National Technical University of Athens

School of Mechanical Engineering

# Computationally Efficient Harmonic-based Reactive Exploration


*Author:*
Panagiotis Grontas

*Supervisor:*
Prof. Kostas J. Kyriakopoulos

A thesis submitted for the degree of

*Diploma in Mechanical Engineering*

# Abstract

Although Harmonic Potential Fields constitute a powerful tool for tackling the autonomous robot exploration problem, yet their applicability is limited in practice by the heavy computational load involved in solving the Laplace equation in real time. In this work, we propose a computationally efficient exploration scheme employing a Fast Multipole accelerated Boundary Element Method scheme, that enjoys both linear complexity w.r.t. the boundary's size as well as linear memory requirements. Furthermore, we devise an adaptive control law for the specified boundary conditions that allows us to tune the robot's behavior without affecting the inherent safety and convergence properties of the underlying potential field. Finally, we validate the performance of the proposed exploration scheme through extensive simulations in realistic environments.

# Περίληψη

Παρά το γεγονός ότι τα αρμονικά δυναμικά πεδία συνιστούν ένα ισχύρο εργαλείο για την αντιμετώπιση του προβλήματος της αυτόνομης ρομποτικής εξερεύνησης, η εφαρμοσιμότητα τους περιορίζεται στην πράξη από το υψηλό υπολογιστικό κόστος που απαιτείται για την επίλυση της εξίσωσης Laplace σε πραγματικό χρόνο. Στην παρούσα εργασία, προτείνουμε μια υπολογιστικά αποδοτική μέθοδο εξερεύνησης η οποία βασίζεται στην Fast Multipole accelerated Boundary Element Method, η οποια χαίρει γραμμικής πολυπλοκότητας αναφορικά με το μέγεθος του συνόρου καθώς και γραμμικές απαιτήσεις μνήμης. Επιπροσθέτως, σχεδιάζουμε έναν προσαρμοστικό νόμο ελέγχου για τις επιβαλλόμενες συνοριακές συνθήκες ο οποίος μας επιτρέπει να ρυθμίσουμε την συμπεριφορά του ρομπότ χωρίς να επηρεάζονται οι εγγενείς ιδιότητες ασφάλειας και σύγκλισης του υποκείμενου δυναμικού πεδίου. Τέλος, επιβεβαιώνουμε τις επιδόσεις του προτεινόμενου αλγορίθμου μέσω εκτενών προσομοιώσεων σε ρεαλιστικά περιβάλλοντα.

# Acknowledgements

Firstly, I would like to thank Prof. Kostas Kyriakopoulos for giving me the opportunity to pursue my diploma thesis at the Control Systems Lab. His commitment to detail and mathematical rigor throughout this thesis, and delivered courses, has inspired me to embrace such a mindset myself. Further, I would like to thank him for many insightful conversations and advice regarding career and future choices.

I want to acknowledge my collaborators and lab-mates Panagiotis Vlantis, Charalampos Bechlioulis and Constantinos Vrohidis for the long hours spent in the lab. Specifically, I want to thank Panagiotis Vlantis for his invaluable support, ideas, directions, for helping me identify mistakes and providing ample guidance. Next, I want to thank Charalampos Bechlioulis for his technical advice and encouragement. Last but not least, I want to thank Constantinos Vrohidis for assisting in a number of issues regarding this thesis, and for the fruitful, and fruitless, mathematical and philosophical conversations.

Finally, I want to thank my family and friends for their motivation and wholehearted support throughout my studies.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Statement

The autonomous robotic *exploration* problem constitutes the main topic of this thesis. The exploration problem consists of a mobile robot, equipped with range sensors allowing it to perceive a subset of its surroundings, that navigates an initially unknown environment with the purpose of incrementally mapping it.

Although the motion of mobile robots is usually subjected to kinematic constraints, this nonholonomic aspect of the exploration problem is typically neglected. Thus, it is assumed that we are capable of directly controlling the velocity of the robot. Moreover, in this work we consider that the robot can localize itself perfectly inside its environment. This assumption is not unrealistic as, in general, the utilization of appropriate *simultaneous localization and mapping* (SLAM) algorithms provides reliable estimates of the robot's configuration.

A variety of range sensors can be employed for the exploration task, such as laser, ultrasonic or infrared, each with its corresponding advantages and shortcomings. Nevertheless, all types of proximity sensors allow the robot to determine the position of obstacles within the range of the sensor. The robot's sensing region is typically a disk, for a $360^o$ angle of view, or a circular sector, corresponding to a limited angle of view. We note that the robot's perceived space is subject to line-of-sight constraints or occlusions, i.e., the sensors cannot observe areas behind obstacles (see Figure 1.1). Overall, as the robot traverses its environment, the onboard range sensors enable it to determine which regions are occupied by obstacles and which ones are free.

The ultimate goal of autonomous exploration is to devise an algorithm capable of navigating a robot in order to map the entirety of its environment, without any human input. The partial map constructed by the robot while exploring can be employed to identify unexplored frontiers, i.e., free regions adjacent to unobserved areas, that will allow acquisition of new information. Furthermore, obstacle avoidance should be maintained in order to ensure

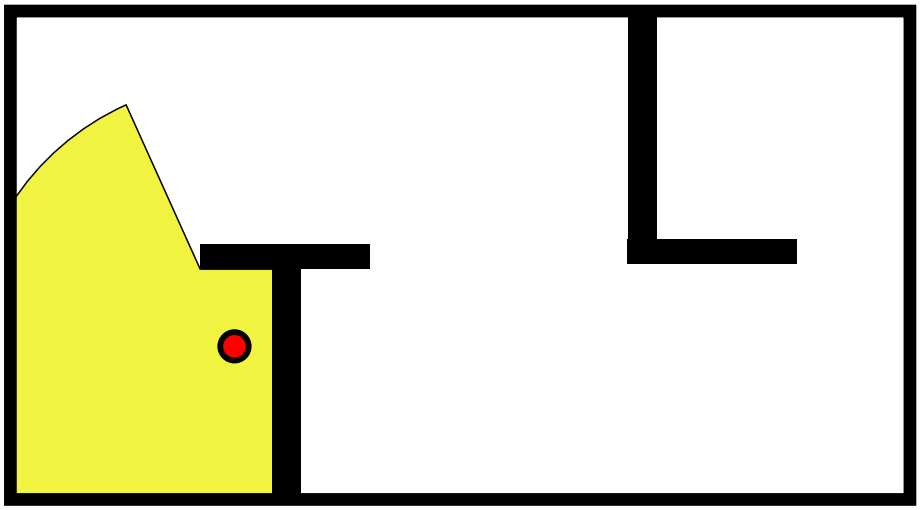


Figure 1.1: The robot's position is shown as a red circle while, the yellow shaded area corresponds to the robot's sensing region.

that the robot remains safe throughout its trajectory. Due to the inherent reactive nature of this problem, arising from the partial knowledge of the environment, it is imperative for exploration schemes to allow for rapid integration of accumulated information, to satisfy the aforementioned safety requirements, as well as to achieve efficient navigation. Hence, computational efficiency is a critical component of any decent exploration algorithm.

## 1.2 Motivation

The availability of accurate representations of the environment constitutes a necessity for the robust and safe execution of a variety of robotic tasks. In particular, path and motion planning algorithms require that a map of the robot's workspace is provided. Although, in some cases, such information can be found in the architectural blueprints of a building, the latter tend to be inaccurate, as they do not account for human-induced changes of the floorplans, such as furniture or offices. Besides, even if these features were incorporated in preexisting maps, they would still remain unreliable because any environment populated by humans is highly dynamic, thus, necessitating frequent updates and refinements of its map.

The aforementioned requirements can be satisfied by employing autonomous robotic exploration schemes to generate faithful maps, without relying on a human operator to manually navigate the robot. The autonomy involved is beneficial, since providing human input may be costly, tedious and, in certain instances, unavailable. In particular, exploration algorithms are well-suited for environments that are unreachable by humans, such as

areas which are hazardous or deprived of adequate wireless communication coverage, hence, restricting the use of teleoperation.

Apart from map construction, robotic exploration can also be applied to different occasions. For example, exploring an initially unknown environment is equivalent to the task of reaching a goal configuration without any prior knowledge or any real-time indication, e.g. by sensors, regarding its location. A particular instance of the previous problem are search-and-rescue operations.

## 1.3 Literature Review

Several methodologies have been proposed in the related literature to address the exploration problem. Specifically, a well-studied approach focuses on determining future robot configurations, and corresponding paths to reach them, that potentially maximize information gain. To that end, a number of candidate points are generated, in a manner similar to frontier-based methods [4], and the predicted information gain associated with each point is approximated by means of an information-theoretic utility function. Typically, utility functions incorporate three measures to evaluate candidate vantage points:

- Map entropy, which is associated with uncertainty in the map, thus driving the robot to explore,
- Localization, that captures the uncertainty regarding the position of the robot, as a consequence of odometric errors, and
- Cost, described by the distance from the robot to each goal configuration.

In [1] a utility function based on entropy reduction of a Rao Blackwellized Particle Filter is used to balance between exploration and loop-closing. The generated candidate configurations using the previous approach, as well as their expected utility are shown in Figure 1.2. Charrow et al. [5] employ the Cauchy-Schwarz quadratic mutual information to

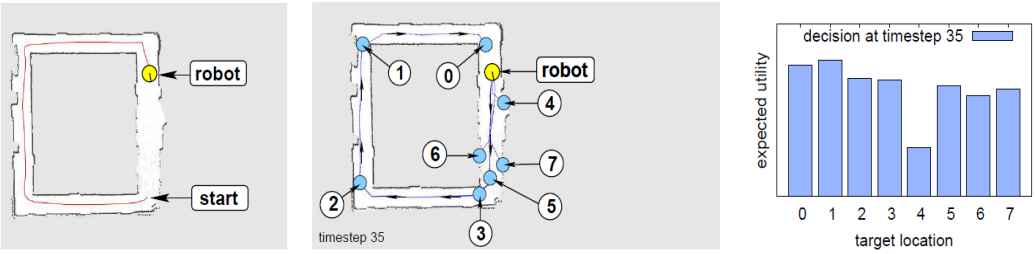



Figure 1.2: Candidate future robot positions and their anticipated utility, as computed by the algorithm described in [1].

approximate the information gain and, also, they develop an approach that accounts for the dependence between distinct sensor readings and apply their algorithm to both ground and aerial vehicles. In a recent work [3], the authors trained a Gaussian Process using Bayesian Optimization to estimate the Mutual Information of candidate vantage points. An in-depth comparison of such methodologies can be found in [6], [7].

Another popular class of algorithms utilizes artificial potential fields to safely guide the robot towards the unexplored regions of the workspace. Generally, artificial potential fields suffer from the existence of spurious local minima that may lead the robot in undesired configurations. On the other hand, harmonic potential fields (HPFs) are free of local unwanted attractors in their domain's interior by design. In [8], the authors use HPFs with Dirichlet boundary conditions calculated through relaxation methods to explore 2D workspaces. The imposed boundary conditions maintain a uniform high (resp. low) value on obstacles (resp. unexplored frontier), thus inducing an orthogonal gradient on the boundary. However, this approach generally exhibits large flat areas of the potential, thus, resulting in very small gradients. Shade et al. [2] employ HPFs to navigate a mobile robot equipped with a stereo camera to explore a 3D environment, represented as an octree. Nevertheless, the construction of the underlying harmonic potential scales poorly w.r.t. the domain's size. Finally, in [9], the computational cost is mitigated by computing the HPF only on a local window around the robot, whereas the Full Multigrid Method was exploited in [10] for path planning.

It should be noted though, that in all aforementioned HPF-based works the potential was calculated using finite difference schemes, which suffer from the need to discretize an entire 2D domain. Moreover, as stated in [6], only one type of sources and sinks can be utilized, thus preventing the prescription of relative strength among boundaries.

## 1.4 Our Approach

In this work, we propose a controller based on HPFs for navigating a mobile robot, to explore an initially unknown, compact workspace, employing the Fast Multipole accelerated Boundary Element Method (FMBEM) [11], for the construction and online update of the HPF used for navigation, that yields linear complexity in terms of computational effort and required memory. We remark that, to the best of the authors' knowledge, this is the first time that FMBEM is employed in a motion planning scheme and provides a substantial improvement in the scalability of HPF-based approaches. Furthermore, we impose Neumann boundary conditions for the solution of the Laplace equation, for which we design suitable adaptive laws, similarly to [12], thus enabling us to improve the robot's behavior, up to a certain degree, without compromising the inherent safety and convergence properties of the underlying HPF.

Next, we present the main contributions of the proposed control scheme:

1. We leverage a number of appealing properties associated with FMBEM. Firstly, the

linear complexity and memory requirements allow us to handle large domains without degradation of the algorithm's performance. Secondly, the use of the Boundary Element Method requires only a discretization of the domain's boundary which, generally, is more sparse and straightforward than discretizing the entire workspace. Finally, the potential and gradient of the HPF can be readily evaluated at any interior point of the domain employing the FMBEM, as opposed to finite differences approaches that obtain these values solely on nodal points.

2. The adaptive law for the boundary conditions enables us to prescribe, up to a certain degree, a specific robot behavior. In the related literature, the common choice of uniform Dirichlet boundary conditions along obstacles and frontiers is restrictive, as it is unable to impose preferences among boundary regions.

3. We provide a rigorous analysis and proofs of our controller's safety and convergence properties, that guarantee the solution of the exploration problem.

The basic idea of using HPFs to guide robotic exploration is shown in Figure 1.3. A vector field is generated on the explored part of the workspace (shown in white), leading the robot towards the unexplored environment (gray) while avoiding obstacles (black).

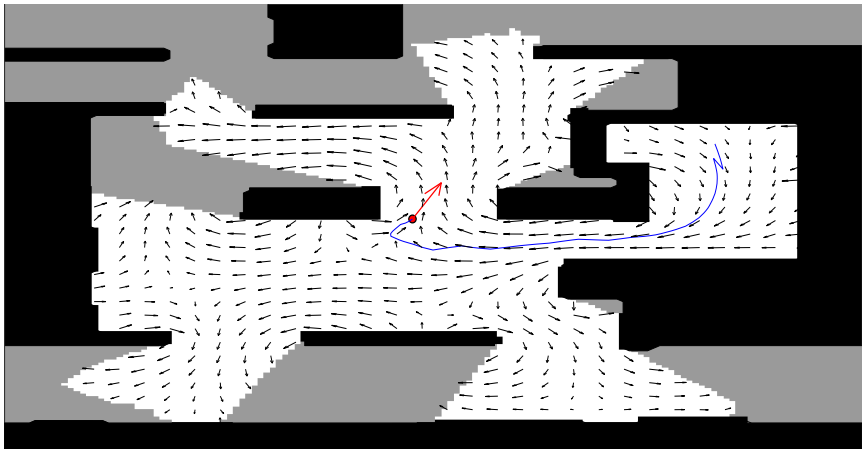


Figure 1.3: A snapshot of the generated occupancy grid map and the vector field navigating the robot, on top of the actual workspace.

## 1.5 Thesis Structure

The remainder of this thesis is organized as follows. In Chapter 2 we formulate the exploration problem in formal mathematical terminology. In Chapter 3 we briefly introduce the Fast Multipole Method, we describe our proposed controller and analyze its stability

properties, while in Chapter 4 we discuss the implementation of our control scheme. In Chapter 5 we demonstrate the performance of our controller through extensive simulation studies and compare the obtained results with the existing literature. Finally, in Chapter 6 we conclude by discussing the degree up to which the set goals were attained, the problems and difficulties that arose in the design process and propose future research directions.

# Chapter 2

# Problem Formulation

In this chapter we formally state the exploration problem. Particularly, in Section 2.1 we introduce the used mathematical notation, and in Section 2.2 we define the fundamental notions and components of the problem and discuss the constraints imposed on the robot's motion and sensing capabilities. In Section 2.3, we describe how the accumulated knowledge about the environment evolves during the execution of the task and, also, we explicitly state the requirements and goals of exploration. Finally, the method employed for the representation of the incrementally constructed map is presented in Section 2.4.

## 2.1 Mathematical Notation

We use $\|v\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$ to denote the euclidean norm of the vector $v \in \mathbb{R}^n$. We, also, define the disk centered at $p \in \mathbb{R}^2$ with radius $r > 0$ as $\mathcal{B}(p, r) = \{q \in \mathbb{R}^2 : \|p-q\| \leq r\}$. Given two sets $\mathcal{A}, \mathcal{B}$ we denote the interior, boundary, and the difference of $\mathcal{B}$ from $\mathcal{A}$ as $\text{int}\mathcal{A}, \partial\mathcal{A}, \mathcal{B} \setminus \mathcal{A}$, respectively. We use $\nabla_p\phi, \nabla_p^2\phi$ to denote the gradient and the Laplace operator of the scalar $\phi$ w.r.t. the vector $p$.

## 2.2 Exploration Model

Let $\mathcal{W}$ be a compact and connected subset of $\mathbb{R}^2$, referred herein as the workspace, and let $\mathcal{W}_f \triangleq \text{int}\mathcal{W}$. We consider a point robot which is allowed to move inside $\mathcal{W}_f$, whose motion obeys the single integrator kinematic model:

$$\dot{p} = u \tag{2.1}$$

where $p = [x, y]^T \in \mathbb{R}^2$ denotes the robot's position and $u \in \mathbb{R}^2$ denotes the corresponding control input vector. We assume that the robot is equipped with sensors that allow it to perceive a subset of its surroundings given by:

$$\mathcal{S}(p) = \{q \in \mathcal{W} : q \in \mathcal{B}(p, r) \wedge \mathcal{L}(p, q) \subseteq \mathcal{W}\} \tag{2.2}$$

with $\mathcal{B}(p, r)$ denoting the disk centered at $p$ with radius $r > 0$ (modeling sensor range constraints) and $\mathcal{L}(p, q)$ denoting the line segment connecting the two points $p$ and $q$ inclusively (corresponding to line-of-sight sensor constraints). Also, we define $\mathcal{P}(t_i, t_f)$ as the path traversed by the robot during the time interval $[t_i, t_f]$; for brevity, when $t_i = 0$ and $t_f > 0$, we will write $\mathcal{P}(t_f)$ instead of $\mathcal{P}(0, t_f)$. Given a continuous robot path $\mathcal{P} \subset \mathcal{W}_f$, we define the explored region of $\mathcal{W}$ as follows:

$$\mathcal{E}(\mathcal{P}) = \cup_{p \in \mathcal{P}} \mathcal{S}(p) \tag{2.3}$$

Considering the boundary of the explored region at some time instance, we note that $\partial\mathcal{E} = \partial\mathcal{E}_F \cup \partial\mathcal{E}_O$, with $\partial\mathcal{E}_F$ and $\partial\mathcal{E}_O$ belonging to the robot's free space and workspace boundaries, i.e., $\partial\mathcal{E}_F \subset \mathcal{W}_f$ and $\partial\mathcal{E}_O \subseteq \partial\mathcal{W}$, respectively. In general, each of $\partial\mathcal{E}_F$ and $\partial\mathcal{E}_O$ consists of zero or more disjoint arcs, i.e.: $\partial\mathcal{E}_F = \cup_{i \in \mathcal{I}_F} \partial\mathcal{E}_{Fi}$ and $\partial\mathcal{E}_O = \cup_{i \in \mathcal{I}_O} \partial\mathcal{E}_{Oi}$ with $\mathcal{I}_F = \{1, 2, \ldots, N_F\}$ and $\mathcal{I}_O = \{1, 2, \ldots, N_O\}$ being an indexing of the respective set. We additionally define a parametrization $\sigma : [0, 1] \mapsto \partial\mathcal{E}$ of the explored region's boundary. To clarify the notions introduced in this section, we provide an intuitive demonstration in Figure 2.1.

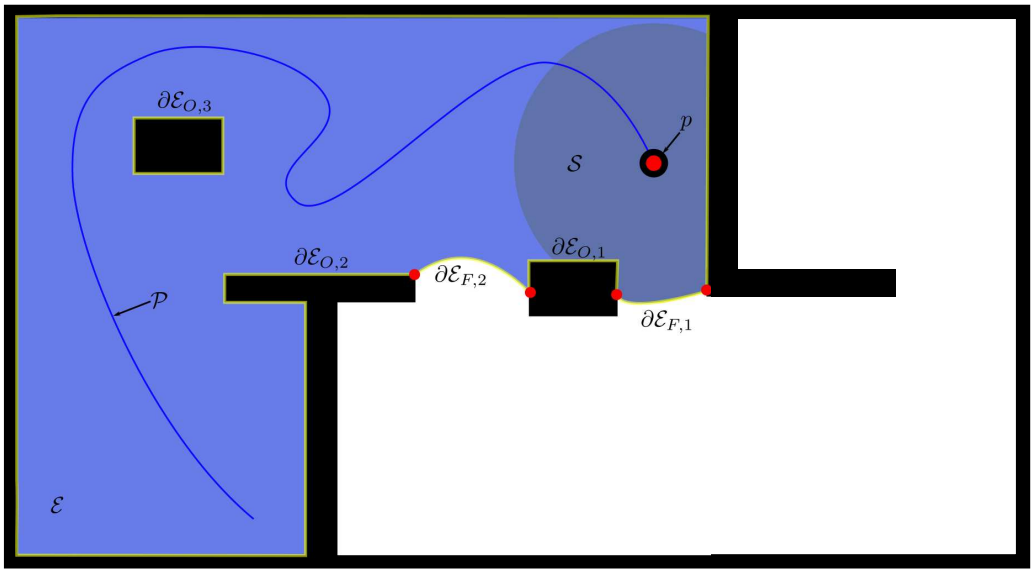



Figure 2.1: This figure illustrates the different components of the exploration process. The blue and gray regions correspond to $\mathcal{E}$ and $\mathcal{S}$ respectively, while the yellow lines represent $\partial\mathcal{E}$, and the different parts comprising the boundary are annotated appropriately. The unexplored parts of the workspace are shown in white.

## 2.3 Evolution of the Boundary

We provide an insight on the evolution of $\partial\mathcal{E}$, along the robot's trajectory. Firstly, we note that any point $q \in \partial\mathcal{E}_F$ that enters the robot's sensing region becomes instantly part of int$\mathcal{E}$. In this manner, as the robot approaches $\partial\mathcal{E}_F$, the latter *expands*, i.e., the area of $\mathcal{E}$ increases. On the other hand, occupied boundary points will remain on the boundary, since it holds that if $q \in \partial\mathcal{W} \cap \mathcal{S}$ then $q \in \partial\mathcal{S}$ (assuming that the sensors provide consistent measurements and the workspace is static). In this context, we will also elaborate on some properties of $\sigma$. Specifically, we define $\sigma$ in such a way that points $q \in \partial\mathcal{E}_O$, are associated with a constant parametrization index $i_\sigma \in [0, 1]$. Conversely, assume that $\partial\mathcal{E}_E \subseteq \partial\mathcal{E}_F$ is a set that undergoes expansion under which $\partial\mathcal{E}_E$ ends up in $\partial\mathcal{E}'_E$. Our parametrization ensures that the two aforementioned sets are mapped in the same part of the domain of $\sigma$, i.e., the sets $\sigma^{-1}(\partial\mathcal{E}_E), \sigma^{-1}(\partial\mathcal{E}'_E)$ are identical. This procedure, as well as the expansion of the boundary are visualized in Figure 2.2.

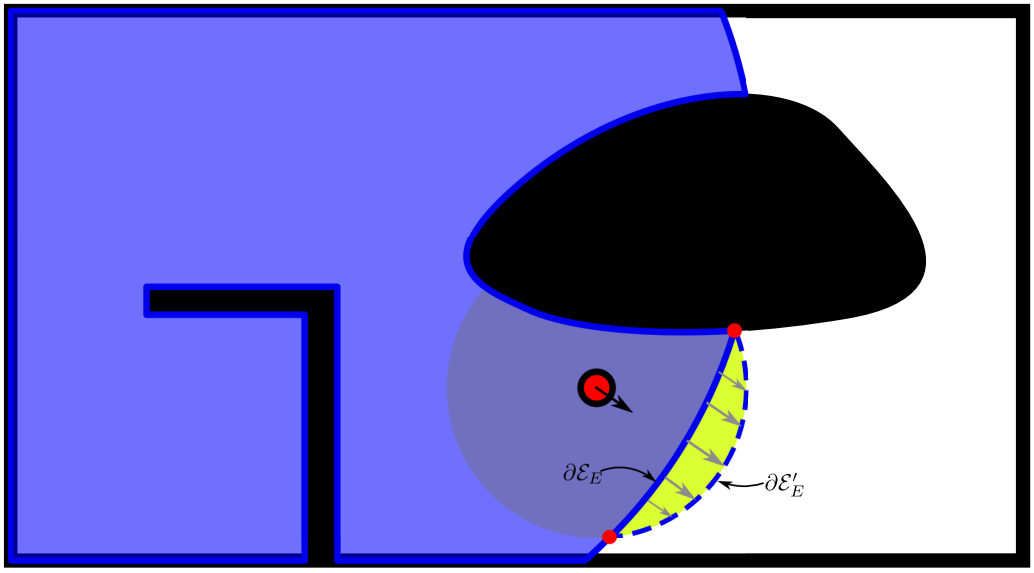



Figure 2.2: The expansion of the boundary and the correspondence of boundary regions before and after the expansion. The robot's motion, and the consequent motion of the sensors, induces the inclusion of the yellow area in the explored region (blue) and, also, causes the region $\partial\mathcal{E}_E$ to end up in $\partial\mathcal{E}'_E$.

In this work, our goal is to design a control law $u = f_p(p, t, \mathcal{E})$ such that there exists a finite time instant $T > 0$ for which:

$$\mathcal{E}(\mathcal{P}(t)) = \mathcal{W}, \ \forall t \geq T \quad . \tag{2.4}$$

In addition, ensuring the safety of the robot's trajectory throughout the exploration is of

paramount importance. The exploration scheme is safe if the following condition is satisfied:

$$p(t) \in \mathcal{W}_f, \ \forall t \geq 0 \ . \tag{2.5}$$

**Remark 1.** *The proposed algorithm can be readily applied for a disk robot through contraction, in the Minkowski difference sense, of $\mathcal{E}$ by the robot's radius (see Figure 2.3). The actual, non-contracted map can be retrieved by simply storing the sensor readings and the robot's trajectory; but, it might differ from $\mathcal{W}$ because some of its regions may be unreachable by the disk robot, e.g. due to narrow doorways.*

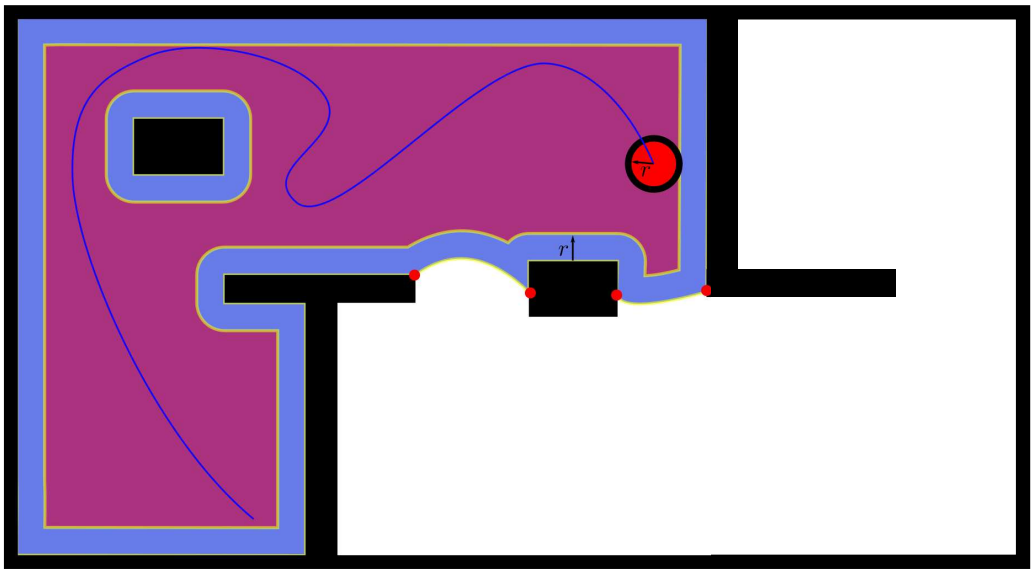



Figure 2.3: The actual and the contracted explored region for a disk robot, with radius $r$, shown in blue and red respectively.

## 2.4 Occupancy Grid Mapping

The classification of boundary points as occupied or free is achieved using an occupancy grid map representation [13]. Particularly, a grid is embedded in the explored region $\mathcal{E}$, consisting of cells $m_i$ centered at $c_i = [c_{x,i}, c_{y,i}]^T$, $i \in \mathcal{I}_G$ with $\mathcal{I}_G = \{1, 2, \ldots, N_G\}$. Every grid cell is assigned a probability of occupancy, denoted $\Pr(m_i)$, which is inherited by any boundary points lying inside the cell, i.e. for some $q \in \mathcal{E}$, $\Pr(q) = \Pr(m(q))$ where

$$m(q) = m_i, \ i \in \mathcal{I}_G : \|c_{x,i} - q_x\| \leq \frac{m_r}{2} \wedge \|c_{y,i} - q_y\| \leq \frac{m_r}{2} \tag{2.6}$$

with $m_r$ denoting the grid resolution. Given $\Pr(q),\ q \in \partial\mathcal{E}$ we can formally define the occupied boundary as follows:

$$\partial\mathcal{E}_O = \{q \in \partial\mathcal{E} : \Pr(q) > \alpha\} \tag{2.7}$$

where the constant $\alpha \in (0.5, 1)$ defines a threshold for classifying a point as occupied. Obviously, it holds that $\partial\mathcal{E}_F = \partial\mathcal{E} \setminus \partial\mathcal{E}_O$.

We denote $\Pr = [\Pr(m_1), \Pr(m_2), \ldots, \Pr(m_{N_G})]^T$ the vector containing the probability of each cell in the grid. Furthermore, $\Pr(m_i)$ is updated through the following equation

$$\dot{\Pr}(m_i) = f_{\Pr}(p, c_i) \tag{2.8}$$

where $f_{\Pr}$ is the inverse sensor model. We omit an explicit definition of $f_{\Pr}$, as it is beyond the scope of this thesis, but we note that for a variety of reasonable choices (see Chapter 9 in [14]), $f_{\Pr}$ is a piece-wise continuous function.

# Chapter 3

# Technical Solution

This chapter elaborates on the proposed control scheme and constitutes the main contribution of this thesis. Specifically, in Section 3.1 we describe our controller and its 3 major components, i.e., the Fast Multipole accelerated Boundary Element Method, the Velocity Control Law, and the Adaptive Law for the imposed boundary conditions. The safety, convergence and stability properties of our control scheme are analyzed in Section 3.2. Finally, in Section 3.3 an analytical comparison between our methodology and prior approaches in the related literature is provided.

## 3.1 Control Design

The main idea is to address the exploration problem by designing a control law $u = f_p(p, t, \mathcal{E})$, based on a HPF $\phi(p, t)$ defined within the explored region $\mathcal{E}$, such that $\partial\mathcal{E}_F$ is rendered attractive and $\partial\mathcal{E}_O$ is rendered repulsive. Such a potential field $\phi(p, t)$ can be constructed by solving the Laplace's equation

$$\nabla_p^2 \phi(p) = 0, \quad \forall p \in \mathcal{E} \tag{3.1}$$

subject to Neumann boundary conditions

$$\frac{\partial \phi}{\partial n} = k\left(\sigma^{-1}(q), t\right), \quad \forall q \in \partial\mathcal{E} \tag{3.2}$$

where $\nabla^2$ denotes the Laplace operator, and $n$ denotes the exterior normal at the corresponding boundary position. Effectively, $k\left(\sigma^{-1}(q), t\right)$ specifies how attractive ($k < 0$) or repulsive ($k > 0$) a point $q \in \partial\mathcal{E}$ should be. In order for the Neumann problem to admit a solution, the boundary conditions should satisfy the constraint $\int_{\partial\mathcal{E}} k ds = 0$. Thus, we refer to a set of boundary conditions satisfying the previous constraint as *compatible*. Notice that solutions of the Laplace equation, i.e., HPFs, bear properties that render them appealing for navigation. Particularly, the Maximum-Minimum Principle dictates that local extrema

can only be found in the domain's boundary, effectively rendering all critical points of $\phi$ in int$\mathcal{E}$ saddles.

To overcome the implications following the fact that the aforementioned dynamical system is infinite-dimensional, we design boundary conditions such that $k = k(\sigma^{-1}(q), t; \hat{k})$ where $\hat{k}$ is a finite-dimensional vector of parameters, referred to as control points, that are used for adjusting $k$. Particularly, the $i$-th component of $\hat{k}$, denoted $\hat{k}_i$, corresponds to the value of the distribution $k$ at a point $q_i$ on the boundary of $\mathcal{E}$, with $i \in \mathcal{I}_{CP} = \{1, 2, \ldots, N_{CP}\}$. For any boundary point $q$, with $q \neq q_i$, we choose $k\left(\sigma^{-1}(q), t\right)$ as a linear combination of nearby control point values. The discretization scheme is illustrated in Figure 3.1. In order to maintain a fine discretization of the boundary, the number of control points $N_{CP}$ varies proportionally to the length of $\partial\mathcal{E}$. In addition, we equip the specified boundary conditions $\hat{k}$ with appropriately designed adaptive laws, such that the attractiveness of the explored boundary is as specified above and no additional (locally) stable equilibria appear in int$\mathcal{E}$.

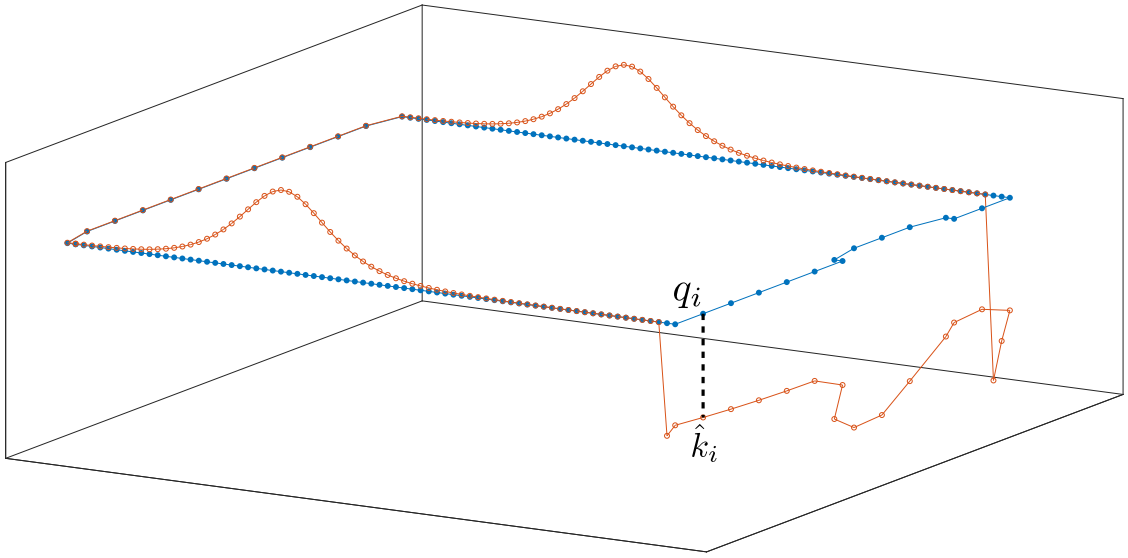



Figure 3.1: The discretization of boundary values using the control points $\hat{k}_i$ on the boundary points $q_i$. The blue line and dots represent $\partial\mathcal{E}$ and the boundary points $q_i$, respectively. The green line illustrates $k$, and the control points $\hat{k}_i$ are shown as green circles.

### 3.1.1 Fast Multipole Boundary Element Method

In this subsection, we briefly describe the FMBEM and its properties (for more details refer to [11], [15]). The main idea of FMBEM is to reduce the computational cost of the conventional BEM, which is $\mathcal{O}(n^2)$ where $n$ is the number of boundary elements used for

approximating the domain's boundary, by computing approximate solutions of the corresponding BVP with a specified error. Hence, a hierarchical subdivision of the domain into cells is derived via quadtree decomposition and the interaction between elements residing in "distant" cells is approximated using multipole expansion. This approach enables us to avoid the analytical computation of every pairwise element interaction at the expense of a bounded approximation error, thus, reducing the computational complexity to $\mathcal{O}(n)$. Furthermore, the need to store the dense and non-symmetric matrix involved in the conventional BEM is circumvented in FMBEM by employing an iterative solver, e.g., GMRES [16], consequently achieving $\mathcal{O}(n)$ memory requirements.

### 3.1.2 Velocity Control Law

We equip our robot with the following control law for its velocity:

$$u = f_p(p, \hat{k}) = -K_u s(p)\ \nabla_p \phi(p, \hat{k})\ , \tag{3.3}$$

where $K_u$ is a scaling constant and $s(p)$ is a function that reduces the velocity magnitude as the robot approaches the boundary. Concretely, we define

$$s(p) = S_{R_1}\left(d(p, \mathcal{E})\right) \tag{3.4}$$

where $d(p, \mathcal{E}) = \min_{q \in \partial\mathcal{E}} \|p - q\|$ is the minimum distance between the robot and the boundary, and $S_a$ is a $C^1$ bump function:

$$S_a(x) = \begin{cases} 1, & x > a \\ 3(\frac{x}{a})^2 - 2(\frac{x}{a})^3, & 0 \leq x \leq a \\ 0, & x < 0 \end{cases}\ . \tag{3.5}$$

Plots of $S_a(x)$ for a selection of parameters are shown in Figure 3.2. The positive constant $R_1$ serves as a distance threshold, slowing down the robot when it is located within distance $R_1$ of the boundary.

### 3.1.3 Adaptive Law

We now present the adaptive laws for the prescribed boundary values $\hat{k}$ which are given by:

$$\dot{\hat{k}} = f_b(p, \hat{k}, \mathrm{Pr}) = c\mu(\hat{k}_t - \hat{k}) + b_e(\hat{k}_t - \hat{k})\ , \tag{3.6}$$

where $c, \mu, \hat{k}_t, b_e$ are functions to be defined later. Essentially, the function $\hat{k}_t$ acts as a reference value for the corresponding control parameter whereas $\mu$ is used for adjusting the convergence rate. Finally, $c$ renders the first term of (3.6) null when the robot is near a critical point of $\phi$ and $b_e$ ensures that the trajectories of the robot remain safe at all time.

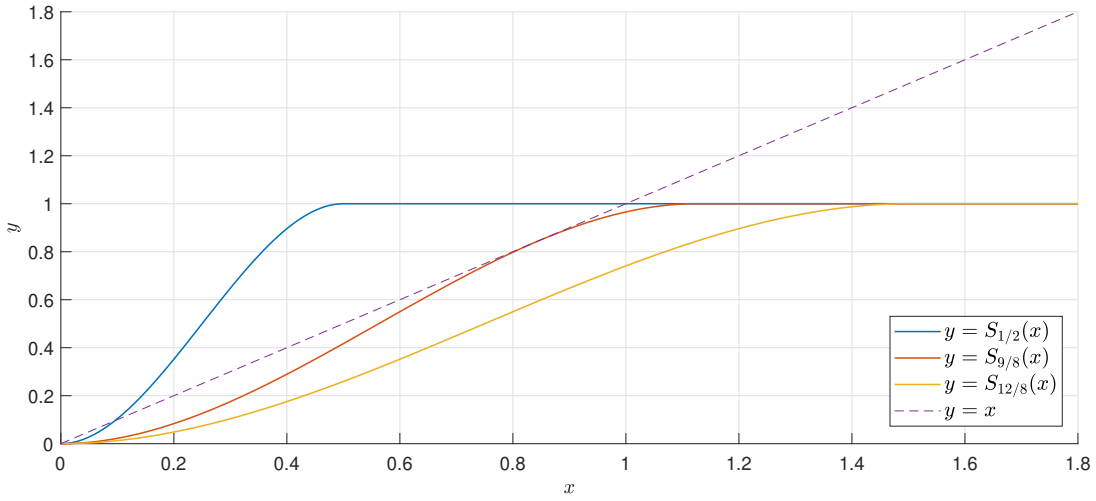



Figure 3.2: The plot of $S_a(x)$ for different parameters $a$. Note that, for $a > 9/8$ the function's value is smaller than its argument.

The desired values $\hat{k}_t$ are defined in terms of the auxiliary function $k'$ which is given by:

$$k' = k'_1 \cdot k'_2 \tag{3.7}$$

where

$$k'_1(q) = \begin{cases} S_{1-\alpha}(\Pr(m(q)) - \alpha), & \Pr(m(q)) \geq \alpha \\ -S_\alpha(\alpha - \Pr(m(q)), & \Pr(m(q)) < \alpha \end{cases} \tag{3.8}$$

$$k'_2(q) = \mathrm{e}^{-\frac{4 \cdot \|p-q\|}{r}} \tag{3.9}$$

The purpose of $k'_1$ is to increase (resp. decrease) the "attractiveness" of the corresponding boundary point proportionally to the probability of it being part of the free boundary (resp. being occupied by an obstacle). Boundary points that are close to the robot are favored through $k'_2$. The purpose of $k'_2$ is to render small areas in the vicinity of the robot more attractive than larger areas far away, e.g. an unexplored corner of the room the robot is currently in, thus sparing the robot from returning back at a later time.

Moreover, to ensure that the desired boundary conditions are compatible, we first define the sets $\partial\mathcal{E}_p = \{q \in \partial\mathcal{E} : k'(q) \geq 0\}, \partial\mathcal{E}_n = \{q \in \partial\mathcal{E} : k'(q) < 0\}$ and the integral of boundary values on each set $I_i = \int_{\partial\mathcal{E}_i} k'(s)ds,\ i \in \{p, n\}$. Then, the desired boundary conditions are chosen as follows:

$$\hat{k}_{t,i} = \frac{\bar{k}}{k'_m \cdot I(q_i)} \cdot k'(q_i),\ \forall i \in \mathcal{I}_{CP} \tag{3.10}$$

where $\bar{k}$ are constant absolute upper bounds for $\hat{k}_t$,

$$I(q) = \begin{cases} I_p, & q \in \partial\mathcal{E}_p \\ -I_n, & q \in \partial\mathcal{E}_n \end{cases} \tag{3.11}$$

scales $k'$ according to which set each point belongs to, and finally, $k'_m = \max_{q\in\partial\mathcal{E}} \left( \| \frac{k'(q)}{I(q)} \| \right)$. At this point, we stress out that we deploy our algorithm after the robot has observed a region of $\partial\mathcal{W}$, which can be achieved for instance by instructing the robot initially to move with a constant fixed velocity until the robot has encountered an obstacle, in order to guarantee that $I_p \neq 0$. Similarly, our control law terminates once $\Pr(q) > \alpha'$, $\forall q \in \partial\mathcal{E}$ where $\alpha' \in (0.5, \alpha)$, which ensures that $I_n \neq 0$ and that $\partial\mathcal{E}$ coincides with $\partial\mathcal{W}$ with a probability of at least $\alpha'$.

Next, we define $c$ appropriately as follows to ensure that the first term of $f_b$ vanishes in the neighborhood of critical points of $\phi$:

$$c = S_{\epsilon_w}(\|\nabla_p \phi\| - \epsilon_1) \tag{3.12}$$

where $\epsilon_1, \epsilon_w$ are small positive constants. Also, we control the rate of the adaptive law through the function $\mu$, which is given by:

$$\mu = S_{\mu_1}\left( \frac{K_u s \|\nabla_p \phi\|^2}{\| \frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k}) \| + \epsilon_2} \right) , \tag{3.13}$$

where $\mu_1 > \frac{9}{8}$ ensures that the function's value is smaller than its argument (the proof is provided in Subsection 3.2.1 and is intuitively illustrated in Figure 3.2), and $\epsilon_2 > 0$ renders the above expression's denominator strictly non-zero. Notice that the term $\frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k})$ is proportional to $\frac{\partial \phi}{\partial \hat{k}} \frac{d\hat{k}}{dt}$, which in turn corresponds to the change of the HPF's values, at $p$, induced by the change of boundary conditions. In order to devise a practical way for calculating the aforementioned derivatives, we recall that the solution of the Laplace equation is obtained by $H \cdot \hat{\phi} = G \cdot \hat{k}$ [17], and the potential at a point in the domain's interior is approximated by

$$\begin{aligned} \phi(p) &= H_p \cdot \hat{\phi} - G_p \cdot \hat{k} \\ &= (H_p \cdot H^{-1} \cdot G - G_p) \cdot \hat{k} \end{aligned} , \tag{3.14}$$

where $\hat{\phi}$ corresponds to the HPF values at the control points, and $H, H_p, G, G_p$ are matrices/vectors of appropriate dimensions, whose details we omit for the sake of brevity. From the previous equation it becomes apparent that, $\frac{\partial \phi}{\partial \hat{k}} \frac{d\hat{k}}{dt}$ can be computed by solving the Laplace equation with boundary conditions $\frac{\partial \phi}{\partial n} = \frac{d\hat{k}}{dt} \propto \hat{k}_t - \hat{k}$. As we can see, the potential depends linearly on the imposed boundary conditions.

Finally, we elaborate more on the term $b_e$ after introducing the following definition. We say that $\phi$ is *safe* at some time instant, when no point belonging to an obstacle's boundary is attractive, i.e., $\hat{k}_i \geq 0,\ \forall i \in \mathcal{I}_{CP}$ such that $q_i \in \partial\mathcal{E}_O$ (assuming that a sufficiently fine discretization of the boundary has been used). Generally, $\phi$ will not always be safe under the proposed control scheme because regions of the free boundary will be replaced by newly discovered obstacles during exploration. Hence, to render the potential field safe in finite time and avoid potential collisions between the robot and newly discovered obstacles, we define $b_e$ as follows:

$$b_e(\mathcal{E}, \hat{k}, \mathrm{Pr}) = \begin{cases} 0, & \hat{k}_i \geq 0,\ \forall i \in \mathcal{I}_{CP} : q_i \in \partial\mathcal{E}_O \\ 1, & \text{otherwise} \end{cases} , \tag{3.15}$$

for reasons that will become apparent in the following section.

## 3.2 Stability Analysis

In this section, we elaborate on the safety and convergence properties of the proposed control scheme. We recall that, the overall dynamical system, whose state is denoted $z$ and consists of $p$, $\hat{k}$, and Pr is described by the following differential equations

$$\dot{z} = \begin{bmatrix} \dot{p} \\ \dot{\hat{k}} \\ \dot{\mathrm{Pr}} \end{bmatrix} = \begin{bmatrix} f_p(p, \hat{k}, \mathrm{Pr}) \\ f_b(p, \hat{k}, \mathrm{Pr}) \\ f_{\mathrm{Pr}}(p, \hat{k}, \mathrm{Pr}) \end{bmatrix} = f_z(z) \ . \tag{3.16}$$

First, we shall state some useful properties regarding $k$ and $b_e$ that will be used in the subsequent analysis.

**Proposition 1.** *Assuming that $k$ is compatible for $t = 0$, and under the adaptive law (3.6), it is guaranteed that $k$ will remain compatible for all time.*

*Proof.* We define $I_c \triangleq \int_{\partial\mathcal{E}} k(s)ds$. Notice that the following holds:

$$\begin{aligned} I_c &= \int_{\partial\mathcal{E}} \sum_{i \in \mathcal{I}_{CP}} \mathcal{N}_{SF,i}(s)\hat{k}_i ds \\ &= \sum_{i \in \mathcal{I}_{CP}} \left( \int_{\partial\mathcal{E}} \mathcal{N}_{SF,i}(s)ds \right) \hat{k}_i = l \sum_{i \in \mathcal{I}_{CP}} \hat{k}_i \end{aligned} \tag{3.17}$$

where $\mathcal{N}_{SF,i}$ is a typical linear shape function, applied to each control point, that integrates to $l$, denoting the length of boundary elements. Then, we compute the time derivative of $I_c$, substituting $\dot{\hat{k}}_i$ from (3.6):

$$\dot{I}_c = \dot{l} \sum_{i \in \mathcal{I}_{CP}} \hat{k}_i + l(c\mu + b_e) \left( \sum_{i \in \mathcal{I}_{CP}} \hat{k}_{t,i} - \sum_{i \in \mathcal{I}_{CP}} \hat{k}_i \right) \ . \tag{3.18}$$

The compatibility of $\hat{k}_t$ implies that the first sum of the second term vanishes, hence, $\dot{I}_c = \left(\dot{l}/l - (c\mu + b_e)\right) I_c$. Consequently, if $I_c = 0$ at $t = 0$ then this holds for all time, thus concluding the proof. □

**Proposition 2.** *If the potential $\phi(p, \hat{k})$ is unsafe, the adaptive law (3.6) guarantees it will become safe in finite time.*

*Proof.* We stress that the definition of $\partial\mathcal{E}_O$, as stated in (2.7), implies that $\hat{k}_{t,i} > 0$, for all $q_i \in \partial\mathcal{E}_O$ with $i \in \mathcal{I}_{CP}$, by virtue of (3.8). Furthermore, assuming that $b_e$ is triggered at some time $t$ it holds that $\hat{k} \to \hat{k}_t$ asymptotically. This means that the boundary value of all occupied control points monotonically approaches the corresponding strictly positive reference value, hence, there must exist some finite time $T > 0$ for which $b_e(t + T) = 0$, implying that $\phi$ is safe. □

**Proposition 3.** *For any two time instants $0 \leq t_1 < t_2$, for which $\mathcal{E}(\mathcal{P}(t_1)) = \mathcal{E}(\mathcal{P}(t_2))$, the set $\{\tau \in [t_1, t_2] : b_e(\tau) = 1\}$ is connected.*

*Proof.* First, if $b_e(t) = 1$ for some time instant $t \in [t_1, t_2]$, Proposition 2 dictates that $b_e(t+T) = 0$ for some finite $T > 0$. In addition, for every $0 \leq t < t'$ such that $b_e(t) = 0$ and $\mathcal{E}(\mathcal{P}(t)) = \mathcal{E}(\mathcal{P}(t'))$ it holds that $b_e(t') = 0$ by construction of the adaptive law. As a result, while $\mathcal{E}$ remains unchanged, $b_e$ will equal 1 until the potential field is rendered safe. □

**Proposition 4.** *Throughout the robot's trajectory, $b_e$ will be triggered finitely many times.*

*Proof.* By construction, any transition of $b_e$ from 0 to 1 corresponds to a sign change of $\hat{k}_{t,i}$ for some $i \in \mathcal{I}_{CP}$. These sign changes are precisely what may render $\phi$ unsafe, and occur when a boundary point, which was initially thought as free, ends up in $\partial\mathcal{E}_O$ during the exploration. Assuming consistent measurements, any boundary point can experience finitely many such sign changes, and in the case of perfect measurements exactly one. Moreover, the number of boundary points $N_{CP}$ is bounded by compactness of the workspace. Thus, $b_e$ will be triggered a finite number of times. □

Next, we present some useful propositions about the boundedness of $\hat{k}_t, \hat{k}, \phi$.

**Proposition 5.** *The functions $\hat{k}_t$ and $\hat{k}$ are bounded for all time.*

*Proof.* By the definition of $\hat{k}_t$ we see that $\|\hat{k}_{t,i}\|$, $\forall i \in \mathcal{I}_{CP}$ is bounded by $\bar{k}$ as long as $I \neq 0$, which is guaranteed. Next, we will show that $\|\hat{k}_i\| \leq \bar{k}$, given that this holds for $t = 0$. Assume that $\hat{k}_i(t) = \bar{k}$ for some $i \in \mathcal{I}_{CP}, t \geq 0$. Then, $\dot{\hat{k}}_i$ must be non-positive because $f_b \propto (\hat{k}_{t,i} - \hat{k}_i)$ and $\hat{k}_{t,i} \leq \bar{k}$; thus, $\hat{k}_i$ cannot exceed $\bar{k}$. Similarly, it holds that $-\bar{k} \leq \hat{k}_i$ and consequently $\hat{k}$ is bounded. □

**Proposition 6.** *The function $\phi(p, \hat{k})$ is lower bounded.*

*Proof.* First, we note that the solution of the Neumann BVP is only unique up to an additive constant. To overcome this ambiguity we impose the constraint $\int_{\partial\mathcal{E}} \phi(s)ds = 0$, which renders the solution unique. Moreover, the well-posedness of the Neumann BVP implies that $\phi(p, \hat{k})$ depends continuously on $\hat{k},\ \forall p \in \mathcal{E}$. Further, $\hat{k} \in [-\bar{k}, \bar{k}]^{N_{CP}}$ which is a compact set. Thus, invoking the extreme value theorem, $\phi(p, \hat{k})$ is lower bounded on the set $\mathcal{E} \times [-\bar{k}, \bar{k}]^{N_{CP}}$. □

In the following, we will show that there are no stable equilibria in the workspace's interior that could potentially trap the robot in undesired configurations.

**Proposition 7.** *All equilibria of* (3.16) *located in* int$\mathcal{E}$ *are unstable.*

*Proof.* First, note that by construction no equilibria can exist while $b_e = 1$ thus we consider only cases where the second term of (3.6) is null in the subsequent analysis. The term $c$, that appears in $f_b$, ensures that for any equilibrium state $z^\star = [p^\star, \hat{k}^\star, \mathrm{Pr}^\star]^T$ for which $p^\star \in \mathrm{int}\mathcal{E}$, there exists a disk such that $f_b(p, \hat{k}) = 0,\ \forall p \in \mathcal{B}(p^\star, \epsilon)$ for some sufficiently small $\epsilon > 0$. Thus, inside this ball the system executes gradient descent on a fixed potential field. Given that $\phi$ is harmonic, any critical points in int$\mathcal{E}$ are strict saddles. Consequently, all equilibria of $f_z$ located in int$\mathcal{E}$ are unstable. □

**Lemma 1.** *The function $f_p$ is locally Lipschitz continuous.*

*Proof.* The function $s(p, \mathcal{E})$ is locally Lipschitz $\forall p \in \mathcal{E}$. Moreover, $\nabla_p\phi$ is analytic and hence locally Lipschitz, $\forall p \in \mathrm{int}\mathcal{E}$. Thus, within int$\mathcal{E}$ $f_p$ is locally Lipschitz as the product of two such functions. Moreover, let $f(p) = g(p)h(p)$ where $g(p) = s(p)$ and $h(p) = \nabla_p\phi(p)$. Then, for any $x, y \in \mathcal{E}$ we note that the following holds:

$$\begin{aligned} \|f(x) - f(y)\| &= \|g(x)h(x) - g(y)h(y)\| \\ &= \|g(x)h(x) + g(x)h(y) - g(x)h(y) - g(y)h(y)\| \\ &= \|g(x)(h(x) - h(y)) + h(y)(g(x) - g(y))\| \\ &\leq \|g(x)\|\|h(x) - h(y)\| + \|h(y)\|\|g(x) - g(y)\| \end{aligned} \tag{3.19}$$

We observe that as $x \to \partial\mathcal{E}$ the first term vanishes because $g(x)$ vanishes and $h(x) - h(y)$ is bounded by construction. Also, note that $h(y)$ is bounded because in int$\mathcal{E}$ the function $\nabla_p\phi$ is analytic and in $\partial\mathcal{E}$ its norm is bounded from above by $\max_{i\in\mathcal{I}_{CP}} \|\hat{k}_i\|$. As such, $f_p$ is locally Lipschitz on $\partial\mathcal{E}$ and consequently everywhere in $\mathcal{E}$. □

**Proposition 8.** *The robot's trajectory under control law* (3.16) *is safe, i.e., collision avoidance is ensured.*

*Proof.* If the potential is safe, then the robot obviously cannot collide with the workspace's boundary. We now consider the case when $\phi$ is not safe, which implies that $b_e = 1$, and we assume that $p \to \partial\mathcal{E}_O$. Consequently, $\dot{p} \to 0$ and according to Lemma 1, $f_p$ is locally

Lipschitz; thus, the robot converges to the boundary asymptotically. By construction of $\dot{\Pr}$ it holds that $\hat{k}_t(\sigma^{-1}(q))$ becomes strictly positive as $p \to q$, for any $q \in \partial\mathcal{W}$. But, according to Proposition 3, $\phi$ will become safe in finite time rendering all obstacles non-attractive, which contradicts our initial assumption. □

**Proposition 9.** *The set $\partial\mathcal{E}_F$ has a non-trivial region of attraction.*

*Proof.* As a consequence of the Maximum Value Theorem of harmonic functions, all local extrema of $\phi$ must reside on the boundary of $\mathcal{E}$. Additionally, $s(p, \hat{k})$ vanishes on $\partial\mathcal{E}$, rendering the extrema of $\phi$ equilibria of (3.16). Since by construction, $\partial\mathcal{E}_F$ is made attractive, all equilibria on it must have a non-trivial region of attraction. □

The completeness of our approach is equivalent to proving that the proposed control scheme drives the robot towards the free boundary, whenever any exists. In this context, the following theorem states that the robot neither can reach $\partial\mathcal{E}_F$ nor needs to, as a concequence of perfect sensing assumption and the non-zero sensing range of the robot.

**Theorem 1.** *Assuming that $b_e = 0$, the robot equipped with the control law* (3.16) *and starting from almost any initial configuration, will move towards and expand $\partial\mathcal{E}_F$. Thus, any compact and connected workspace $\mathcal{W}$ will be explored in finite time.*

*Proof.* Consider the Lyapunov function candidate $V = \phi(p, \hat{k})$. Differentiating w.r.t. time and substituting the control scheme (3.3), as well as the adaptive law (3.6) with $b_e = 0$, we obtain:

$$\begin{aligned} \dot{V} &= (\nabla_p \phi)^T \cdot \frac{dp}{dt} + \frac{\partial \phi}{\partial \hat{k}} \cdot \frac{d\hat{k}}{dt} \\ &= -K_u s \|\nabla_p \phi\|^2 + \frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k}) c\mu \\ &\leq -K_u s \|\nabla_p \phi\|^2 + \|\frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k})\| c\mu \ . \end{aligned} \tag{3.20}$$

We recall that choosing $\mu_1 > 9/8$ implies:

$$\begin{aligned} \frac{K_u s \|\nabla_p \phi\|^2}{\|\frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k})\| + \epsilon_2} \geq S_{\mu_1} \left( \frac{K_u s \|\nabla_p \phi\|^2}{\|\frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k})\| + \epsilon_2} \right) \implies \\ K_u s \|\nabla_p \phi\|^2 \geq \left( \|\frac{\partial \phi}{\partial \hat{k}} \cdot (\hat{k}_t - \hat{k})\| + \epsilon_2 \right) \mu \\ \geq \|\frac{\partial \phi}{\partial \hat{k}} (\hat{k}_t - \hat{k})\| c\mu \end{aligned} \tag{3.21}$$

which in turn implies that $\dot{V} \leq 0$, with the equality holding only for critical points of $\phi$ or boundary points. According to Proposition 6, $V$ is lower bounded and we also showed that it is a non-increasing function of time. As such, $V$ has a finite limit as $t \to \infty$.

Additionally, $\dot{V}$ is Lipschitz w.r.t. time (the proof is provided in Subsection 3.2.1) and therefore uniformly continuous. Hence, invoking Barbalat's Lemma we deduce that $\dot{V}$ will vanish, which can happen iff either $s(p) = 0$ or $\nabla_p \phi = 0$. The former case means that $p \in \partial\mathcal{E}$ and since $p \notin \partial\mathcal{E}_O$, as shown in Proposition 8, this implies that the robot will move towards $\partial\mathcal{E}_F$. The latter case corresponds to the robot converging to a critical point of $\phi$. But, these critical points have been proven to be unstable saddles and, therefore, this can happen only for a set of initial configurations that has zero Lebesgue measure. As such, the robot will navigate towards the minimizers of its potential field, which lie on the free boundary. Therefore, it obviously holds that for all $q \in \partial\mathcal{E}_F$, there exists a finite time instant $T \in [0, \infty)$ such that $q \in \mathcal{S}(p(T))$, since $\mathcal{W}$ is bounded. Thus, any point that belongs to $\partial\mathcal{E}_F$ will eventually be explored in finite time. □

### 3.2.1 Complementary Proofs

**Lemma 2.** *For any $a > \frac{9}{8}$ it holds that $S_a(x) < x,\ x \in (0, +\infty)$.*

*Proof.* First, note that if $S_a(x) < x$ for $x \in (0, a]$, then the inequality holds for all $x \in (0, \infty)$. We define the function

$$\begin{aligned} g(x) &= x - S_a(x) \\ &= x \cdot (2 \cdot \frac{x^2}{a^3} - 3 \cdot \frac{x}{a^2} + 1) \end{aligned} \tag{3.22}$$

and we will show that $g(x) > 0,\ \forall x > 0$. Note that the sign of $g$ is determined by the second degree polynomial

$$h(x) = 2 \cdot \frac{x^2}{a^3} - 3 \cdot \frac{x}{a^2} + 1 \quad , \tag{3.23}$$

whose minimum is obtained at $x^\star = \frac{3}{4} \cdot a$ and is equal to $h(x^\star) = 1 - \frac{9}{8a}$. Thus, choosing $a > \frac{9}{8}$ ensures $h(x) > 0 \implies g(x) > 0$. □

**Lemma 3.** *The function $\dot{V}$ is locally Lipschitz w.r.t. time.*

*Proof.* The first term of $\dot{V}$ is clearly locally Lipschitz in $\text{int}\mathcal{E}$, because $s(p)$ is Lipschitz and so is $\|\nabla_p \phi\|^2$ being the composition of two such functions. Following arguments similar to Lemma 1 we can show that this term is Lipschitz on $\partial\mathcal{E}$. Moreover, since $\|\nabla_p \phi\|$ is bounded in $\mathcal{E}$, the same applies for $\|\nabla_p \phi\|^2$ and therefore the whole term is bounded.

We proceed in the analysis of the properties of the second term of $\dot{V}$. The function $\hat{k}_t$ will be shown to be continuous and piece-wise $C^1$. First, we note that $f_{\text{Pr}}$ is a piece-wise continuous function that exhibits jump discontinuities when $c_i \in \partial\mathcal{S}$ for some $i \in N_G$. Hence, Pr is continuous and piece-wise $C^1$ and, obviously, the same holds for $k_1'$. Also, $k_2'$ is continuous and piece-wise $C^1$ being the composition of $e^{-x}$, which is $C^\infty$, and $\|p - q\|$, which is piece-wise $C^\infty$. Thus, $k'$ is continuous and piece-wise $C^1$ as the product of two

such functions. The function $I$ is also continuous and piece-wise $C^1$ as the integral of such a function. Lastly, by construction $k'_m$ is continuous and piece-wise $C^1$ w.r.t. time since max is a continuous function and its argument is a fraction of such functions, given that it is guaranteed that $I > 0$. Thus, $\hat{k}_t$ is continuous and piece-wise $C^1$ being a composition of $k', I$ and $k'_m$. Clearly, $S_{\epsilon_w}(\|\nabla_p \phi\| - \epsilon_1)$ is also continuous and piece-wise $C^1$ because it is a composition of two such functions. Further, $\frac{\partial \phi}{\partial \hat{k}}$ is a vector whose elements are $C^1$ w.r.t. $p, \hat{k}$ in int$\mathcal{E}$ and, by construction, bounded in $\mathcal{E}$. Finally, the argument of $S_{\mu_1}(\cdot)$ is clearly continuous, piece-wise $C^1$ and bounded while $S_{\mu_1}$ is $C^1$. Hence, $\mu$ is continuous and piece-wise $C^1$.

Overall, both terms of $\dot{V}$ were shown to be continuous and piece-wise $C^1$. Therefore, $\dot{V}$ is continuous and piece-wise $C^1$, and as such locally Lipschitz w.r.t. time. □

## 3.3 Analytical Comparison with Related Literature

Next, we discuss the advantages, as well as the shortcomings, of the proposed control scheme compared to the existing literature. Initially, we focus on methodologies that are similar to ours, i.e., based on HPFs.

As previously stated, an integral disadvantage associated with HPFs is the need to solve Laplace's equation in real-time, in order to obtain the necessary harmonic vector field employed for navigation. In particular, the construction of the underlying HPF is costly, even prohibitively so for large domains, and scales poorly. As a result, implementations of such approaches [8],[18], rely on relaxation methods to retrieve a partially-converged numerical solution, thus, sparing the computation time required for full convergence. Moreover, the computational complexity involved in the aforementioned numerical techniques scales poorly w.r.t. the domain's size. These shortcomings are alleviated in [19],[9], by computing the HPF only on a local window around the robot, and in [10] by employing the Full Multigrid Method. In addition, the finite differences schemes employed by the aforementioned works suffer the following difficulties:

- Discretization of an entire 2D domain is necessary,
- The HPF value is only available on nodal points, and
- The derivatives of the HPF need to be approximated.

The previous shortcomings are circumvented in our control scheme by employing the FMBEM. Specifically, the FMBEM enjoys linear complexity in terms of computational effort and required memory, which comes at the expense of obtaining approximate solutions of the corresponding BVP with a specified error. The boundedness of the approximation error enables us to exempt our methodology from the utilization of heuristic approaches to reduce the computational cost. Furthermore, the Boundary Element Method involves discretizing solely the boundary of the domain, which, in general, is more straightforward.

Finally, direct evaluation of the HPF value and its derivatives is possible at any interior point, owing to the boundary integral representation of the solution.

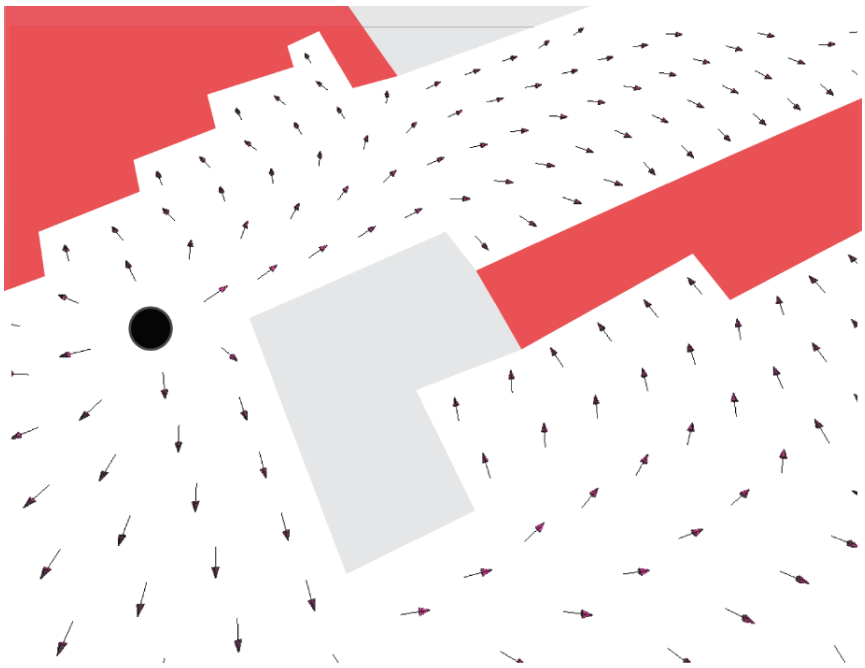


Figure 3.3: The negated gradient vector field induced by the mixed boundary conditions employed in [2]. Unexplored and occupied space is shown in red and gray, respectively. Notice that the vector field tends to drive the robot near obstacles.

Another point in favor of our controller is the utilization of Neumann boundary conditions, as opposed to the Dirichlet or mixed boundary conditions adopted in the existing literature. Notably, in [8],[18] the authors impose a uniform high (resp. low) value on obstacles (resp. unexplored frontiers) which induces an orthogonal gradient on the boundary. As a downside, this approach generally exhibits large flats areas of the potential, thus, resulting in very small gradients. Shade et al. alleviate this by employing mixed boundary conditions. Specifically, a constant low value is enforced on boundaries between known and unknown space, a high value is prescribed on the position of the robot, while the normal derivative on occupied boundaries is fixed to zero. Nonetheless, the vanishing normal derivative on obstacles causes robot's trajectory to pass near them (see Figure 3.3). Finally, the aforementioned approaches assign uniform attractiveness/repulsiveness on frontiers/obstacles, hence, preventing the prescription of a relative importance between boundary regions. On the contrary, the proposed control law maintains a significant gradient in the robot's vicinity and, also, the non-vanishing normal derivative in nearby obstacles imposes a considerable repulsion on the robot (see Figure 3.4). Additionally, we can designate preferences among

boundary regions by utilizing non-constant Neumann boundary conditions. This issue is, also, tackled in [19], [20] through the use of a modified version of Laplace's equation.

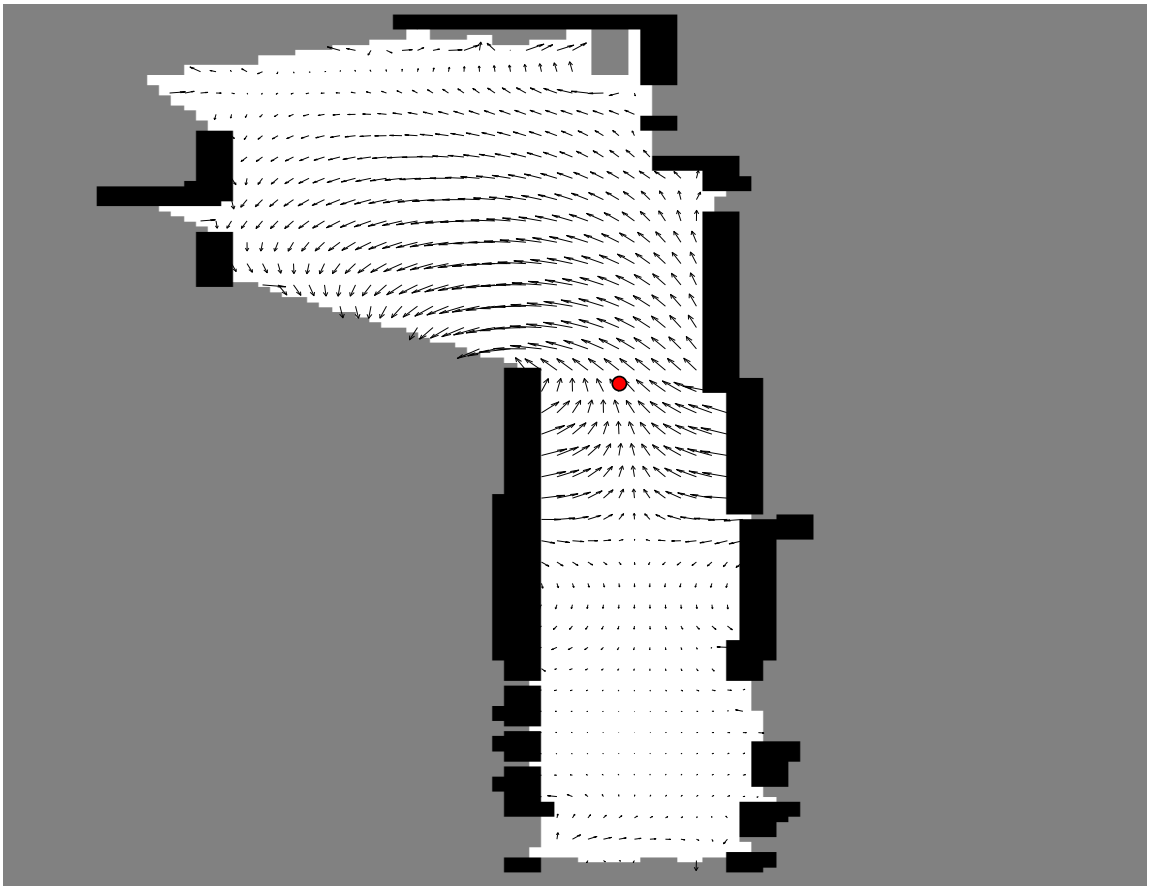


Figure 3.4: The negated gradient vector field induced by our control law. Unexplored and occupied space is shown in gray and black, respectively. We observe that neither the gradient, nor the normal derivative on obstacles vanish in the robot's vicinity. Note: The reason some vectors are pointed towards obstacles is that $\phi$ is not safe (but will become eventually, as proven previously).

The algorithms that do not utilize HPFs, e.g. [3], [5], [21] focus on generating a number of candidate goal configurations, and associated paths to reach them, which are subsequently evaluated using an information-theoretic utility function. An inherent shortcoming of this approach is its non-reactive nature. Particularly, once a goal configuration has been selected the robot is blindly driven towards it. Although some methodologies [22] are capable of reacting to dynamic obstacles that may occur along the way, the information accumulated is only utilized in the next planning operation, i.e., when the goal is reached. In contrast,

changes in the constructed map are constantly incorporated in our control scheme. In addition, our controller solves both the path and motion planning problem as opposed to competing algorithms. Nevertheless, it should be noted that approaches in the related literature actively focus on the integrated exploration problem, i.e., they take into account the uncertainty in the robot's position. The proposed controller cannot readily solve the aforementioned problem and requires modifications towards that direction.

# Chapter 4

# Implementation

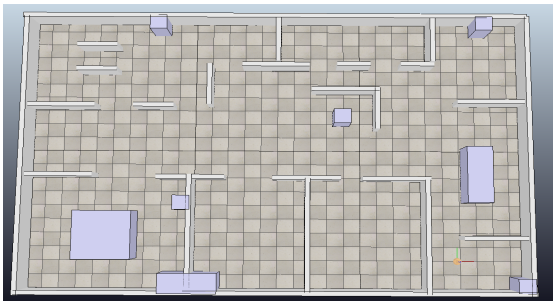
A crucial part of the current thesis consists in the creation of an appropriate simulation environment, as well as the implementation of the proposed control law. This chapter elaborates on the different necessary modules used in the presented exploration scheme. Specifically, Section 4.1 demonstrates the simulation environment in VREP involving the workspace, sensors and a simplistic robot representation, Section 4.2 describes the basic preprocessing of sensor measurements, while Section 4.3 elaborates on the Occupancy Grid Mapping algorithm. The methodology devised for extracting the boundary of a given occupancy map is described in Section Section 4.4, followed by Section Section 4.5 where an algorithm for determining the correspondence between points of consecutive extracted boundaries is established. Finally, in Section 4.6 we outline the basics of the FMBEM algorithm and demonstrate several computational results.

## 4.1 Mobile Robot Simulation in VREP

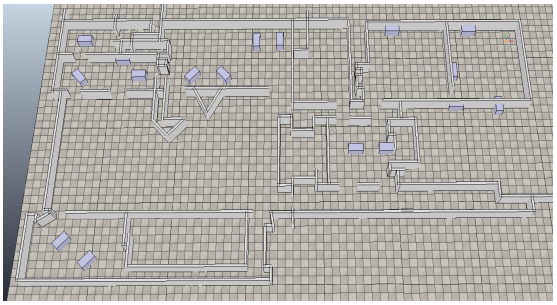
The simulation of the mobile robot exploration was performed using the Virtual Robot Experimentation Platform [23], abbreviated VREP. The robot simulator VREP, is ideal for fast algorithm development, prototyping and verification as it features an integrated development environment and easy-to-use API frameworks for a number of programming languages, namely Matlab and C++. In the following, the main components of the simulation environment implemented in VREP are described.

Initially, a simulated workspace is created using cuboid and cylindrical shapes that make up the walls of the environment. More realistic details can be included by adding furniture, e.g. chairs and tables. Two of the workspaces recreated in VREP and used for the subsequent simulations are demonstrated in Figure 4.1.

The robot is represented using a "Dummy" (see Figure 4.2), which is the simplest object offered in the simulator. A more intricate representation, e.g. a robot with wheels, would be unnecessary since the robot is treated as a point, with single integrator dynamics, in the



(a) First sample workspace.



(b) Second sample workspace.

Figure 4.1: A subset of the workspaces used in the simulations.

preceding analysis.

The robot/dummy is equipped with "Proximity Sensors" that allow it to perceive obstacles within the sensor's range and line-of-sight. Proximity sensors are implemented as rays (see Figure 4.2), spread evenly around the robot. The number of rays used is equivalent to the sensor's angular resolution. Each ray provides appropriate readings about whether it hits an obstacle and at what distance it is located. These measurements are transferred to Matlab, using the remote API, to be further processed.

For any sensible angular resolution, adding the appropriate number of rays manually in the simulation environment is both tedious and impractical. In particular, a resolution of 1 ray/$^o$ would require 360 rays. Moreover, altering the properties of the rays, e.g. changing the maximum range, would be a prohibitively time-consuming task. To that end, a script, written in the Lua programming language, was developed and embedded on the dummy object, which generates rays attached on the dummy at the beginning of the simulation; the number and properties of the rays are defined programmatically by the user. This approach allows for flexible adjustment of the simulation parameters, and straightforward simulation in any new workspace by simply copying and pasting the dummy along with the embedded script.

## 4.2 Preprocessing of Sensor Measurements

Before incorporating sensor measurements in the mapping algorithm an appropriate preprocessing is necessary. Specifically, proximity sensor measurements, i.e. distance of an obstacle along a particular ray, are expressed in each ray's coordinate frame and, thus, need to be appropriately transformed in the absolute coordinate frame before being integrated in the mapping algorithm. Knowing the robot's position $p$, the distance measurements $z$ and the ray's angle $\theta$, w.r.t. the absolute coordinate frame (see Figure 4.3), the absolute position of

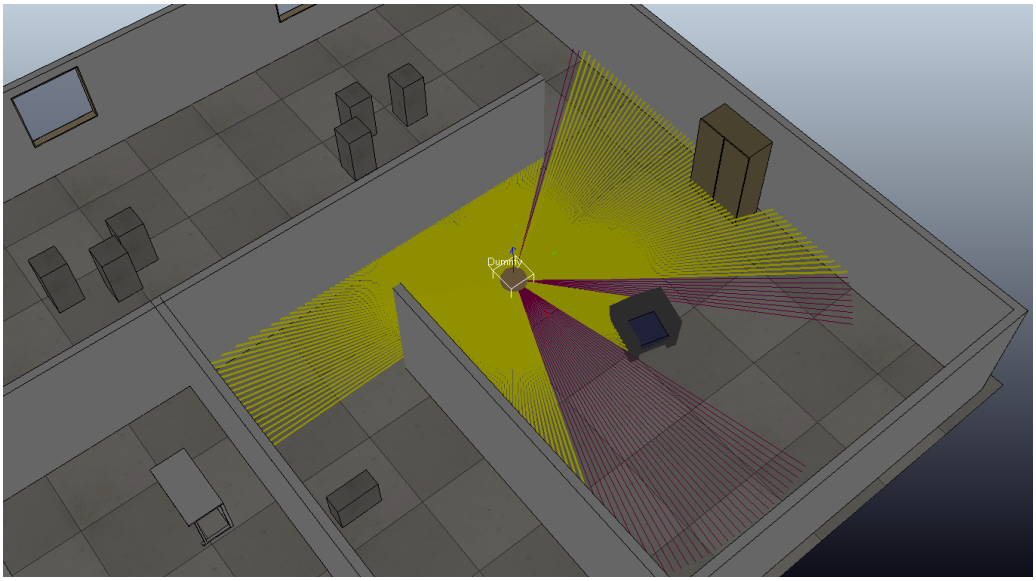



Figure 4.2: The "Dummy" robot represented as a sphere with the proximity sensor rays attached on it. Yellow-colored rays indicate the presence of an obstacle within the range of a particular ray.

the obstacle $p_{obs}$ can be easily computed as follows:

$$p_{obs} = p + \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} z \\ 0 \end{bmatrix} \ . \tag{4.1}$$

This computation is performed for all the rays. Figure 4.4 illustrates a trajectory executed by the robot, along with the corresponding sensor readings.

## 4.3 Occupancy Grid Mapping

Any robot operating autonomously in a particular environment requires a map in order to navigate and plan successfully. Moreover, in many cases the central goal of exploration algorithms is to generate accurate maps of a given workspace. In the current work, we employ the well-studied Occupancy Grid Mapping (OGM) algorithm [13] to represent the explored regions of the workspace. The reason for choosing OGM among other mapping algorithms is both the simplicity and efficiency it provides, as well as being a very common choice in the exploration literature. Also, OGM is capable of accounting for possible errors in sensor readings, but requires that the robot pose is known or estimated, e.g. by a SLAM algorithm. Next, we will elaborate on the details and implementation of OGM. The theoretical part of the algorithm is a brief summary of the corresponding chapter in [14]; the reader is directed there for a more detailed analysis.

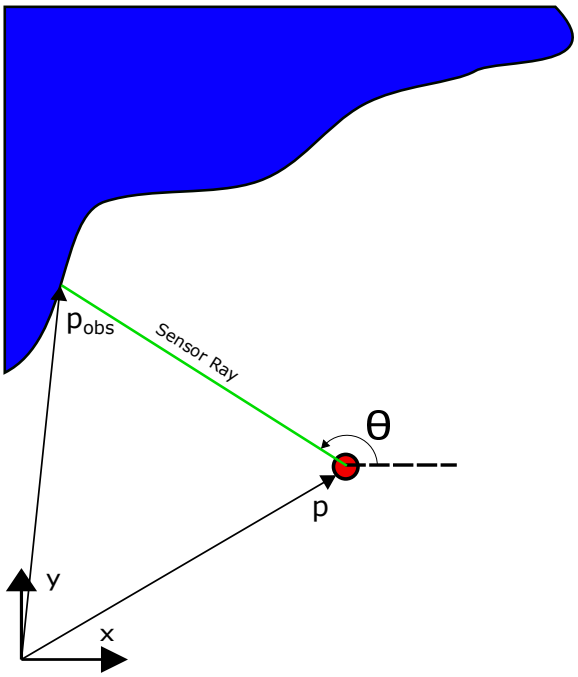



Figure 4.3: Transformation of sensor measurements from the ray's coordinate frame to the absolute coordinate frame.

As described in Section 3.1 a grid is embedded in the workspace, and the OGM algorithm is used to update the probability of occupancy of grid cells observed by the robot. For the following analysis, the notation used in [14] is adopted; in particular, time is represented as discrete steps, the robot's position at time step $t$ is denoted $x_t$, while the measurements obtained at that time are denoted $z_t$. The probability that the cell $m_i$ is occupied, given the trajectory $x_{1:t}$ and sensor readings $z_{1:t}$ is denoted $\Pr(m_i; x_{1:t}, z_{1:t})$. We define the log-odds probability at time step $t$ as

$$l_{t,i} = \log\left(\frac{\Pr(m_i; x_{1:t}, z_{1:t})}{1 - \Pr(m_i; x_{1:t}, z_{1:t})}\right), \ \forall i \in \mathcal{I}_G \ . \tag{4.2}$$

Thus, the update law for $l$ is

$$l_{t,i} = \begin{cases} l_{t-1,i} + f_l(m_i, x_t, z_t) - l_0, & c_i \in \mathcal{S}_t \\ l_{t-1,i}, & \text{otherwise} \end{cases}, \ \forall i \in \mathcal{I}_G \tag{4.3}$$

where $f_l$ is the inverse sensor model, $l_0$ is the prior of occupancy, and $\mathcal{S}_t$ is the sensing region of the robot at $t$. We recall that $c_i$ denotes the center of $m_i$. We choose $l_0 = 0$, which prescribes that an unforeseen grid cell may be occupied or free with equal probability.

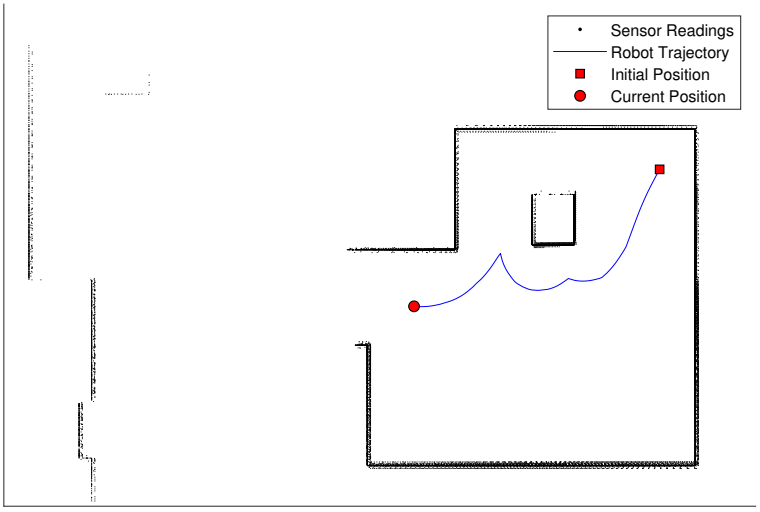



Figure 4.4: A representative trajectory of the robot and the sensor readings collected along it.

Regarding the choice of $f_l$, it is noted that a wide range of approaches are proposed in literature. For instance, in [14] the use of supervised learning algorithms, such as logistic regression or neural networks, to approximate $f_l$ is discussed. In the current work, we adopt the simple, yet commonly-utilized, definition of $f_l$ that follows:

$$f_l = \begin{cases} l_{occ}, & m_i \text{ measured as occupied} \\ l_{free}, & m_i \text{ measured as free} \end{cases}, \tag{4.4}$$

where $l_{occ} > 0, l_{free} < 0$ are constants. As such, the log odds probability of cells that are indicated by the sensor readings to be free (resp. occupied) is increased by $l_{free}$ (resp. $l_{occ}$).

From a practical standpoint, it is necessary to determine which cells in $\mathcal{S}_t$ are measured as free and which ones as occupied. Apparently, the cell in which the robot is included should be indicated as free, while the cell containing the end point of the ray should be labeled as either free or occupied depending on the reading; any cells in between should be considered as free, since the ray passes through them unobstructed. Hence, it remains to identify the cells affected by each ray. To that end, we employ Bresenham's line algorithm [24] which selects cells in a $n$-dimensional grid, that form a close approximation of the line segment connecting two points. This procedure is demonstrated in Figure 4.5. Colored cells represent the grid approximation of the ray, generated through Bresenham's algorithm. Yellow colored cells are identified as free, while the red colored one is identified as occupied

by the particular ray.

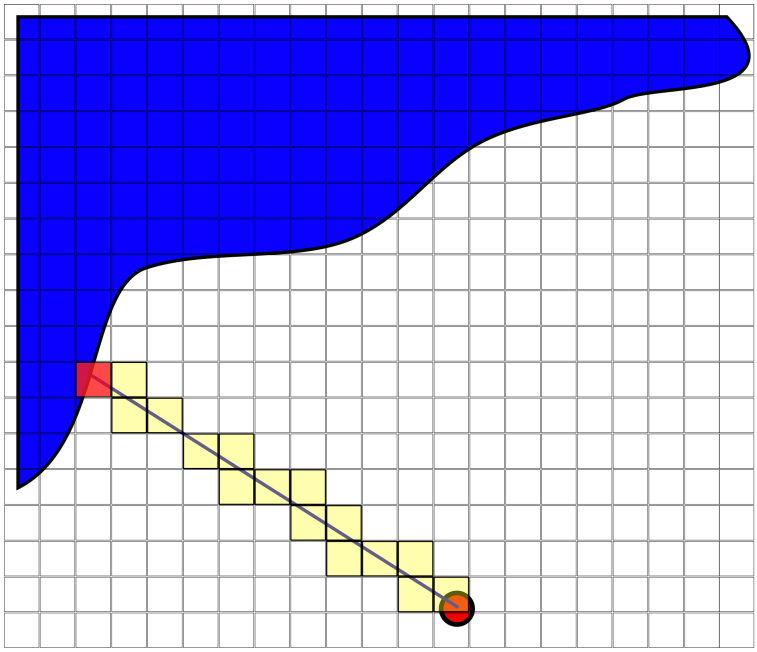
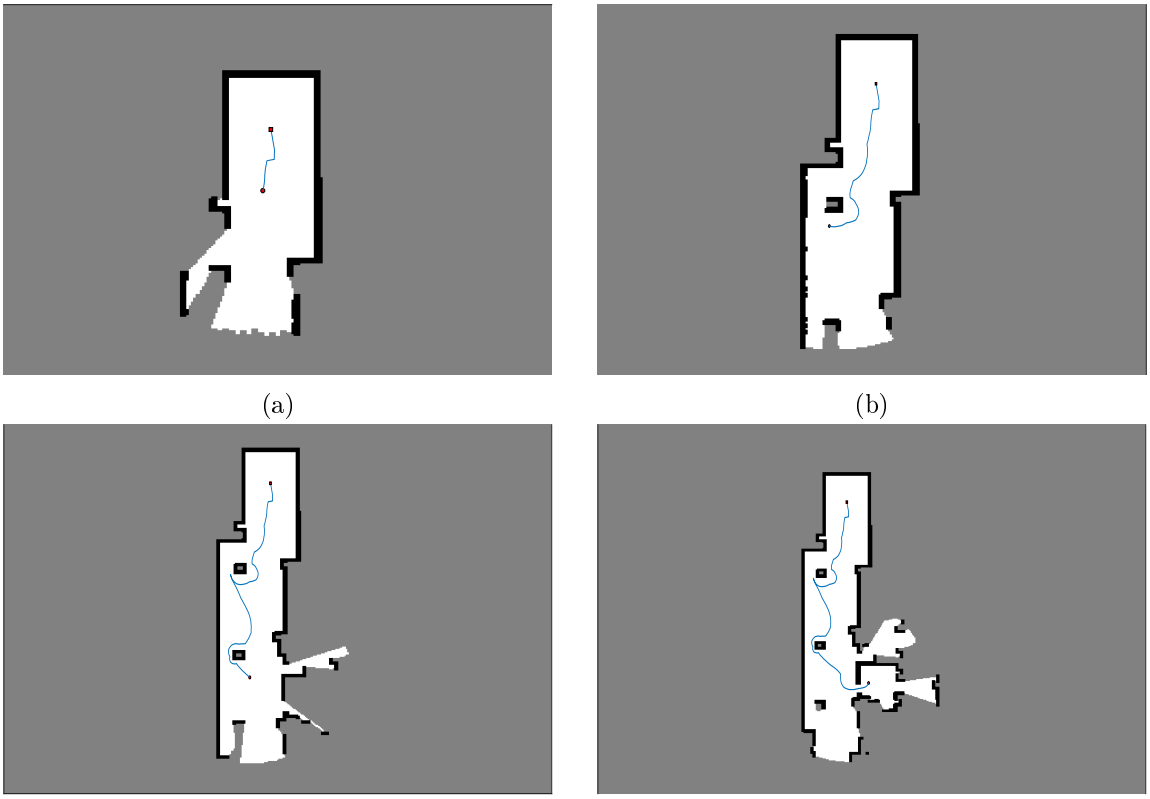


Figure 4.5: The grid approximation of a sensor ray using Bresenham's algorithm. Cells of yellow color are identified as free, while the red-colored cell is indicated as occupied.

Through the aforementioned procedure, the probability of all observed cells in the grid is appropriately updated in each time step. As the robot navigates the unknown workspace, new information is incrementally incorporated in the map. Four consecutive instances of the occupancy grip are presented in Figure 4.6.

Finally, it is noted that the occupancy grid map is stored in matrix format; each matrix element is associated with a grid cell, based on its position (row and column), and holds its probability of occupancy. In addition to being a straightforward implementation, this representation allows for effortless manipulation of the map using image processing techniques, e.g. dilation and convolution, as will be shown in Section 4.4.



(a) (b) (c) (d)

Figure 4.6: Four instances of the generated occupancy grid map along a robot's trajectory. White, black and gray cells represented free, occupied and unknown space respectively. The robot's trajectory is shown as a continuous blue line.

## 4.4 Boundary Extraction

Given the map generated by the OGM algorithm, it is necessary for its boundary to be extracted as required by the FMBEM solver. This boundary is composed of two disjoint regions, as explained in Section 3.1. In particular, the frontier $\partial\mathcal{E}_F$ consists of the free grid cells that are adjacent to cells of unknown occupancy, i.e. cells whose probability of occupancy is equal to 0.5, whereas the occupied boundary $\partial\mathcal{E}_O$ comprises the occupied grid cells neighboring with free cells. Upon identifying the cells that make up $\partial\mathcal{E}$, their centers are used as a point-wise approximation of $\partial\mathcal{E}$. The algorithm employed for extracting the boundary of the occupancy map is outlined in Algorithm 1.

Step 1 amounts to performing morphological dilation of obstacle cells using a square structuring element, i.e. a matrix $\mathbb{R}^{n\times n}$ composed of ones, with size $n = 2$. This oper-

**Algorithm 1** Boundary Extraction

1: Dilation of obstacles.
2: Isolation of the free connected region.
3: Identification of boundary components.
4: Grouping and ordering of the center points of cells.

ation bears a dual purpose. First, and most importantly, we note that small features of the workspace may not contribute enough grid cells/points to construct a closed arc representation. For instance, a small table leg may occupy only one grid cell, thus limiting its description to a single isolated point. But, in order for the Laplace equation to admit a solution the boundary must be composed of closed arcs, which can be ensured by dilating the occupied cells. Secondly, augmenting the size of obstacles through dilation can be applied to account for the actual dimensions of a disk robot, choosing the size of the structuring element appropriately.

As a consequence of erroneous measurements, it is possible for grid cells outside the actual workspace to be labeled as free (see Figure 4.7). Such cells need to be identified and removed, as they would cause the extracted boundary to be inconsistent; this is tackled in Step 2. To that end, the connected region of free cells containing the robot needs to be isolated. Apparently, any cells labelled as free that do not belong to this region cannot have been observed and, hence, correspond to faulty measurements. Implementation-wise, we create a binary matrix, with the same size as the occupancy grid, whose value at free cells is 0, and everywhere else is 1. Then, a flood fill algorithm is applied, placing its seed at the robot's position in the grid, thus rendering the value of all cells in the connected free region true. Consequently, any cells whose value remains false must correspond to invalid measurements.

Following these preprocessing steps, we are ready to extract the boundary of the map. The regions $\partial\mathcal{E}_O, \partial\mathcal{E}_F$ will be identified separately, before being merged to construct $\partial\mathcal{E}$. To find $\partial\mathcal{E}_O$, we first create the matrix $M_{free}$, with dimensions identical to the occupancy grid, whose entries are 0 everywhere except on elements which correspond to free grid cells, whose value is set to 1. Then, the edge detection kernel

$$M_{conv} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.5}$$

is convolved with $M_{free}$ to produce $M_o$. Note that occupied and unknown cells adjacent (in the 4-connected pixel connectivity sense) to free cells will receive a positive value in $M_o$, as can be easily verified. Therefore, any matrix elements with positive value in $M_o$ make up the outline of the free region. Out of these elements, the ones associated with occupied cells apparently constitute $\partial\mathcal{E}_O$. We remark that, not all occupied cells belong to $\partial\mathcal{E}_O$ because

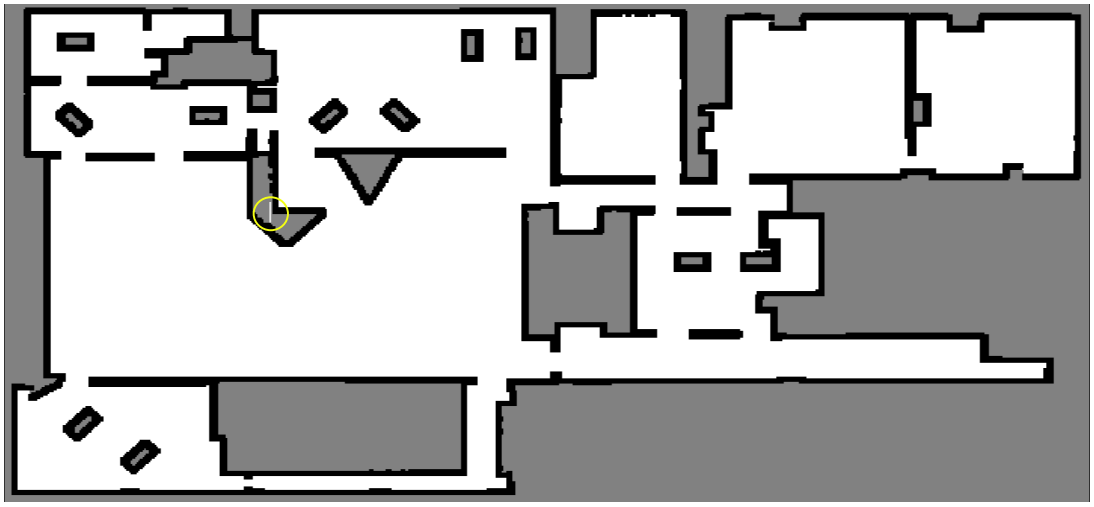


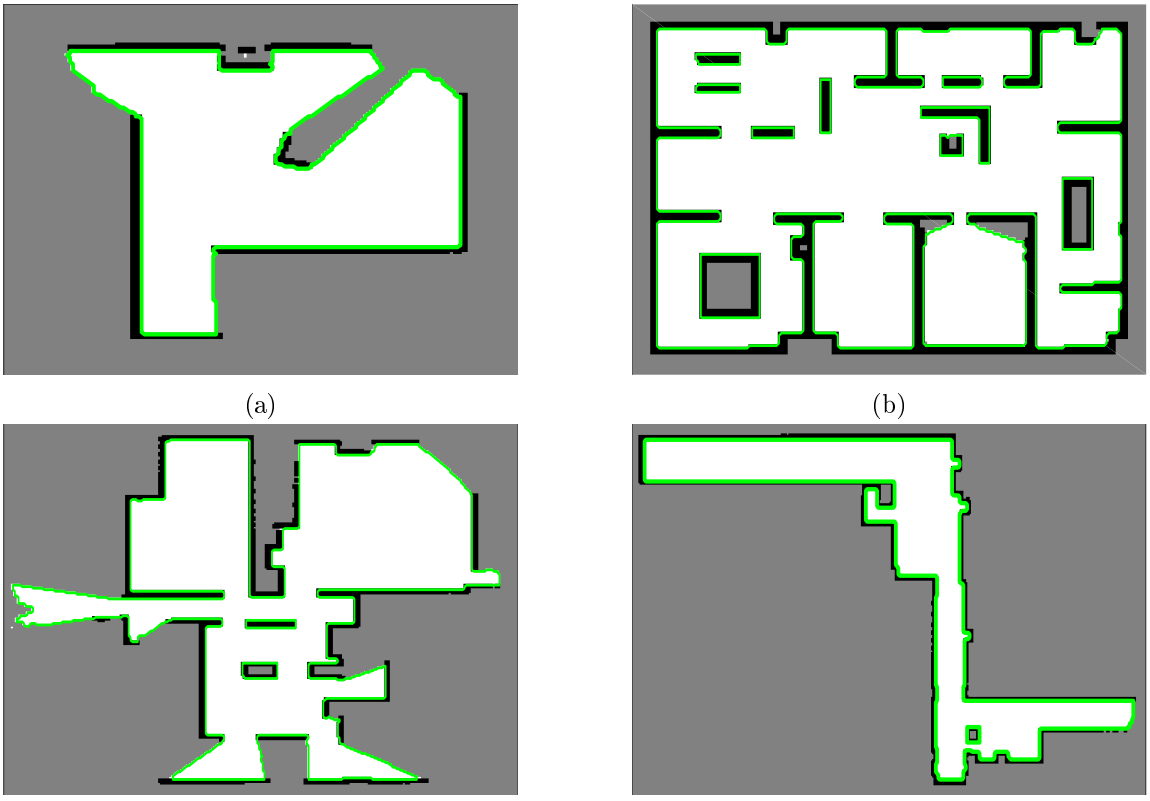
Figure 4.7: The yellow cirle indicates free cells that do not belong to to connected region of the robot's workspace.

due to the incremental construction of the occupancy grid, or due to dilation, it is possible for an occupied cell to have only occupied or unknown neighbors.

A similar procedure is followed for the extraction of $\partial\mathcal{E}_F$. We, initially, create the matrix $M_u$ and set the value of unknown cells to 1, while the remaining cells are set to 0. Afterwards, the new matrix $M_f$ is generated through the convolution of $M_u$ with $M_{conv}$. By the same reasoning as previously, elements that correspond to free cells and have a positive value in $M_f$ make up $\partial\mathcal{E}_F$. Consequently, both components of $\partial\mathcal{E}$ have been identified in the occupancy grid map.

Finally, having tracked the individual cells that form $\partial\mathcal{E}$, it is imperative to group them appropriately in order to trace the boundary correctly. We recall that $\partial\mathcal{E}$ is composed of multiple arcs and, thus, boundary cells have to be divided depending on which arc they belong. Moreover, the cells, and the corresponding center points, that make up a particular arc need to be arranged in a specific order, i.e. clockwise for inner boundaries and counterclockwise for the outer boundary, so as to properly describe it. The aforementioned tasks are performed using the Matlab function BWBOUNDARIES [25].

An efficient and robust derivation of $\partial\mathcal{E}$ is of paramount importance in the proposed control scheme, since it forms the basis for solving the Laplace's equation using the FMBEM. Hence, the boundary extraction methodology was rigorously tested in a multitude of different workspaces. The occupancy grid map and the corresponding extracted boundary is illustrated in Figure 4.8, in 4 different occasions.



(a) (b) (c) (d)

Figure 4.8: The occupancy grid map and the resulting boundary of the explored region $\partial\mathcal{E}$, depicted as a green line.

## 4.5 Boundary Value Projection

In this section, we describe the difficulty arising when updating the boundary values $\hat{k}$ while $\partial\mathcal{E}$ is changing, as well as how this implication is overcome. Given two boundaries extracted in consecutive time steps, it is clear that some regions will be identical while others will differ, due to the expansion of the boundary, as shown in Figure 4.9. Assigning $\hat{k}_i$ to points that overlap with the previous boundary is straightforward; we simply copy the previous value. On the contrary, non-overlapping points need to have their value inferred according to their position on the boundary. Specifically, each new boundary arc (i.e. a set of connected points that does not overlap with the previous boundary) is associated with an arc on the past boundary, which resulted in the new one through expansion (as indicated by the arrows in Figure 4.9).

There are two types of boundary regions which no longer exist in the subsequent bound-

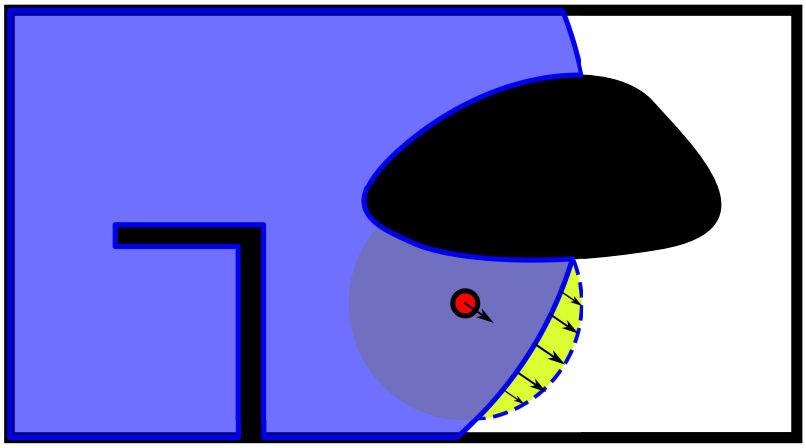


Figure 4.9: The expansion of the explored region with the subsequent change in the boundary. The previous explored region is shown in blue while the area added in the current time step in yellow. The dashed line illustrates the new boundary region.

ary. Firstly, an arc undergoing expansion may completely disappear in the subsequent boundary, if it encloses only free space which is explored (see Figure 4.10). In this occasion no action needs to be taken, as the previous boundary points involved are no more part of the new boundary.

Secondly, an expanding arc will, in most cases, remain on the new boundary and we need to match it to the original arc. If there exists a single changed boundary region, determining the aforementioned association would be unambiguous; but, since multiple changed regions may arise, a more elaborate method is required. In particular, we note that the two endpoints of an expanded arc belong to the unchanged boundary and hence, they can be used to identify the correspondence between previous and new changed regions. Any two changed regions in the previous and new boundary whose endpoints are shared, correspond to the same boundary arc before and after expansion, respectively. This concept is demonstrated in Figure 4.11.

Having established a way of matching altered boundary regions, we devise a method to project the boundary values from the previous boundary to the new. Initially, for a particular changed region we label one endpoint as the start and the other as the end of the arc, based on the arc's positive direction (i.e. clockwise/counterclockwise for inner/outer boundaries). Consequently, points are parameterized based on the ratio $r_p = \frac{\ell_p}{\ell_T}$, where $\ell_p$ denotes the length of the arc between the starting endpoint and the particular interior point, while $\ell_T$ denotes the total arc length. We approximate the boundary value in the parametrized space using linear interpolation between the values of previous boundary points. Then, the value of new boundary points is derived by sampling the piece-wise approximation at the corresponding parametrization value (see Figure 4.12).

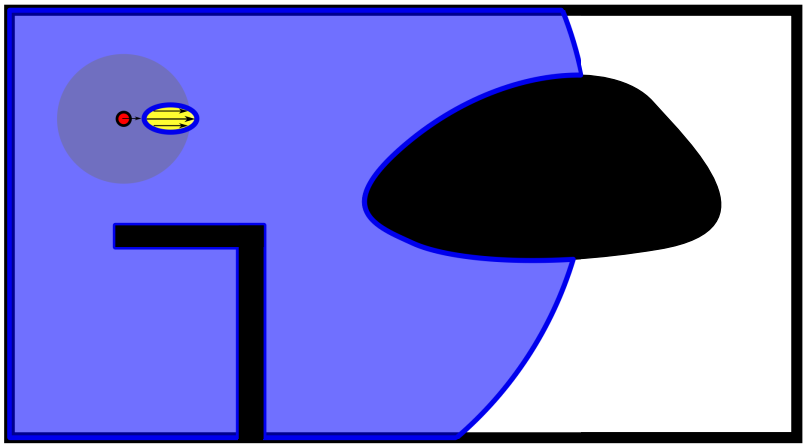


Figure 4.10: An example of a boundary arc that disappears after being expanded. The elliptical inner boundary will altogether cease to exist once the yellow region is incorporated in the explored region (blue).

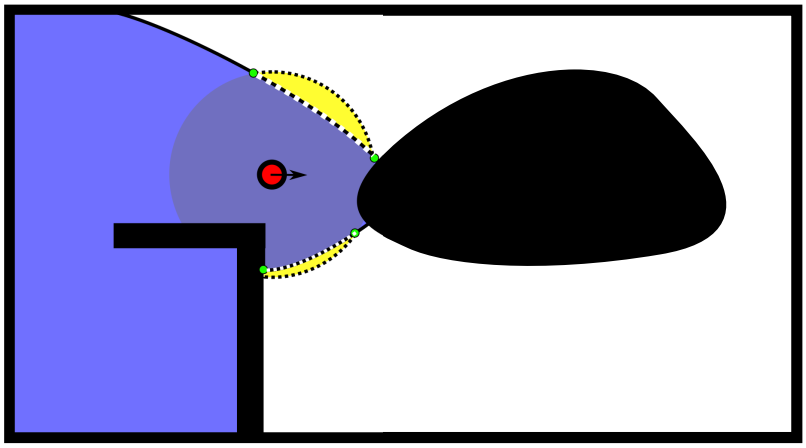


Figure 4.11: Identification of the correspondence between previous and new changed regions. The former is depicted with white and the latter with black dashed lines. The endpoints used to perform the matching are represented as yellow circles.

Overall, given a point in the new changed boundary, we determine its boundary value through the following steps:

- We establish the changed boundary region it belongs,
- We calculate its parametrization,
- We match the new changed region to the corresponding one in the previous boundary,

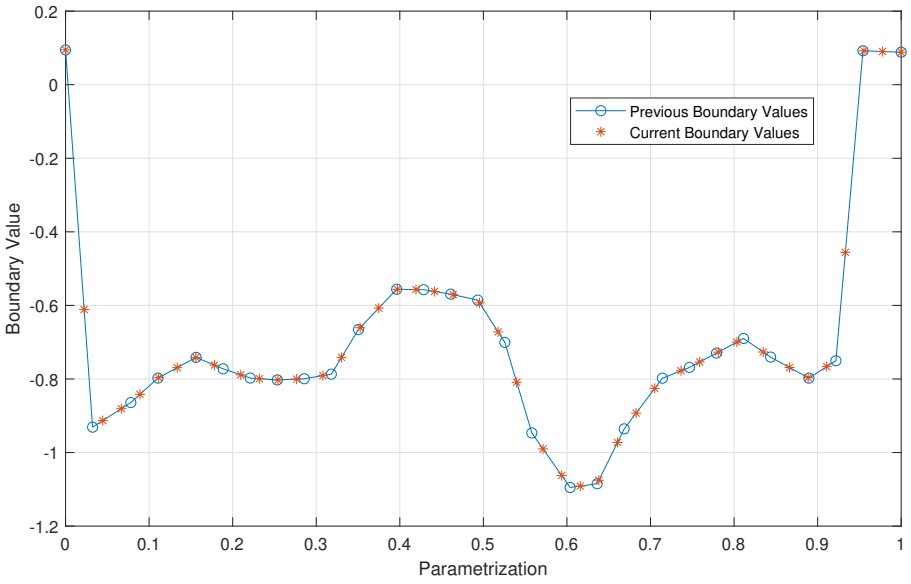



Figure 4.12: The piece-wise linear approximation of the boundary values distribution on the previous boundary (blue continuous), as well as the sampled points of the new boundary (red asterisks).

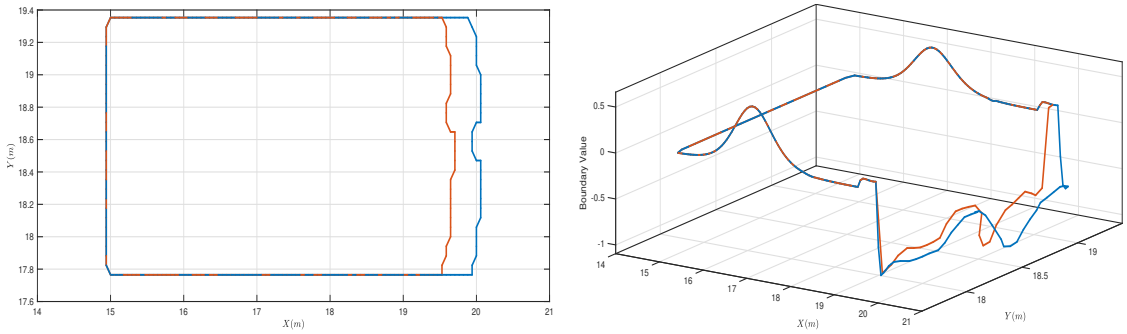
- We sample the approximate boundary value function of the previous boundary on the new point's parametrization.

The final results of boundary value projection are shown in Figure 4.13. We note that the adaptive law has not been applied yet; the figure solely illustrates how $\hat{k}$ is mapped on the changing boundary.

Although this algorithm is straightforward, a practical implementation requires considerable effort, especially taking into account the implications associated with the representation of the boundary.

## 4.6 Fast Multipole Boundary Element Method

In this section, we elaborate more on the application of the Fast Multipole Method in potential problems described by the Boundary Element Method (BEM). We present the main idea of FMBEM along with useful and intuitive computational results. This discussion serves as an outline meant to provide insight on the method, and the interested reader can refer to [15], [11] for an in-depth analysis regarding the analytical derivation, the precise details of the algorithm as well as implementation notes.




(a) The boundaries extracted in two subsequent time steps.

(b) The projection of boundary values from the previous boundary to the new one.

Figure 4.13: Applying the boundary value projection algorithm to determine the boundary values as the geometry of the boundary changes. The previous boundary is shown in red while the new one in blue.

### 4.6.1 Algorithm

The conventional BEM consists in recasting the Laplace equation

$$\nabla_p^2 \phi = 0, \ p \in \Omega \tag{4.6}$$

as a boundary integral equation

$$\phi(q) = -\int_{\partial\Omega} \left[ v(q,p) \frac{\partial \phi(q)}{\partial n_q} - \phi(q) \frac{\partial v(q,p)}{\partial n_q} \right] ds_q \tag{4.7}$$

using Green's identity. We note that $\Omega, \partial\Omega$ is the solution domain and its boundary, while $v(p,q)$ denotes the fundamental solution of Laplace's equation given by:

$$v(p,q) = \frac{1}{2\pi} \ln(\|p-q\|) \ . \tag{4.8}$$

Next, $\partial\Omega$ is discretized into a finite number of boundary elements that are used to approximate the boundary's geometry and the distribution of boundary quantities, i.e. $\phi$, $\frac{\partial\phi}{\partial n}$, upon it. A common choice is the constant element which comprises of a line segment connecting two endpoints upon which boundary quantities are assumed constant. Thus, discretizing $\partial\Omega$ with $N$ elements, denoted $\partial\Omega_i, \ i = 1, \ldots, N$, we express (4.7) in the form

$$-\frac{1}{2}\phi_i + \sum_{j=1}^{N} \left( \int_{\partial\Omega_j} \frac{\partial v}{\partial n} ds \right) \phi_j = \sum_{j=1}^{N} \left( \int_{\partial\Omega_j} v ds \right) \phi_{n,j}, \ i = 1, \ldots, N \tag{4.9}$$

where we write $\phi_n = \frac{\partial\phi}{\partial n}$ for the sake of brevity, and denote $\phi_i, \phi_{n,i}$ the corresponding boundary quantity on the $i$-th element. The integrals that appear on the previous equation

can be computed analytical or numerically for each element. Moreover, we note that one of the boundary quantities is prescribed on each element while the other is unknown. Hence, by appropriately rearranging the previous equations we can express (4.9) as a system of linear equations:

$$A \cdot x = b \tag{4.10}$$

where $x \in \mathbb{R}^N$ is the vector of unknown boundary quantities, $A \in \mathbb{R}^{N \times N}$ is the matrix of coefficients, and $b \in \mathbb{R}^N$ is a constant known vector. Unfortunately, forming $A$ requires $N^2$ operations to compute the integrals in (4.7) for every pair-wise element combination and, also, $N^2$ memory requirements to store it, being dense and non-symmetric. These facts hinder the use of the conventional BEM in large domains. (A more rigorous analysis and derivation of the BEM can be found in [17].)

The main idea of FMBEM is to reduce the aforementioned computational cost by exploiting analytical results, to avoid the need to evaluate all the coefficients of the matrix $A$, devise an efficient method to directly compute the matrix-vector product $A \cdot x$ and employ an iterative method to derive the system's solution. Without delving in the mathematical details, which can be found in [11], we note that the integrals

$$\int_{\partial\Omega_j} \frac{\partial v(p,q)}{\partial n} \phi(q) ds_q \tag{4.11}$$

$$\int_{\partial\Omega_j} v(p,q) \frac{\partial \phi}{\partial n}(q) ds_q \tag{4.12}$$

can be approximated with a specified error using a complex polynomial expansion with an appropriate number of terms. Thus, the influence of the element centered at $p$ on the element $\partial\Omega_j$ can be computed by evaluating the corresponding polynomial, termed multipole expansion, given that the two elements are sufficiently "far" apart. The crucial advantage offered by this approach is that the derived polynomial coefficients depend solely on the element that acts as source, i.e. the one centered at $p$. Hence, we can group elements that are located in close proximity and derive a joint expression for their influence on far away elements simply by adding their respective polynomial coefficients. To formally define the concepts of "far away" and "close proximity", a hierarchical subdivision of the domain is derived using quadtree decomposition and we consider elements residing on the same cell to be close, and elements residing on non-adjacent cells to be far away. This reasoning allows an FMBEM algorithm with $\mathcal{O}(N \log N)$ computational complexity to be devised.

The previous method suggests that in order to compute (4.11), (4.12) for a particular element $\partial\Omega_i$ it is necessary to identify all far away cells and evaluate their corresponding polynomials and, also, incorporate the influence of nearby elements through direct evaluation of the integrals (analytical or numerical). A further improvement can be achieved through the following approach. We can derive a polynomial approximation, for each cell, termed local expansions that incorporates the influence of all far away cells with a bounded error specified by number of polynomial terms included. Thus, computation of the integrals

boils down to a single evaluation of the local expansion and direct evaluation of nearby elements. This is the core of the FMBEM that enables linear computational complexity and memory requirements.

Overall, the steps of the FMBEM algorithm, described from a high-level perspective, are the following:

1. Discretize $\partial\Omega$ using boundary elements and imposing appropriate boundary conditions.
2. Derive a quadtree decomposition of the boundary elements.
3. Compute the multipole expansion coefficients for each populated cell in the tree structure. In this step further analytical considerations are employed to minimize the number of required operations. In particular, the coefficients on coarser levels of the tree can be determined with minimal computational burden, given the coefficients on finer levels. Thus, this step is performed on a fine to coarse level basis.
4. Compute the local expansion coefficients starting from coarse decomposition levels.
5. Evaluate integrals using local expansions to account for far away cells and direct evaluation for nearby elements.
6. Perform the matrix-vector product $A \cdot x$ and the respective iteration of the iterative solver.
7. Repeat until convergence.

### 4.6.2 Numerical Results

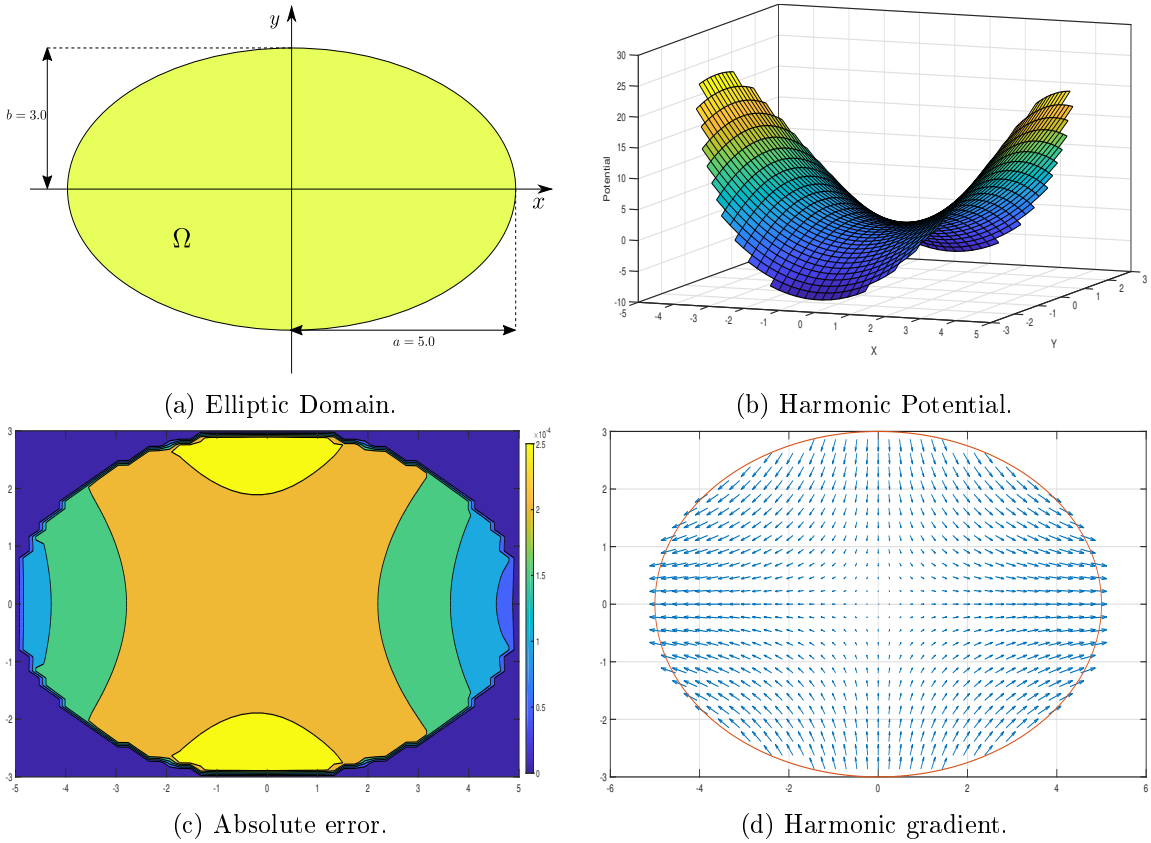
In this subsection we present computational results acquired by applying the FMBEM in several different domains. Particularly, we compute the harmonic potential and vector field in domains with specific boundary conditions, that admit analytical solutions in order to test the correctness of the implementation. The first two case studies are based on examples from [17], while the third one is derived from [11].

The first domain we study is an ellipse, whose dimensions are shown in Figure 4.14a, with appropriate Neumann boundary conditions. Specifically, the prescribed value of the normal derivative along the boundary is given by

$$\bar{\phi}_n = \frac{2(b^2x^2 - a^2y^2)}{\sqrt{b^4x^2 + a^4y^2}} \tag{4.13}$$

and the corresponding analytical solution is

$$\phi = x^2 - y^2 + C \ . \tag{4.14}$$




(a) Elliptic Domain.

(b) Harmonic Potential.

(c) Absolute error.

(d) Harmonic gradient.

Figure 4.14: The elliptic domain test case.

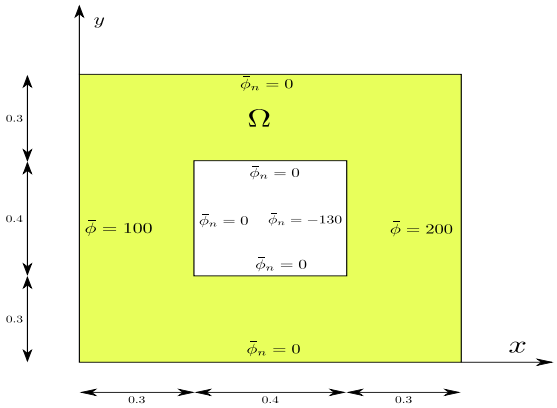
Also, the gradient of $\phi$ can be easily computed:

$$\nabla_{(x,y)}\phi = \begin{bmatrix} 2x \\ -2y \end{bmatrix} . \tag{4.15}$$

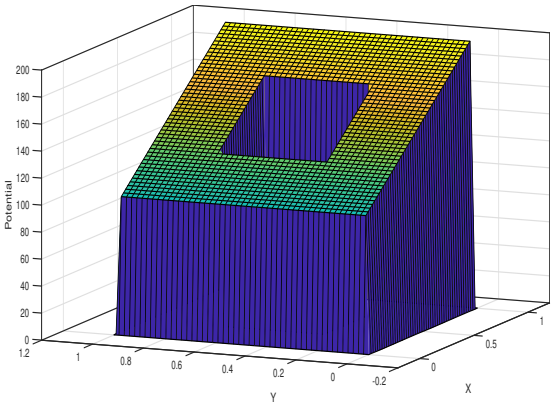
The numerical solution is obtained through the FMBEM and Figure 4.14 illustrates the numerically computed potential, the respective error w.r.t. the analytical solution, as well as the harmonic gradient field. The number of boundary elements employed to discretize the boundary is 500 and the computation time is $0.1362s$.

The next domain to be considered is rectangular, including a rectangular hole in its center. The domain's geometry and the imposed boundary conditions are illustrated in Figure 4.15a. The exact solution of the aforementioned problem is:

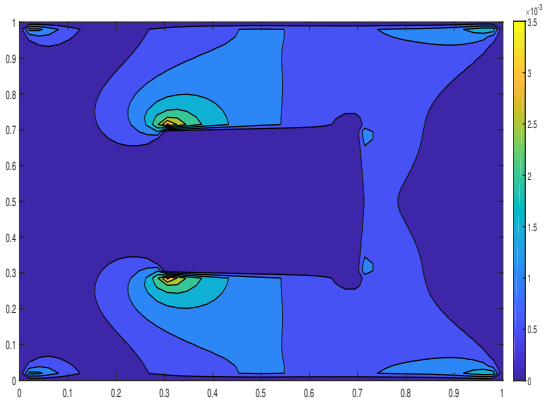
$$\phi = 100(1 + x) \tag{4.16}$$



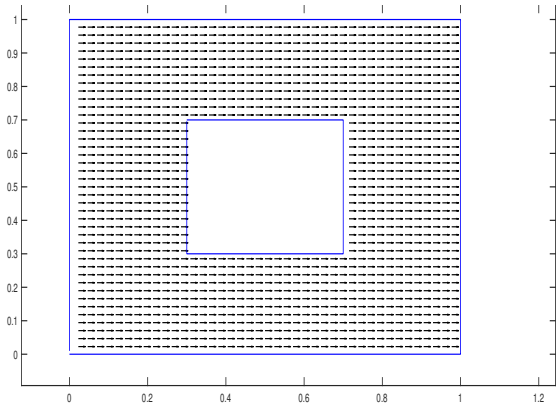

(a) The rectangular domain and the imposed boundary conditions.




(b) Harmonic Potential.



(c) Absolute error.



(d) Harmonic gradient.

Figure 4.15: The rectangular domain test case.

and the gradient obviously is:

$$\nabla_{(x,y)}\phi = \begin{bmatrix} 100 \\ 0 \end{bmatrix} . \tag{4.17}$$

Similarly, we provide a graph of the potential, the error and the harmonic vector field in Figure 4.15, where the FMBEM computation employed 2000 boundary elements and required $0.9752s$ of computation time.

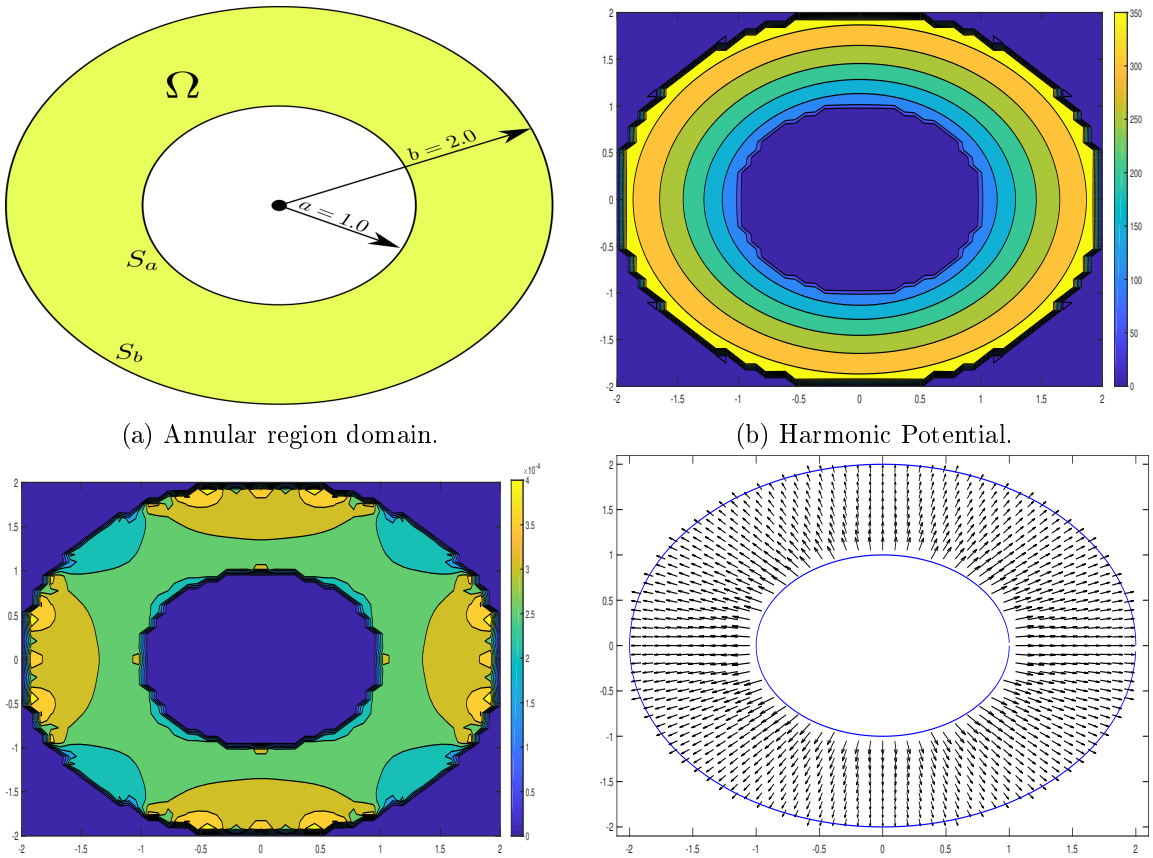
The final case study constitutes the annular region depicted in Figure 4.16a. We impose the boundary conditions $\phi = \phi_a$ on the inner boundary $S_a$, and $\phi_n = \phi_{n,b}$ on the boundary $S_b$. In this setting, the analytical solution is given by:

$$\phi(r) = \phi_a + \phi_{n,b} b \ln\left(\frac{\|\boldsymbol{r}\|}{a}\right) \tag{4.18}$$

where $\boldsymbol{r}$ is the radial coordinate in polar coordinate-system center at $[0,0]^T$. The gradient of the potential field is calculated as follows:

$$\nabla_{(x,y)}\phi = \phi_{n,b} b \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|^2} \tag{4.19}$$

Discretizing the boundary with 5000 boundary elements, we provide the same graphs as previously in Figure 4.16. The solution time for this case was $3.3401s$.




(a) Annular region domain.

(b) Harmonic Potential.

(c) Absolute error.

(d) Harmonic gradient.

Figure 4.16: The annular region domain test case.

# Chapter 5

# Results

This chapter provides an extensive demonstration of our algorithm's performance through simulation studies in realistic environments. Initially, in Section 5.1, we discuss the setup utilized to perform the simulations as well as some practical implementation aspects. The main results are presented in Section 5.2, where we specify the chosen parameters of our control law, we illustrate the ground truth maps employed in the subsequent simulations and demonstrate the robot's trajectory, as well as the generated occupancy map. Moreover, we discuss and demonstrate the usefulness of the term $k_2'$ in (3.7). Finally, we compare our methodology with a competing one from the related literature in Section 5.3.

## 5.1 Hardware Setup

Simulations were executed on an Asus laptop with an Intel i5 processor operating at 1.6GHz, 6GB RAM on a Windows operating system. The simulation environment, i.e. maps, sensors, and robot, was realized in VREP; the specific version being VREP PRO EDU 3.6.1. The control law was implemented in MATLAB 2018a, as well as the visualization of the results. Finally, the FMBEM solver was implemented in the C++ programming language.

At this point, we note that some functions of the FMBEM solver were initially coded in MATLAB and translated in C++ using the Coder toolbox offered by MATLAB. Specifically, certain parts of the code can be conveniently modified to run in parallel using the latter programming language. The parallelized C++ code generated by MATLAB was considerably faster than its serial counterpart.

## 5.2 Simulation Results

### 5.2.1 Controller Parameters

Although the design of our control scheme ensures safety and full exploration, yet an appropriate choice of parameters can lead to improved performance. The parameters were heuristically tuned by performing multiple simulations in different environments and comparing the obtained results with respect to:

- Exploration time,
- Path length,
- Map quality, and
- Average computation time.

In particular, the parameter $K_u$ that is used to scale the robot's velocity, directly affects the exploration time. While increasing $K_u$ generally results in reduced time requirements, very large values occasionally lead the robot to rush in the direction of obstacles, before the influence of $s$ takes effect to guarantee appropriate deceleration. Map quality is directly determined by $m_r$, which defines the fineness of the occupancy map and, consequently, its accuracy. Nonetheless, given that the boundary of the explored region is obtained as a point-wise approximation using the centers of grid cells, increasing $m_r$ would result in an increased number of boundary elements and, thus, heavier computational load. Finally, the parameters associated with probabilities, i.e. $\alpha, \alpha'$, played a minor role because in practice $\Pr(m_i)$ quickly converges to its true value. The selected values for the parameters are shown in Table 5.1.

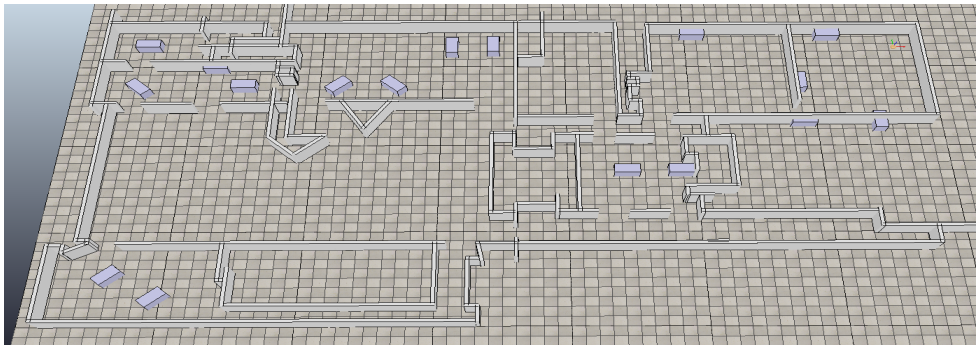
| $r$ | $m_r$ | $\alpha$ | $\alpha'$ | $K_u$ |
|---|---|---|---|---|
| $6m$ | 13 cells/$m$ | 0.8 | 0.7 | 1.2 |
| $R_1$ | $\bar{k}$ | $\epsilon_1, \epsilon_w$ | $\epsilon_2$ | $\mu_1$ |
| 0.2 | 1.0 | 0.01 | 0.01 | 10/8 |

Table 5.1: The values specified for our control scheme's parameters.

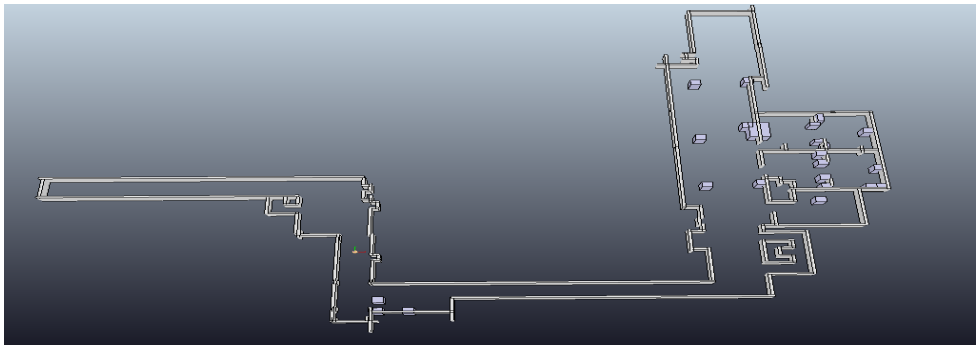
### 5.2.2 Environments

In order to verify and demonstrate the efficacy of our control law, we deploy it in three different environments. These environments were selected so as to provide a realistic representation of actual exploration conditions. Particularly, the first two environments are adapted from the Radish Repository and correspond to the Fort AP Hill [26] and Claxton

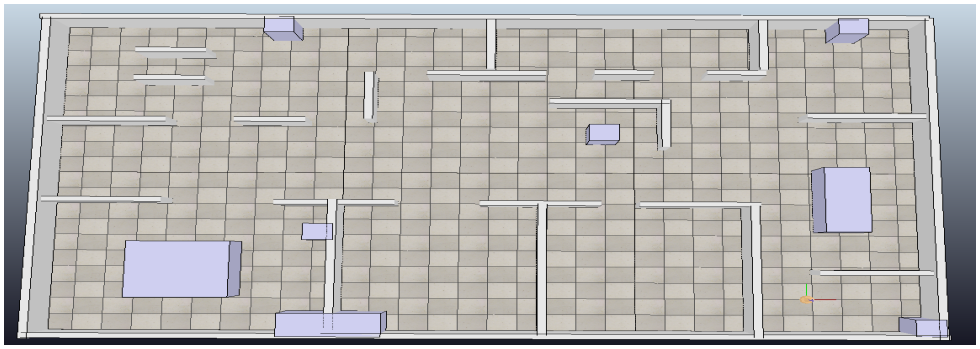
CS building [27] datasets. The third map is a synthetic structured map similar to an office environment. The VREP implementation of each workspace is shown in Figure 5.1.



(a) Fort AP Hill VREP map.



(b) Claxton CS Building VREP map.



(c) Synthetic office map.

Figure 5.1: The VREP environments used for the simulation results.

### 5.2.3 Generated Trajectories and Maps

Next, we validate and discuss our controller's performance by considering the maps and trajectories generated and evaluating the path length and total exploration time for each one of the three scenarios. Additionally, to demonstrate the usefulness of $k_2'$ we also include simulation results without employing it in (3.7).

A representative trajectory in the Fort AP hill environment and the resulting occupancy map is shown in Figure 5.2.

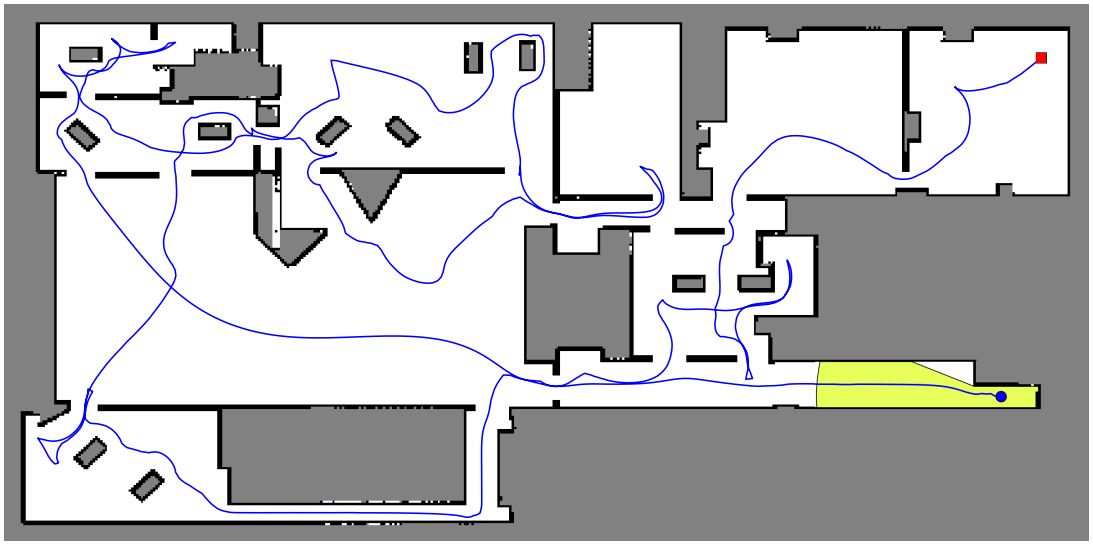
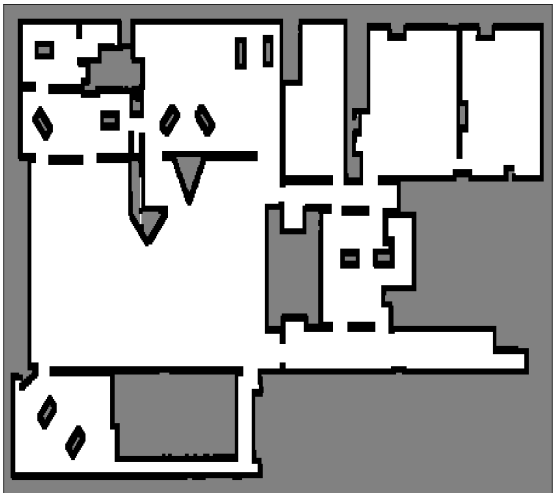


Figure 5.2: Generated occupancy grid map and the corresponding robot trajectory for the AP hill environment. The initial and final position of the robot are shown as a square and circle respectively. The yellow shaded area corresponds to the sensing region in the final position. (Dimensions: $35m \times 18m$)

Also, the robot's sensing region in the final position is included to establish a comparison between the range of the sensors and the size of the workspace. The figure indicates that complete exploration is achieved since the entirety of the environment is explored, verifying our theoretical conclusions. It can be noted that a small number of isolated cells on the boundary remain free, which is a consequence of the way the boundary is extracted. In particular, as discussed previously, the obstacle cells are augmented (in the dilation sense) to ensure the robustness of the boundary extraction algorithm. The augmented occupancy grid map and the extracted boundary are illustrated in Figure 5.3. Clearly, the difference between the two maps is trivial and, also, in practice such an augmentation of the obstacles would be necessary, so as to account for the robot's non-zero dimensions.



(a) Augmented Fort AP Hill occupancy grid map.


(b) Extracted boundary (dimensions in meters).

Figure 5.3: Augmentation of the obstacles to achieve robustness of boundary extraction.

Similar graphs are included for the remaining two environments in Figure 5.4 and Figure 5.5. Observing the occupancy grid maps and comparing them to the ground truth models in VREP, we deduce that the proposed controller generates accurate representations of the environment. Moreover, we stress out that the robot maintains, in general, a considerable distance from nearby obstacles. Indeed, in partially known environments, such as the ones studied, it is unsafe for the robot to be in close proximity to obstacles since unexpected map changes may occur, as a consequence of possible errors in past measurements and the unpredictability of future ones.

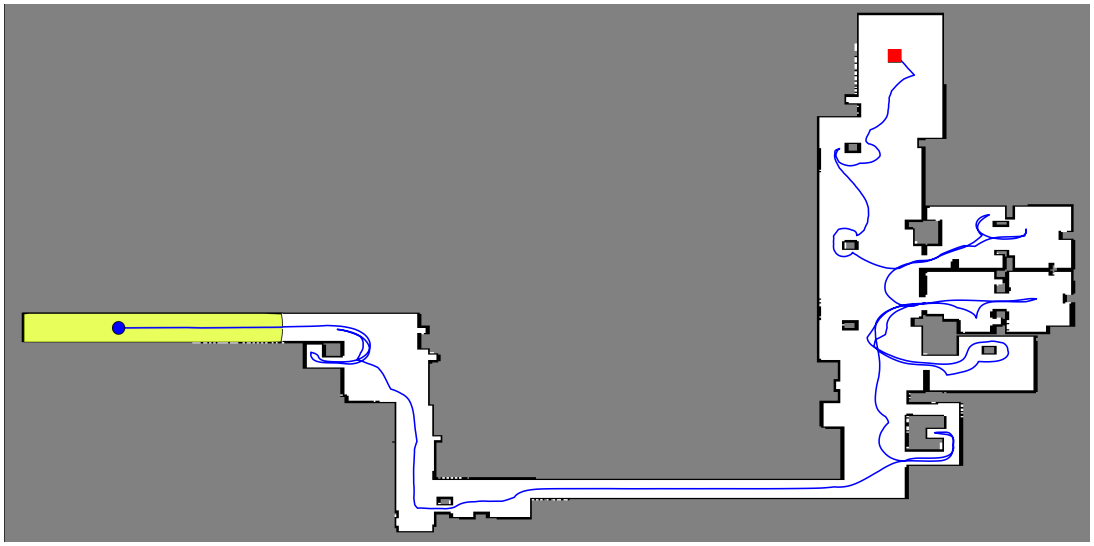


Figure 5.4: Generated occupancy grid map and the corresponding robot trajectory for the Claxton CS Building. (Dimensions: $34m \times 43m$)

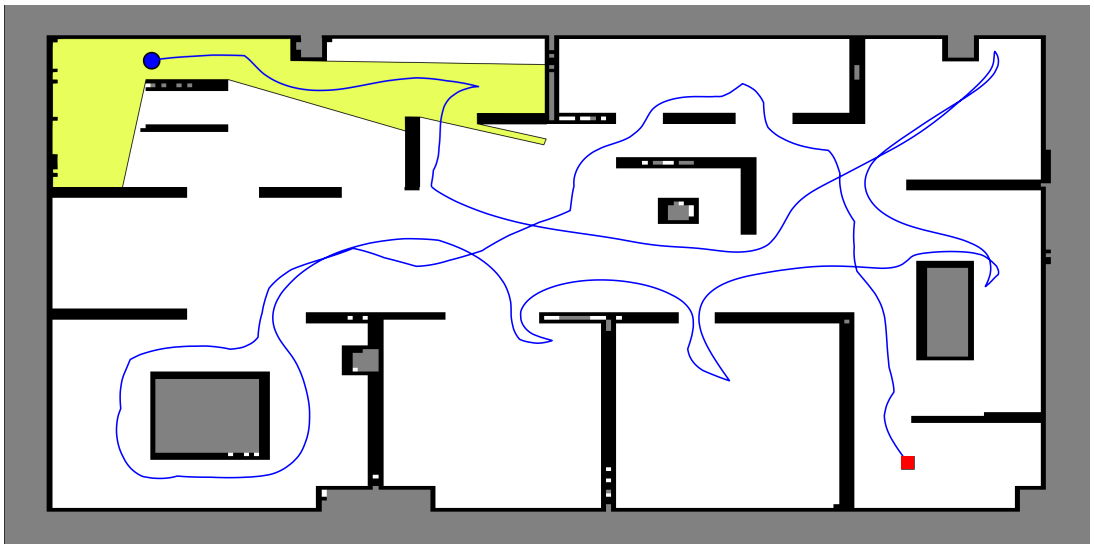
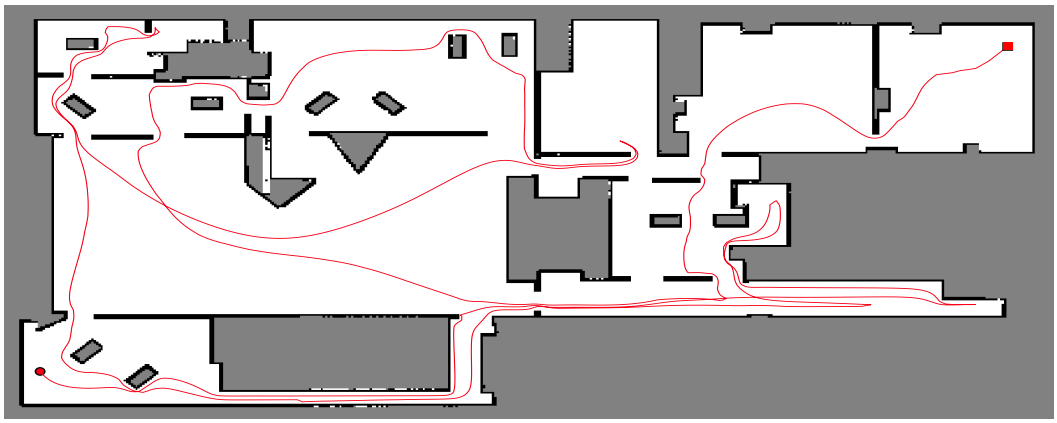
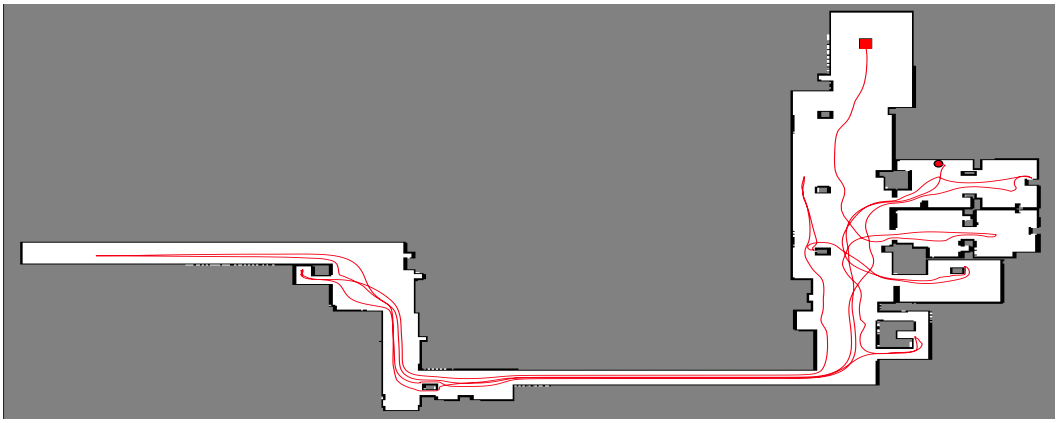


Figure 5.5: Generated occupancy grid map and the corresponding robot trajectory for the synthetic office environments. (Dimensions: $16m \times 11m$)

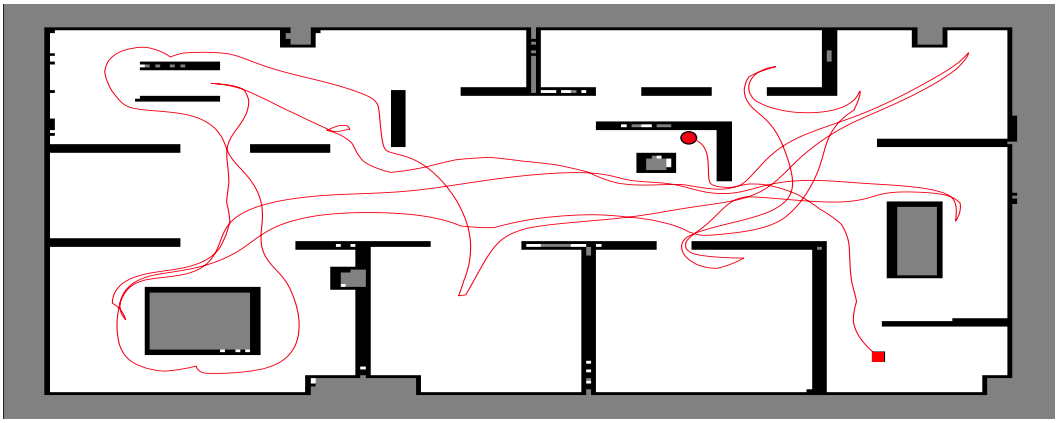
Additionally, to demonstrate the usefulness of $k_2'$ we also include simulation results without employing it in (3.7), as shown in Figure 5.6.



(a) Fort AP Hill.



(b) Claxton CS Building.



(c) Synthetic office environment.

Figure 5.6: Simulation results without the inclusion of $k_2'$ in (3.7).

As can be clearly observed, in the absence of $k_2'$ the robot tends to visit certain regions of $\mathcal{W}$ multiple times, resulting in longer and more complicated paths. This is a consequence of the fact that the normal derivative is practically uniform along each of $\partial\mathcal{E}_F$ and $\partial\mathcal{E}_O$, since $\hat{k}_t$ depends solely on the probability of occupancy of the corresponding boundary point. Hence, the robot is attracted to the larger regions $\partial\mathcal{E}_{Fi},\ i \in \mathcal{I}_F$ leaving small unexplored patches in the areas it currently explores, which require the robot to return at a later time. Furthermore, comparing the responses with and without $k_2'$, we note that in the latter case the generated trajectories tend to pass closer to obstacles, which can be explained by the following reasoning. Generally, the length of $\partial\mathcal{E}_O$ is significantly larger than that of $\partial\mathcal{E}_F$, especially near the end of the exploration task and, also, the imposed boundary conditions are mostly uniform. But, to ensure the compatibility of $k$, it is obvious that the normal derivative on $\partial\mathcal{E}_O$ will be considerably smaller (in absolute terms) than on $\partial\mathcal{E}_F$, to account for their disproportionate lengths. Thus, the repulsiveness of obstacles is significantly reduced, leading the robot to navigate in their close proximity. This problem is circumvented by the inclusion of $k_2'$ because obstacles located far away have negligible influence on the compatibility integral, allowing nearby obstacles to exert substantial repulsiveness.

Finally, in Table 5.2 we report the path length of the generated trajectory as well as the exploration time, averaged over 10 runs starting from randomly chosen initial configurations for each environment. The numbers in parentheses correspond to the results derived without employing $k_2'$ in (3.7).

Table 5.2: Evaluation of the proposed controller. Numbers in parentheses correspond to simulations without $k_2'$ in (3.7).

| Environment | Path Length ($m$) | Time ($s$) |
|---|---|---|
| Fort AP Hill | 202.7 (207.1) | 1336.1 (1543.4) |
| Claxton CS | 165.2 (259.2) | 519.8 (1194.4) |
| Synthetic | 74.3 (114.1) | 199.5 (526.1) |

## 5.3 Computational Comparison with Related Literature

In this section we compare our control scheme to an algorithm from the related literature. In particular, the algorithm proposed in [3], which will be abbreviated BayOpt, employs Bayesian optimization to train a Gaussian Process that predicts the Mutual Information of candidate vantage points. This approach can be considered state-of-the-art as it belongs to the top 4% of papers of its field (96th percentile in Scopus).

To contrast the two approaches, we recreate an environment presented in [3] and analyze the resulting trajectories and the average computation time required per iteration. Unfortunately, a comparison of path lengths cannot be established because these results are

expressed as the number of exploration steps in the aforementioned work, without explicitly stating the length of each step. Moreover, we use the same parameters, wherever relevant, as the ones presented in BayOpt to maintain an unbiased comparison. Specifically, the sensing radius is set to $r = 1m$ and the grid resolution to $m_r = 10$ cells$/m$.

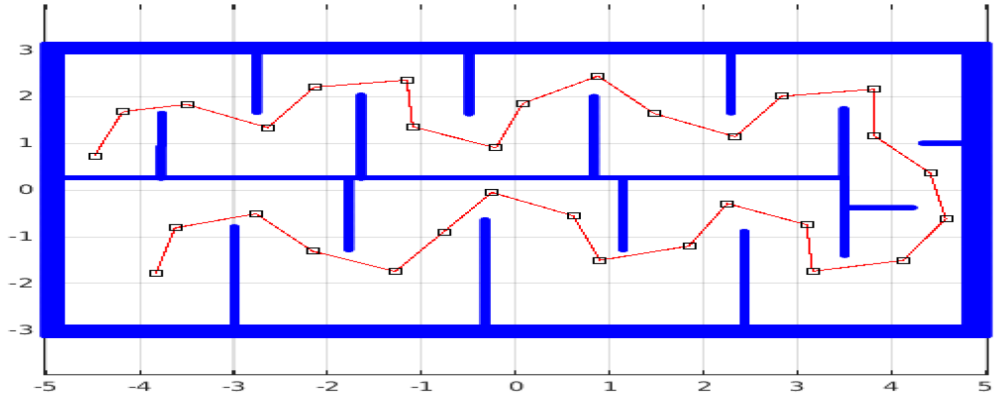
The examined environment is shown in Figure 5.7a, along with the path followed by the competing approach. The path generated by our controller is demonstrated in Figure 5.7b, and it can easily be perceived to be longer, as the robot traverses the map twice. Nonetheless, observing the path in Figure 5.7a and given that $r = 1m$, it can be deduced that certain parts of the environment are left unexplored, for instance, but not limited to, the bottom left corner which is clearly more than $1m$ from any point along the robot's trajectory. Hence, the greater length of our path is a consequence of the more accurate mapping of the environment. Additionally, increasing the sensing radius to $1.5m$ significantly reduces the path length of our approach (see Figure 5.7c). On the contrary, such a modification may not benefit the competing methodology, as it relies heavily on the candidate configurations to be generated within the sensing region but in considerable close proximity to the robot. Lastly, our approach exceeds BayOpt in terms of the resulting path's smoothness, which determines, up to a certain degree, how feasible is its execution by an actual robot.

The average and standard deviation of the computation time per step are reported in Table 5.3. Our control scheme's and BayOpt's data are sampled over 20 and 100 trials, respectively, starting from random initial configurations. The computational cost of the proposed control law is substantially lower, outperforming BayOpt by a factor of 2.

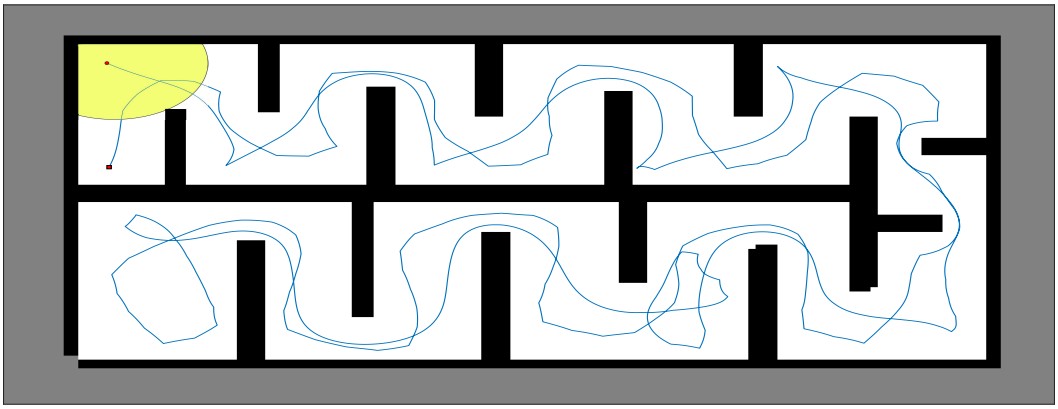
Table 5.3: Comparison between our control scheme and the one proposed in [3].

| Computation Time Per Step ($s$) | BayOpt | Harmonic |
|---|---|---|
| Average | 3.71 | 1.21 |
| Standard Deviation | 0.15 | 0.41 |

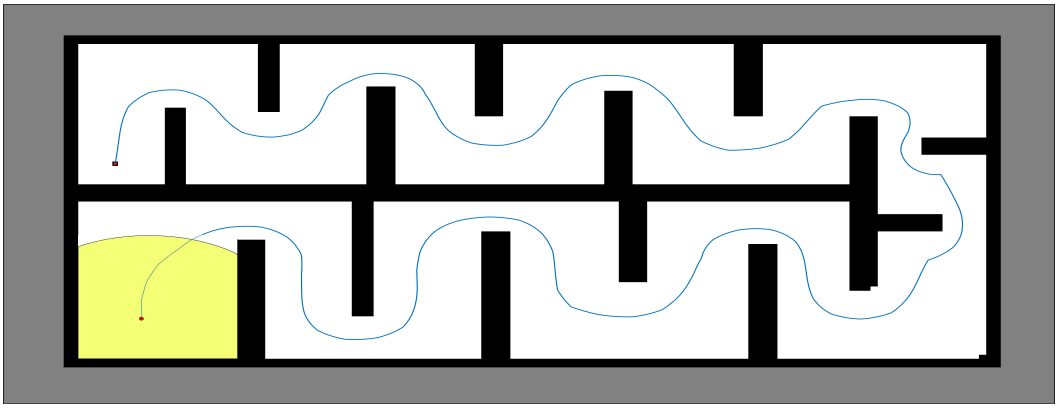
Overall, our control scheme compares favorably to BayOpt in a number of aspects, with the most prevalent being computational efficiency. This is further supported by the considerably smaller domain's studied in [3], compared to the ones presented in this work.




(a) The path followed by the BayOpt algorithm.



(b) The path followed by our algorithm with $r = 1m$.



(c) The path followed by our algorithm with $r = 1.5m$.

Figure 5.7: Comparison between our control law and the BayOpt algorithm, in the synthetic maze map presented in [3].

# Chapter 6

# Discussion

The current chapter serves as a conclusion for this thesis. In Section 6.1 we assess the overall results and consider whether the defined goals were attained. Section 6.2 elaborates on the difficulties that arose both on a theoretical, as well as on an implementation level. Finally, we conclude by proposing future research directions in Section 6.3.

## 6.1 Assessment

Contemplating on the analytical and computational results demonstrated in this work, we can conclude that the goals described in Chapter 2 have been accomplished. In particular, we developed a rigorous model of the exploration process and devised a controller, based on harmonic potential fields, which under the realistic assumption of consistent sensor measurements, can provably achieve full exploration of any compact and connected workspace in finite time, from almost any initial configuration. We devised an adaptive control law that adjusts the imposed boundary conditions in real-time, hence, ensuring obstacle avoidance and prescribing the robot's behavior up to a certain degree. Furthermore, we employed a FMBEM scheme for solving the Laplace equation efficiently, which allowed us to significantly reduce the computational and memory requirements of our navigation scheme. The efficacy of our control law was demonstrated through extensive simulation studies in realistic environments. Finally, comparisons were made with state-of-the-art algorithms drawn from the related literature, and it was shown that our approach compares favorably in a number of aspects.

## 6.2 Difficulties

The main source of difficulties in developing the proposed control law originated in the theoretical part of this work. A recurring issue was establishing the absence of stable equilibria in the interior of the workspace. For instance, an initial approach, inspired by the

work in [12], consisted in deriving the system's Jacobian on equilibrium points and using properties of the Hessian matrix of harmonic potential fields to prove instability using Lyapunov's indirect method. Nevertheless, discontinuities arising during the exploration process hindered this approach. Multiple ideas were considered before arriving at the solution presented in the corresponding chapter.

Ensuring the safety of the control law constituted another significant analytical difficulty. Initially, we pursued the design of a continuous adaptive law which proved to be futile when faced with rendering newly-observed obstacles repulsive. To see the complications arising in this setting, replace the discontinuous term $b_e$ with a continuous one, whose magnitude increases as the robot approaches obstacles but vanishes away from them. This reasonable choice suffers from the existence of a boundary, between the regions where the safety term is operating or not, upon which equilibria cannot be characterized as unstable, using the previously presented approach, since the adaptive laws will operate arbitrarily close to them. Hence, the use of a discontinuous term was unavoidable but not without its own difficulties, such proving that this term switches a finite number of times and that it operates for finite time intervals.

As far as the implementation of the controller is concerned, a number of different modules had to be realized, with the FMBEM being undoubtedly the most demanding one. Although its analytical derivation is considerably involved, drawing heavily from complex analysis and calculus, the algorithmic part of the method is its most complicated component, incorporating multiple interconnected functionalities based on the corresponding analytical results. Fortunately, free and open-source code is provided by the author of [11], as a complement to the book, which can serve as a useful guideline.

Last but not least, the boundary extraction algorithm was crucial for the overall control scheme and, hence, necessitated careful implementation. Deriving the boundary of the occupancy grid map required both robustness, being necessary for the Laplace equation numerical solver, as well as accuracy. Thus, the implemented algorithm was tested in two different settings. Initially, it was deployed "offline" on manually generated maps to test its effectiveness, followed by online deployment on multiple maps to identify possible errors.

## 6.3 Future Directions

The autonomous exploration problem constitutes a well-studied research topic associated with a number of extensions that enjoy considerable interest from the robotics community. Future research directions include considering 3D workspaces since the Fast Multipole Boundary Element Method remains applicable in three dimensions and, also, implementing our algorithm on an actual robot in order to evaluate its performance in real world conditions. Finally, a significant future research direction is extending the proposed methodology to address the integrated exploration problem, thus, actively taking into account the uncertainty in the robot's configuration in the planning process.

# Bibliography


[1] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," in *Robotics: Science and Systems*, 2005.

[2] R. Shade and P. Newman, "Choosing where to go: Complete 3d exploration with stereo," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2806–2811.

[3] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with bayesian optimization," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1816–1822.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, July 1997, pp. 146–151.

[5] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, pp. 4791–4798, 06 2015.

[6] M. Juliá, A. Gil, and s. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, 05 2012.

[7] F. Amigoni, "Experimental evaluation of some exploration strategies for mobile robots," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 2818–2823.

[8] E. Prestes, M. A. P. Idiart, P. M. Engel, and M. Trevisan, "Exploration technique using potential fields calculated from relaxation methods," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 4, Oct 2001, pp. 2012–2017 vol.4.

[9] V. A. M. Jorge, R. Maffei, G. S. Franco, J. Daltrozo, M. Giambastiani, M. Kolberg, and E. Prestes, "Ouroboros: Using potential field in unexplored regions to close loops," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2125–2131.

[10] R. Silveira, E. Prestes, and L. Nedel, "Fast path planning using multi-resolution boundary value problems," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 4710–4715.

[11] Y. Liu, *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, 2009.

[12] P. Vlantis, C. Vrohidis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Robot navigation in complex workspaces using harmonic maps," in *2018 IEEE International Conference on Robotics and Automation*, May 2018, pp. 1726–1731.

[13] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, pp. 116 – 121.

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

[15] R. Beatson and L. Greengard, "A short course on fast multipole methods," in *Wavelets, Multilevel Methods and Elliptic PDEs*. Oxford University Press, 1997, pp. 1–37.

[16] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, July 1986.

[17] J. T. Katsikadelis, "Chapter four - numerical implementation of the bem," in *The Boundary Element Method for Engineers and Scientists (Second Edition)*, second edition ed., J. T. Katsikadelis, Ed. Oxford: Academic Press, 2016, pp. 59 – 112. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128044933000047

[18] E. P. e Silva, M. Trevisan, M. A. P. Idiart, and P. M. Engel, "Bvp-exploration: further improvements," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 4, Oct 2003, pp. 3239–3244 vol.3.

[19] R. Maffei, V. A. M. Jorge, E. Prestes, and M. Kolberg, "Integrated exploration using time-based potential rails," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3694–3699.

[20] E. Prestes and P. M. Engel, "Exploration driven by local potential distortions," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 1122–1127.

[21] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 661–666.

[22] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 487–494.

[23] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[24] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[25] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003.

[26] A. Howard, "Fort ap hill," 2010, Accessed: 2019-07-15. [Online]. Available: https://dspace.mit.edu/handle/1721.1/62267

[27] M. Bailey, "Claxton cs building, university of tennessee," 2010, Accessed: 2019-07-21. [Online]. Available: https://dspace.mit.edu/handle/1721.1/62283