations. Eigen was also used to handle multi-dimensional matrices representing the voxel map, which stored spatial information such as Regions of Interest (ROIs), free space, and obstacles.
- Point Cloud Library (PCL): Essential for processing 3D data, PCL enabled tasks such as filtering, segmentation, and visualization of the point clouds generated by the RGB-D camera.
- OctoMap: It provided a highly efficient way to store and process spatial data using octrees, a hierarchical data structure that divides 3D space into cubic regions. Octrees offer significant advantages in terms of memory usage and computational speed, as they represent sparse environments compactly while maintaining sufficient detail where needed. OctoMap was particularly crucial for implementing raytracing operations, allowing the algorithm to determine visible voxels, simulate fields of view, and account for occlusions caused by obstacles. Its dynamic nature enabled the real-time updating of the map as the robot explored its environment.
- **Version Control and Collaboration:** For version control and team collaboration, Git Lab was used. This ensured efficient management of the code-base, enabling seamless tracking of changes and continuous integration of new features.



3.2 Hardware

In this project, the hardware components played a crucial role in the real-world implementation of the system.

- **Simulation Hardware:** For simulation purposes, we utilized a conventional desktop computer equipped with a licensed version of MATLAB. This simple setup allowed us to model and analyze various scenarios before deploying the system in a physical environment.
- **Real-World Implementation Hardware:** The real-world deployment involved several key hardware components:
 - Intel® RealSense™ Depth Camera D435i: This advanced depth camera combines stereo depth sensing with an integrated inertial measurement unit (IMU). The D435i offers a wide field of view ($87^\circ \times 58^\circ$) and a depth resolution of up to 1280×720 pixels, making it suitable for both indoor and outdoor applications. The addition of the IMU enhances the camera’s ability to maintain depth awareness during movement, which is particularly beneficial for applications requiring simultaneous localization and mapping (SLAM).



Figure 3.1: Intel RealSense Depth Camera D435i

- Kinova Gen3 Robotic Arm: The D435i camera was mounted on the end-effector of the Kinova Gen3, a versatile and modular robotic arm designed for research and development. The Gen3 features 6 degrees of freedom (DoF), allowing for precise and flexible manipulation. It is equipped with smart actuators that provide torque sensing, enabling better interaction with its environment. The arm supports various programming environments, including C++ and Python, and is compatible with simulation tools like Gazebo and MoveIt, facilitating seamless integration into our system.



Figure 3.2: Kinova Gen3

- TIM (Train Inspection Monorail) Robot: The Kinova Gen3 robotic arm was installed on Train Inspection Monorail robot (TIM), CERN’s semi-autonomous vehicle designed for real-time monitoring and inspection of the Large Hadron Collider (LHC) tunnel. TIM operates along a 27-kilometer monorail track suspended from the tunnel’s ceiling and is equipped with various instruments to survey the tunnel structure, monitor environmental conditions, and ensure safety. The integration of our system into TIM allowed for autonomous exploration and data collection within the LHC tunnel environment.



Figure 3.3: TIM Robot

3.3 2D Simulation

3.3.1 2D Simulated Environment

The 2D simulation environment was developed in MATLAB, starting with a simple grid-based map model to better understand the problem and the underlying algorithms. This approach allowed for a gradual comprehension of the system's functionality and provided a manageable framework to test and refine the core concepts. The grid map represents the spatial layout of the search area, with each cell corresponding to a specific region, and its state dynamically updated as the agent explores the environment. To facilitate visualization and understanding of the exploration process, cells were assigned specific colors based on their status, as shown in Figure 3.4:

- White: Unexplored free space.
- Black: Unexplored occupied space.
- Green: Free space.
- Red: Occupied space.
- Purple: Frontier free space (cells on the boundary between known and unknown space).
- Yellow: Region of Interest, representing high-priority areas for the agent to search.
- Blue: Object (Goal to find).

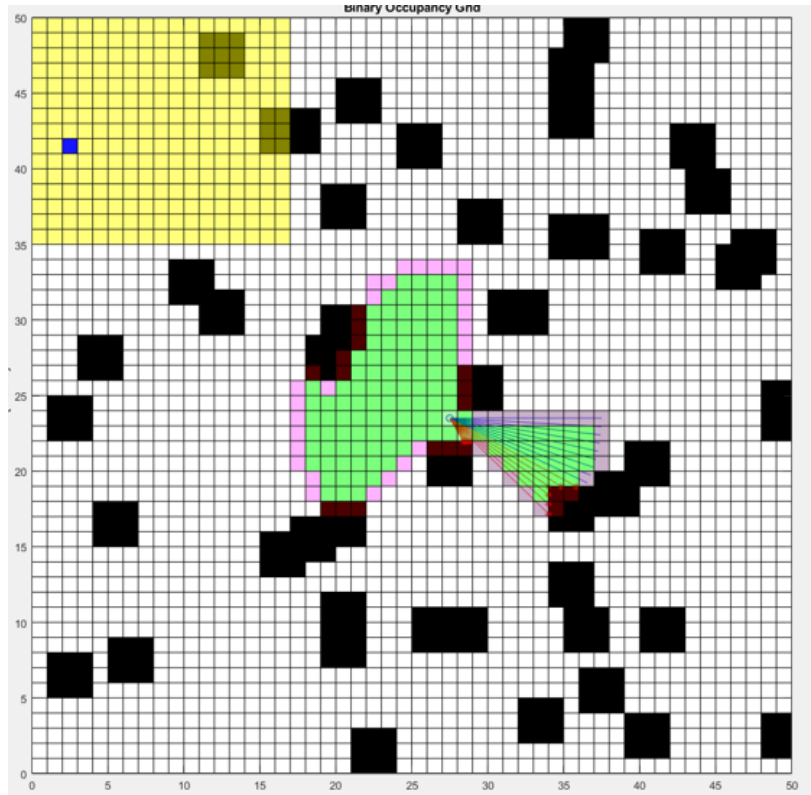


Figure 3.4: 2D 50x50 Occupancy Gridmap

The simulated agent is equipped with a sensor modeled to have a field of view, allowing it to perceive its surroundings within a specified range and angle. The FoV is simulated using raytracing, which ensures that the sensor cannot see through obstacles. Rays are cast from the agent's position outward within its FoV, marking cells as visible and free until an obstacle is encountered. This approach accurately mimics the physical behavior of sensors, where occlusions limit visibility as shown in Figure 3.5.

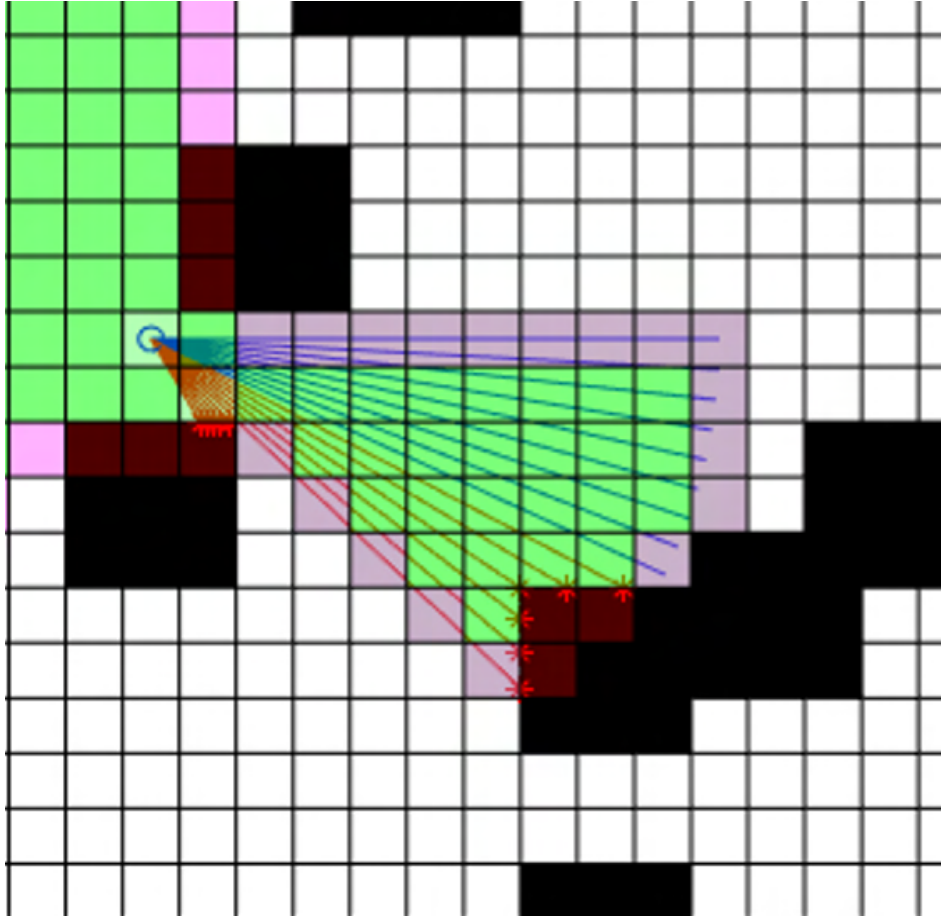


Figure 3.5: Sensor's FoV using ray tracing

The environment also includes predefined obstacles, represented as black cells, to introduce challenges for the agent's exploration strategy. Regions of Interest (yellow cells) are strategically placed around the object (blue), to evaluate the agent's ability to prioritize areas of higher significance during the search process. As the agent moves and gathers information, cells are updated to reflect their observed status, transitioning from unexplored (white, black, yellow or blue) to explored (green, red or purple).

This setup provided a simplified yet effective framework for testing and validating the core functionalities of the search algorithm, including the simulation of sensor capabilities, the representation of spatial information, and the dynamic updating of the map.

3.3.2 Metrics for Evaluation

To evaluate and compare the performance of the proposed algorithms, a set of quantitative metrics was defined. These metrics capture various aspects of the exploration and search process, including efficiency, coverage, and movement cost. By analyzing these metrics, we can assess how well each algorithm performs in different scenarios and under various constraints. Below is a detailed description of each metric:

1. **Percentage of the Explored Map:** This metric tracks the proportion of the total map that has been explored over the course of the simulation. It provides insight into the efficiency of the algorithm in reducing unexplored areas and achieving comprehensive coverage. Measuring this percentage across iterations also allows for an evaluation of the exploration rate, helping identify algorithms that balance speed and thoroughness.
2. **Percentage of the Explored ROI:** Since regions of interest (ROIs) are critical in focused exploration tasks, this metric measures the proportion of the ROI that has been explored over iterations. This allows us to determine how effectively an algorithm prioritizes high-value areas and how quickly it achieves the specific goal of covering the ROI.
3. **Iterations (Positions Only):** This metric represents the number of positional steps the agent requires to locate the target object. Each step corresponds to a movement from one position to the next best position based on the algorithm's evaluation criteria. By isolating the positional aspect of the exploration process, this metric provides a focused measure of how effectively the algorithm directs the agent toward the target.

The goal is to minimize this value, as fewer positional steps indicate that the algorithm is efficient in selecting optimal positions, thereby reducing the time and resources needed to locate the object. A lower number of iterations reflects the algorithm's ability to prioritize high-information areas and converge quickly on the target.

4. **Total Iterations (Positions and Orientations):** This metric extends the concept of positional steps by also accounting for the orientation adjustments made at each position. It measures the total number of decisions and actions, including both movement to a new position and the evaluation of multiple orientations to select the optimal field of view.

The combined consideration of positions and orientations provides a comprehensive view of the algorithm's performance, encompassing both spatial efficiency and its ability to direct the agent's gaze toward unexplored or critical areas. Minimizing this metric indicates that the algorithm is not only effective in selecting optimal positions but also efficient in determining orientations, resulting in faster and more targeted exploration of the environment.

5. **Distance to the Object Over Iterations:** Tracking the distance from the agent to the target object over the course of the exploration highlights how effectively the algorithm directs the agent toward the objective. A steady decrease in this distance indicates that the algorithm efficiently narrows the search area and converges on the target.

6. **Total Distance Travelled:** This metric measures the cumulative distance the agent travels throughout the exploration process. Each movement from one position to another contributes to the total distance, reflecting the spatial efficiency of the search strategy. Minimizing the total distance travelled is critical, as it directly impacts the resource consumption of the robot, such as energy usage and time. An effective search algorithm should aim to reduce unnecessary movements by selecting positions that maximize information gain while maintaining a direct trajectory toward the target. By minimizing the total distance, the robot can locate the objective more quickly and with fewer resources, which is especially important in energy-constrained or time-sensitive applications.
7. **Total Angular Distance Travelled:** In addition to translational movement, this metric tracks the cumulative angular adjustments the agent makes during the search process. Each orientation change, measured in degrees, contributes to the angular distance travelled. Excessive angular movement can result in inefficiencies, consuming time and energy that could otherwise be allocated to forward progress or mapping efforts.

Minimizing angular distance is essential for efficient planning, as it ensures the robot is not wasting resources on unnecessary rotations. A well-designed search strategy should optimize the balance between position selection and orientation adjustments, reducing angular movements while maintaining effective coverage of the environment.

In summary, the metrics outlined above provide a comprehensive framework for evaluating the performance of the proposed algorithms in object searching tasks. Each metric focuses on a specific aspect of the exploration process, such as coverage efficiency, resource consumption, and goal-directed behavior, enabling a thorough comparison of different strategies. However, it is important to note that the field lacks a universally agreed-upon evaluation metric. Many studies adopt their own methodologies, often emphasizing specific factors such as time, computational cost, or distance traveled, depending on the particular use case or research focus.

To address this variability, we have chosen a diverse set of metrics that collectively offer a well-rounded understanding of algorithm performance. This approach provides a balanced and informative perspective, facilitating a deeper understanding of their strengths, weaknesses, and practical applicability in real-world scenarios.

3.3.3 Basic Search Algorithms

To develop a comprehensive understanding of the object searching problem, we began by creating five distinct algorithms, ranging from the simplest to progressively more complex approaches. Starting with basic algorithms allowed us to grasp the core challenges and principles of object searching in a controlled and structured manner. These foundational algorithms serve as a stepping stone for more sophisticated solutions and help establish a clear framework for evaluation.

To guide the exploration process and better understand the principles of object searching, we decided that these initial algorithms would only consider frontier cells as valid next positions. This decision was based on the general idea that frontiers represent the boundary between known and unknown regions, making them the areas where the most valuable information about the environment can be obtained. By focusing on frontiers, the algorithms could prioritize expanding the explored area while maximizing information gain from each movement.

Frontier cells, as described in Section 3.3.1, are those that lie adjacent to both explored (free or occupied) and unexplored cells. This adjacency is limited to a cross pattern (up, down, left, and right), excluding diagonally adjacent cells for simplicity and computational clarity. By constraining the exploration to these boundary cells, the algorithms systematically reduced the unexplored space while ensuring efficient navigation through the environment.

To simplify the Next-Best-View (NBV) process and reduce its computational complexity, we divided it into two separate steps: first, selecting the best position, and then determining the optimal orientations for that position. By decoupling these decisions, we ensured that the problem remained tractable and allowed for systematic evaluation of each component. The algorithms we will test focus on different strategies for selecting the best position, while the method for determining orientations remains consistent across all of them.

Once the agent moves to the next best position, the orientation selection is done as follows: The process is based on evaluating eight predefined orientations, separated by 45 degrees, to cover all possible directions around the chosen position. The goal is to maximize visibility and information gain from the immediate surroundings. The agent will prioritize looking towards neighboring cells that are either frontiers or unexplored, ensuring that its field of view is directed towards regions likely to provide valuable updates to the map, Figure 3.6.

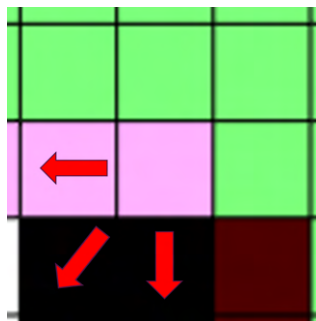


Figure 3.6: Orientation selection in basic algorithms

1. Closest Frontier Algorithm

The Closest Frontier Algorithm selects the frontier cell that is nearest to the agent's starting position. This approach leads the agent to explore areas immediately surrounding its initial point, gradually expanding the radius of exploration as more frontiers become available. By prioritizing proximity, the algorithm ensures a systematic and incremental expansion of the explored region, allowing for a detailed mapping of the area close to the starting point. This method is particularly effective for environments where local exploration is prioritized or where the immediate surroundings hold significant importance. A visual example of this process is shown in Figure 3.7.

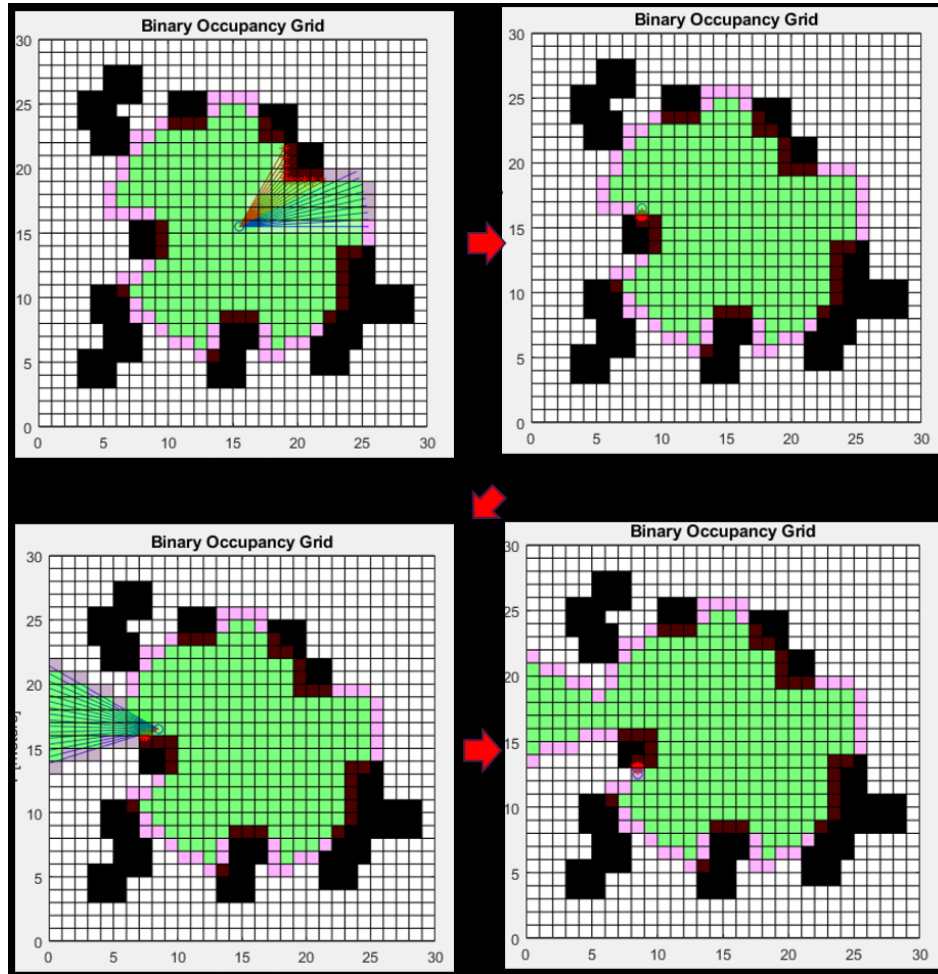


Figure 3.7: Closest frontier example

2. Furthest Frontier Algorithm

In contrast to the closest frontier approach, the Furthest Frontier Algorithm selects the frontier cell that is the farthest from the starting position. This strategy encourages a more aggressive exploration style, directing the agent to explore distant areas of the map early in the process. By attempting to move away from the already explored region, the algorithm can often achieve greater information gain with each position selected. This is particularly

beneficial in scenarios where the objective is to quickly cover a large portion of the map or where regions far from the starting point are more likely to contain valuable information. A visual example of this process is shown in Figure 3.8.

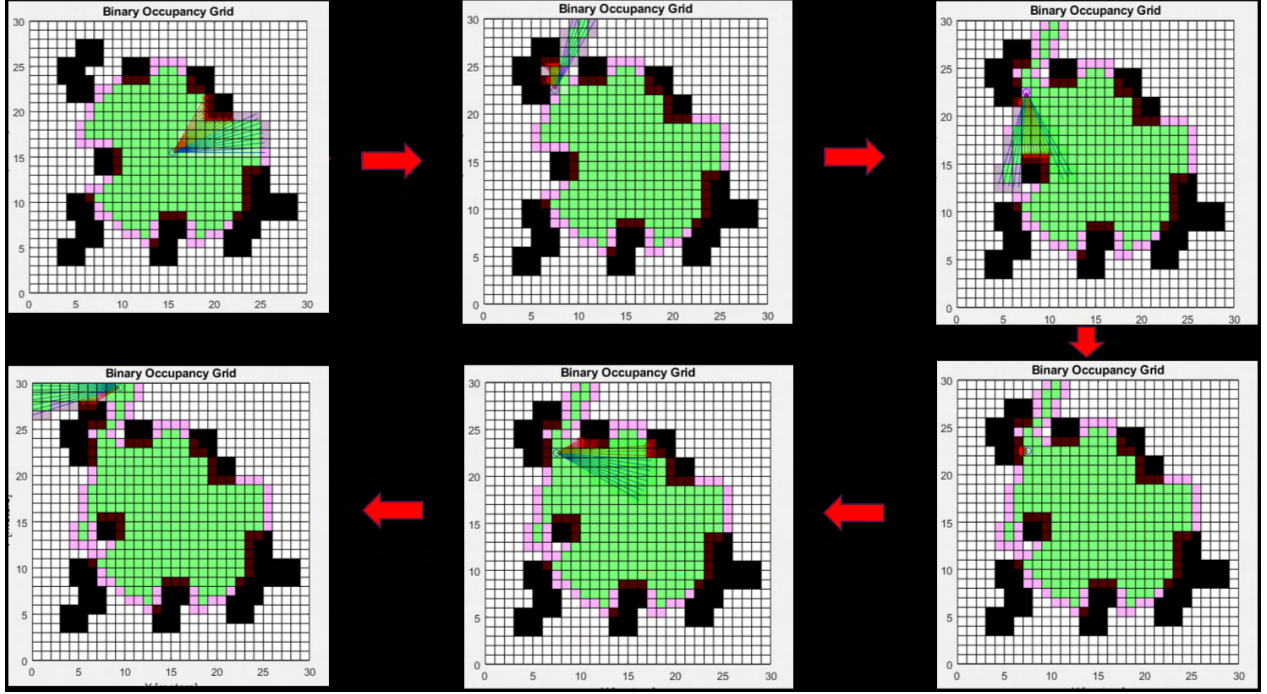


Figure 3.8: Furthest frontier example

3. Information Gain Algorithm

The Information Gain Algorithm selects the frontier cell based only on the potential amount of new information it can provide. For each candidate frontier cell, the algorithm calculates the number of unexplored cells within a predefined radius around it. The frontier with the highest number of unexplored neighboring cells is chosen as the next-best position. This approach prioritizes maximizing the information gain in every step, focusing the exploration effort on areas that are likely to yield the greatest updates to the map. However, it does not consider the distance cost, which can result in less efficient paths in environments with widely distributed frontiers. A visual example of this process is shown in Figure 3.9.

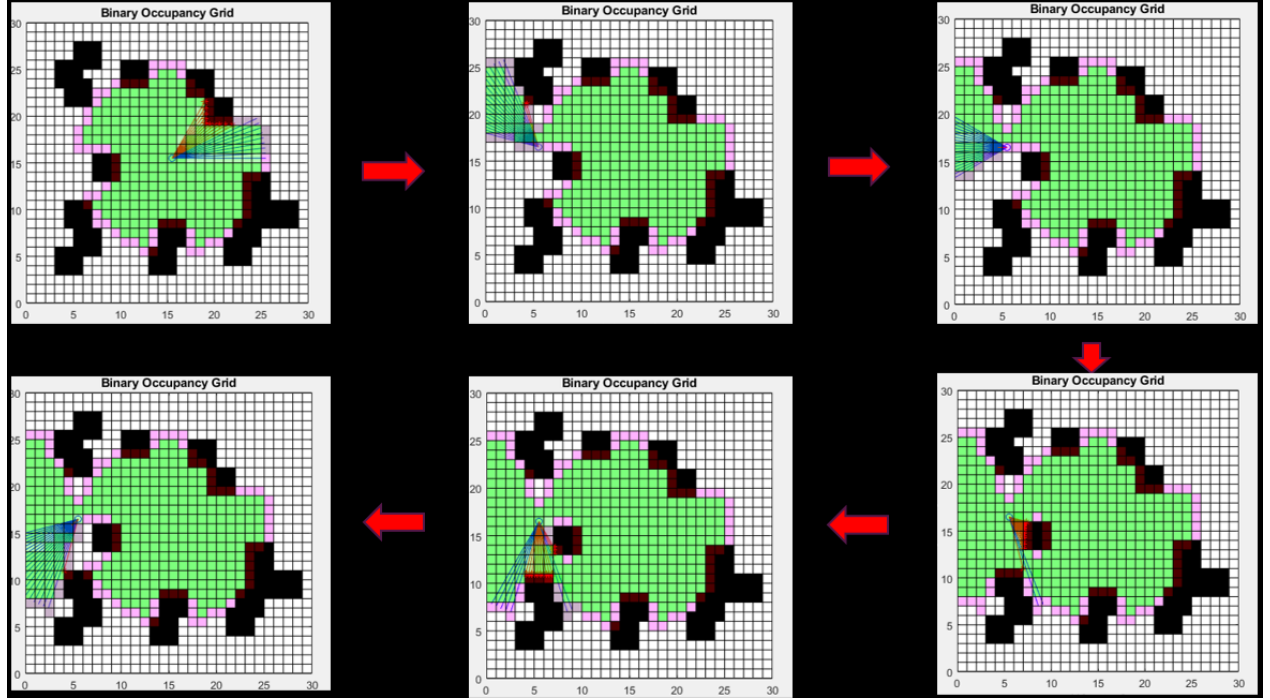


Figure 3.9: Information Gain Algorithm example

4. Information Gain + Displacement Cost Algorithm

Building on the information gain approach, the Information Gain + Displacement Cost Algorithm balances information gain with the cost of moving to a candidate position. The decision is based on a cost function that combines two factors, Equation 3.1, higher the score, better the candidate position.

$$\text{score} = \alpha \left(\frac{\text{NumInexploradas}}{\text{MaxUnexploredCells}} \right) - \beta \left(\frac{\text{dist}}{\text{MaxDistanceInMap}} \right) \quad (3.1)$$

Where:

- NumInexploradas is the number of unexplored cells within a predefined radius around the candidate frontier cell.
- dist is the Euclidean distance between the agent's current position and the candidate frontier cell.
- MaxUnexploredCells and MaxDistanceInMap are normalization factors to ensure the components are on a comparable scale. MaxUnexploredCells is the maximum amount of cells that could be unexplored around a determined position while MaxDistanceInMap is the maximum distance reachable in the grid map, is directly dependant to the map size.
- α and β are weighting factors that determine the relative importance of information gain and displacement cost.

This algorithm allows for a more balanced exploration strategy. By considering both the potential information gain and the cost of reaching a frontier, it ensures that the agent does not expend excessive effort moving to distant locations with marginal benefits. The weighting factors can be adjusted depending on the specific goals of the exploration task, such as prioritizing efficiency over coverage or vice versa.

5. ROI-Based Algorithm

The ROI-Based Algorithm builds upon the Information Gain + Displacement Cost Algorithm by incorporating an additional term in the cost function to guide the search towards predefined regions of interest. This enhancement leverages prior knowledge of the ROIs to focus the exploration on areas of higher priority, balancing the general exploration with a directed search.

The updated cost function is defined as follows, Equation 3.2:

$$\text{score} = \alpha \left(\frac{\text{NumInexploradas}}{\text{MaxUnexploredCells}} \right) - \beta \left(\frac{\text{dist}}{\text{MaxDistanceInMap}} \right) - \gamma \left(\frac{\text{distProb}}{\text{MaxDistanceInMap}} \right) \quad (3.2)$$

Where:

- NumInexploradas: The number of unexplored cells within a predefined radius around the candidate frontier cell.
- dist: The Euclidean distance from the agent's current position to the candidate frontier cell.
- distProb: The Euclidean distance from the candidate frontier cell to the mean position of the ROI.
- MaxUnexploredCells and MaxDistanceInMap: Normalization factors for scaling each term.
- α , β and γ : Weighting factors that define the importance of information gain, displacement cost, and distance to the ROI, respectively.

The ROI's mean position is calculated by averaging the coordinates of all cells identified as part of the ROI. This mean position serves as the target for the additional term in the cost function, encouraging the algorithm to prioritize frontier cells closer to the ROI. By minimizing distProb, the system is guided towards areas of higher significance, ensuring that the exploration remains purposeful and aligned with the search objectives.

This approach combines efficient exploration with task-oriented guidance, balancing the need to explore new areas with the goal of directing the agent's efforts towards critical regions. It is particularly effective in scenarios where ROIs are known to contain high-value targets, enabling the system to adapt dynamically to the environment while maintaining a clear focus. A visual example can be found in Figure 3.10.

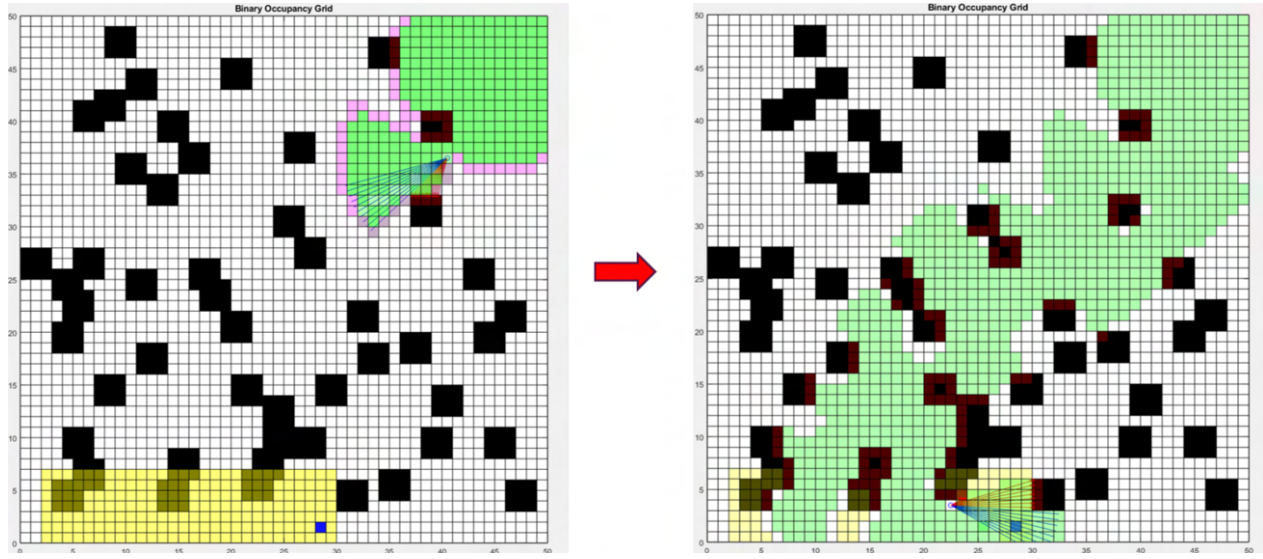


Figure 3.10: ROI Algorithm example

3.3.4 Gradient Representation

In the last approach, ROI-Based Algorithm, the calculation of the mean position of the Region of Interest was used to guide the agent towards high-priority areas. However, this method proved insufficient in scenarios where the ROI had an irregular shape, was significantly large, or consisted of multiple disjoint regions, as shown in Figure 3.11. To address these limitations, we developed a gradient-based approach that integrates the spatial information of the ROIs, detected obstacles, and unexplored areas into a unified map. This approach allows for a more nuanced and precise navigation strategy.

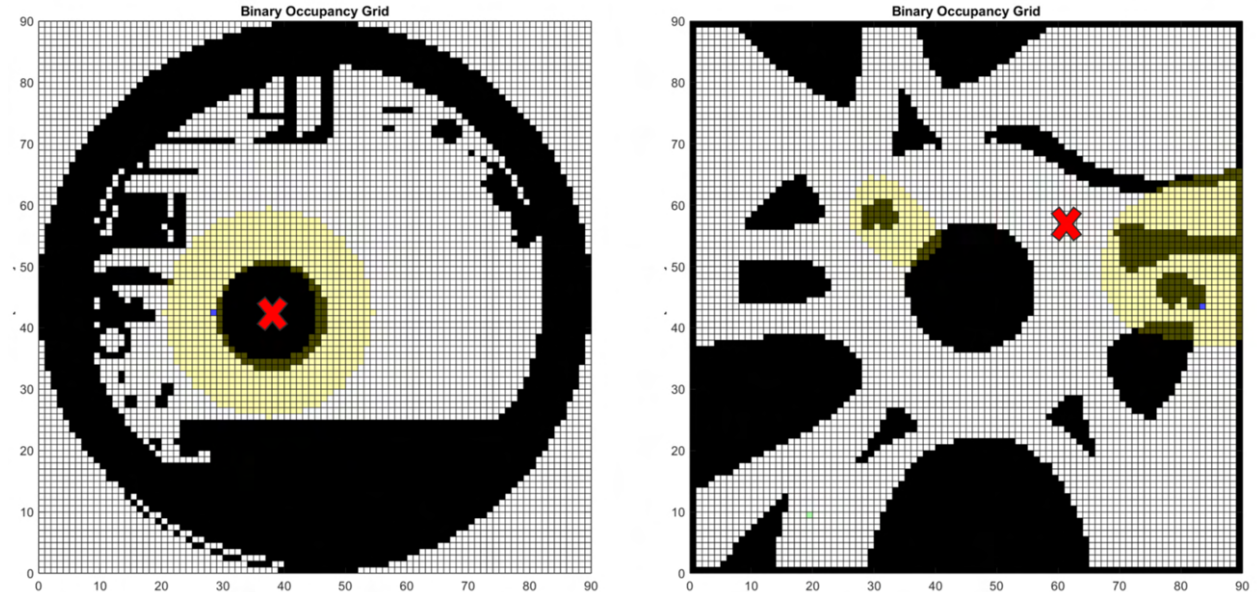


Figure 3.11: Average ROI Position Error

The gradient map is a spatial representation that assigns a numerical value to each cell in the environment based on its proximity to certain key features:

1. **ROI Proximity:** To prioritize movement toward ROIs, the gradient map assigns higher values to cells closer to the ROI, Figure 3.12:

- For each cell (i, j) , the Euclidean distance to all ROI cells is computed:

$$d = \sqrt{(r_{\text{ROI}} - i)^2 + (c_{\text{ROI}} - j)^2}, \quad (3.3)$$

where r_{ROI} and c_{ROI} are the row and column indices of the ROI cells. The minimum distance, d_{min} , is selected for each cell.

- The gradient value for a cell is then calculated using the inverse-square formula:

$$G_{\text{ROI}}(i, j) = \frac{1}{d_{\text{min}}^2 + \epsilon} * \epsilon \quad (3.4)$$

where ϵ_{ROI} is a smoothing parameter. For ROIs, a relatively large ϵ_{ROI} is used to ensure a gradual transition of influence throughout the map. This helps to extend the desirability of ROI-proximal regions, effectively guiding the agent across the entire map toward the ROI.

- Cells that belong to the ROI are assigned a fixed high value:

$$G_{\text{ROI}}(i, j) = 1 \quad \text{if } \text{cellState}(i, j) = \text{ROI}. \quad (3.5)$$

2. **Obstacle Avoidance:** Obstacles influence the gradient map by penalizing cells near them, encouraging the agent to maintain a safe distance to maximize its field of view, Figure 3.13:

- For each cell (i, j) , the Euclidean distance to all obstacle cells is computed:

$$d_{\text{obs}} = \sqrt{(r_{\text{obs}} - i)^2 + (c_{\text{obs}} - j)^2}. \quad (3.6)$$

where r_{obs} and c_{obs} are the row and column indices of the obstacles cells. The minimum distance, $d_{\text{obs min}}$, is selected for each cell.

- Cells closer to obstacles are assigned lower values:

$$G_{\text{obs}}(i, j) = \frac{1}{d_{\text{obs min}}^2 + \epsilon} * \epsilon, \quad \text{if } d_{\text{obs min}} < d_{\text{threshold}}, \quad (3.7)$$

where ϵ_{obs} is a smaller smoothing parameter. A localized smoothing effect is desired for obstacles to limit their influence to nearby cells. In addition, in order to only affect cells which are close to obstacles, we use a threshold $d_{\text{threshold}}$, which is the maximum distance within which obstacle effects are considered. Beyond this threshold, the gradient value is set to zero to ensure that obstacle influence remains localized.

This ensures that the agent avoids positions too close to obstacles, as these restrict the field of view and hinder exploration efficiency.

- Cells identified as obstacles are assigned the lowest value:

$$G_{\text{obs}}(i, j) = 1 \quad \text{if } \text{cellState}(i, j) = \text{obstacle}. \quad (3.8)$$

3. **Unexplored Areas:** To promote exploration, the gradient map assigns values to cells near unexplored areas, Figure 3.14:

- For each cell (i, j) , the Euclidean distance to unexplored cells is calculated:

$$d_{\text{unexp}} = \sqrt{(r_{\text{unexp}} - i)^2 + (c_{\text{unexp}} - j)^2}. \quad (3.9)$$

where r_{unexp} and c_{unexp} are the row and column indices of the obstacles cells. The minimum distance, $d_{\text{unexp min}}$, is selected for each cell.

- Cells closer to unexplored areas are given higher values:

$$G_{\text{unexp}}(i, j) = \frac{1}{d_{\text{unexp min}}^2 + \epsilon} * \epsilon, \quad (3.10)$$

where ϵ_{unexp} is moderately sized to ensure unexplored areas influence nearby cells effectively. This incentivizes the agent to continue exploring unknown regions while maintaining a balanced approach.

- Cells which are unexplored are assigned a fixed high value:

$$G_{\text{unexp}}(i, j) = 1 \quad \text{if } \text{cellState}(i, j) = \text{unexp}. \quad (3.11)$$

After each component of the gradient map (ROI proximity, obstacle avoidance, and unexplored areas) is calculated individually, the maps are combined into a single composite map, with each component weighted according to its relative importance 3.12. To enhance continuity and reduce abrupt changes, the resulting maps are smoothed using a Gaussian filter, as shown in Figure Figure 3.15. This smoothing improves navigation behavior by providing a more continuous and navigable representation of the environment.

$$G_{\text{combined}}(i, j) = w_{\text{ROI}} \cdot G_{\text{ROI}}(i, j) - w_{\text{obs}} \cdot G_{\text{obs}}(i, j) + w_{\text{unexp}} \cdot G_{\text{unexp}}(i, j), \quad (3.12)$$

where:

- $G_{\text{combined}}(i, j)$: The combined map value for the cell at (i, j) .
- w_{ROI} : Weighting factor for the ROI gradient map, prioritizing cells near ROI regions.
- w_{obs} : Weighting factor for the obstacle gradient map, modulating the influence of obstacle proximity.
- w_{unexp} : Weighting factor for the unexplored gradient map, prioritizing cells near unknown regions.

This combined gradient map provides a detailed representation of the environment, enabling the agent to prioritize regions of interest while avoiding obstacles and efficiently exploring new areas.

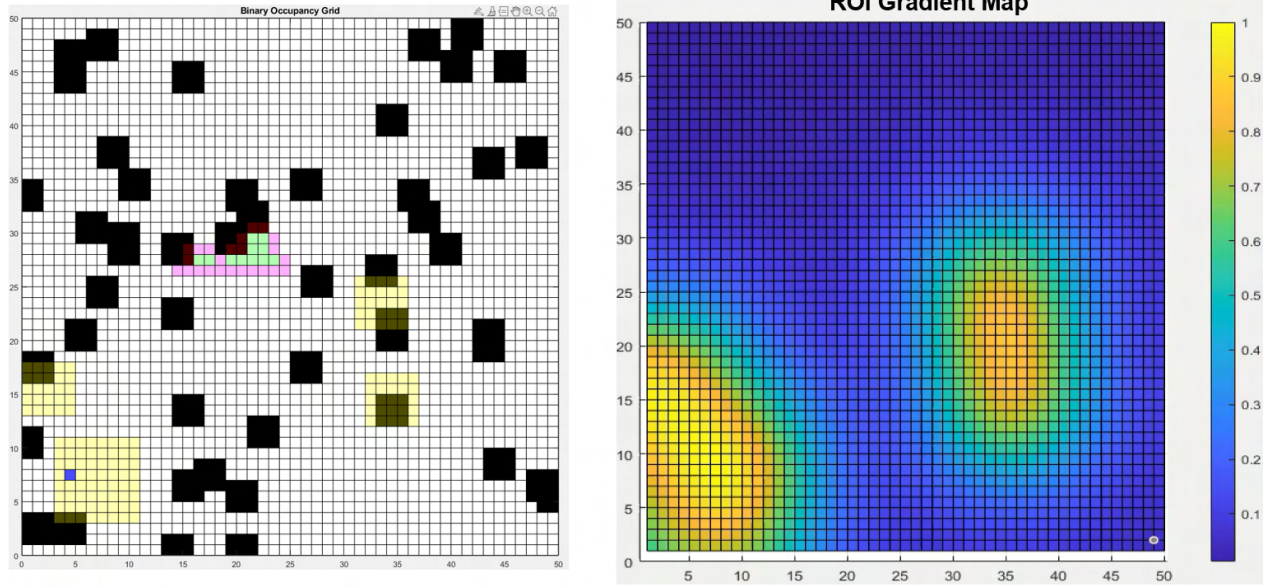


Figure 3.12: ROI Gradient Map

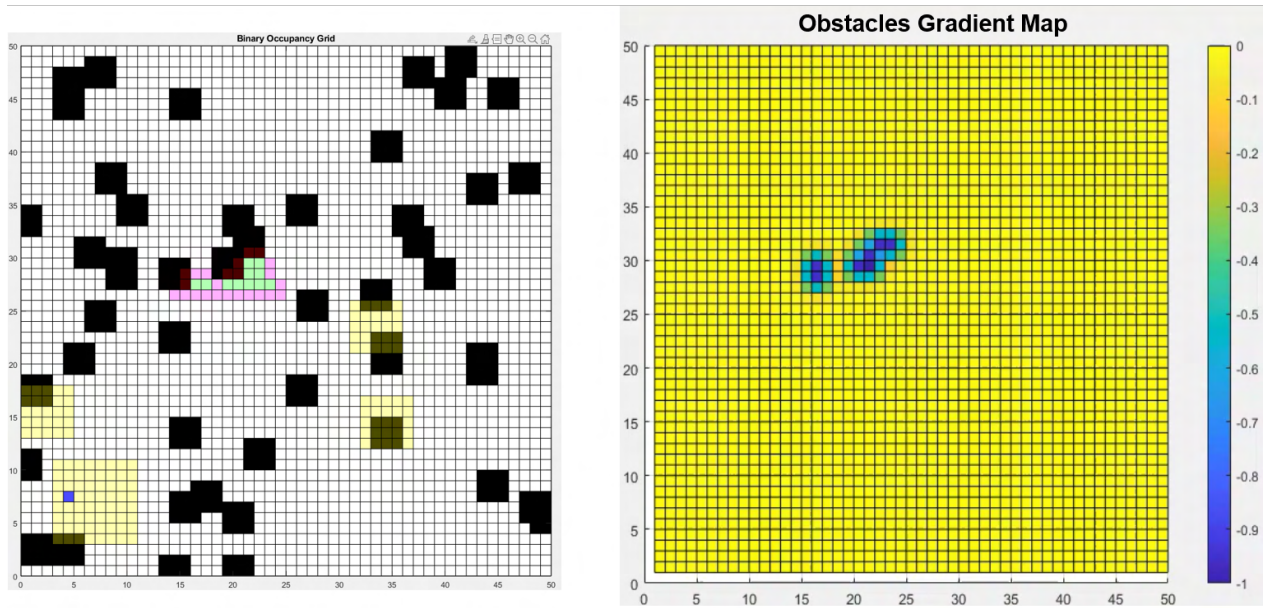


Figure 3.13: Obstacles Gradient Map

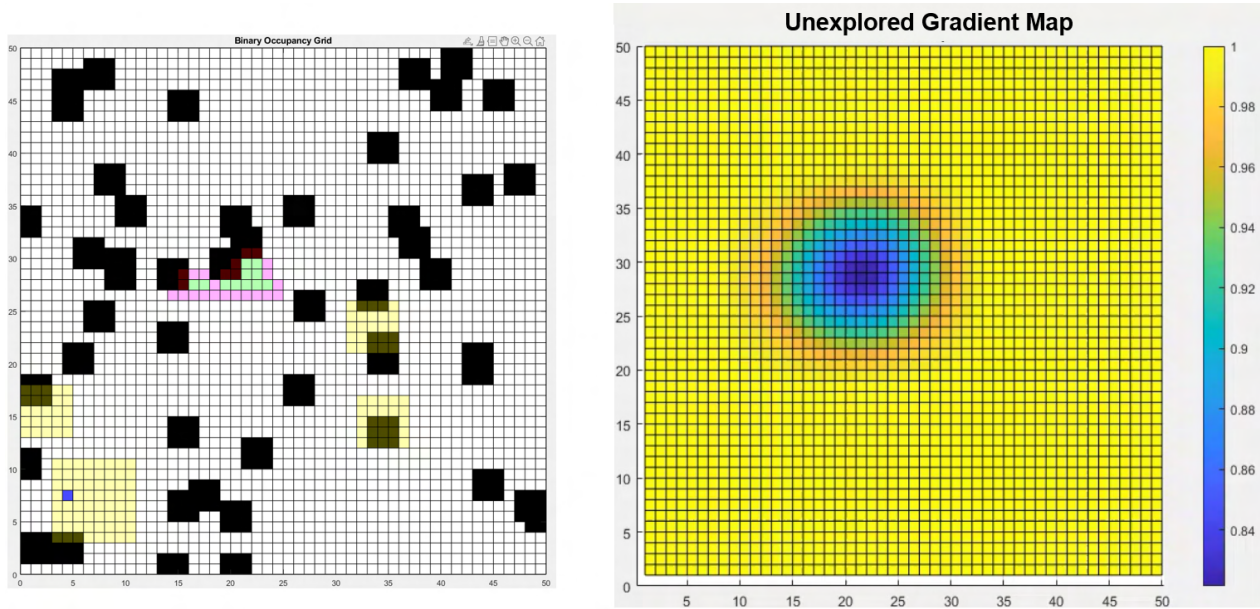


Figure 3.14: Unexplored Gradient Map

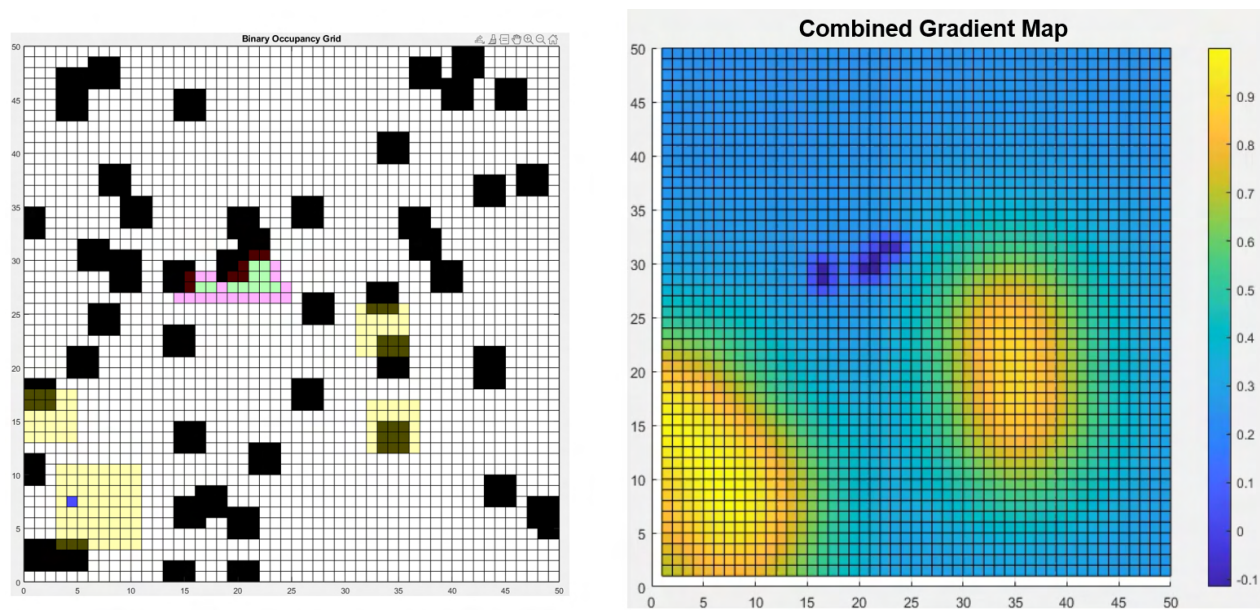


Figure 3.15: Combined Gradient Map

Integration into the Cost Function

To evaluate candidate frontier cells, the combined gradient map value is integrated into the cost function used to determine the Next-Best-View (NBV). The cost function is defined as:

$$\text{Score} = \alpha \cdot G_{\text{combined}}(i, j) - \beta \cdot C_{\text{disp}}(i, j), \quad (3.13)$$

where:

- α : Weighting factor for the composite map value, emphasizing the desirability of high-gradient cells.
- $G_{\text{combined}}(i, j)$: Combined gradient map value for the candidate cell.
- β : Weighting factor for the displacement cost.
- $C_{\text{disp}}(i, j)$: Displacement cost, calculated as the Euclidean distance between the agent's current position and the candidate cell:

$$C_{\text{disp}}(i, j) = \sqrt{(x_{\text{current}} - x_{\text{candidate}})^2 + (y_{\text{current}} - y_{\text{candidate}})^2}. \quad (3.14)$$

To balance exploration efficiency and resource consumption, the displacement cost is smoothed using a sigmoid function:

$$C_{\text{smoothed}}(i, j) = \frac{1}{1 + e^{-k \cdot (C_{\text{disp}}(i, j) - d_{\text{threshold}})}}, \quad (3.15)$$

where k controls the steepness of the transition and $d_{\text{threshold}}$ defines the inflection point.

Algorithm Workflow

The workflow for the gradient-based approach is as follows:

1. **Gradient Map Generation:** Separate gradient maps for ROIs, obstacles, and unexplored areas are constructed and smoothed using Gaussian filters.
2. **Combined Map Calculation:** The individual gradient maps are combined using predefined weights to create a unified representation of the environment.
3. **Frontier Evaluation:** Each candidate frontier cell is evaluated by computing its score using the cost function defined in Equation 3.13.
4. **Selection of the Next Position:** The candidate cell with the highest score is selected as the next position for the agent, ensuring efficient navigation and prioritization of high-value regions.

Advantages of the Gradient-Based Approach

The gradient-based approach addresses limitations of earlier mean-based ROI guidance methods by providing a flexible and robust framework. By seamlessly integrating multiple environmental features, the gradient map ensures that the agent navigates intelligently,

balancing exploration and goal-directed behavior. This method allows the agent to prioritize complex regions of interest while avoiding obstacles and continuing to uncover new areas in an efficient manner.

3.3.5 Approach with Field of View Simulation - HOPE-OS 2D

While the gradient-based approach provided a robust framework for guiding the agent toward regions of interest, it exhibited certain limitations that hindered its overall effectiveness. One significant issue was its tendency to direct the agent too close to the ROI. While proximity to the ROI may seem advantageous, this behavior often limited the agent's field of view, reducing its ability to maximize both the quantity and quality of information gathered. By positioning itself too near the ROI, the agent could inadvertently obscure parts of the region or fail to observe critical unexplored areas, as occlusions and limited visibility range became more pronounced.

Additionally, the algorithm's reliance on exploring frontier cells restricted the range of possible movements and search strategies. While frontiers are valuable for expanding the explored area, they are not always the optimal choice. Non-frontier free cells, particularly those strategically located at a moderate distance from the ROI, can provide significant informational value by enhancing visibility and coverage of the surrounding environment.

These limitations are illustrated in Figure 3.16, while the desired behavior is shown in Figure 3.17

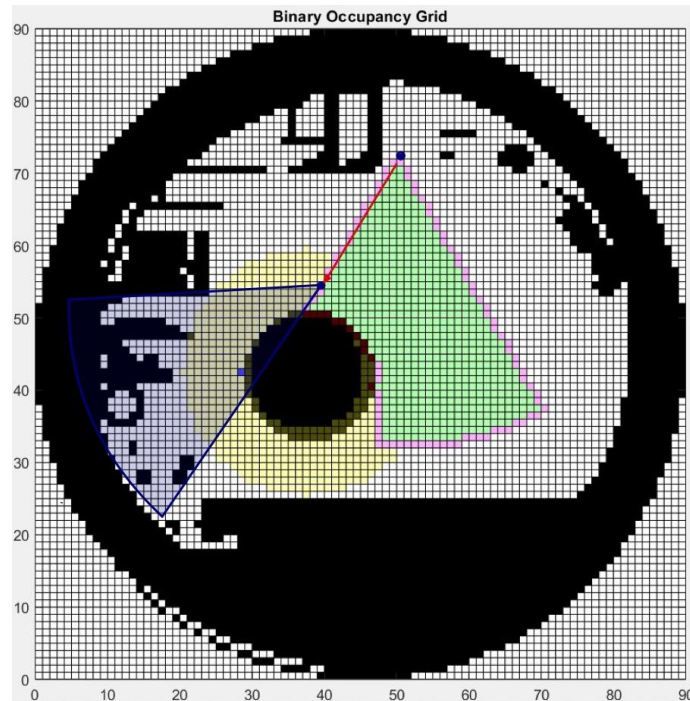


Figure 3.16: Gradient Approach NBV

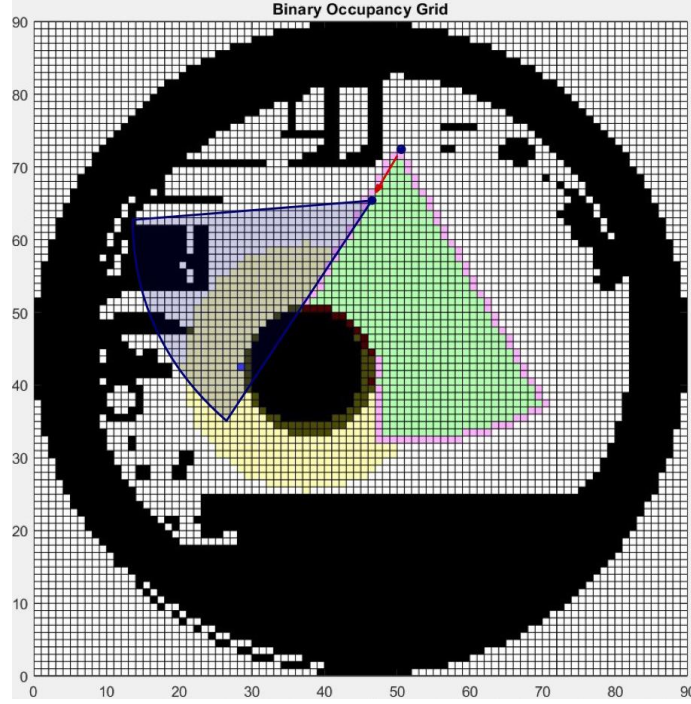


Figure 3.17: Desired NBV

Figure 3.16 demonstrates the behavior of the gradient-based algorithm, where the agent approaches the ROI too closely, resulting in reduced visibility and suboptimal information gain. In contrast, Figure 3.17 highlights the desired behavior, where the agent maintains a strategic distance from the ROI, optimizing its FoV to capture a larger portion of the ROI and its surroundings. This balanced approach not only enhances the efficiency of the exploration process, but also ensures that the agent collects more comprehensive data with each movement.

Clustering for Computational Efficiency

One of the key challenges in implementing the algorithm was the computational cost of evaluating all potential positions in the free space. Allowing the agent to explore not only frontier cells but all free cells in the map significantly expanded the search space. While this increased the flexibility of the algorithm, it also introduced a considerable computational burden, as the number of candidate positions grew rapidly with the size of the map.

To address this challenge, we implemented a clustering approach based on k-means to reduce the number of candidate positions. The clustering process groups the free cells into clusters, with each cluster represented by its centroid. This enables the algorithm to evaluate a manageable subset of representative positions, rather than exhaustively considering every free cell. The centroids are then treated as candidate positions for the Next-Best-Position (NBP) evaluation, Figure 3.18.

The choice of k-means clustering is particularly justified in this context for the following reasons:

1. **Computational Efficiency:** By reducing the number of candidate positions, clustering significantly decreases the computational complexity of the evaluation process. This is especially important in large maps where the number of free cells can be prohibitively high.
2. **Spatial Representation:** Clustering naturally captures the spatial distribution of free cells, ensuring that the candidate positions are evenly distributed across the map. This avoids over-sampling in densely populated regions and ensures that the agent considers positions across the entire free space.
3. **Dynamic Adaptation:** As the agent explores the environment and the distribution of free cells changes, the clustering process dynamically adapts, ensuring that the selected candidates remain representative of the current map state.

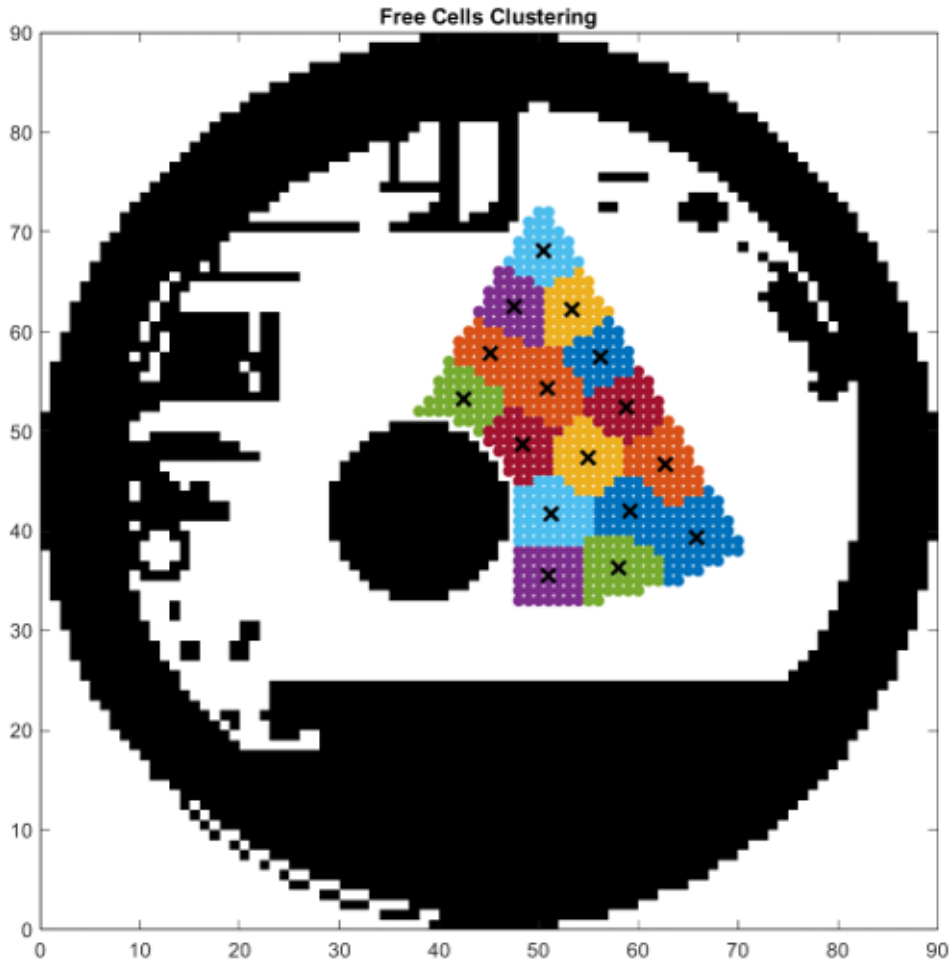


Figure 3.18: Illustration of k-means clustering applied to the free space of the map. The centroids (marked as crosses) represent candidate positions for evaluation.

Reclustering Process

To efficiently narrow down the search area and reduce computational cost, a reclustering strategy is employed. The process begins by dividing the free space into clusters using the k-means algorithm, the clustering resolution is adjusted dynamically based on the number of free cells and a target number of clusters. This ensures that the clusters become progressively smaller and more precise in areas of interest, maintaining a balance between computational cost and search precision.

The reclustering process refines the search space by progressively adjusting the size and number of clusters, ensuring that computational efficiency is maintained while focusing on the most promising areas. Below is a detailed explanation of each step:

1. **Initial Setup:** At the beginning, all free cells ($N_{\text{free_cells}}$) are treated as potential candidate positions for the agent. The algorithm starts with a small initial cluster size ($s_{\text{cluster}} = s_{\text{min}}$) to divide the space into finer regions, Figure 3.19 left.
2. **Dynamic Adjustment of Cluster Size:** The number of clusters (k_{current}) is calculated using the formula:

$$k_{\text{current}} = \left\lceil \frac{N_{\text{free_cells}}}{s_{\text{cluster}}^3} \right\rceil \quad (3.16)$$

where:

- $N_{\text{free_cells}}$: Total number of free cells in the search space.
- s_{cluster} : Current size of each cluster.

To ensure the number of clusters stays constant, the algorithm uses a loop to incrementally increase the cluster size (s_{cluster}). The loop runs as follows:

```
clustersize = 1;
target_clusters; % Desired number of clusters
while true
    num_clusters = ceil(N_free_voxels / clustersize);
    if num_clusters <= target_clusters
        break; % Stop when the desired number of clusters is reached
    else
        clustersize = clustersize + 1; % Increment cluster size
    end
end
```

This dynamic adjustment ensures that the cluster size increases iteratively, reducing the number of clusters (k_{current}) until it falls below or equals a predefined threshold (k_{target}), thereby, the loop terminates when $k_{\text{current}} \leq k_{\text{target}}$. At this point, the cluster size (s_{cluster}) is optimal for refining the search space, balancing computational effort with resolution, as shown in Figure 3.19 right.

3. **Reclustering for Higher Resolution:** After determining the optimal cluster size and evaluating the clusters using a scoring function (explained in Section 3.3.5), the

top-performing clusters are selected for further refinement. This selection ensures that the algorithm focuses on the most promising regions of the search space.

The clustering process is then repeated within the top-performing clusters, following the same procedure as the initial clustering, equation 3.16, but with an important difference: The size of the candidate cells ($N_{\text{free_cells}}$) now is lower, because it only takes into consideration the free cells of the top-performing clusters. Therefore, if the number of candidates is lower, and we want to achieve $k_{\text{current}} \leq k_{\text{target}}$ in equation 3.16, the size of the clusters s_{cluster} is going to be smaller. The reclustering result is shown in Figure 3.20 left.

This dynamic reclustering mechanism ensures that:

- **Dynamic Increasingly Resolution:** As the region of candidate positions becomes smaller (due to retaining only the top-performing clusters), the size of the clusters decreases, resulting in finer resolution. This guarantees that even within smaller regions, the algorithm maintains a detailed evaluation of candidate positions.
- **Adaptive Refinement:** Larger initial search areas require more reclustering steps to achieve the desired resolution, as the cluster size is progressively reduced to meet the $k_{\text{threshold}}$ condition. This makes the algorithm scalable and efficient, adapting to the size and complexity of the search space.

The iterative reclustering process stops once the size of the clusters is small enough to satisfy the resolution criteria. At this point, the refined candidate positions are evaluated to determine the Next Best Position (NBP) using the gradient map values and FoV simulations, as detailed in the subsequent sections. The NBV result is shown in Figure Figure 3.19 right.

By clustering the free cells, the algorithm ensures that the computational effort is concentrated on regions that are most likely to yield valuable information, while the dynamic adjustment of cluster sizes maintains scalability and efficiency. The iterative nature of the process allows for a gradual refinement of the search space, balancing exploration and precision in the selection of the agent's next position. This preprocessing step lays the foundation for the subsequent evaluation of potential fields of view, as described in next Section 3.3.5.

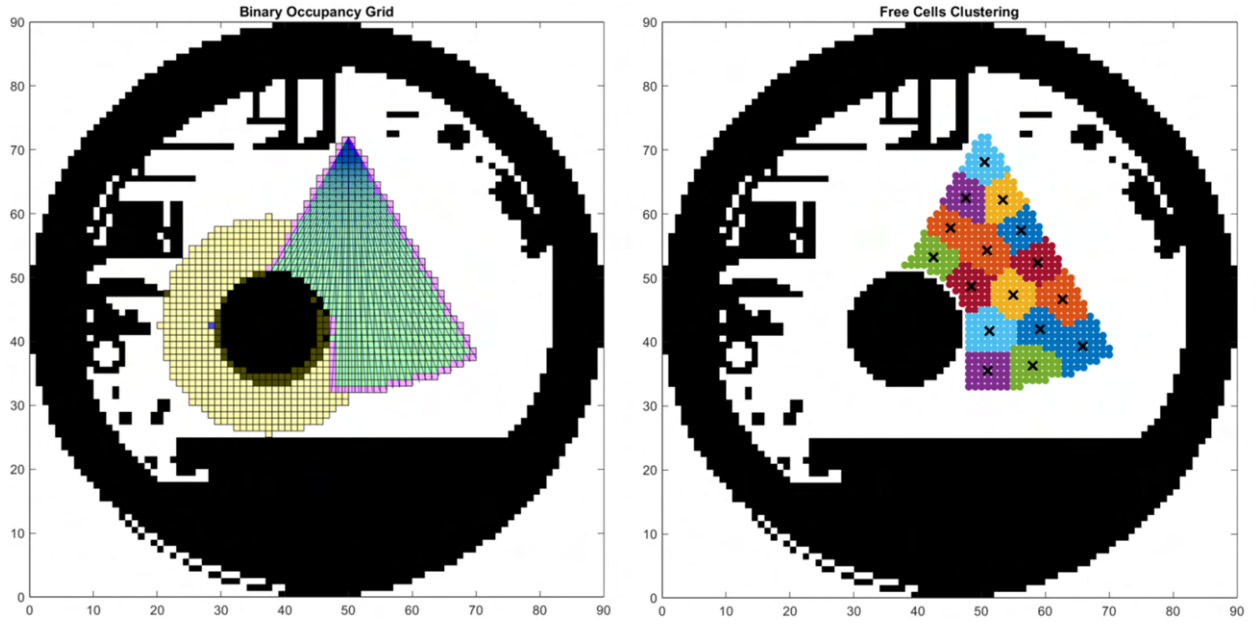


Figure 3.19: Clustering process

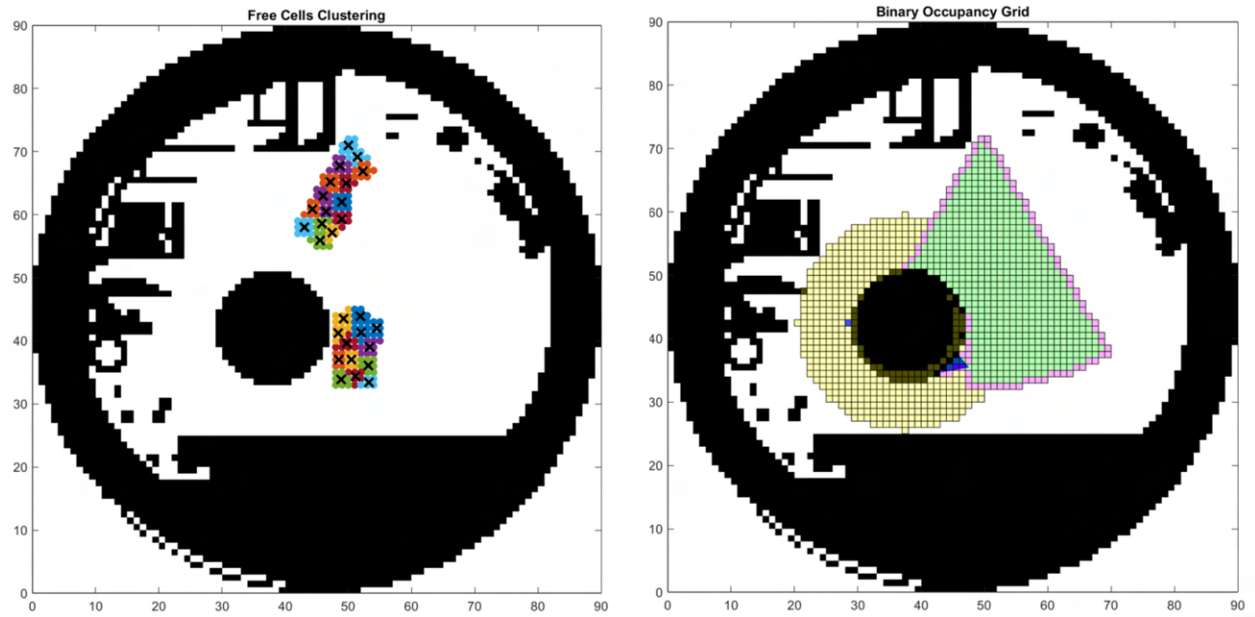


Figure 3.20: Reclustering process

Incorporating Field of View Simulation

To overcome the limitations of purely gradient-based approaches mentioned at the beginning of this Section 3.3.5, we enhanced the algorithm by integrating FoV simulations. This allows the algorithm to evaluate candidate positions more effectively by considering the actual visibility from each position. The 360° FoV simulation uses ray tracing and the current map information to model the sensor's field of view, accounting for factors such as occlusions caused by obstacles and sensor range limitations, as shown in Figure 3.21.

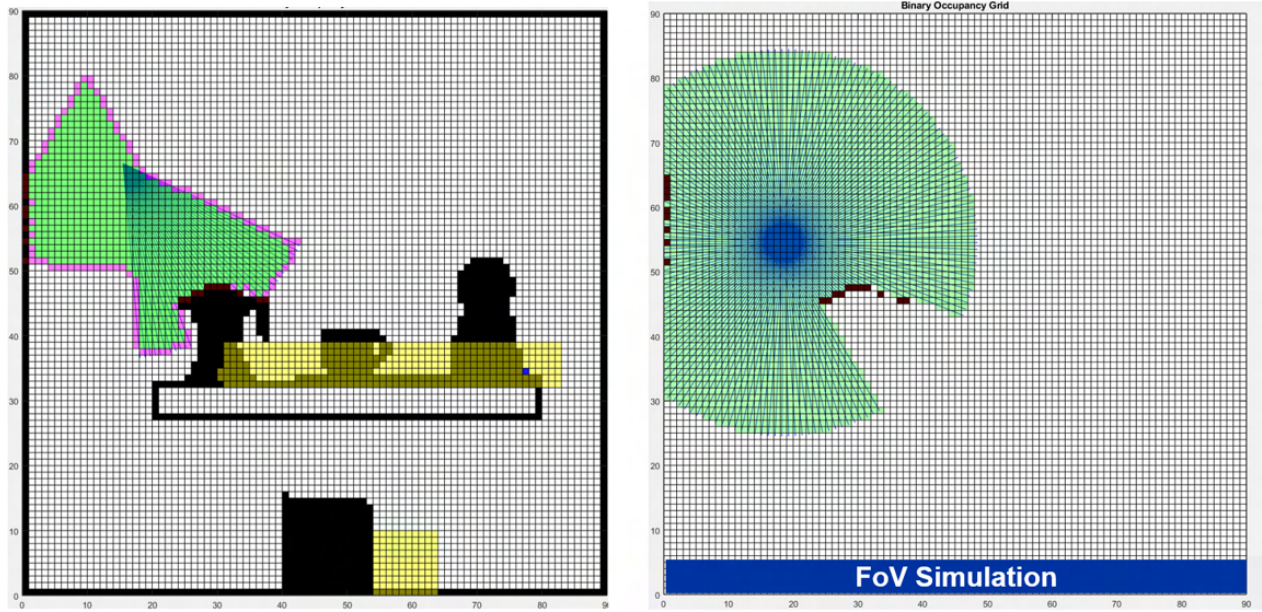


Figure 3.21: Candidate Position 360° FoV Simulation

The process for incorporating FoV simulations is as follows:

1. For each candidate position, the algorithm simulates the 360° FoV using ray tracing to determine which cells could be visible with the current occupancy information of the map.
2. The simulation excludes cells that have already been explored, focusing solely on unexplored areas and ROIs.

Once the visible unexplored cells are identified through the FoV simulation, their values in the gradient map are used to evaluate the quality of the candidate position. Since the FoV simulation already excludes explored cells and simulates obstacles occlusion, the only gradient map relevant at this stage is the ROI gradient map. By focusing solely on the ROI gradient map, the evaluation process is streamlined, targeting areas of high informational value.

- **Total Score:** The algorithm calculates the sum of the gradient values of all visible unexplored cells in the ROI gradient map. This metric represents the total quantity of potential information gain.
- **Average Score:** The algorithm computes the mean gradient value of the visible

unexplored cells. This metric reflects the quality of the information that can be obtained from the candidate position.

These two metrics were chosen because they complement each other in assessing the candidate positions. The total score focuses on maximizing the overall information gain, which is essential for efficient exploration. In contrast, the average score, being influenced by extreme values, serves to detect whether high-priority regions (e.g., ROI cells) are present within the visible area. Together, they provide a balanced evaluation, ensuring that the selected position maximizes both the quantity and quality of information observed.

The candidate positions are evaluated using a refined cost function that incorporates the results of the FoV simulation:

$$\text{Score} = \alpha \cdot \left(\omega_{\text{total}} \cdot \frac{\text{Total Score}}{\text{Max Information Gain}} + \omega_{\text{avg}} \cdot \frac{\text{Average Score}}{\text{Max Average Gain}} \right) - \beta \cdot \text{Displacement Cost}, \quad (3.17)$$

where:

- ω_{total} and ω_{avg} are weighting factors that balance the contributions of the total and average scores to the final evaluation.
- Max Information Gain and Max Average Gain are normalization constants, ensuring the total and average scores scale between 0 and 1.
- Displacement Cost penalizes positions further away from the agent's current location, calculated as:

$$\text{Displacement Cost} = \frac{1}{1 + e^{-\lambda \cdot (\text{Distance} - d_{\text{threshold}})}}, \quad (3.18)$$

where λ is a smoothing parameter, and $d_{\text{threshold}}$ defines the point where displacement starts to be significantly penalized.

- α and β are overall weighting factors that determine the relative importance of information gain and displacement cost in the final score.

The refined cost function combines both the quantity and quality of information that can be observed from a position, as well as the movement cost required to reach it. By weighting the total and average scores, the algorithm ensures that positions offering both significant information gain and high-priority insights (e.g., proximity to ROI cells) are prioritized. Meanwhile, the displacement cost encourages efficient navigation by discouraging unnecessary movement unless it is justified by high informational value.

Enhanced orientation Selection

Incorporating a more advanced orientation selection mechanism became essential to address the limitations of the previous approach, which relied solely on evaluating the neighboring cells, Section 3.3.3. The enhanced methodology leverages the simulation of the FoV to determine the optimal orientation of the agent's sensor at a given position. This approach allows the algorithm to evaluate orientations based on their visibility and informational value, improving overall exploration efficiency.

Orientation Selection Workflow:

1. **Initial Orientation Sampling:** The algorithm begins by simulating the FoV for 12 evenly spaced orientations, separated by 30° each. This sampling provides a broad initial assessment of potential orientations, ensuring that all directions around the agent are evaluated, as shown in Figure XX.
2. **Evaluation of Orientations:** Each orientation is evaluated based on the gradient map of the ROI, same process as the one explained in Section 3.3.5. In Figure 3.22 there is a visual example of the sensor's FoV simulation, where the left image shows the initial observation of the world and the right image shows how the FoV is simulated from a candidate position.
3. **Refinement:** From the initial 12 orientations, the two best-performing ones are selected based on their evaluated scores. For these two orientations, the algorithm further refines the search by simulating FoVs for orientations at $\pm 10^\circ$ and the original orientation. This step increases the resolution in the directions with the highest potential informational gain.

A final refinement is performed by simulating FoVs for the two best orientations from the previous step, this time with adjustments of ± 5 . This ensures a fine-grained evaluation around the most promising directions, enhancing the accuracy of the selected orientation.

4. **Selection of the Best Orientation:** The orientation with the highest combined score is selected as the optimal direction for the sensor, as shown in equation 3.19, similar to equation 3.17. This ensures that the agent maximizes both the quantity and quality of information gained while minimizing unnecessary computational overhead.

$$\text{Score} = \alpha \cdot \left(\omega_{\text{total}} \cdot \frac{\text{Total Score}}{\text{Max Information Gain}} + \omega_{\text{avg}} \cdot \frac{\text{Average Score}}{\text{Max Average Gain}} \right) \quad (3.19)$$

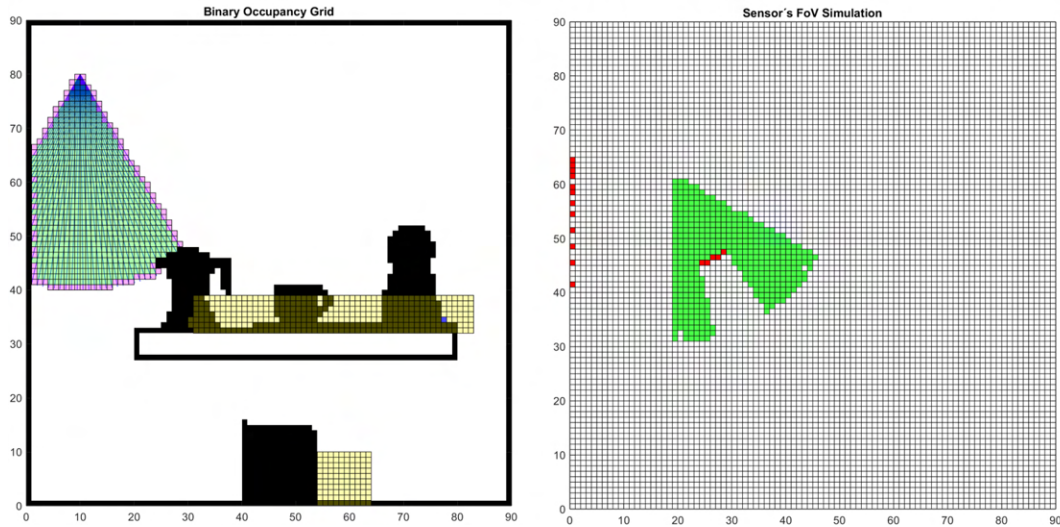


Figure 3.22: Sensor's FoV simulation for orientation selection

The enhanced orientation selection method offers several key advantages that significantly improve the overall performance of the exploration algorithm. Computational efficiency is achieved by progressively narrowing the search space through hierarchical sampling and refinement, ensuring a balance between computational cost and resolution. Improved orientation evaluation function, which increases the likelihood of detecting high-priority regions such as regions of interest.

This enhanced orientation selection process works synergistically with the position selection mechanism, ensuring that the agent not only navigates to the most informative locations but also optimally directs its sensor to maximize informational gain. Together, these advancements significantly enhance both the efficiency and effectiveness of the exploration algorithm.

Algorithm Workflow

To consolidate and summarize the concepts explained so far, the proposed exploration algorithm integrates position and orientation selection into a systematic workflow. This section outlines the key steps, accompanied by a description of their functionality and supported by a visual representation through 12 images, as shown in Figure 3.23:

1. **Initial Agent Observation:** The algorithm starts by capturing the agent's initial view of the environment. This provides a baseline map showing explored, unexplored, and obstacle regions, which serves as the starting point for further analysis.
2. **Generation of the ROI Gradient Map:** A gradient map is generated, emphasizing proximity and importance of ROIs. This map directs the algorithm's focus to high-priority regions for exploration.
3. **Clustering:** The free space is divided into clusters using the k-means algorithm. The number of clusters is dynamically adjusted to balance computational cost and resolution. This step reduces the number of candidate positions while ensuring comprehensive coverage of the search space.
4. **Field of View Simulation:** For each cluster centroid, the algorithm simulates the agent's possible field of view using ray tracing. This identifies the cells that would be visible from a candidate position, considering occlusions caused by obstacles and sensor range limitations.
5. **Evaluation of Visible Cells:** The visible unexplored cells are evaluated using their value in the ROI gradient map.
6. **Reclustering:** The top-performing clusters are selected, and the search space is refined by re-clustering within these regions. The process dynamically reduces cluster size, increasing resolution while focusing on promising areas. Steps 4 and 5 are repeated for the refined clusters.
7. **Sampling Orientations:** After selecting the best position, the algorithm samples 12 evenly spaced orientations (30° intervals). This ensures a comprehensive initial assessment of potential directions.

8. **Initial Orientation Selection:** The sampled orientations are evaluated using the ROI gradient map. The two best-performing orientations are retained for further refinement. The orientations are evaluated using the ROI gradient map and sensor's FoV simulation, as shown in equation 3.19
9. **Resampling Orientations at $\pm 10^\circ$:** Additional FoV simulations are conducted for the two best orientations at $\pm 10^\circ$ and the original angle. This step increases resolution in the most promising directions.
10. **Second Orientation Refinement at $\pm 5^\circ$:** From the refined orientations, the two best are resampled at $\pm 5^\circ$, providing a fine-grained evaluation of top directions.
11. **Selection of the NBV:** The orientation with the highest combined score is selected as the optimal direction for the sensor. The scoring function incorporates both quantity and quality of information gained (Equation ??).
12. **Agent Movement and Exploration:** The agent moves to the selected Next Best View (NBV) position and explores the new environment. The updated map reflects newly visible areas, and the process restarts from Step 2.

Summary

This workflow combines position and orientation selection hierarchically and iteratively, ensuring efficient exploration. By refining the search space and optimizing sensor orientations, the algorithm balances computational cost, resolution, and informational gain.

The accompanying visual representation through 12 images depicts the process, Figure 3.23:

- Image 1: Initial agent observation.
- Image 2: ROI gradient map generation.
- Image 3: Clustering process.
- Image 4: FoV simulation example.
- Image 5: Visible cells evaluation using the gradient map.
- Image 6: Reclustering process.
- Image 7: Sampling of 12 orientations.
- Image 8: Selection of the best orientations.
- Image 9: Resampling at $\pm 10^\circ$.
- Image 10: Second resampling at $\pm 5^\circ$.
- Image 11: Selection of the NBV.
- Image 12: Agent movement and exploration.

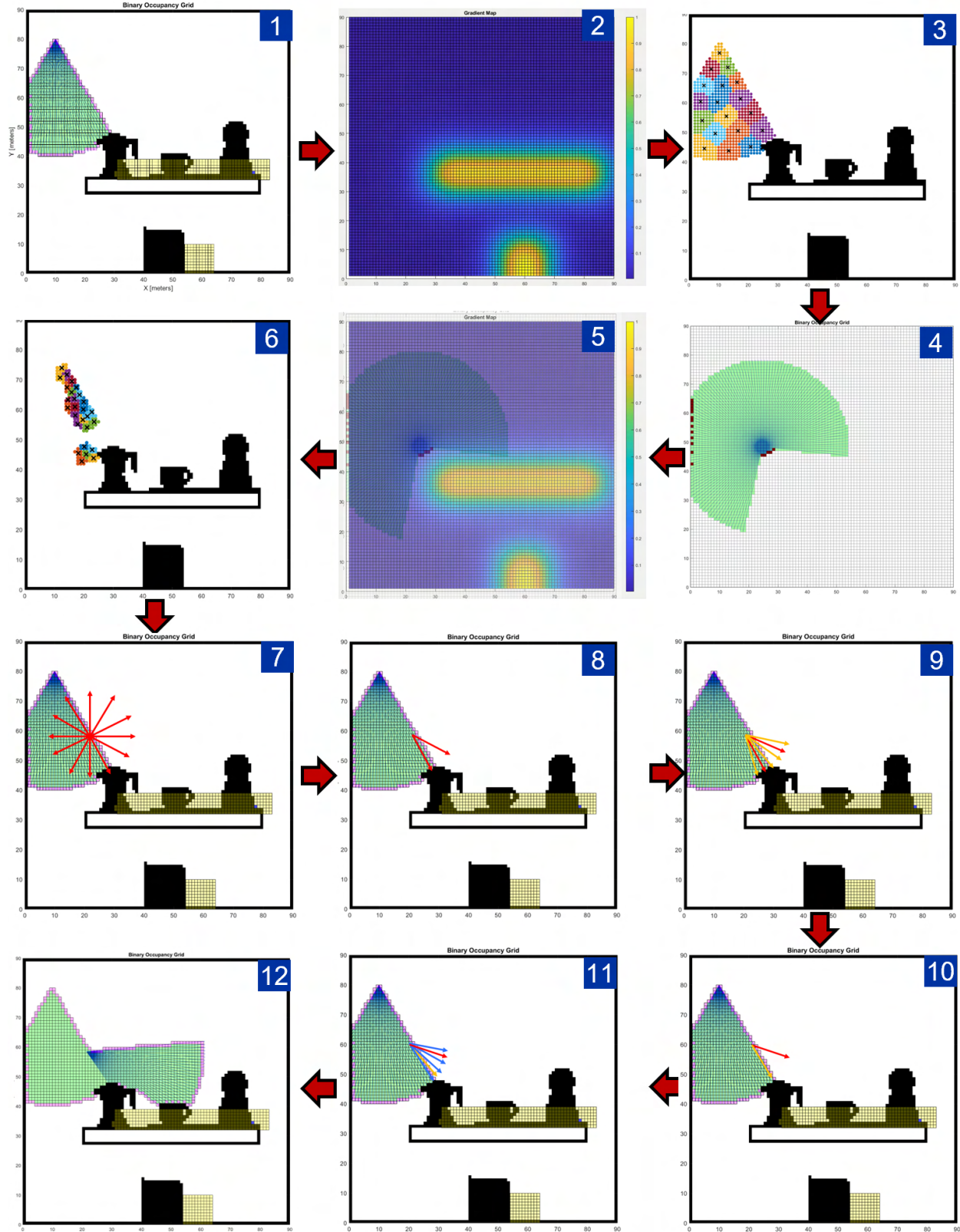


Figure 3.23: Algorithm Workflow Scheme

Advantages of the Enhanced Approach

By incorporating FoV simulations and expanding the candidate search space, the enhanced algorithm achieves the following:

- **Maximized Information Gain:** Evaluating the field of view at each candidate position ensures optimal visibility and prioritization of high-value regions like ROIs.
- **Expanded Search Flexibility:** Considering all free cells, rather than restricting to frontier cells, increases adaptability while maintaining computational efficiency through k-means sampling.
- **Balanced Decision-Making:** The refined cost function integrates information gain and displacement costs, enabling a harmonious balance between thorough exploration and goal-directed movement.

3.3.6 Human Decision Process

To evaluate the effectiveness of the proposed algorithm, the methods outlined in Section 3.3.2 were employed. However, a lack of established benchmarks in the domain made it challenging to determine the algorithm’s performance in absolute terms. To address this, we devised an innovative approach: comparing the algorithm’s decision-making capabilities against those of humans. This comparison provided a practical and intuitive benchmark to assess the algorithm’s intelligence and efficiency.

Design of the Human Experiment A custom-made video game was developed in MATLAB to facilitate the comparison. The game allowed human players to select the best next position and orientation for exploring a map and locating a hidden object. The gameplay mirrored the algorithm’s task, presenting players with the same information available to the algorithm, such as explored regions, obstacles detected, and the ROIs.

The game was designed to be simple, engaging, and competitive, ensuring participant motivation and attention. To make the experience more enjoyable, a playful narrative was introduced: participants were tasked with recovering the stolen cake for CERN’s 70th anniversary by finding clues hidden across various gridmaps. The narrative added an element of fun and purpose, while competition fostered enthusiasm and a drive for optimal performance. Players competed against each other to achieve the best scores and, intriguingly, against the "evil machine"—our algorithm.

Incorporating a narrative and competitive elements offered several psychological benefits:

- **Increased Engagement:** A storyline and gamified competition captured player’s interest, encouraging sustained participation and focused decision-making.
- **Enhanced Performance:** The desire to outperform others and the algorithm motivated participants to strategize effectively and explore the maps with care.
- **Stress Reduction:** Framing the experiment as a game helped reduce anxiety and cognitive overload, enabling participants to approach the task in a relaxed and enjoyable manner.

Participants were instructed to optimize their gameplay based on three key metrics:

- Distance Travelled: Minimizing the total distance covered during the exploration.
- Number of Iterations: Reducing the number of movements and orientation changes required to find the target.
- Angular Distance Travelled: Limiting the total angular adjustments made during the search process.

Importantly, participants were not timed during their decision-making process. This ensured that players could focus on optimizing their performance without the pressure of a ticking clock, allowing for a more thoughtful and deliberate exploration strategy, the game interface and Logo is shown in Figure 3.24.

Summing up, this human-comparison framework offered a novel and insightful way to evaluate the algorithm's intelligence and efficiency. By analyzing how human players approached the same task, we gained valuable perspectives on the algorithm's strengths, limitations, and potential for further improvement. Moreover, this approach could serve as the foundation for developing standardized benchmarks, offering a novel way to evaluate and compare exploration algorithms in a manner that bridges human problem-solving and machine intelligence.

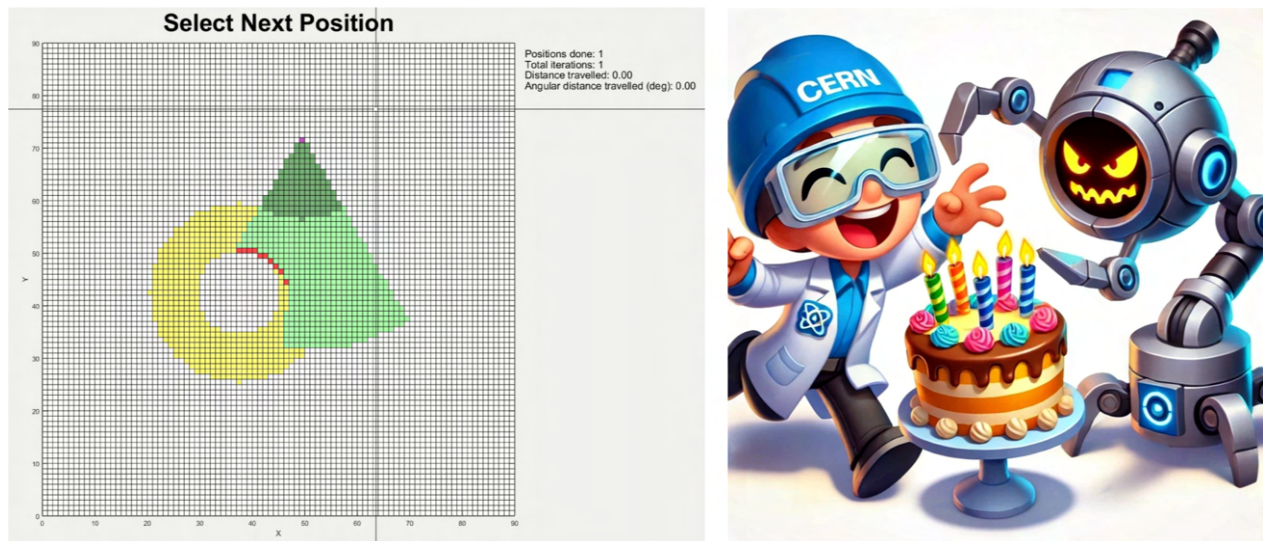


Figure 3.24: Human Decision Game Interface and Logo

3.4 HOPE-OS 3D Simulation

The transition to 3D object searching builds upon the principles and strategies outlined in the Advanced 2D FoV Simulated Gradient Algorithm, Section 3.3.5, adapted for the complexities of a voxel-based 3D environment. The enhanced methodology involves a spherical FoV simulation for position evaluation and a frustum-shaped FoV for orientation selection.

3.4.1 3D Simulated Environment

The voxel map serves as the 3D equivalent of the 2D grid map described in Section 3.3.1, functioning under similar principles but adapted for three-dimensional environments. Each voxel represents a discrete unit of space in the environment, storing information about its state: unexplored, free, occupied, or a region of interest. This structure enables efficient representation and management of spatial data, allowing the algorithm to track the agent's progress and update the map dynamically. Like the grid map, the voxel map facilitates the integration of gradient values for prioritizing exploration, leveraging the same strategies to identify optimal positions and orientations.

Figure 3.25 illustrates the 3D voxel map used for simulation, representing a section of the LHC tunnel at CERN. The voxel map captures the environment in discrete units, with each voxel color-coded to convey its state: green for free space and red for occupied space. The region of interest is prominently marked around the pipe, reflecting the areas where the object is most likely to be found. The initial FoV of the sensor is clearly visible, demonstrating the agent's current visibility constraints and the detected regions of the map. Notably, the target object is highlighted within a red circle, indicating its precise location.

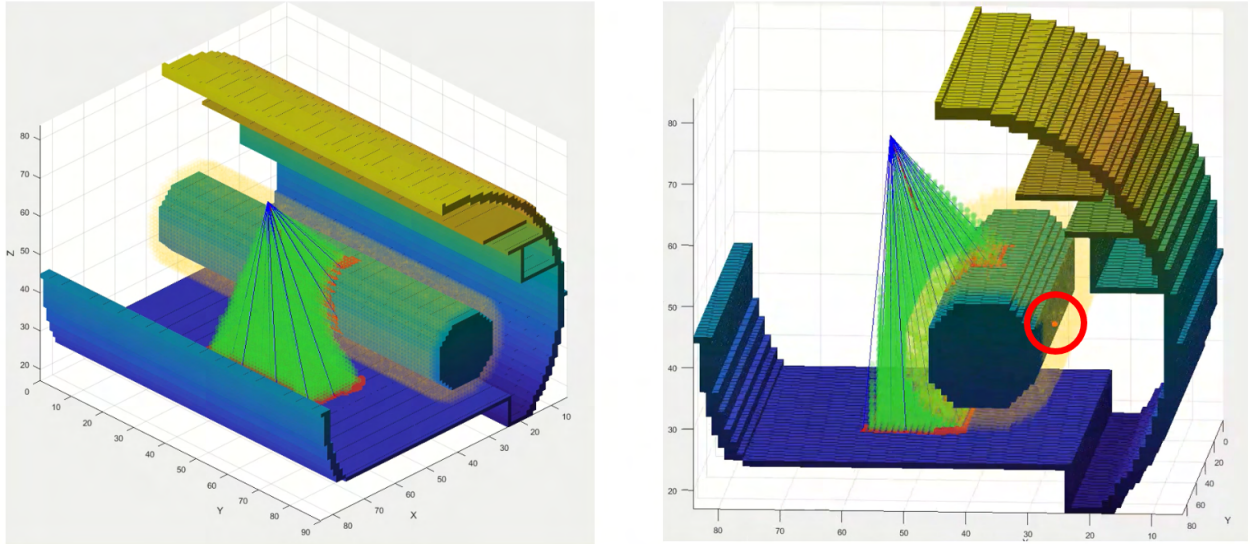


Figure 3.25: 3D Simulated Environment

To ensure the simulated environment closely resembles a real-world scenario, the voxel map is scaled to represent actual physical dimensions. Each voxel corresponds to a fixed volume in the real world, for example, with an edge length of 5 cm. This transformation from voxel units to real-world measurements ensures that cost functions, gradient computations, and other calculations are expressed in standard international units rather than abstract voxel units. This approach not only enhances the realism of the simulation but also makes the algorithm adaptable to different map resolutions and varying environmental scales.

3.4.2 Algorithm Workflow

The exploration process is very similar to the one explained in Section 3.3.5, it begins with the agent's observation of the environment, where the initial voxel states are updated based on the sensor's field of view, Figure 3.25. After this observation, a 3D ROI gradient map is generated, similar to the approach in the 2D gridmap described in Section 3.3.5, but adapted for the voxel map structure.

In this 3D approach, an additional obstacle gradient map is incorporated into the cost function. This obstacle map acts as a safety measure, discouraging the selection of positions that are too close to detected obstacles. The gradient values for both the ROI and obstacle maps are calculated based on the following principles:

1. ROI and Obstacles Gradient Map Generation

The ROI gradient map is designed to guide the agent toward regions of interest. The gradient values are computed using a combination of terms to provide both smooth transitions and sharp prioritization near ROI cells:

$$g(d) = \frac{1}{d^2 + q} \cdot q \quad (3.20)$$

$$f(d) = \frac{1}{d^2 + h} \cdot h \quad (3.21)$$

where:

- d is the distance to the ROI.
- q and h are constants that control the shape and smoothness of the respective gradient functions.
- $g(d)$ is a gradient function that emphasizes the influence of nearby ROI cells with low scaling factor q .
- $f(d)$ is a complementary gradient function with a high scaling factor h , providing an additional layer of smoothness.

This two functions are combined to calculate the ROI gradient map.

$$G_{\text{ROI}} = \frac{w \cdot g(d) + f(d)}{w + 1} \quad (3.22)$$

where:

- w is the ponderation factor, when w is increased, it gives more importance to $g(d)$ function, emphasizing the influence of nearby ROI cells.
- G_{ROI} represents the combined ROI gradient value, balancing the contributions of $g(d)$ and $f(d)$ based on distance.

For better comprehension, these functions are plotted in Figure 3.26, using the following values: $q=0.002$, $h=1$ and $w=4$. The resultant gradient map can be observed in Figure 3.27 left.

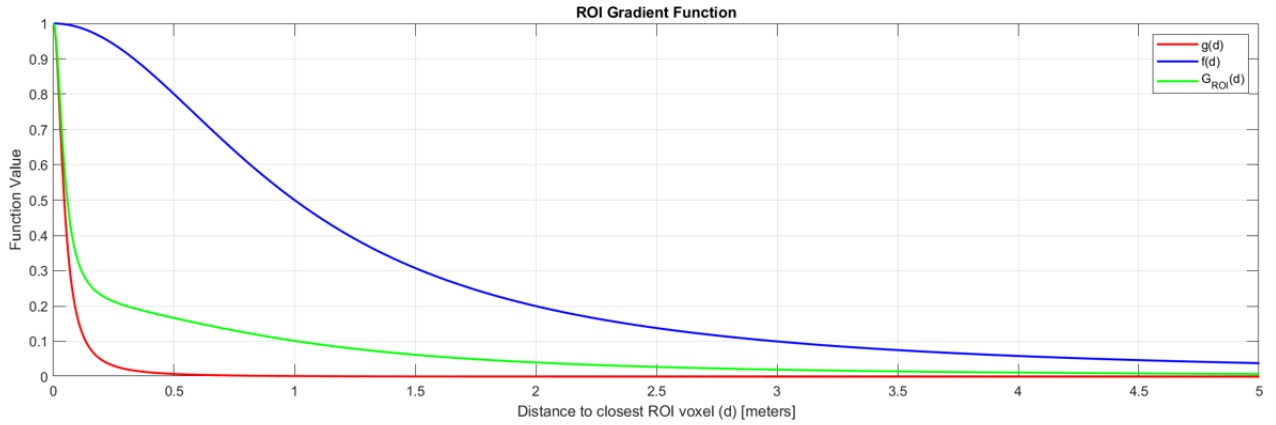


Figure 3.26: 3D ROI Gradient function

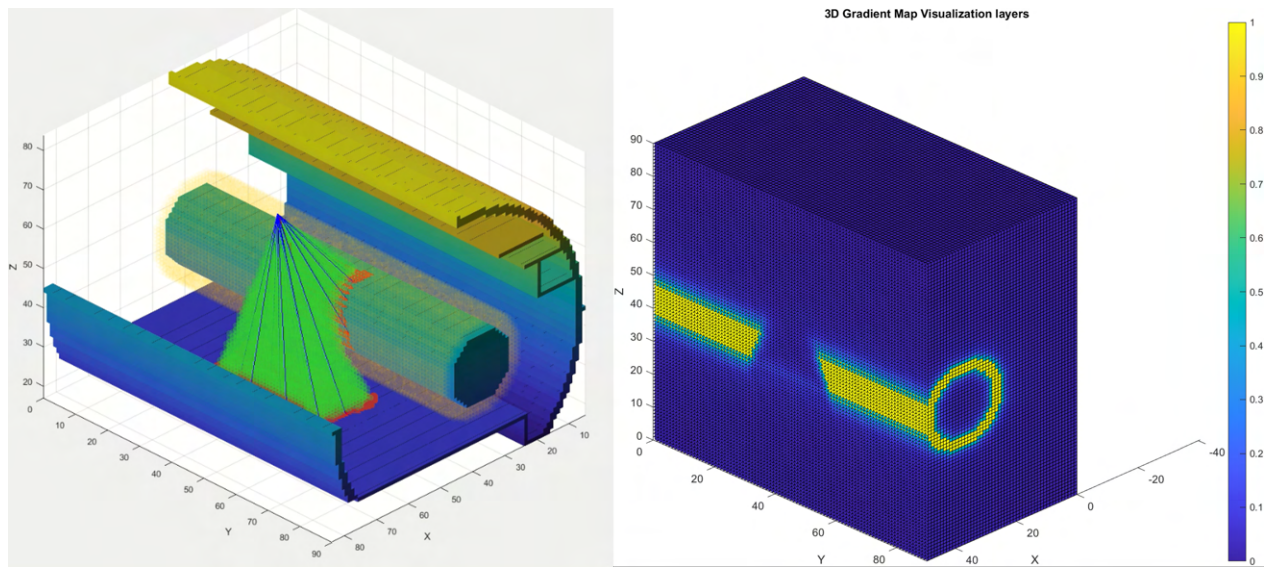


Figure 3.27: ROI Gradient 3D visual representation

Obstacle Gradient Map To discourage movement near obstacles, the obstacle gradient is computed as:

$$G_{\text{Obstacle}} = \frac{1}{1 + e^{k \cdot (d - d_{\text{threshold}})}} \quad (3.23)$$

where:

- d is the distance to the ROI or obstacle cell.
- k is a scaling factor that determines the sharpness of the gradient.
- $d_{\text{threshold}}$ defines the range of the security distance from obstacles.

For better comprehension, this function is plotted in Figure 3.28, using the following values: $k_1=0.25$ and $k_2=40$. The resultant gradient map can be observed in Figure 3.29 left.

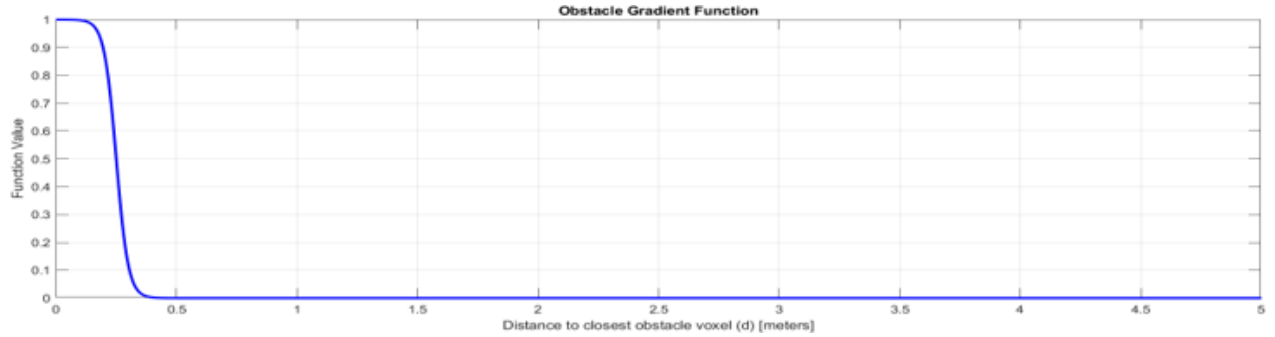


Figure 3.28: 3D Obstacles Gradient function

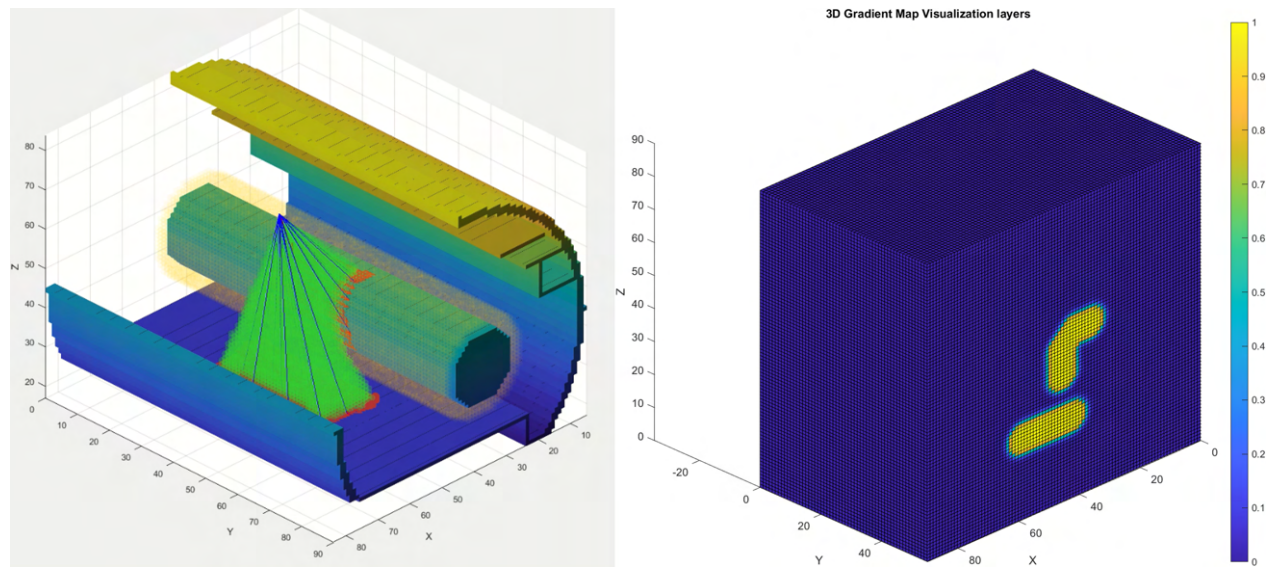


Figure 3.29: Obstacles Gradient 3D visual representation

2. Evaluating Positions with FoV Simulation and Cost Function

Once the ROI and obstacles gradient map is constructed using the current map information, the next step involves identifying candidate positions for exploration. This process begins with clustering and reclustering the free space, using k-means algorithm, as detailed in the earlier Section 3.3.5. The clustering process narrows down the potential search space to the most promising regions, ensuring computational efficiency and focus on high-priority areas. Visual examples of the clustering and reclustering stages can be observed in Figure 3.30.

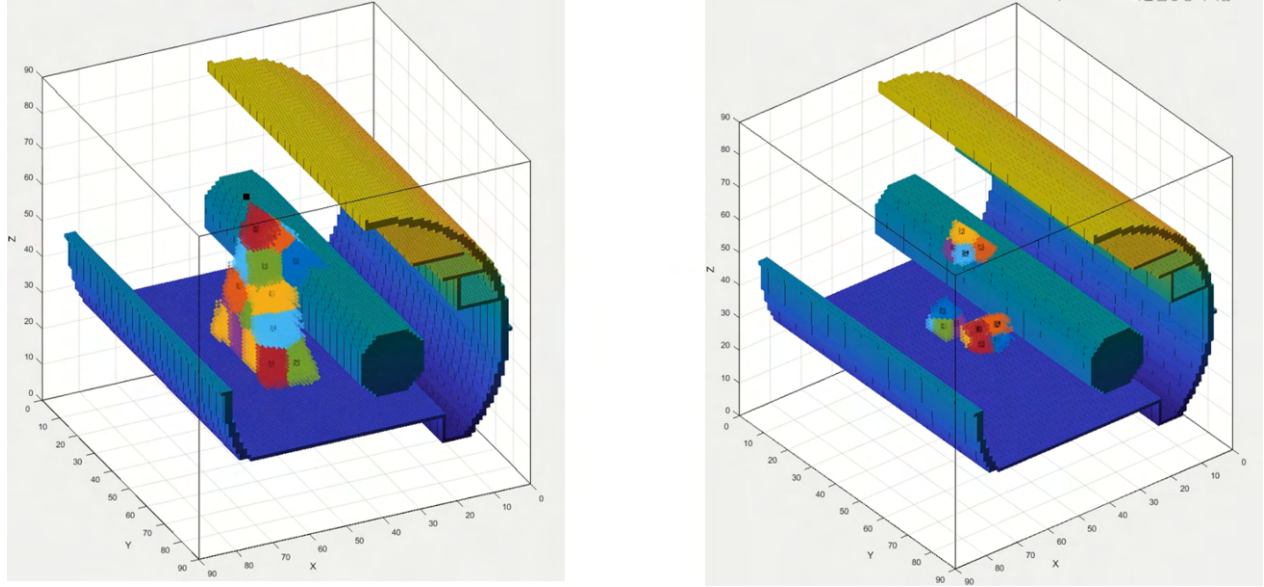


Figure 3.30: Clustering and reclustering in 3D

The evaluation of the positions obtained after the clustering process is very similar to the one explained in Section 3.3.5 and is done as follows: For each candidate position, a spherical FoV is simulated, representing the sensor's ability to observe its surroundings. In Figure 3.31 there is a visual example of how the spherical FoV is simulated using ray tracing. The FoV is limited by the sensor's range and considers occlusions caused by obstacles, leveraging the current occupancy information. This simulation identifies which cells would be visible from the candidate position, providing the basis for evaluating the position's informational value.

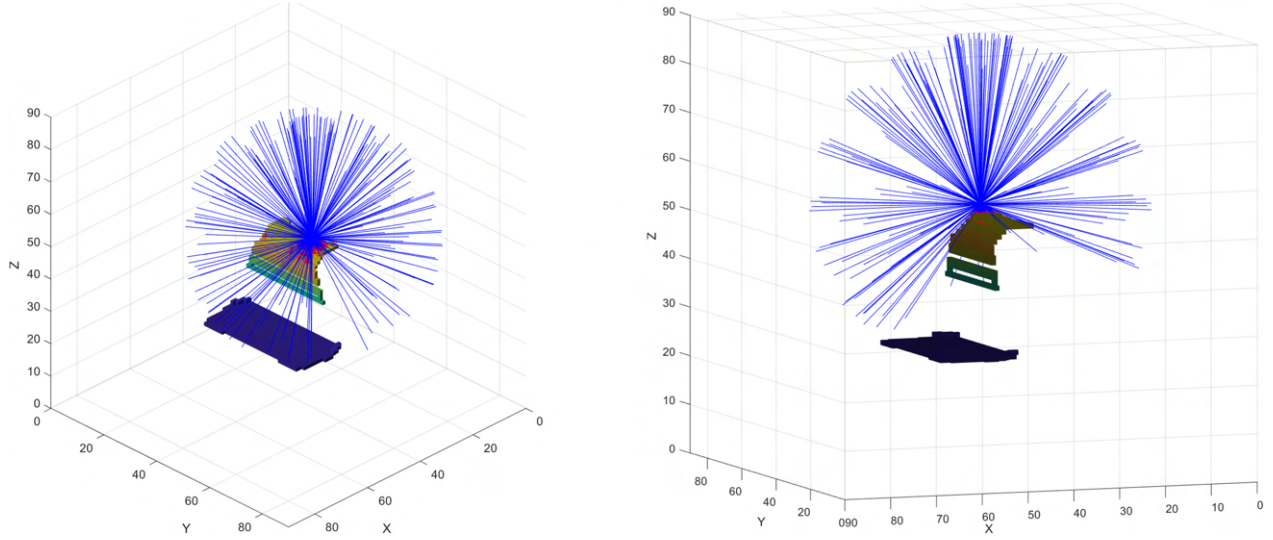


Figure 3.31: Spherical FoV 3D simulation

After obtaining which voxels could be visible from a certain position, we exclude voxels that have already been explored, focusing solely on unexplored areas and regions of interest. Once the visible unexplored cells are identified, their values in the ROI gradient map are used to evaluate the quality of the candidate position, using the same process as the one explained in Section 3.3.5, total score and average score. These metrics complement each other, with the total score emphasizing the quantity of information gain and the average score identifying high-priority regions within the visible area.

As a safety measure, the obstacle gradient map is incorporated into the evaluation function. This map penalizes candidate positions near detected obstacles by discouraging selections too close to them.

Finally, the displacement cost (DispCost) is designed to penalize candidate positions based on their distance from the agent's current location. This ensures that the algorithm considers movement efficiency when selecting the next position. The displacement cost increases as the distance increases, discouraging the selection of distant positions unless they offer significantly higher informational value. It is calculated using the following equation:

$$\text{DispCost} = -\left(\frac{1}{d^2 + \epsilon} \cdot \epsilon\right) + 1 \quad (3.24)$$

where:

- d is the Euclidean distance between the current agent position and the candidate position.
- ϵ is a small constant that adjusts the slope of the penalty function, ensuring smoother transitions.

For better comprehension, this function is plotted in Figure 3.32, using the following value: $\epsilon=0.05$.

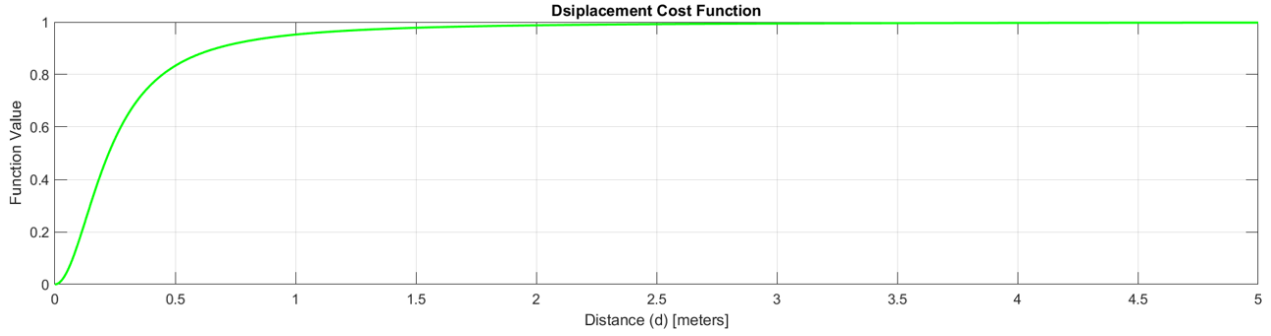


Figure 3.32: Displacement Cost Function

The final cost function to evaluate candidate positions combines multiple components to ensure a balanced and efficient exploration strategy. It is expressed as:

$$\text{Score} = (\alpha \cdot \text{ROI Gradient} - \beta \cdot \text{Displacement Cost} - \gamma \cdot \text{Obstacle Proximity Cost}) \quad (3.25)$$

where:

- **ROI Gradient:** A weighted combination of the total and average scores derived from the ROI gradient map, representing the quantity and quality of potential information gain from a candidate position.
- **Displacement Cost:** Penalizes positions farther from the agent's current location. It is smoothed using a sigmoid function to balance the cost of movement with exploration efficiency.
- **Obstacle Proximity Cost:** Discourages selecting positions close to detected obstacles, ensuring safer navigation. It is derived from the obstacle gradient map.
- α, β, γ : Weighting factors that determine the relative importance of the ROI gradient, displacement cost, and obstacle proximity cost in the final score.

This cost function ensures that the agent prioritizes positions offering high informational value, minimizes unnecessary movement, and avoids unsafe proximity to obstacles.

3. 3D Orientation selection

After selecting the next best position, it is necessary to choose the best orientation for that position. In the 3D adaptation of the orientation selection process explained in Section 3.3.5, the sensor's field of view is modeled as a frustum with a limited range and angular coverage, representing the real-world capabilities of the sensor. The goal is to optimize the sensor's orientation to maximize the quantity and quality of information captured from the environment, while efficiently considering computational constraints.

The sensor's orientation is determined by sampling a range of possible angles, simulating the FoV for each, and evaluating their informational value based on the ROI gradient map.

Unlike previous methods, this updated process can account for multiple optimal orientations, optimizing the robot's movement by taking into consideration more than one orientation per position.

The orientation selection process is done as follows:

1. **Initial Orientations Sampling:** A set of initial orientations is sampled evenly around the agent's current position. For example, 26 orientations may be sampled, spaced 45° apart in horizontal plane and 45° in the vertical plane, to provide comprehensive initial coverage of the possible viewing directions, 3.33.

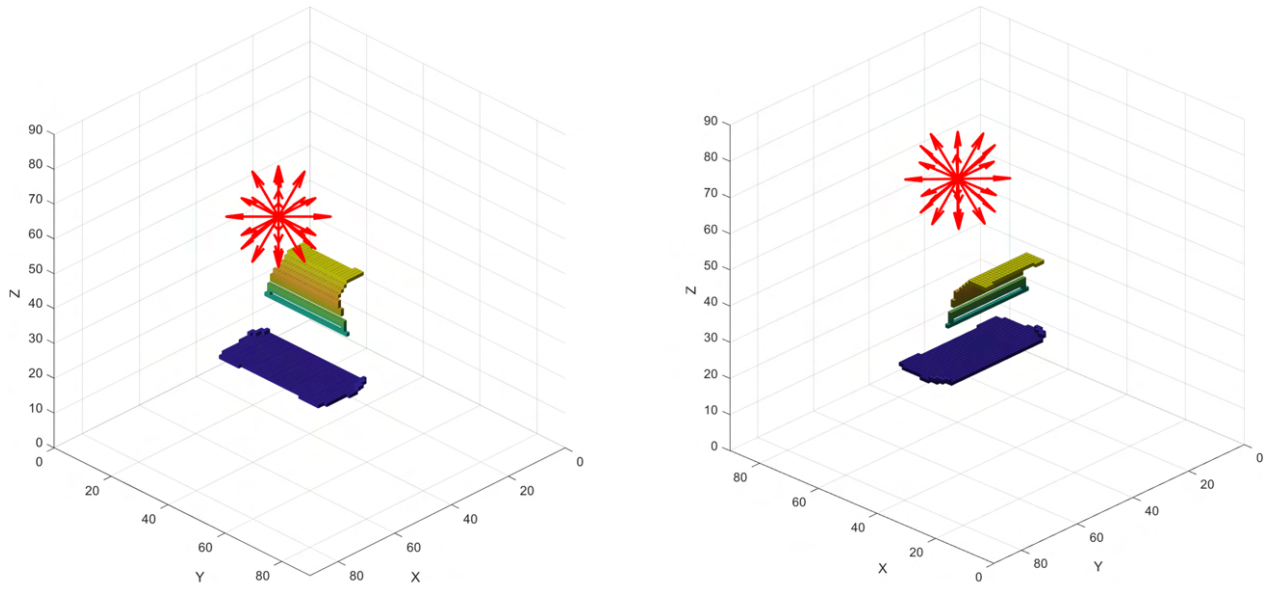


Figure 3.33: 3D Orientations Sampling

2. **Field of View Simulation:** For each sampled orientation, a 3D FoV is simulated using ray tracing. This simulation determines the cells within the FoV that are visible from the sensor's current position, considering the occupancy map's current state, including obstacle occlusions and sensor range limits. In Figure 3.34 we can observe how the sensor's frustum is simulated in order to obtain the possible visible voxels.
3. **Evaluation of the Orientations:** Each orientation is evaluated based on the value of the visible voxels in the ROI gradient map, same process as the one explained in Section 3.3.5.

After calculating scores for all sampled orientations, the algorithm identifies orientations with scores above a certain threshold (e.g., 80% of the maximum score). This step ensures that only highly informative orientations are considered for further refinement.

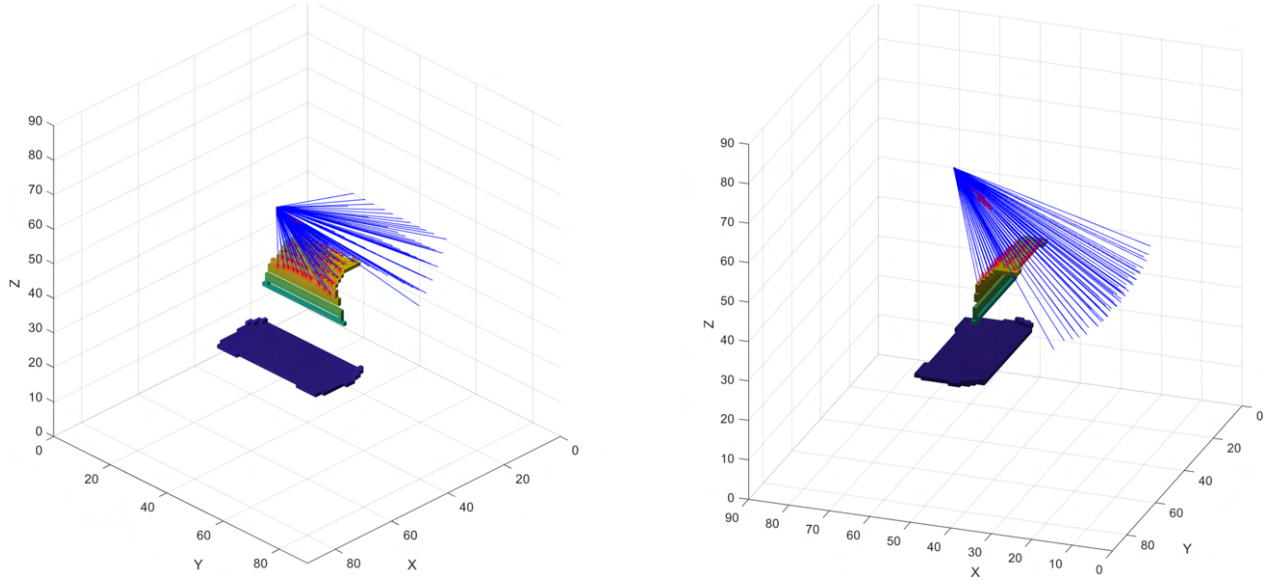


Figure 3.34: Sensor's Frustum Simulation

4. Orientation Resampling: For the selected orientations, the ones which have a score above the threshold, additional refinement is performed. Orientations are resampled at smaller angular intervals (e.g., $\pm 15^\circ$ around the best-performing orientations), as show in Figure 3.35 where two initial orientations were resampled. This hierarchical refinement ensures high-resolution evaluation of promising directions without unnecessary computational cost.

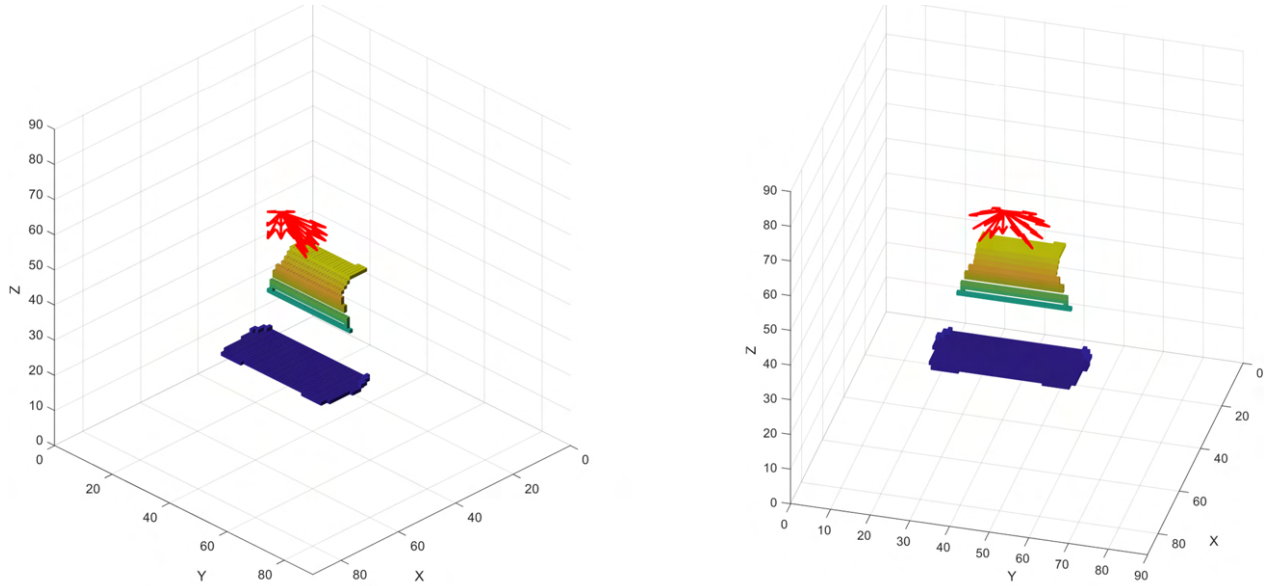


Figure 3.35: Sensor's Frustum Orientations Resampling

5. Selection of Optimal Orientations: The final refined orientations are selected based on

their scores and the threshold, prioritizing those with the highest weighted gradient values.

6. **Reordering for Efficiency:** The selected orientations are reordered to minimize angular distance between successive orientations. This step reduces unnecessary sensor rotation, optimizing exploration efficiency.

3.4.3 Summary of the HOPE-OS 3D Exploration Workflow

To summarize everything explained in last Section 3.3.5, the exploration process can be condensed into the following steps, which form a continuous loop. Each step contributes to optimizing the agent’s movement and sensor orientation for efficient search process. Below is a schematic representation of the workflow:

1. ROI and Obstacles Gradient Map Generation:

- Generate a gradient map to guide the agent toward regions of interest while penalizing positions near detected obstacles for safety reasons.
- The gradient values are computed based on inverse distance functions.

2. Evaluating Positions with FoV Simulation and Cost Function:

- Sample candidate positions through clustering and reclustering to optimize computational efficiency using k-means.
- Simulate the sensor’s spherical FoV for each candidate position using ray tracing, identifying visible unexplored cells and considering occlusions caused by obstacles.
- Evaluate candidate positions using a cost function 3.25

3. 3D Orientation Selection:

- Sampling and resampling initial orientations across a 360° spherical range, simulating the sensor’s frustum-shaped FoV for each orientation.
- Evaluate orientations based on the visible ROI gradient values, focusing on both the total and quality of information.
- Retain multiple high-performing orientations based on a threshold to maximize coverage.

This loop repeats continuously as the agent explores the environment, dynamically updating the voxel map and recalculating gradients, positions, and orientations. The iterative nature ensures that the exploration remains adaptive and efficient, balancing information gain, safety, and computational cost.

3.5 Real World Implementation

The transition from simulation to real-world implementation requires careful consideration of both the hardware and software components to ensure seamless integration. This section outlines the methods used to adapt the exploration algorithm to the physical robot system.

3.5.1 Frame Transformation

To enable seamless integration between the robot's hardware and the exploration algorithm, it was necessary to establish a proper frame transformation. The voxel map operates in the world frame, which is fixed relative to the robot's base frame. This fixed relationship ensures that any defined ROIs are accurately positioned with respect to the robot's environment. Using forward kinematics, the position of the robot's end effector is dynamically obtained during operation. Additionally, a fixed transformation exists between the end effector and the camera mounted on it, ensuring the correct alignment of the camera's data with the voxel map.

3.5.2 Octree Generation

The octree structure, implemented using the OctoMap library, was chosen for its efficiency and flexibility in representing three-dimensional occupancy data. The OctoMap framework provides several advantages, including optimized memory usage and the ability to perform ray tracing to determine free and occupied regions. This capability is crucial for updating the map with the latest observations from the robot's RGB-D sensor. Once the occupancy data is generated in the octree, it is transferred to a voxel map or occupancy matrix. This voxel map is used for all subsequent decision-making processes, including cost function evaluations. It also serves as the repository for exploration states and ROI information.

An octree is a tree data structure used to partition a three-dimensional space into smaller, manageable cubic volumes called voxels. Each node in the octree represents a cube in the 3D space, and these cubes are recursively subdivided into eight smaller cubes (or children) as needed, Figure 3.36. This hierarchical structure allows octrees to efficiently represent and query 3D occupancy data, making them particularly suitable for large-scale environments where memory efficiency and fast access to spatial data are crucial.

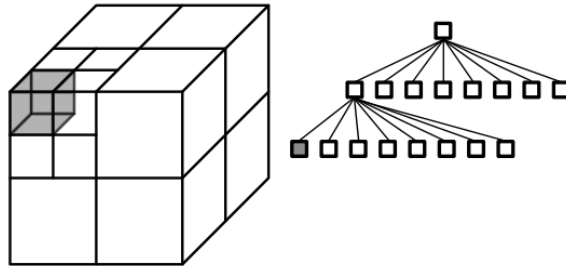


Figure 3.36: Octree visual representation

In Figure 3.37, the left image illustrates the octree representation highlighting detected obstacles. The right image shows the regions marked as free space, derived through ray tracing from the sensor’s perspective. These visualizations demonstrate how the octree structure captures both occupied and free regions.

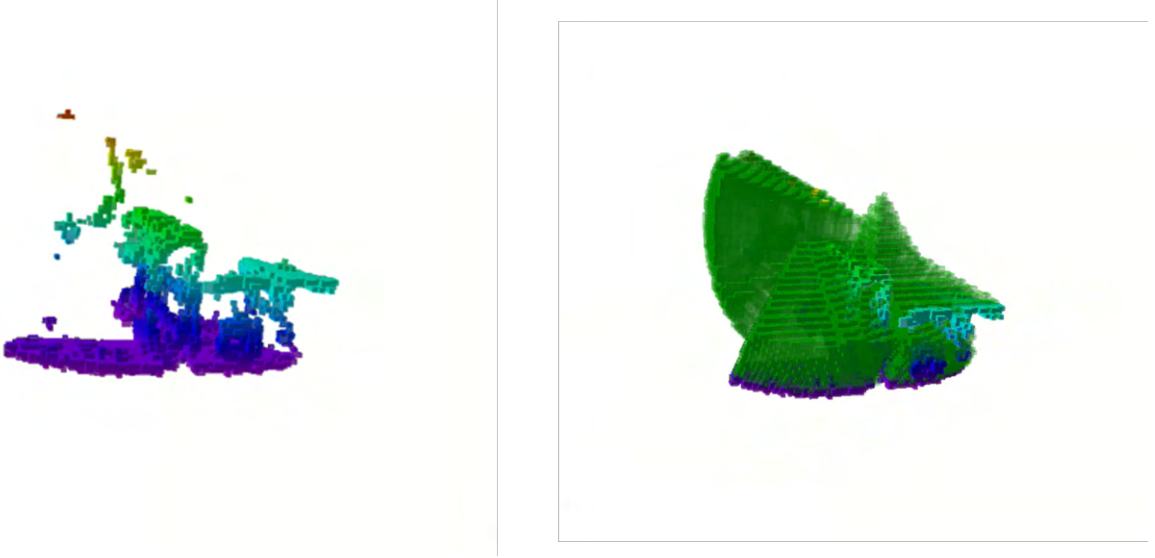


Figure 3.37: Octree generation example

3.5.3 CRF Implementation

The real-world implementation was carried out using the CERN Robotic Framework (CRF). Two primary classes were developed to handle the integration:

1. **Octomap Class:** This class is responsible for generating the octree structure and simulating the sensor’s FoVs. It handles ray tracing to determine free and occupied regions, providing the foundational spatial data required by the decision-making process.
2. **Exploration Class:** This class orchestrates the algorithm’s decision-making workflow. It integrates the data from the voxel map and the sensor’s simulated FoV to evaluate candidate positions and orientations. Based on the computed cost function, it selects the next best position and orientation for the robot.

The general implementation workflow is illustrated in Figure 3.38, where the interplay between the various components such as octree generation, exploration, path following, and CNN-based detection, Figure 3.39. The RGB-D camera provides a real-time point cloud, which is processed into the occupancy octree and subsequently guides the exploration process.

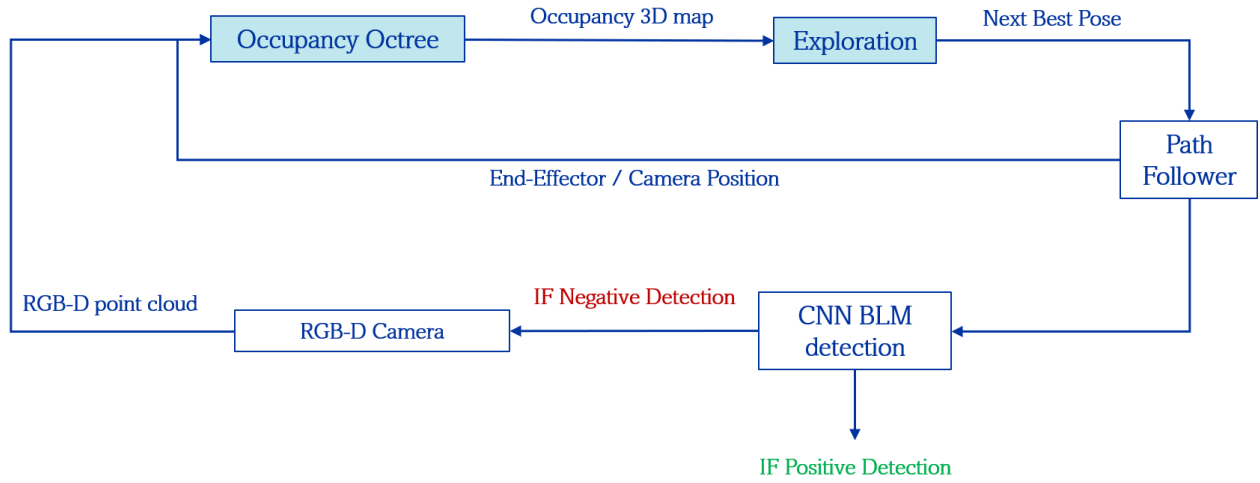


Figure 3.38: CRF Object searching implementation scheme

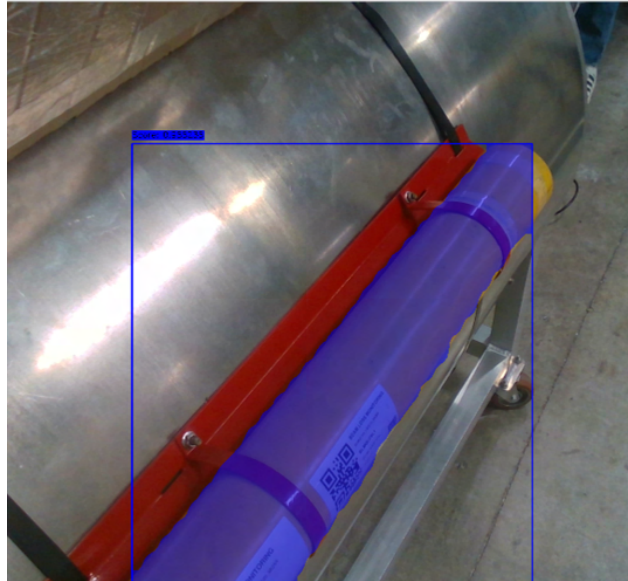


Figure 3.39: CNN detection of the BLM

Chapter 4

Results and Discussion

The results presented in this section encompass a comprehensive evaluation of the exploration algorithms developed and discussed in previous chapters. These evaluations were conducted across a range of scenarios, including both 2D and 3D simulations, as well as limited real-world experiments. While the simulations provided extensive and controlled environments to thoroughly test the algorithms, the real-world implementation was constrained by certain practical limitations, which will be discussed in detail later in this section. Despite these constraints, the results obtained offer valuable insights into the performance and potential of the proposed methods.

4.1 2D Simulations

4.1.1 Basic Algorithms

The initial experiments focused on the five basic algorithms described in Section 3.3.3. These tests were performed using more than 10,000 randomly generated 50x50 grid maps, ensuring diversity in the evaluation scenarios. To maintain fairness and repeatability, the maps were designed to be consistent across all algorithms, with identical starting positions for the agent, as well as randomly but consistently generated positions for the target object and the ROI. The simulated sensor had a range of 10 units and a FoV amplitude of 60 degrees, ensuring realistic visibility constraints during exploration. This extensive simulation dataset allowed for a robust comparison of the algorithms under controlled yet varied conditions, laying the groundwork for evaluating their strengths and weaknesses.

Figure 4.1 illustrates an example of the generated grid maps, showcasing the typical elements: free spaces, obstacles, and ROI. These maps provided the basis for evaluating the following algorithms, explained in detail in Section 3.3.3:

1. Closest Frontier Algorithm: Prioritizes the nearest frontier cell to the agent's current position, gradually expanding the explored region.
2. Furthest Frontier Algorithm: Directs the agent toward the farthest frontier cell, encouraging aggressive exploration of distant areas.

3. Information Gain Algorithm: Selects the frontier cell with the highest number of unexplored neighbors, maximizing information gain without considering movement cost.
4. Information Gain + Displacement Cost Algorithm: Balances information gain with the cost of movement, ensuring efficient exploration paths.
5. ROI-Based Algorithm: Enhances the previous approach by integrating a term that guides the agent toward predefined ROIs.

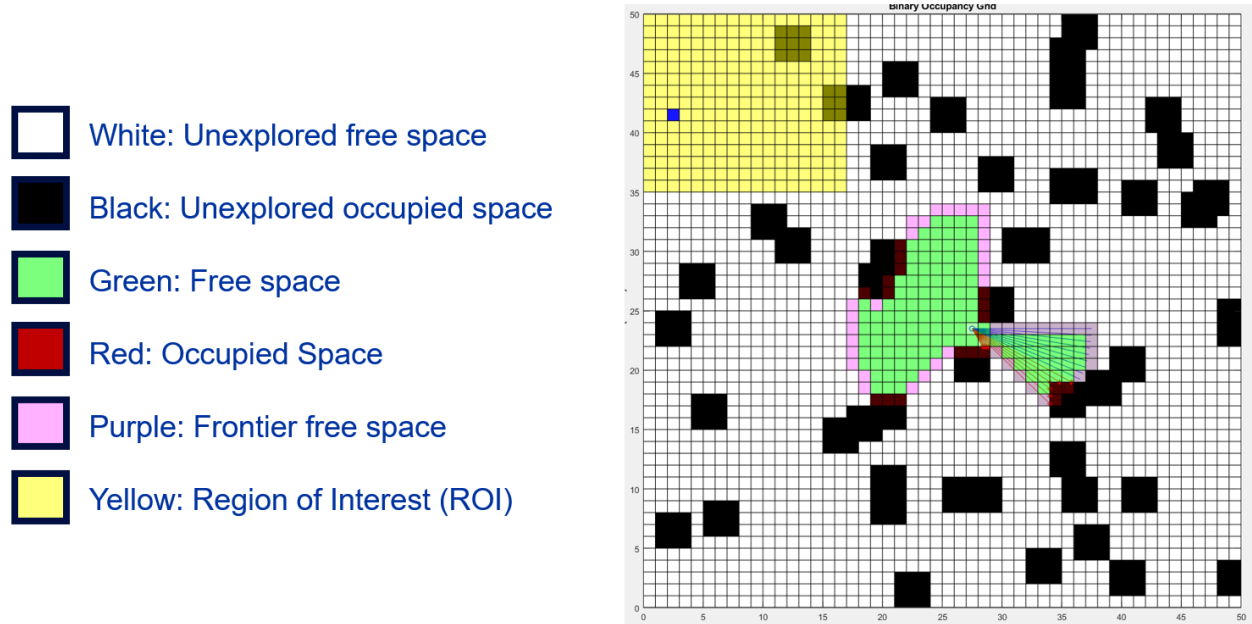


Figure 4.1: 2D 50x50 Gridmap

Following the evaluation framework outlined in Section 3.3.2, the performance of the basic algorithms was assessed using a comprehensive set of metrics. These metrics, designed to capture key aspects of the exploration process, include:

1. Map Coverage: The percentage of the total map explored during the simulation, reflecting the algorithm's ability to achieve comprehensive coverage.
2. ROI Coverage: The percentage of the ROI explored, measuring how effectively the algorithm prioritizes high-value regions.
3. Iterations: The number of positional steps required to locate the target object, indicating the efficiency of position selection.
4. Total Iterations: The combined count of positional steps and orientation adjustments, providing a holistic view of the algorithm's operational demands.
5. Distance to Target: The agent's distance to the target object over iterations, showcasing its ability to converge on the objective.
6. Total Distance Travelled: The cumulative distance the agent moves during exploration, highlighting spatial efficiency.

7. Total Angular Distance Travelled: The cumulative angular adjustments made, representing the efficiency of orientation selection.

The first comparison focuses on the total number of iterations required to solve the maps, evaluated in two ways: Figure 4.2 (top) considering only positional steps and Figure 4.2 (bottom) accounting for both positional steps and orientation adjustments. This analysis provides a general overview of how efficiently each algorithm selects positions and orientations, highlighting their ability to minimize the number of decisions needed to locate the target. Algorithms that require fewer iterations demonstrate superior decision-making, optimizing both movement and exploration strategies.

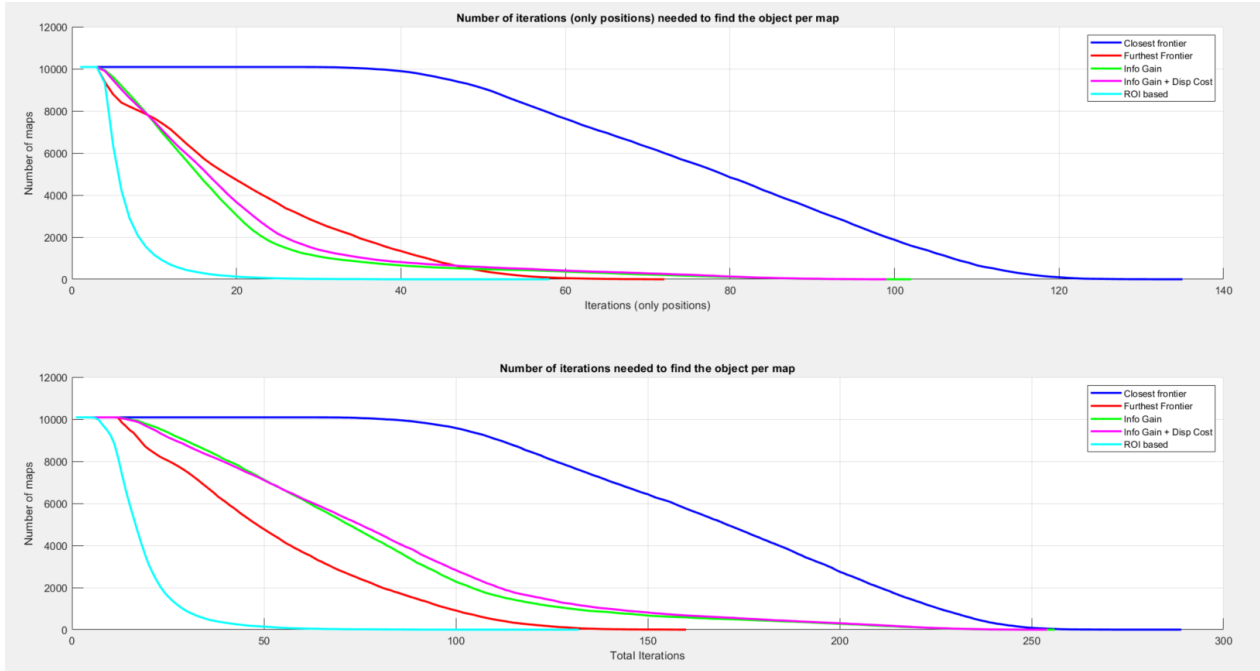


Figure 4.2: Iterations required to solve the maps: (Top) positions only, (Bottom) positions and orientations.

When analyzing both plots together, we will focus on top plot, because is the most relevant for determining the quality of the positions selected by each algorithm. This metric directly reflects how efficiently each algorithm guides the agent toward the target object.

As expected, the ROI-Based Algorithm (cyan) outperforms all others, demonstrating its capability to prioritize high-value regions effectively. The Information Gain Algorithm (red) slightly outperforms the Information Gain + Displacement Cost Algorithm (magenta), likely because the displacement cost consideration leads to the selection of intermediate positions, which can increase the number of necessary steps. Surprisingly, the Furthest Frontier Algorithm (green) performs relatively well in these scenarios, highlighting its ability to explore large areas efficiently in this specific map type. Lastly, the Closest Frontier Algorithm (blue) performs the worst, as expected, due to its overly conservative approach that limits rapid exploration and forces excessive positional steps.

Overall, the ROI-Based Algorithm stands out as the most efficient, while the positional inefficiencies of the Closest Frontier Algorithm underscore its limitations in this context.

2D Basic Algorithms Behavior

To evaluate the performance and behavior of the algorithms and gain a deeper understanding of their functionality, the following graphs were used 4.3.

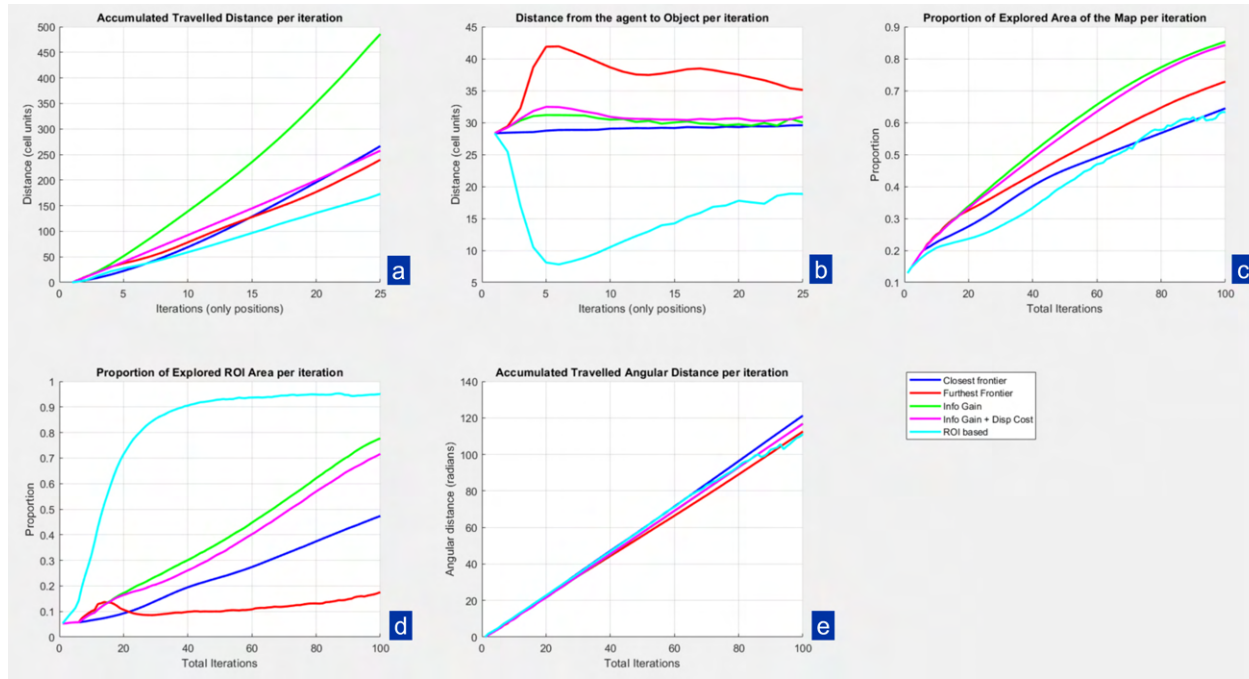


Figure 4.3: Behavior plots of 2D algorithms

- (a) Accumulated Traveled Distance per Iteration: This graph measures the accumulated distance traveled by the agent over iterations, focusing on spatial efficiency. It highlights the average cumulative movement required by each algorithm to locate the target object.
- (b) Distance from Agent to Object per Iteration: This plot tracks how effectively each algorithm directs the agent toward the target object by monitoring the distance between them over positional iterations.
- (c) Proportion of Explored Map Area per Iteration: This graph shows the cumulative percentage of the map explored by the agent, illustrating the exploration efficiency of each algorithm.
- (d) Proportion of Explored ROI Area per Iteration: This plot focuses on the coverage of high-priority Regions of Interest, showing how well the algorithms prioritize these critical areas.
- (e) Accumulated Traveled Angular Distance per Iteration: This graph tracks the cumulated angular adjustments made by the agent, providing insights into the efficiency of orientation adjustments during the exploration process.

These metrics collectively allow for a comprehensive evaluation of the algorithms, providing quantitative measures of exploration efficiency, information prioritization, and movement cost.

The performance of the algorithms varies significantly across the evaluated metrics, providing key insights into their behavior:

- **Furthest Frontier and Closest Frontier Algorithms:** These algorithms exhibit the poorest performance. They explore the least amount of the map and the ROI per iteration, highlighting their inefficiency. Closest Frontier’s conservative approach leads to slow exploration, while Furthest Frontier’s focus on distant areas results in excessive travel without effectively prioritizing high-value regions.
- **Information Gain and Information Gain + Displacement Cost Algorithms:** Both perform similarly, with Information Gain slightly better at exploring both the map and the ROI. However, Information Gain is less efficient in terms of distance traveled, as seen in plot (a), where it requires significantly more movement to locate the object. Adding Displacement Cost improves this by balancing exploration with movement efficiency, although ROI exploration is slightly reduced.
- **ROI-Based Algorithm:** This algorithm stands out as the most efficient. It minimizes the distance traveled between iterations, initially focusing on getting closer to the object and subsequently exploring the ROI. While it explores less of the overall map per iteration compared to others, it quickly and effectively prioritizes the ROI, making it ideal for targeted exploration.

It is important to note that the x-axis in the plots is truncated to focus on iterations where a statistically significant number of simulations are represented. This approach not only ensures reliable data but also enables us to study the behavior of the algorithms throughout the iterative process. Maps requiring higher iterations are rare, and their small sample size makes the data unreliable for meaningful conclusions. Therefore, the evaluation is limited to the iteration range where statistical significance is ensured, allowing for a robust and accurate understanding of the algorithms’ performance over time.

2D Basic Algorithms Comparison

In order to compare the performance of the algorithms effectively, a robust approach is to analyze their final outcomes through boxplots. These plots encapsulate critical metrics such as the total distance traveled, total iterations, proportion of explored map, and ROI coverage required to locate the object. By summarizing these values into statistical distributions, box-plots provide a clear and concise way to evaluate the efficiency and effectiveness of each algorithm, highlighting differences in performance while identifying potential outliers and variations.

Unlike the previous plots, which focused on understanding the behavior of the algorithms, these box-plots summarize key statistics that directly determine the effectiveness of the algorithms, Figure ??.

Each box-plot shows the distribution of values, with the red line representing the median and the box encapsulating the inter, quartile range (IQR). The whiskers indicate the range of

the majority of the data, while the red crosses represent outliers—data points that deviate significantly from the rest of the observations. These outliers may result from specific edge cases in the simulation, highlighting extreme or uncommon scenarios.

To ensure the reliability of these comparisons, statistical significance plays a vital role. A convergence test is performed, plotting how key statistics such as the mean, median, and inter-quartile range vary as more simulations are conducted on different maps. If these metrics stabilize with increasing simulations, it confirms their statistical significance and validates the use of the box-plot data for algorithm comparison. An example of this convergence test is shown in Figure 4.4.

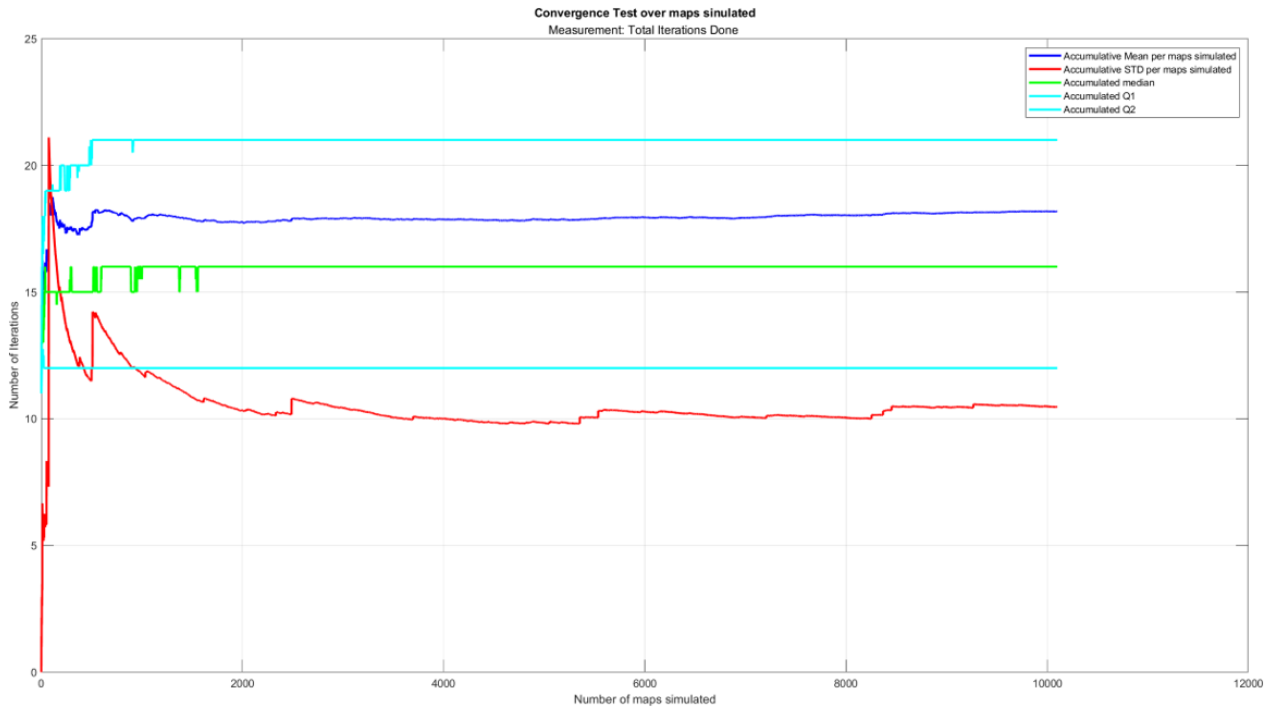


Figure 4.4: Convergency Test

Here's a brief explanation of each box-plot, Figure 4.5:

- (a) Total Travelled Distance per Map: It represents the total distance traveled by each algorithm to find the object in every map. It reflects the spatial efficiency of the algorithms, with lower values indicating more optimal paths.
- (b) Total Travelled Angular Distance per Map: This plot shows the total angular distance (in radians) traveled by the agent's sensor during the exploration. It highlights how efficiently the algorithms adjusted their orientation to locate the target.
- (c) Total Explored Area of the Map: It displays the proportion of the total map that was explored by each algorithm before finding the object. It provides insight into how comprehensive each algorithm's exploration strategy was.
- (d) Iterations per Map (Only Positions, Bottom Left): This plot illustrates the number

of positional steps required by each algorithm to find the object. It shows how well each algorithm optimizes the choice of positions.

- (e) Total Iterations per Map: This combines both positional steps and orientation adjustments to evaluate the overall efficiency of the algorithms, including both movement and sensor alignment.
- (f) Total Explored ROI Area per Map: This box-plot represents the proportion of the ROI area explored before finding the object. It evaluates how effectively each algorithm focuses on regions of interest.

The boxplots are labeled numerically to correspond with the tested algorithms:

1. Closest Frontier Algorithm
2. Furthest Frontier Algorithm
3. Information Gain Algorithm
4. Information Gain + Displacement Cost Algorithm
5. ROI-Based Algorithm

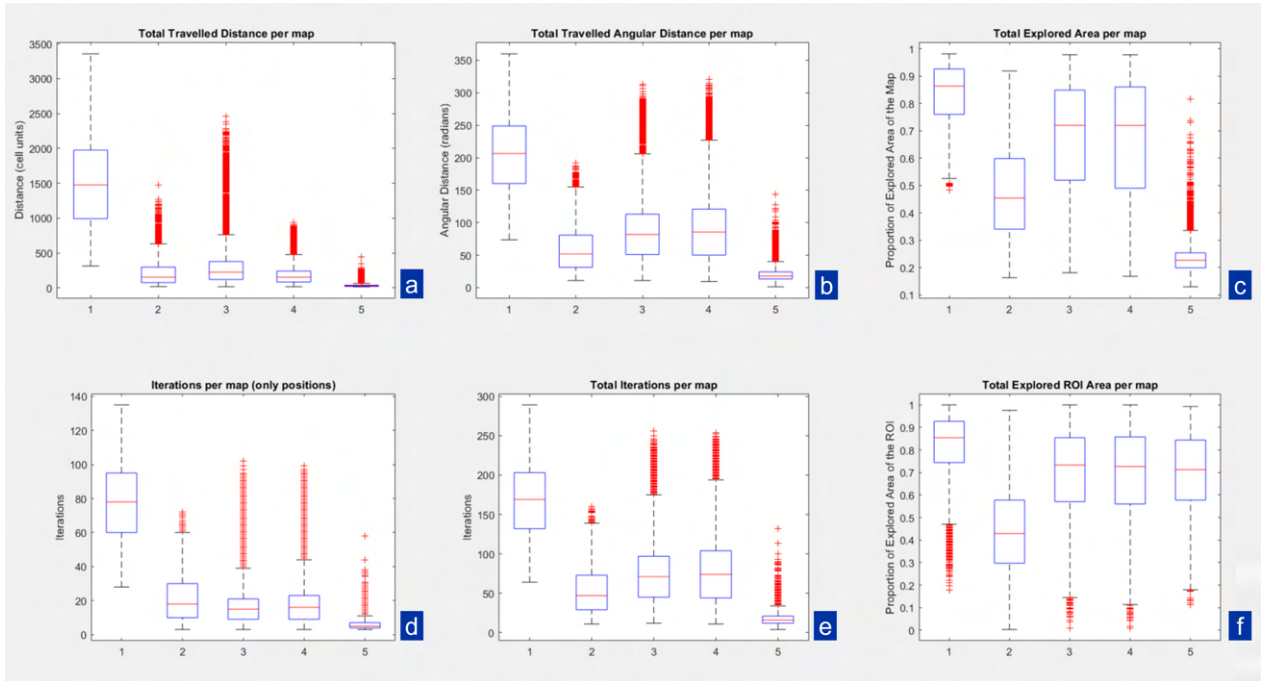


Figure 4.5: Comparison of 2D basic algorithms

In conclusion, it is surprising to observe that the Furthest Frontier algorithm performs significantly better than the Closest Frontier algorithm, and even outperforms Information Gain and Information Gain + Displacement Cost in some parameters. This behavior may be attributed to the characteristics of the generated maps, which are not excessively large, allowing the Furthest Frontier approach to explore efficiently without incurring extreme

penalties.

Regarding angular distance, it is not a critical factor in this evaluation, as all algorithms utilize the same orientation selection method and do not explicitly attempt to optimize it. This ensures that comparisons remain focused on positional performance and exploration strategies.

What stands out clearly is that the ROI-Based algorithm is by far the best-performing approach across all metrics. It achieves superior efficiency in terms of distance traveled and rapid ROI exploration. Even its outliers, marked in red, demonstrate performance that is still remarkably effective compared to the outliers of other algorithms. This highlights the robustness and reliability of the ROI-Based strategy in prioritizing high-value areas while maintaining an efficient overall exploration process.

4.1.2 FoV Simulation (HOPE-OS 2D) vs Gradient Algorithm Test

After evaluating the limitations and strengths of the previously tested algorithms, we transitioned to the advanced FoV simulation and gradient-based approach, as described in Section 3.3.5, to address key challenges observed in the earlier methodologies. The inclusion of gradient maps was a logical progression from the ROI-Based Algorithm. While the mean position of the ROI was effective in guiding exploration, it proved insufficient for irregularly shaped, large, or disjoint regions, as described in Section 3.3.4. Gradient maps offer a more precise and nuanced representation of ROI influence, enabling the algorithm to handle such cases effectively.

Additionally, the integration of FoV simulations significantly enhances decision-making for both position and orientation selection. By simulating the sensor's actual visibility range, the algorithm can optimize its movements to maintain an ideal distance from the ROI, avoiding the pitfalls of overly close proximity, which restricts visibility and hampers exploration efficiency, as described in Section 3.3.5.

The simulations will be conducted using only five 90x90 grid maps, see Figure 4.6, each designed with larger obstacles to increase complexity. The sensor will have a range of 40 units (cells) and a FoV amplitude of 60 degrees, reflecting a more expansive sensing capability compared to the initial tests. Each map will be repeated 100 times since the use of the k-means clustering algorithm introduces non-determinism, as clustering results vary with each iteration. The goal is to analyze the algorithm's behavior when fundamental parameters are varied and subsequently compare its performance with the Human Decision Process in the next section.

It is important to note that optimizing all the parameters governing the algorithm's decision process is highly complex due to the large number of variables involved and the potential dependency on the specific map type. Therefore, this test does not aim to fully optimize the algorithm but instead provides insight into how its performance changes when fundamental parameters, specifically, adjusting score function parameters that balances the importance of ROI and displacement cost.

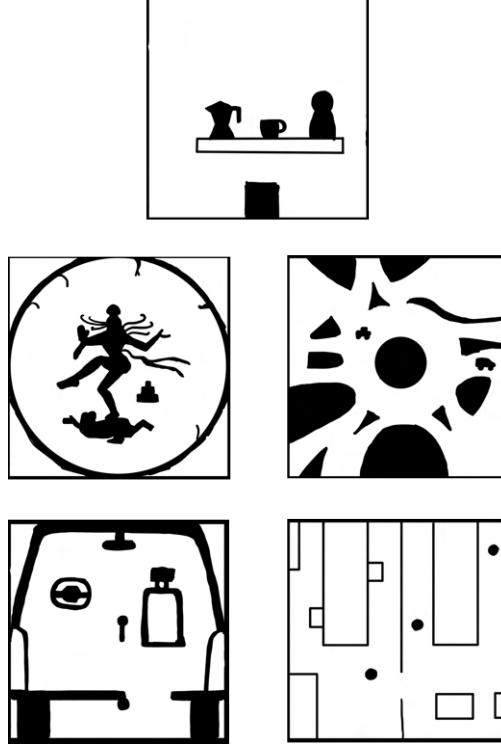


Figure 4.6: Grid maps for 2D testing

The same evaluation methods outlined in Section 3.3.2 are employed to compare the performance of the algorithm in this set of simulations. It is also important to note that the distance units used in these experiments remain in cell units of the grid map.

Additionally, to simulate the realistic movement constraints of a robot, the agent is restricted from "teleporting" to distant positions. If the target position is more than 15 cells away, the agent will move to an intermediate position within 15 cells along the path to the target.

For the this test, we aim to evaluate the impact of altering the weighting factors in the cost function used to determine the Next Best Position (NBP), Equation 3.17. The cost function is defined as:

$$\text{Score} = \alpha \cdot \left(\omega_{\text{total}} \cdot \frac{\text{Total Score}}{\text{Max Information Gain}} + \omega_{\text{avg}} \cdot \frac{\text{Average Score}}{\text{Max Average Gain}} \right) - \beta \cdot \text{Displacement Cost},$$

where:

- ω_{total} and ω_{avg} weight the contributions of the total and average scores within the gradient map evaluation.
- α controls the relative importance of informational value derived from the gradient map.
- β adjusts the penalty associated with the displacement cost.

We modify α and β parameters while ROI gradient parameters are fixed to $\omega_{\text{total}} = 0.9$ and $\omega_{\text{avg}} = 0.1$.

By adjusting α and (β), displacement cost relevance is changed, thus, the test aims to achieve a better tradeoff between exploration efficiency and prioritization of ROI regions.

Reminder: The distance measurement unit is the grid-map cell unit.

Legend Figure 4.8:

1. HOPE-OS 2D 1, $\alpha = 0.4$ and $\beta = 0.6$
2. HOPE-OS 2D 2, $\alpha = 0.5$ and $\beta = 0.5$
3. HOPE-OS 2D 3, $\alpha = 0.6$ and $\beta = 0.4$

From these results, see Figures 4.7 & 4.8, we can draw several conclusions regarding the impact of α and β on the algorithm's performance.

As expected, reducing the influence of displacement cost (i.e., increasing α and reducing β) leads the agent to prioritize more informative viewpoints, even if they are farther away. This effect is clearly visible in Figure 4.8c where the configuration with $\alpha = 0.6$ and $\beta = 0.4$ (HOPE-OS 2D 3) results in the highest median travel distance. The same trend is observable in the line plot of Figure 4.7c, where cumulative distance per iteration grows faster for this setting. Conversely, when β is higher (HOPE-OS 2D 1), the robot prefers shorter movements and overall distance traveled decreases.

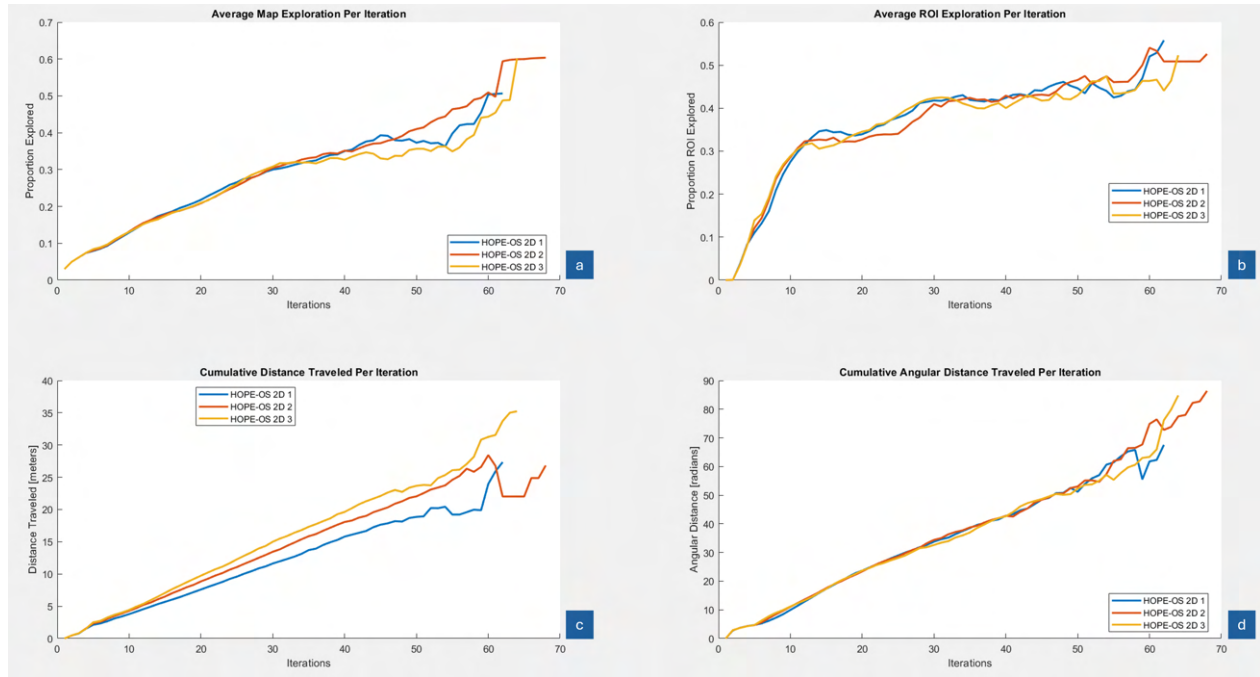


Figure 4.7: Algorithm behavior when α and β variations are applied

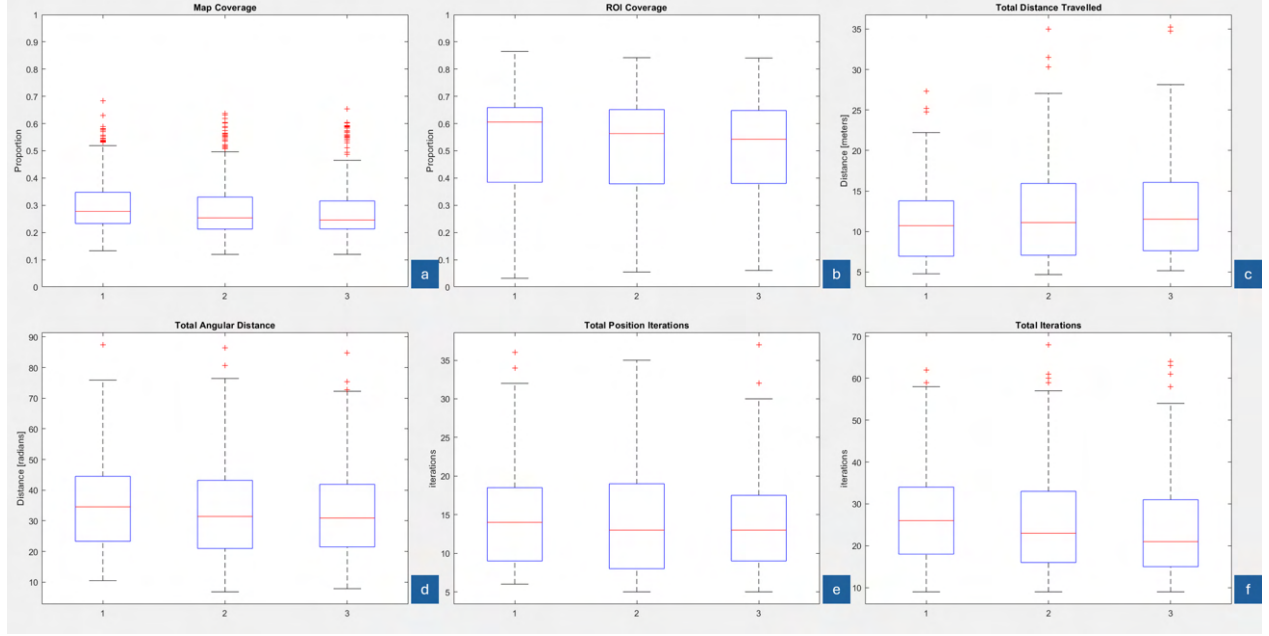


Figure 4.8: Algorithm results when α and β variations are applied. HOPE-OS 2D 1 (1), HOPE-OS 2D 2 (2), HOPE-OS 2D 3 (3)

However, this additional distance does not imply inefficiency. In fact, all three configurations maintain comparable levels of ROI Coverage (Figure 4.8b) and Map Coverage (Figure 4.8a), with HOPE-OS 2D 3 showing slightly better median values in both. This indicates that longer paths are often justified by greater visibility and information gain, especially when initial observations are uninformative or when ROIs are large and spatially distributed.

Regarding Total Iterations (Figure 4.8f) and Total Position Iterations (Figure 4.8e), the results remain relatively stable across all parameter sets. This suggests that the algorithm converges at a similar rate regardless of the weight configuration, which reinforces the idea that increased travel is used strategically, not wastefully. The same conclusion can be drawn from the average map and ROI exploration per iteration (Figures 4.7a and 4.7b), where all configurations follow nearly identical trends.

In summary, increasing α and reducing β causes the robot to explore more actively and cover greater distances, but this behavior leads to better ROI inspection without compromising iteration count or angular efficiency. This tradeoff can be tuned to suit different application needs

4.1.3 FoV Simulation (HOPE-OS 2D) Vs Human Decision Process

Comparing the algorithm with itself and its various parameter adjustments provides valuable insights into how we can optimize its performance. However, such comparisons do not offer a benchmark against established models or alternative strategies. Given the absence of a universal benchmark or widely accepted comparison method for this type of object searching algorithm, we have chosen to evaluate its performance against a human decision process, as

detailed in Section 3.3.6.

In this test, both the algorithm and human players are tasked with locating an object in the five predefined maps shown in Figure 4.6. Both the algorithm and the human participants have access to the same information about the map: the occupancy data provided by the sensor and the location of the ROI. Additionally, human participants are also restricted to a movement radius of 15 cells per step, aligning with the same limitation imposed on the algorithm to avoid unrealistic "teleportation" behavior.

The maps are designed with larger and recognizable objects, allowing human players to use their cognitive abilities to abstract information and make assumptions about the map's structure and potential object locations. This capability provides humans with a significant advantage, as they can apply higher-level reasoning and strategy that the algorithm lacks. Furthermore, human participants improve their performance over time, learning to optimize their search strategies as they become more familiar with the task.

However, this test design mitigates the learning advantage to some extent. With only five maps, humans do not have sufficient time to fully optimize their strategies, creating a more balanced comparison. While it is true that human's performance improves as they gain experience, the limited dataset prevents them from fully exploiting their learning potential. In contrast, the algorithm does not learn or adapt its strategy over time, meaning its performance remains consistent across all tests.

In summary, by comparing the performance of the algorithm against human participants, we aim to evaluate the algorithm's capability not only in optimizing exploration strategies but also in approximating human-like decision-making when searching for objects in complex environments. This test provides a benchmark for assessing the algorithm's overall efficiency, adaptability, and robustness in scenarios that require strategic exploration and prioritization.

Legend Figure 4.10:

1. Human Decision Process (30 participants).
2. HOPE-OS 1, $\alpha = 0.4$ and $\beta = 0.6$.

Analysis of Figure 4.9

- Map Exploration (a): Humans tend to explore a larger amount of the map per iteration than HOPE-OS. This can be attributed to their ability to make strategic decisions and prioritize broader coverage. The algorithm lags behind in map exploration, possibly due to its systematic reliance on calculated scores rather than a strategic overview.
- ROI Exploration (b): HOPE-OS and Humans perform very similar in the beginning of the search. However, as the situation gets more complex, humans explore a larger part of the ROI per iteration, indicating their capability to prioritize high-value regions more effectively.
- Distance Traveled (c): Human decision makers exhibit a greater distance traveled per iteration. This suggests that humans end up not prioritizing movement optimization and often choose suboptimal paths. In contrast, the algorithm maintains a more consistent

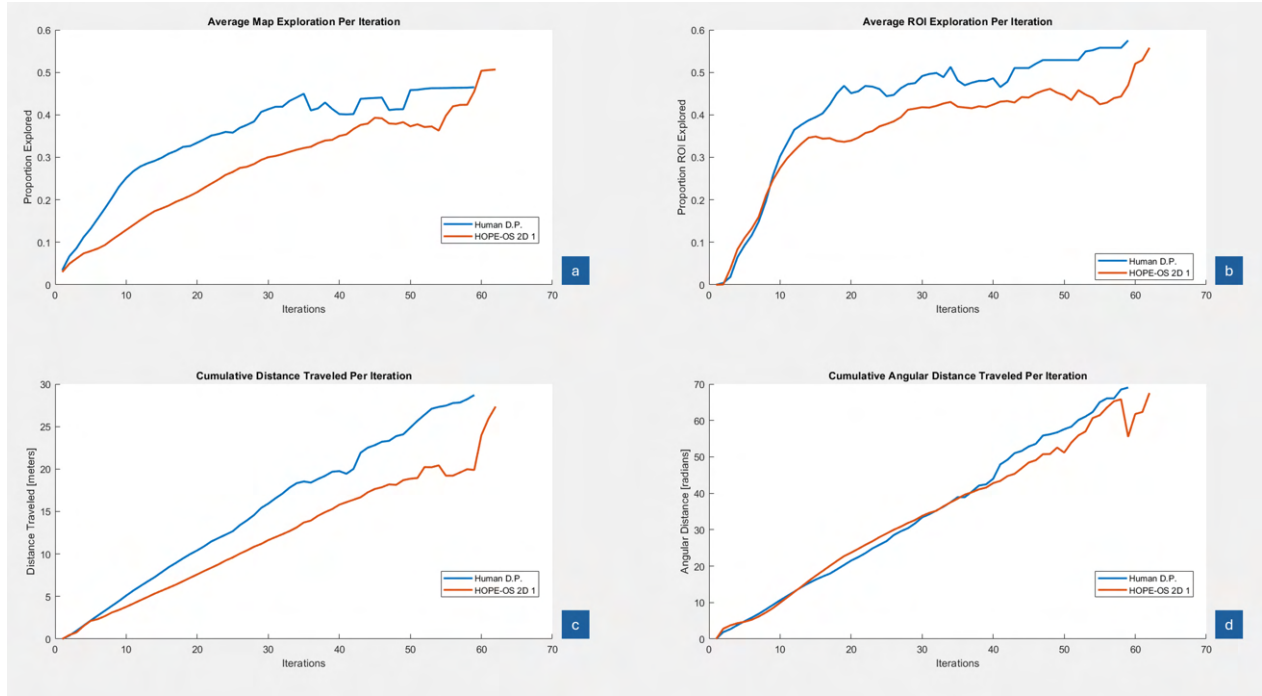


Figure 4.9: Behavior plot: HOPE-OS 2D Algorithm vs Human Decision Process

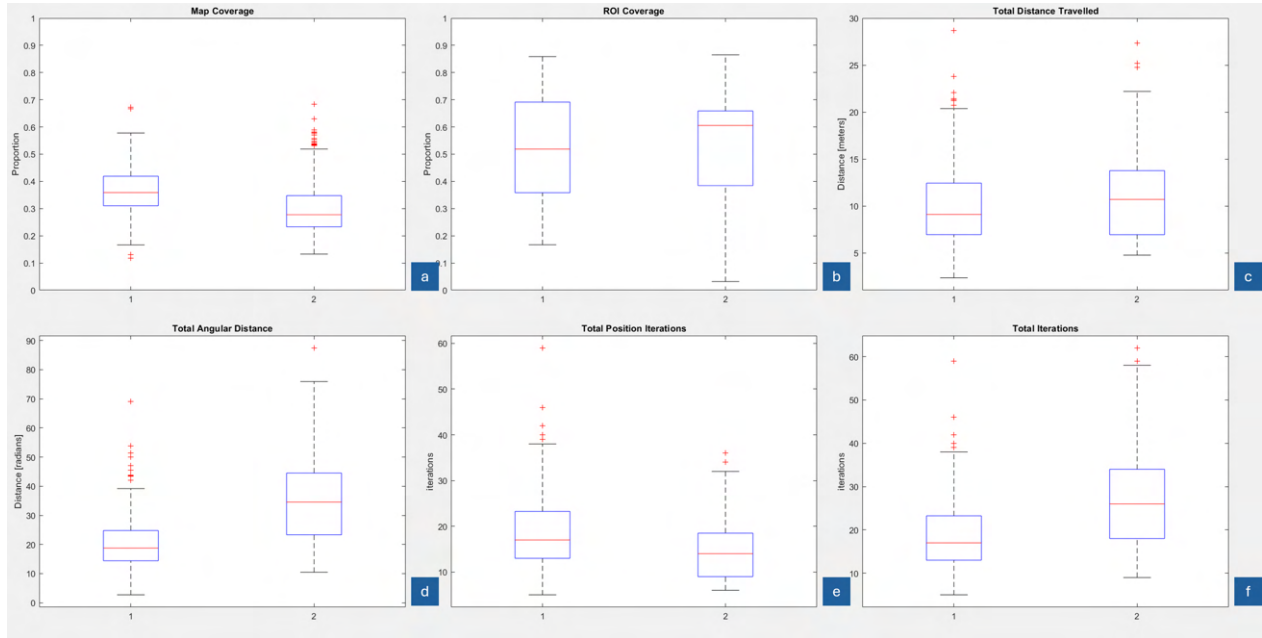


Figure 4.10: Results plot: HOPE-OS 2D Algorithm (2) vs Human Decision Process (1)

trajectory, showing its ability to optimize movement cost.

- Angular Distance (d): In terms of angular distance, both the human decision process and the algorithm perform similarly, with consistent angular adjustments across iterations. This indicates that orientation decisions are relatively straightforward and less influenced by human intuition.

Analysis of Figure 4.10

- Map Exploration (a): Humans slightly explore more area of the map until the object is found, meaning that sometimes they prioritize amount rather than quality, strategically optimizing the search.
- ROI Exploration (b): Humans require a little less ROI exploration to locate the object, demonstrating their ability to reason about which ROI regions are likely to contain the object. The algorithm explores more ROI on average, as it systematically evaluates regions without the benefit of abstract reasoning.
- Distance Traveled (c): Although the cumulative distance curve in Figure 4.9c suggests that human participants travel more over time, this is due to the fact that they require more total iterations to complete the task. In contrast, Figure 4.10c shows that the algorithm has a higher median total distance traveled. This apparent discrepancy arises because cumulative plots are measured over all iterations, including both position and orientation steps—whereas the total distance boxplot reflects only the final displacement until success. The algorithm tends to make longer, more targeted moves per position step, resulting in fewer total iterations but greater displacement overall.
- Angular Distance (d): Humans exhibit lower angular distance values because they needed less total iterations.
- Iterations (e and f): The HOPE-OS algorithm outperforms humans in terms of median position iterations required to locate the object. Moreover, it also has lower dispersion, showing higher robustness.

These results highlight the efficiency and robustness of HOPE-OS compared to human strategies. While humans benefit from high-level reasoning, abstract understanding of spatial structure, and prior experience, they also tend to over-explore and take longer paths. In contrast, HOPE-OS follows a cost-efficient, heuristic-guided strategy that enables it to find the object with fewer steps, despite having no learning or reasoning.

Notably, HOPE-OS approaches human-level performance in key metrics such as ROI and Map coverage or traveled distance, while outperforming in number of iterations. This suggests that the algorithm is not only systematic, but also strategically effective within its domain.

These insights suggest areas for improvement in the algorithm. Incorporating higher-level abstractions or heuristics that mimic human reasoning could enhance its performance. For instance, introducing a mechanism to strategically prioritize regions based on broader patterns or environmental cues could provide a better balance between methodical decision-making and adaptive exploration.

In conclusion, this comparison is highly encouraging, as it compares the algorithm to human intelligence and capabilities. Despite the inherent advantages humans have, such as intuition, abstraction, and strategic reasoning, the algorithm manages to achieve results that come remarkably close and even outperforms in some metrics. This highlights the significant potential and promise of HOPE-OS, paving the way for further refinements that could narrow the performance gap even further.

4.2 3D Simulations

The 3D tests closely mirror the experiments conducted in the 2D environment, evaluating the same performance metrics but within a three-dimensional setting. To ensure a realistic and statistically robust evaluation, the algorithm was tested on seven voxel-based environments, each measuring $4.5\text{ m} \times 4.5\text{ m} \times 3.5\text{ m}$ with a voxel resolution of 0.05 m , totaling approximately 71 m^3 per map. These environments were designed with varying levels of structural complexity and occupancy patterns, including open areas, narrow passages, and irregular obstructions, see Figure 4.11.

Each map was evaluated under multiple initial agent positions and different object/ROI configurations, resulting in 1644 unique test cases. Not all scenarios provided informative initial observations, some offered very limited visibility due to occlusions or unfavorable starting viewpoints. This variability was intentional, as it allowed us to assess the algorithm’s capacity to generalize across different conditions and adapt to poor initial sensing without relying on prior knowledge.

The sensor used in the simulation featured a realistic perception setup, with a maximum range of 2.5 m and a frustum with a field of view of 60° in the vertical plane and 90° in the horizontal plane. This ensured that the evaluation conditions accurately reflected the perception constraints the system would face in deployment.

The parameters used in these tests were the following:

- ROI Gradient Function: For equations 4.1, 4.2 and 4.3 we set the variables $q = 0.002$, $h = 1$ and $w = 4$

$$g(d) = \frac{1}{d^2 + q} \cdot q \quad (4.1)$$

$$f(d) = \frac{1}{d^2 + h} \cdot h \quad (4.2)$$

$$G_{\text{ROI}} = \frac{w \cdot g(d) + f(d)}{w + 1} \quad (4.3)$$

- Obstacles Gradient Function: For equation 4.4 we set $k = 40$ and $d_{\text{threshold}} = 0.1$.

$$G_{\text{Obstacle}} = \frac{1}{1 + e^{k \cdot (d - d_{\text{threshold}})}} \quad (4.4)$$

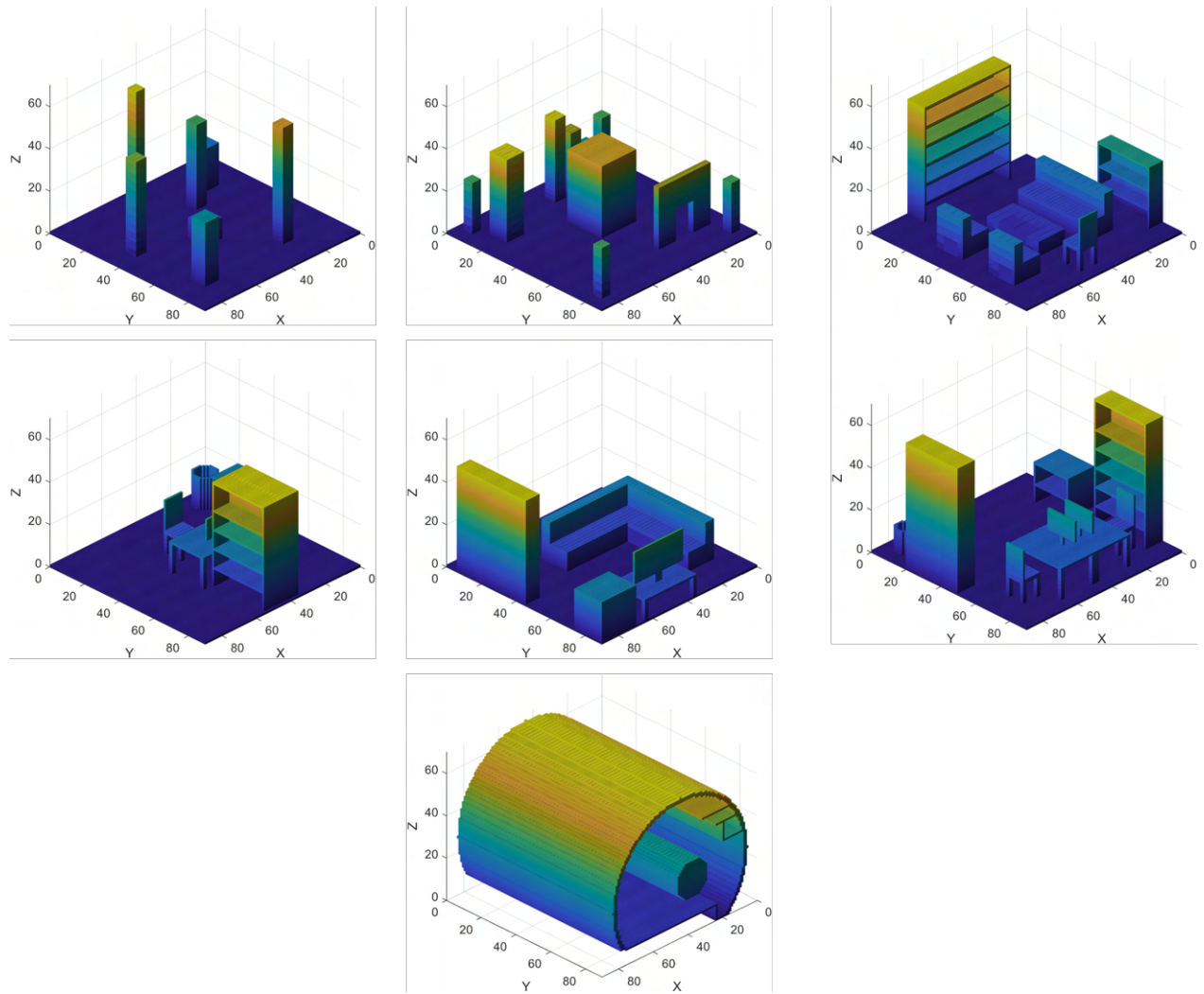


Figure 4.11: Environments for 3D testing

- Displacement Cost Function: For equation 4.5 we set $\epsilon = 0.3$

$$\text{DispCost} = -\left(\frac{1}{d^2 + \epsilon} \cdot \epsilon\right) + 1 \quad (4.5)$$

- Global Cost Function: For equation 4.6 we set $\alpha = 0.26$, $\beta = 0.24$ and $\gamma = 0.5$.

$$\text{Score} = (\alpha \cdot \text{ROI Gradient} - \beta \cdot \text{Displacement Cost} - \gamma \cdot \text{Obstacle Proximity Cost}) \quad (4.6)$$

Adjusting these parameters is a highly challenging task, as they are all interdependent. The parameters were fine-tuned empirically by observing the agent's behavior during simulations, iteratively adjusting the values until a configuration that appeared to work reasonably well was identified.

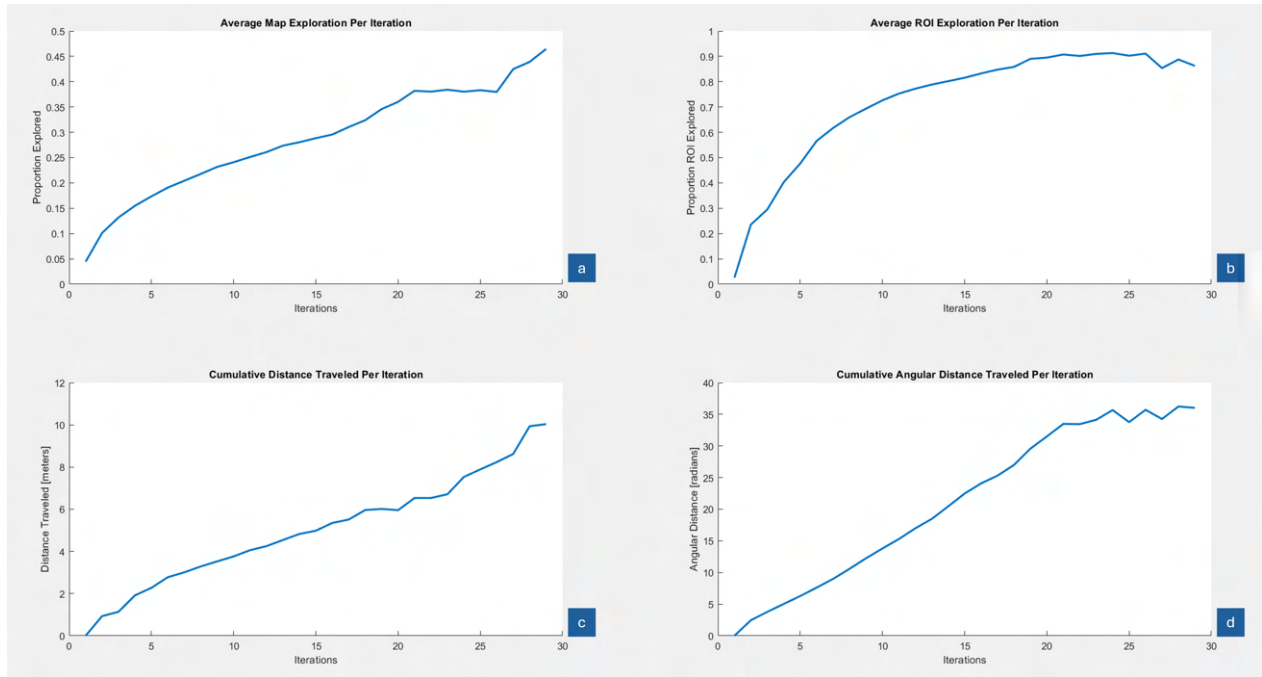


Figure 4.12: 3D Algorithm behavior

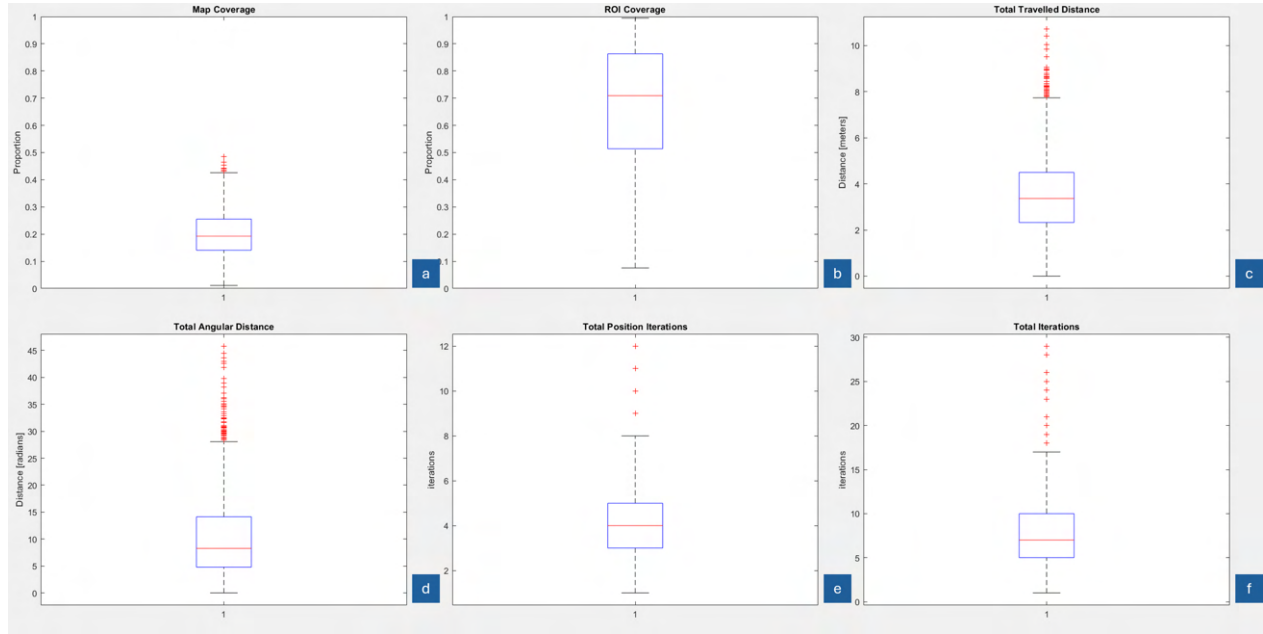


Figure 4.13: 3D Algorithm results

The algorithm’s performance across the 3D simulations is summarized in Figures 4.12 and 4.13, which respectively show cumulative behavior across iterations and aggregated statistics over all 1644 test cases.

In Figure 4.12a, we observe a steady increase in average map exploration per iteration, which reflects the agent’s ability to incrementally uncover the environment. The curve maintains consistent growth, even as the number of iterations increases, demonstrating the algorithm’s sustained exploratory behavior. However, the slope flattens in later stages, suggesting diminishing returns as most easily accessible areas are explored.

Figure 4.12b displays ROI exploration per iteration, where the algorithm achieves rapid early gains. By iteration 10, more than 70% of the ROI has been explored on average. This early convergence indicates that the gradient map effectively guides the agent toward high-priority regions. Interestingly, the curve begins to saturate around iteration 20, suggesting that remaining ROI voxels are harder to access, often occluded or located behind obstacles.

In terms of cumulative distance traveled (Figure 4.12c), the algorithm shows gradual but consistent growth, with a noticeable jump near iteration 28. This likely corresponds to a final repositioning movement to access a previously hidden region. Despite this, the total distance remains modest, which aligns with the displacement cost component of the score function that penalizes long movements. Similarly, cumulative angular distance (Figure 4.12d) grows in a mostly linear fashion.

Figure 4.13 provides the overall statistical view via boxplots:

- Map Coverage (a): The algorithm explores a relatively small portion of the environment (median 20%). This is expected due to the ROI-driven nature of the planner, which avoids unnecessary exploration of low-priority regions.

- ROI Coverage (b): The ROI coverage is high (median 73%), confirming that the gradient guidance is effective in directing the robot’s attention toward the most relevant areas.
- Total Traveled Distance (c): The algorithm maintains low overall displacement (median 3.7m), with a narrow interquartile range. This reflects efficient navigation and good control over unnecessary movement.
- Total Angular Distance (d): Angular displacement is also low (median 8.5 radians), with limited outliers. This further confirms that viewpoint selection is both targeted and stable.
- Total Position Iterations (e): The number of position updates remains small, with a median of just 4 and a maximum of 12, indicating that few moves are needed to achieve sufficient visibility and find the object.
- Total Iterations (f): Including orientation steps, the median total iteration count is 7, with a maximum of 29, highlighting the algorithm’s ability to reach the goal with minimal planning cycles.

These results demonstrate that HOPE-OS maintains strong performance in 3D environments, with high ROI inspection rates. The algorithm achieves target localization with few steps, limited rotational adjustments, and minimal exploration of irrelevant areas. Despite the added complexity of 3D space, the method scales effectively without compromising efficiency, confirming its robustness under realistic occlusion constraints and diverse environmental conditions.

4.3 Real World Tests

The real-world tests were carried out in a mockup environment designed to replicate a segment of the CERN Large Hadron Collider (LHC), as depicted in Figure 4.14. Due to technical difficulties, the planned experiments involving the TIM robot and the Kinova robotic arm could not be executed. Instead, a UR10 robot was utilized for these tests. The UR10 was mounted on a custom-built base created by the team, although the base’s mobility features were not employed, and the robot remained stationary throughout the experiments. It can be seen in Figure 4.14, and also the BLM (object to find) attached to the left side of the pipe. This change, however, does not impact the relevance of the tests, as the algorithm was specifically designed to be adaptable to different robotic platforms, ensuring its functionality across a variety of systems.

The implementation was carried out using the CERN Robotic Framework (CRF). However, the current version of the CRF lacks a collision-avoidance trajectory planner. To mitigate potential collision risks, the robot’s possible positions were constrained to a virtual horizontal plane positioned at a height sufficient to ensure it could not collide with the LHC pipe structure. This setup allowed the robot to safely execute its exploration tasks without introducing complex trajectory planning.

Another significant challenge was the lack of joint limit handling in the CRF trajectory planner. Without this feature, the UR10 robot frequently reached its joint limits during operation,

causing it to get stuck and disrupting the tests. These interruptions made data collection difficult and limited the scope and reliability of the results obtained from the real-world experiments.



Figure 4.14: Real world scenario: UR10 in the mock-up of the LHC

Despite these challenges, the test results, while not highly conclusive, provide a preliminary indication of the algorithm’s performance in a real-world environment. The collected data, though sparse, validate the basic feasibility of the algorithm under practical conditions. Furthermore, a demonstration video showcasing the robot solving a search task to locate a BLM has been uploaded to YouTube to visually demonstrate the algorithm’s functionality in this setting, see the video in this [link](#).

The range used for the RealSense camera during the real-world tests was 2 meters, with a field of view of $87^\circ \times 58^\circ$. The parameters used in the gradient map functions and cost calculations were identical to those applied in the 3D simulations described in the previous section. Due to the challenges and limitations discussed earlier, only six successful trials were conducted where the robot operated without interruption. These results are presented in Figure 4.15, providing insight into the algorithm’s performance in real-world conditions.

The voxel map generated during the real-world tests had the same dimensions as those used in the 3D simulations: $90 \times 90 \times 90$, with a voxel edge size of 0.05m. The origin of the map (0,0,0) was set relative to the base of the robot, simplifying the definition of the Region of Interest (ROI). The ROI was defined by performing the necessary frame transformations: first, from the ROI to the robot’s base frame, and then from the base frame to the global frame. This approach ensured accurate and consistent ROI localization within the voxel map.

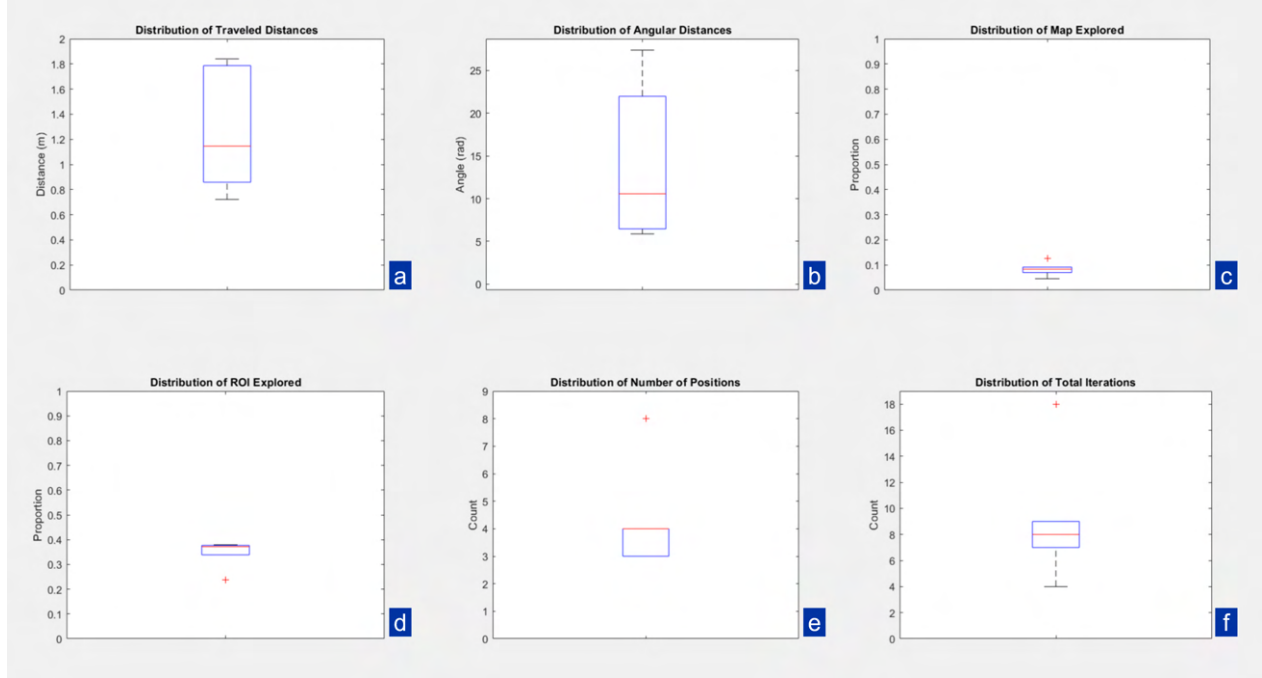


Figure 4.15: Box-plots of performance metrics from real-world tests

From the results of the real-world tests depicted in Figure 4.15, we can draw the following observations:

- **Total Traveled Distance (a):** The median traveled distance is approximately 1.2m, with a relatively small inter-quartile range, indicating consistent performance across the tests. The highest value is less than 2m, thereby we can observe that the algorithm's decision making process algorithm is effective and minimizes the displacement necessary to find the object.
- **Total Angular Distance (b):** The angular distance traveled has a median value of approximately 10rad, which aligns well with expectations given the sensor's limited field of view.
- **Total Map Exploration (c):** The proportion of the map explored remains low, with a median of approximately 0.1 (10%). This is consistent with the ROI-focused approach of the algorithm, which prioritizes regions of interest over exhaustive map coverage. The narrow range of values here suggests stable behavior in this metric across trials.
- **Total ROI Exploration (d):** The total ROI exploration proportion centers around 0.4, with minimal variation. This consistency reflects the algorithm's ability to effectively identify and focus on high-priority regions, even under real-world constraints.
- **Number of Positions and Total Iterations (e & f):** The robot typically required a median of 3 positions to complete the task, with a few outliers reaching up to 8 positions. These values make were expected and are actually low compared to the 3D simulations results, Figure 4.13. Similarly, the median number of total iterations (including position and orientation adjustments) was around 6.

Figure 4.16 illustrates the sequence of poses taken by the algorithm as it progresses toward locating the BLM. Each image represents a key step in the decision-making process

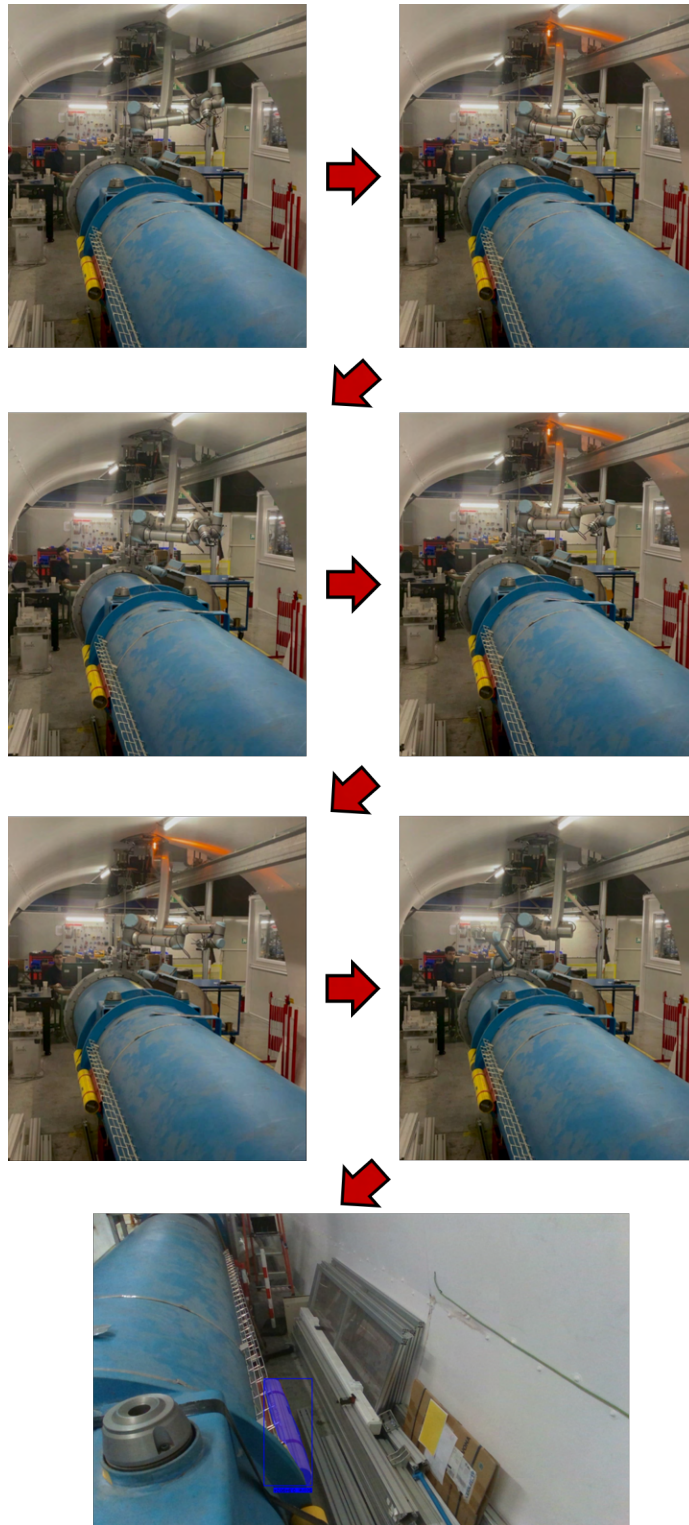


Figure 4.16: Real world test. Decision making example.

In conclusion, while the number of real-world tests conducted was limited due to technical challenges, the results are promising. The algorithm demonstrated its ability to navigate and prioritize high-value regions effectively, even within the constraints of a complex and realistic environment. Despite occasional inefficiencies caused by joint limits or sensor errors, the overall performance suggests that the algorithm has strong potential for real-world deployment. Additionally, even though the code is not optimized, the decision-making process only takes around 1s, meaning it could be optimized to work in a continuous loop.

Although further tests with a larger dataset are necessary to draw statistically significant conclusions, these initial trials provide a solid foundation and validate the feasibility of the approach in practical applications.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

Object searching remains a critical yet complex problem in robotics that requires addressing key challenges such as uncertainty, decision-making, and the seamless integration of perception with exploration strategies. Its applications range from industrial maintenance to search and rescue missions. Despite its significance, there is no universally defined or standardized solution for this challenge, particularly in dynamic and diverse environments.

This thesis has presented a novel heuristic-based approach—HOPE-OS—that addresses the core challenge of determining the Next Best View (NBV) using gradient-based region prioritization and field of view simulation. By assuming manually defined ROIs, the algorithm focuses entirely on the NBV problem. This design choice allowed the development of a lightweight, robot-independent and map-agnostic framework that dynamically builds voxel maps and reasons about occlusion without relying on prior environmental knowledge. Gradient maps guided the search process toward target regions, while FoV simulations enabled realistic evaluation of candidate viewpoints, incorporating both sensor constraints and visibility.

The proposed framework has demonstrated strong performance across a wide range of conditions. In 2D environments, HOPE-OS achieved exploration efficiency comparable to human participants. In 3D simulations, including highly occluded, large-scale environments, the algorithm maintained its performance and adaptability, efficiently locating targets with minimal map coverage and movement.

Preliminary real-world tests confirmed the algorithm’s feasibility, completing successful object searches under realistic sensor input and motion constraints. These results collectively demonstrate that HOPE-OS offers a scalable, generalizable, and practical solution to object search, with competitive performance despite its simplicity.

5.1.1 Summary of Contributions

The key contributions of this work include:

- A novel heuristic planning algorithm that combines ROI gradient fields with occlusion-aware FoV simulation for NBV selection.
- A unified scoring function that balances information gain, displacement cost, and obstacle proximity.
- A clustering-based viewpoint sampling method that adapts search resolution based on gradient values.
- A
- A novel benchmarking protocol comparing human and algorithmic decision-making in 2D object search tasks.
- Demonstration of robust and generalizable performance in diverse 3D simulations and preliminary real-world testing.
- A computationally lightweight solution with lower time complexity than previous approaches, making it suitable for real-time use and deployment on resource-constrained platforms.

5.1.2 Strengths and Limitations

HOPE-OS stands out for its simplicity, real-time performance, and ability to generalize across platforms and environments. Its parameterized scoring mechanism allows tuning exploration behavior without retraining or remapping. Unlike more complex frameworks such as POMDP-based solvers or deep learning models, HOPE-OS requires no prior data, training phase, or semantic labeling, and it operates effectively even with minimal information.

The main limitation lies in the reliance on manually defined ROIs. Achieving optimal performance requires careful adjustment of weighting factors in the cost function and gradient maps. Additionally, voxel-based mapping, while simple and effective, may become memory-intensive in very large environments. Integrating octree-based representations or semantic map priors is a logical next step, as outlined in Section 5.2.

5.2 Future Work

The proposed algorithm has demonstrated significant potential, but several avenues for improvement and expansion remain to be explored. These enhancements could address current limitations and further adapt the methodology for complex and diverse scenarios, particularly in real-world applications. Below are the key areas for future work:

- Automated Region of Interest Generation

Currently, the ROIs are defined manually, which limits the algorithm’s autonomy and scalability. Automating the generation of ROIs would enable the system to identify

high-priority regions dynamically based on environmental features or predefined criteria. Methods such as semantic segmentation, object recognition, and clustering could be used to identify ROIs automatically, reducing human intervention and improving efficiency.

- Object Correlation for ROI Generation

Incorporating object correlation techniques could enhance the automated generation of ROIs. For instance, the algorithm could leverage prior knowledge of object relationships to infer likely locations for a target. Techniques discussed in the state of the art, such as Bayesian frameworks or graph-based methods, could be employed to correlate objects and create meaningful ROIs based on the contextual relationships between them.

- Semantic Knowledge for Probable Locations

Integrating semantic knowledge into the algorithm could significantly improve its ability to predict probable object locations. For example, knowing that specific objects are often found in certain areas (e.g., tools near a workstation or valves near pipelines) would allow the algorithm to prioritize these areas more effectively. Such knowledge could be encoded using ontologies, pre-trained machine learning models, or other high-level reasoning tools to complement the existing decision-making process.

- Implementation of Belief States and Uncertainty

To improve robustness, the algorithm could incorporate belief states and uncertainty modeling. For example, even when a region is marked as explored and free, there should be an associated uncertainty metric that reflects the confidence level in the assessment. This would be particularly useful when integrating additional sensors such as LiDAR or when operating in environments where sensor noise or occlusions are prevalent. Integrating SLAM (Simultaneous Localization and Mapping) techniques could also enhance the algorithm's performance in real-world scenarios by providing a continuously updated map of the environment.

- Displacement Cost Depending on Kinematics

The current displacement cost is calculated based on Euclidean distance, which does not account for the robot's kinematics or potential obstacles in its trajectory. Future developments of the algorithm should incorporate a trajectory planning component to evaluate the actual cost of displacement. This would involve integrating collision avoidance strategies and considering the robot's joint limits and mobility constraints. By doing so, the algorithm would not only reduce travel time and energy consumption but also ensure safe navigation in cluttered or constrained environments.

- Continuous Decision-Making During Motion Currently, the algorithm makes decisions at discrete intervals: the robot moves to a position, captures its environment, updates the map, and then decides where to move next. This approach, while effective, could be significantly improved by allowing the algorithm to make continuous decisions during motion. By evaluating new potential NBV positions while the robot is en route, the

algorithm could avoid unnecessary movements to already explored areas, dynamically optimize the path, and reduce overall search time. This constant evaluation loop would increase the information gathered per iteration and improve the algorithm's adaptability in complex or dynamic environments, where new data may rapidly change the decision-making process.

- Transition to Octree-Based Spatial Representation

Currently, the algorithm operates using voxel maps to represent the environment, which work well for smaller or moderately sized scenarios. However, voxel maps face scalability limitations when applied to very larger environments due to their fixed resolution and high memory requirements. Transitioning to octree-based spatial representation offers a solution to this issue.

Bibliography

- [1] David Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.
- [2] K. Zheng. *Generalized Object Search*. Brown University, 2023. Doctoral dissertation, Providence, Rhode Island.
- [3] Y. Wang, S. James, E. K. Stathopoulou, C. Beltrán-González, Y. Konishi, and A. Del Bue. Autonomous 3-d reconstruction, mapping, and exploration of indoor environments with a robotic arm. *IEEE Robotics and Automation Letters*, 4(4):3340–3347, Oct 2019.
- [4] D. Joho, M. Senk, and W. Burgard. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems*, 59(5):319–328, 2011.
- [5] S. Y. Goldsmith and R. Robinett. Collective search by mobile robots using alpha beta coordination. In *International Workshop on Collective Robotics*, pages 135–146, 1998.
- [6] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
- [7] D. Deng, R. Duan, J. Liu, K. Sheng, and K. Shimada. Robotic exploration of unknown 2d environment using a frontier-based automatic-differentiable information gain measure. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1241, May 2023.
- [8] T. Wohlfahrt. Autonomous exploration and mapping with mobile robots using next-best-view planning and semantic annotation. Master’s thesis, German Aerospace Center (DLR), 2015.
- [9] D. Holz and et al. Simultaneous exploration and localization for mobile robots using a combination of heuristic-based and frontier-based approaches. In *ISR/Robotik 2010 (41st International Symposium on Robotics)*, 2010.
- [10] S. Selin and M. K. Habib. Frontier-based exploration for autonomous robot. *arXiv preprint*, arXiv:1806.03581, 2018.
- [11] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon “next-best-view” planner for 3d exploration. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1462–1468, 2016.
- [12] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto. An efficient

- sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, Apr 2020.
- [13] M. Naazare, F. G. Rosas, and D. Schulz. Online next-best-view planner for 3d-exploration and inspection with a mobile manipulator robot. *IEEE Robotics and Automation Letters*, 7(2):3779–3786, Apr 2022.
 - [14] S. M. Pollock. A simple model of search for a moving target. *Operations Research*, 18(5):883–903, 1970.
 - [15] L. D. Stone. *Theory of Optimal Search*. Elsevier, 1976.
 - [16] M. Mangel. Search for a randomly moving object. *SIAM Journal on Applied Mathematics*, 40(2):327–338, 1981.
 - [17] T. Kollar and N. Roy. Using semantic knowledge for robot object search. *Robotics and Autonomous Systems*, 57(12):1235–1246, Dec 2009.
 - [18] Y. Ye and J. K. Tsotsos. Sensor planning for 3d object search. volume 73, pages 145–161, 1997.
 - [19] A. Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J. K. Tsotsos, and E. Korner. Active 3d object localization using a humanoid robot. *IEEE Transactions on Robotics*, 27(1):47–64, 2010.
 - [20] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa. Object search by manipulation. *Autonomous Robots*, 36(1):153–167, 2014.
 - [21] A. Somani, N. Ye, D. Hsu, and W. S. Lee. Despot: Online pomdp planning with regularization. In *Advances in Neural Information Processing Systems*, pages 1772–1780, 2013.
 - [22] J. K. Li, D. Hsu, and W. S. Lee. Act to see and see to act: Pomdp planning for object search in clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
 - [23] G. Hollinger, S. Singh, J. Djughash, and A. Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009.
 - [24] Y. Rizk, M. Awad, and E. W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.
 - [25] D. Shah, B. Osinski, B. Ichter, and S. Levine. Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint*, arXiv:2207.04429, 2022.
 - [26] N. Xu. Active object searching on mobile robot using reinforcement learning. In *2nd International Conference on Computing and Data Science (CDS)*, pages 296–300, Stanford, CA, USA, 2021.
 - [27] L. E. Wixson and D. H. Ballard. Using intermediate objects to improve the efficiency of visual search. *International Journal of Computer Vision*, 12(2-3):209–230, 1994.

-
- [28] T. Wandzel and et al. Simplified search strategies for object detection. *Robotics and Automation Letters*, 2019.
 - [29] B. Bejjani and et al. Simplifying object search in 2d environments. *International Journal of Robotics Research*, 2021.
 - [30] Y. Xiao, S. Katt, A. ten Pas, S. Chen, and C. Amato. Online planning for target object search in clutter under partial observability. In *Proceedings of the International Conference on Robotics and Automation*, 2019.
 - [31] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, Aug 2013.
 - [32] Kaiyu Zheng, Yoonchang Sung, George Konidaris, and Stefanie Tellex. Multi-resolution pomdp planning for multi-object search in 3d. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8723–8730. IEEE, 2021.
 - [33] Kaiyu Zheng, Anirudha Paul, and Stefanie Tellex. A system for generalized 3d multi-object search. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.
 - [34] W. Yang and et al. Mapping rgb images to navigation actions: Data-driven approaches. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
 - [35] D. S. Chaplot and et al. Semantic visual navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
 - [36] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3164–3174, Jun 2020.
 - [37] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada. Learning longhorizon robot exploration strategies for multi-object search in continuous action spaces. *arXiv preprint*, arXiv:2205.11384, 2022.
 - [38] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot. Navigating to objects in the real world. *arXiv preprint arXiv:2212.00922*, Dec 2022.

Annexes

Annex I - Sustainable Development Goals (SDGs)

The research developed in this thesis aligns with several Sustainable Development Goals defined by the United Nations as part of the 2030 Agenda for Sustainable Development, Figure 1. While the work itself does not target a specific social or humanitarian application, the methodologies proposed, such as active perception, Next Best View (NBV) planning, and mobile robot navigation, hold clear potential for broader societal and environmental benefits.



Figure 1: SDGs goals

The following SDGs are particularly relevant:

- SDG 9: Industry, Innovation, and Infrastructure The proposed object search framework supports the integration of advanced technologies in industrial settings. This work promotes smarter, safer, and more efficient infrastructure management, especially in

constrained or complex environments.

- **SDG 11: Sustainable Cities and Communities** Mobile robots equipped with intelligent exploration capabilities can be deployed for structural inspection, domestic service, or emergency response in urban environments. The application of such systems contributes to more resilient, automated, and sustainable urban ecosystems.
- **SDG 12: Responsible Consumption and Production** The system architecture emphasizes efficient resource usage, both in terms of computation and deployment (reusing testbeds such as the TIM rail robot). This design philosophy reduces hardware redundancy and energy consumption, aligning with sustainable engineering principles.

The proposed methods are modular and reusable, and thus adaptable to a variety of real-world robotic applications that may directly serve goals related to accessibility, efficiency, and environmental awareness.

Annex II - Impact Evaluation and Ethical-Legal Considerations

This section addresses the broader implications of the research in environmental, social, and legal-ethical dimensions.

- **Environmental Impact:** The optimization strategies introduced (e.g., maximizing information gain, minimizing redundant movement) result in more efficient exploration behaviors. This translates to reduced power consumption, lower operational costs, and longer mission lifetimes for battery-powered robots. In large-scale deployments, such improvements contribute to a smaller environmental footprint.
- **Social Impact:** The framework may enable robots to perform tasks that would otherwise expose humans to risk, such as infrastructure inspection in hazardous or confined spaces like LHC tunnel. This reinforces the value of robotic assistance in improving safety, reducing exposure to danger, and augmenting human capability.
- **Economic Impact:** By reducing manual labor needs and improving the efficiency of spatial data collection, this work has potential value for industries seeking cost-effective automation solutions. It lays a foundation for scalable, adaptive robotic systems that can be integrated into existing workflows without major infrastructural overhauls.

Legal Considerations

Any real-world deployment of the proposed system must comply with applicable regulations regarding:

- The use of autonomous systems in shared environments
- Data collection and privacy when visual sensors are involved
- Certification for safety, specially in human-robot interaction scenarios

Furthermore, open-source components such as OctoMap and perception frameworks must be used in accordance with their respective licenses. Any proprietary hardware (e.g., Jetson, RealSense) must be integrated respecting intellectual property and usage agreements.

Ethical Considerations

The ethical deployment of robotic systems demands transparency, inclusivity, and accountability:

- **Transparency:** Decisions made by autonomous agents should be explainable and auditable, especially in safety-critical applications.
- **Inclusivity:** Technologies should be designed with accessibility in mind, ensuring they serve diverse populations and do not reinforce existing inequalities.
- **Sustainability:** Energy-efficient and modular architectures, as proposed in this work, contribute to long-term responsible technology development.

While the work presented is primarily technical, its potential impact extends well beyond algorithmic contributions. By addressing exploration and perception challenges in robotics through efficient, scalable, and adaptable methods, this thesis opens the door to future systems that not only advance engineering but also contribute to a more sustainable and equitable technological landscape.

Annex III - Temporal Planning and Execution Budget

Temporal Planning

The following Gantt chart summarizes the timeline of this Master's Thesis, covering research, simulation, real-world implementation, and report writing from June to November 2024.

The full Gantt chart is shown on the next page in landscape format.

[illegible]

Execution Budget

The execution budget includes the cost of materials, software, hardware components, and labor. Shared infrastructure such as the TIM robot or Kinova arm is included for reference. Hardware costs were amortized to reflect shared infrastructure and realistic usage. Major robotic systems (UR10, Kinova Gen3 and TIM Platform) were amortized over 5 years. Low-cost accessories were included in full. The breakdown is as follows, Figure 2.

EXECUTION BUDGET					
1. Material					
1.1 Software					
Code	Description	Quantity	Price	Subtotal	Total
1.1.1	Windows 11	1	€ 139.00	€ 139.00	€ 139.00
1.1.2	MATLAB academic license	1	€ -	€ -	
1.2 Hardware					
Code	Description	Quantity	Price	Subtotal	Total
	Intel RealSense D435i	1	€ 180.00	€ 180.00	€ 18,700.16
	UR10 Robot	1	€ 5,000.00	€ 5,000.00	
	Kinova Gen3 Robotic Arm	1	€ 2,000.00	€ 2,000.00	
	TIM Robot Platform	1	€ 10,000.00	€ 10,000.00	
	NVIDIA Jetson	1	€ 589.00	€ 589.00	
	Computer HP	1	€ 750.00	€ 750.00	
	Monitors ThinkVision	2	€ 78.34	€ 156.68	
	Mouse HP	1	€ 11.99	€ 11.99	
	keyboard DELL	1	€ 12.49	€ 12.49	
2. Labor					
Hours		Pay/hour		Total	
375		€ 11.00		€ 4,125.00	
				EXECUTION BUDGET (Without Taxes)	€ 22,964.16
TOTAL EXECUTION BUDGET					
Description			Total amount		
Software			€ 139.00		
Hardware			€ 18,700.16		
Labor			€ 4,125.00		
Material Execution Budget			€ 22,964.16		
Indirect Cost			€ 3,444.62		
Subtotal			€ 26,408.78		
Tax (IVA 21%)			€ 5,545.84		
TOTAL			€ 31,954.63		

Figure 2: Execution Budget