Patrolling and target tracking using cooperative artificial agents

Pedro Miguel de Oliveira Murta Carrasqueira

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors:  Prof. Cláudia Alexandra Magalhães Soares
              Prof. João Pedro Castilho Pereira Santos Gomes

Examination Committee
Chairperson:  Prof. José Fernando Alves da Silva
Supervisor:   Prof. Cláudia Alexandra Magalhães Soares
Member of the Committee: Prof. Rodrigo Martins de Matos Ventura

November 2018
**Resumo**

Os últimos avanços tecnológicos têm impulsionado a ambição humana em relação à inteligência artificial cooperativa. Em particular, cada vez mais, somos testemunhas de tentativas para construir equipas de agentes que procurem um comportamento inteligente. Uma das áreas relacionadas com este campo é a colocação de uma rede de agentes e as rotas que devem seguir de forma a monitorizar todas as actividades numa determinada região. Esta tese utiliza um conjunto de sensores com limites em alcance e velocidade para dar resposta a três missões: Cobertura de Área, Patrulha, e Seguimento de Alvos.

Uma nova extensão do algoritmo de Lloyd para o k-means, k-disks, que utiliza a distância a bolas, permite encontrar uma estratégia de cobertura de um conjunto de pontos âncora numa abordagem de optimização que garante uma taxa óptima de cobertura baseada em optimização alternada. A Cobertura da Área utiliza o algoritmo k-disks e a inicialização k-means++.

O conceito de massa para os pontos âncora permite determinar o percurso para Patrulha e Seguimento de Alvos, onde o objectivo é manter valores constantes para as massas.

A abordagem é robusta e permite resolver problemas em áreas de quaisquer dimensões e formas desde que se tenha uma definição rigorosa dos pontos âncora. O algoritmo é autónomo e permite alternar entre as missões sem qualquer configuração específica.

**Palavras-chave:** k-means, sensores móveis, Cobertura de Área, Patrulha, Seguimento de Alvos, monitorização de actividade.
Abstract

The latest technological strides have been boosting the human ambition towards cooperative artificial intelligence. In particular, we increasingly witness large efforts for building teams of artificial agents behaving in an intelligent way. One of the research questions under this scope is the deployment of a network of agents and the coordination of their paths in order to efficiently monitor all activities within a region of interest. This thesis uses a set of sensors with limited sensing range and speed to address three missions: Area Coverage, Patrolling, and Target Tracking and Escorting.

A novel extension of Lloyd’s $k$-means algorithm, termed $k$-disks, uses the distance to balls and allows to find a covering strategy of a set of Anchor Points in an optimization based approach based on alternating minimization. The Area Coverage mission is performed using $k$-disks, our proposed extension and $k$-means++ initialization algorithm.

The concept of masses for the Anchor Points allows to plan a path for the missions, where the objective is to keep constant mass values.

Our approach is robust and applicable to problems in regions of any dimension and shape, depending on the definition of the Anchor Points. The algorithm is able to run autonomously and to effortlessly change its mission from Patrolling to Target Tracking and Escorting.

Keywords: $k$-means, mobile sensors, Area Coverage, Patrolling, Target Tracking and Escorting, activity monitoring.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resumo</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td>Nomenclature</td>
<td>xv</td>
</tr>
<tr>
<td>Glossary</td>
<td>xv</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem statement and objectives</td>
<td>1</td>
</tr>
<tr>
<td>1.2 State of the art</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Thesis Outline</td>
<td>6</td>
</tr>
<tr>
<td>2 Background</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Optimization and convex optimization problems</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 FISTA - Fast Iterative Shrinkage-Thresholding Algorithm</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 Proximal operator</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3 Lipschitz Constant</td>
<td>9</td>
</tr>
<tr>
<td>2.2 k-means clustering</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Lloyd’s algorithm</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 k-means++</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Voronoi diagram</td>
<td>12</td>
</tr>
<tr>
<td>3 Area discretization and problem transformation</td>
<td>13</td>
</tr>
<tr>
<td>3.1 Point Placement</td>
<td>13</td>
</tr>
<tr>
<td>3.1.1 Sensor probabilistic model</td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 $\varepsilon$-full area coverage</td>
<td>14</td>
</tr>
<tr>
<td>3.1.3 Anchor Point $A$ location</td>
<td>14</td>
</tr>
<tr>
<td>3.1.4 Number of anchor points</td>
<td>16</td>
</tr>
<tr>
<td>3.2 k-disks: k-means problem extension</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1 k-means++ and Lloyd’s algorithm to the problem extension</td>
<td>19</td>
</tr>
<tr>
<td>3.3 Results on the point distribution</td>
<td>21</td>
</tr>
</tbody>
</table>
3.3.1 Analysis of the point distribution ........................................ 21
3.3.2 Examples on three dimensions ......................................... 22
3.3.3 Analysis of the number of points ..................................... 23

4 Area Coverage ................................................................. 27
4.1 Approach ............................................................................. 27
  4.1.1 Mass of the balls ........................................................ 27
  4.1.2 Lipschitz constant ....................................................... 28
4.2 Algorithm for Area Coverage mission .................................. 29
4.3 Results and discussion for Area Coverage mission ................. 30
  4.3.1 Area Coverage with high density of sensors ..................... 30
  4.3.2 Area Coverage with low density of sensors ..................... 32
  4.3.3 Area Coverage with different sensors ............................. 32
  4.3.4 Area Coverage in the 3 dimensional case ....................... 33
  4.3.5 Monte Carlo Experiment .............................................. 34

5 Patrolling .............................................................................. 37
5.1 Approach ............................................................................. 37
  5.1.1 Mass of the balls ........................................................ 38
  5.1.2 Lipschitz constant ....................................................... 39
  5.1.3 Scanning path ............................................................. 40
5.2 Algorithm for Patrolling mission ........................................... 42
5.3 Results and discussion of Patrolling mission ......................... 42
  5.3.1 Analysis of the mass update parameters ......................... 42
  5.3.2 Analysis of parameter $d_j$ .......................................... 47
  5.3.3 Monte Carlo experiments .............................................. 48

6 Target Tracking and Escorting .................................................. 51
6.1 Intruder trajectories .......................................................... 51
  6.1.1 Linear trajectory ........................................................ 51
  6.1.2 Outward spiral ........................................................... 51
  6.1.3 Bounded random trajectory ........................................ 52
6.2 Approach ............................................................................. 53
  6.2.1 Mass of the balls ........................................................ 53
  6.2.2 Lipschitz constant ....................................................... 54
6.3 Algorithm for Target Tracking and Escorting mission ............. 55
6.4 Results and discussion for Target Tracking and Escorting ........ 56
  6.4.1 Analysis of linear trajectory ........................................ 56
  6.4.2 Analysis of outward spiral trajectory ............................. 57
  6.4.3 Analysis of bounded random trajectory ......................... 58
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.4</td>
<td>Analysis for target with larger velocity than the sensors</td>
<td>60</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Analysis with two targets</td>
<td>61</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Monte Carlo experiments</td>
<td>62</td>
</tr>
</tbody>
</table>

7  Discussion and Perspectives  

7.1  Achievements  
7.2  Future Work  

Bibliography
List of Tables

3.1 Number of anchor points without tolerance ........................................... 23
3.2 Number of anchor points for small tolerance in a squared area .................. 24
3.3 Number of anchor points for small tolerance in a rectangular area .............. 25
3.4 Number of anchor points for small tolerance in 3D .................................. 25
# List of Figures

1.1 Taxonomy for multi-robot target detection and tracking. ........................................ 2  
3.1 Example of symmetric quasi-optimal point placement ............................................. 16  
3.2 Example of point placement with tolerance ........................................................... 18  
3.3 Examples of point placement with different tolerances in a square area ................. 22  
3.4 Examples of point placement in a circular area ....................................................... 23  
3.5 Examples of point placement with different tolerances in 3 dimensions ................. 24  
4.1 Area Coverage mission that achieves $\varepsilon$-full area coverage ............................. 30  
4.2 Example without all Anchor Points covered but with $\varepsilon$-full area coverage .......... 31  
4.3 Simulation without $\varepsilon$-full area coverage ...................................................... 31  
4.4 Examples of Area Coverage in a large area ............................................................. 32  
4.5 Examples of Area Coverage with two types of sensors ......................................... 33  
4.6 Area Coverage in a large area with $R_{\tau}$ ............................................................. 34  
4.7 Monte Carlo experiments for the Area Coverage mission ......................................... 35  
5.1 Example of point placement with symmetry ............................................................ 39  
5.2 Path scanning illustration .......................................................................................... 41  
5.3 Number of iterations until all points are covered for different $m$ ............................. 43  
5.4 Histograms of ball scan for patrolling mission for different $m$ parameter .............. 44  
5.5 Number of iterations until all points are covered for different $n$ ............................. 44  
5.6 Histograms of ball scan for patrolling mission for different $n$ parameter .............. 45  
5.7 Number of iterations until all points are covered for different $\alpha$ .......................... 46  
5.8 Histograms of ball scan for patrolling mission and different $\alpha$ parameter .......... 47  
5.9 Number of iterations until all points are covered for different $d_j$ ........................... 48  
5.10 Histogram representing patrolling mission with different movement restrictions ........ 49  
5.11 Variation of mean number of iterations until all points are scanned in Patrolling ........ 50  
6.1 Intruder linear trajectory ........................................................................................... 52  
6.2 Intruder outward spiral trajectory ............................................................................. 52  
6.3 Intruder bounded random trajectory ....................................................................... 53  
6.4 Tracking of linear trajectory .................................................................................... 56
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>k</em>-means</td>
<td>Clustering problem that aims at assigning a set of points to <em>k</em> centers.</td>
</tr>
<tr>
<td>Anchor Point</td>
<td>Point inside a region of interest that is able to represent the coverage of a region on its surroundings.</td>
</tr>
<tr>
<td>Area Coverage</td>
<td>Mission with the objective of defining an initial deployment of a set of sensors/agents/robots so that they are well distributed inside the region of interest and maximize the probability of detecting a target.</td>
</tr>
<tr>
<td>FISTA</td>
<td>Fast Iterative Shrinkage-Thresholding Algorithm. Algorithm able to find the solution of convex optimization problems.</td>
</tr>
<tr>
<td>Patrolling</td>
<td>Mission with the objective of detecting a target inside the region of interest.</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of interest.</td>
</tr>
<tr>
<td>Target Tracking and Escorting</td>
<td>Mission with the objective of accompanying an intruder that is inside the region of interest while at the same time being able to detect new targets.</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Autonomous patrolling is becoming a subject of great practical and scientific importance. From oceanic and land surveillance to patient monitoring or even simple waiter robots, examples abound. However, technological limits on the capabilities of the sensors, either in velocity or sensing range, make the problem of finding an optimal trajectory using the least amount of network resources a very difficult problem.

Covering small areas is not interesting because the number of necessary sensors will never be prohibitive, thus a low cost solution could be easy to find. The necessity of covering large areas, on the other hand, implies a large number of sensor nodes, especially, if they are stationary, and so it is not a viable solution. Mobile sensors reduce the costs by being able to cover different areas at different time steps, therefore reducing the number of necessary sensors. The employment of a cooperative coverage strategy with these sensors is not trivial and usual leads to greedy methods. The best route that each particular sensor should take in order to achieve a good coverage is of great importance in order to achieve as small costs as possible.

In areas with high sensor density is relatively easy to find acceptable solutions when it comes to tackling the problem of efficiently covering an area and how to reduce the costs of such operations. It is much more difficult to find a solution for an area with small density of sensor nodes, where some of them keep tracking a target and simultaneously the network is able to cover the remaining area.

1.1 Problem statement and objectives

In this thesis we want to solve three different missions in a known region of interest (ROI).

A continuous ROI is very difficult to deal with and therefore impractical. The first objective of this thesis is to discretize the ROI. From the available possibilities we will define a set of points that we will call Anchor Points and represent by set \( \mathcal{A} \). Each element of the Anchor Point set has to give information about the detection probability in its surroundings.

Once \( \mathcal{A} \) is defined, we want to find an initial deployment for a set of sensors \( \mathcal{S} \), where the goal is to find the placement that minimizes the distances from every \( a_i \in \mathcal{A} \) to one of the \( s_j \in \mathcal{S} \) taking into consideration range limitations \( R_j \) of each individual \( s_j \in \mathcal{S} \). If possible, a placement that guarantees
the coverage of the ROI with a probability higher than a known value $\varepsilon$ has to be found. This objective will be defined as Area Coverage.

The next objective we will define as Patrolling. Using the set of mobile sensors $S$, the objective is to find a path that allows to visit every $a_i \in A$ taking into consideration not only range limitations but also the movement restrictions $d_j$ for each $s_j \in S$ at different time steps.

Finally, the last objective is Target Tracking and Escorting where we also want to keep Patrolling of the remaining ROI assured. In this mission the objective is to have one of the $s_j \in S$ escorting a target or intruder (i.e., keeping it at a observation distance), while the remaining network $S$ is able to visit every Anchor Point $a_i \in A$.

1.2 State of the art

A survey conducted by Robin and Lacroix [1] proposes a taxonomy that divides the different research areas associated with the problem of multi-robot target detection and tracking. The division is made according to the scheme of Figure 1.1.

Figure 1.1: Proposed taxonomy by Robin and Lacroix [1].

Figure 1.1 shows the wide variety of problems that can be tackled related to this subject. They are divided in two sub-groups: Target Tracking and Target Detection. The former assumes that a target is present and has been identified, whereas no such knowledge is assumed in the latter.

Target Detection is sub-divided in two categories which distinguish the type of sensors employed, either mobile or static. The static scenario requires a prohibitive number of necessary sensors in large
regions. However, the study of their solutions is interesting to find a good initial deployment. The objective behind this Area Coverage mission is therefore to find the initial positions of the sensors that allow covering the largest area of a known region of interest (ROI).

This area can be associated with Wireless Sensor Networks (WSN), which have received significant attention due to the potential of applications in different areas such as communications, computing intelligence, surveillance or targeting systems [2, 3]. They are composed by a group of sensor nodes that are capable of communicating and extracting data from the ROI in order to collectively accomplish a global goal. The sensors usually have a limited range for communication and for sensing, thus increasing the difficulty of WSN related problems. Most of these problems take into consideration static configurations [4–6]. However, mobile sensors are sometimes used [7].

A common problem in WSN is how to place the sensors in order to achieve a better coverage of the area. Coverage problems can be grouped as: Area Coverage [5–7], Point Coverage [4] and Boundary Coverage. Area Coverage usually demands that all the area be covered by at least one sensor node, and it can be extended to the case where the area has to be covered by $K$ sensor nodes, known as $K$-coverage [5]. Point Coverage arises from the interest in covering only specified points within a certain region of interest. It is straightforward to see that a point coverage problem can easily be transformed into an area coverage problem [4]. Boundary Coverage deals with the problem of placing the sensors such that the boundary is always covered.

In [5] the main objective is to find the initial deployment of a set of sensors such that all the area is covered by at least $K$ sensors i.e., $K$-coverage, while also aiming to minimize the required sensing range. Their formulation is non-convex; however, the authors proposed an algorithm which is proved to converge to a local minimum of the problem. For their approach, the area is divided into partitions according to the concept of $K$-order Voronoi diagram. Each partition has $K$ sensors responsible for it and the sensors positions will be the centroids over all the partitions they are responsible for covering. This approach is able to deploy sensors in 2D areas with obstacles and 3D surfaces. However, the solution for sensors that have limited range is to employ more sensors until the number of sensors allows to obtain $K$-coverage, which may be impractical in real situations.

In [8] the concept of sensors is not taken into consideration; however, the objective is to optimally place static circles in a ROI to cover a set of points representative of the ROI such that the distance between the center of the circle and the points is minimized. A min-max-min combinatorial problem formulation is used to solve the problem. A division in convex sub-problems is used to find the optimal position for the centers in any dimension. The sensors range is, in most approaches, associated with balls, so this formulation can also be associated with a Static Surveillance solution. Once again, it is static and no answer is given for the fact that sensors have limits in their range.

Yang et al. [4] solve a point coverage problem related with the cost minimization of a WSN where the objective is to find the set of sensors that achieve full coverage of an area with some probability at each time step. The implemented division of the ROI is very interesting because they are able to divide it in a finite set of points that, if covered, achieve coverage with the necessary probability. However, they only solve the problem in high sensor density areas where they are static, therefore it is limited.
Robin and Lacroix [1] also subdivide the mobile target positioning in three different categories: Hunting, Probabilistic Search and Capture.

Hunting aims at locating a target in an environment with lack of resources or information like the ROI map or obstacles [1]. Map absence is usually tackled using local perception of the area. These strategies are robust to situations like faulty sensors or loss of communication, but they are rarely efficient. Some approaches use local communication and task assignments while others are based in a self-organization model to collaboratively achieve coverage of the environment.

Probabilistic Search uses uncertainty and priority to define probabilities over the environment. A big advantage behind this approach is that the positions at each time step are not deterministic, which makes it difficult for an intruder to find any pattern and therefore discovering how the sensors will move. Two approaches are commonly used for the problem, either Markov Chains [9] or Stackelberg Games [10].

Markov Chains usually make a representation of the world as a graph where each vertex represents a partition of the area. In [9] an adversarial simulation between an intruder and a patrolling path are defined using a Markov Chain. The objective is to define the probabilities such that the probability of success of four models for the intruders designed by the authors is minimized. According to those probabilities at each time step the robots navigate autonomously in the ROI. The algorithm outperforms other similar algorithms but is still not a cooperative strategy and the goal is only to find the weights of the Markov Chain at the beginning, which will possibly lead to non-visited vertexes and never takes into consideration the time dependency or points that were already previously visited.

As mentioned above, another developed strategy is the employment of Stackelberg games, where an adversarial environment is considered. An example is given in [10] where patrollers, targets and attackers are considered in a maritime environment. The objective is to define the position of the patroller to ensure safety of a target using a Stackelberg Game between the patroller and the attacker. The method used is greedy and the option selected is always the one with the highest payoff. The payoffs are calculated using a weighted sum of a group of functions that represent target utility, reach ability, criticality of route and attractiveness. This greedy method does not take into consideration a cooperative situation and does not aim at a global minimum.

While Probabilistic Search does not give guarantees of covering all the area at an acceptable rate, Patrolling is defined as keeping coverage of a zone by defining a cyclic route [1] which guarantees that every point will be covered, usually the same number of times. There are two main approaches using graph representation of the area: defining an Hamiltonian cycle or Partition of the area. The Hamiltonian cycle is similar to the widely studied Traveling Salesman Problem where the objective is to define the route that minimizes the cost and place the sensors along the defined route. The Partition based strategy divides the area in the same number of partitions as the number of sensors where each sensor is responsible for one of the partitions. Patrolling guarantees that every zone will be covered, it is completely deterministic and easy for an attacker to find a perfect timing for the attack.

For Robin and Lacroix [1], Capture mission aims at clearing or securing a known area, capturing all the targets whose positions are not known. In this technique it is assured that the targets will be found no matter what. For this mission the approaches are mainly centralized, graph or geometric-based, and
the targets are usually stationary.

Another category that is important to address is the employment of both mobile and static sensors as in [7]. The idea behind this method is to complement the data taken from static sensors with the data of mobile sensors. In [7] the mobile sensors have restrictions in their movement and are forced to always be moving. The next position is found taken in consideration a discrete set of possibilities and a set of functions who try to represent different characteristics of the environment such as collaboration between close mobile sensors, avoid boundaries or take into consideration the possible gain of the next positions. A weighted sum of the functions values is made according to the mission objectives. This solution is a mix between static surveillance and capture and is also applicable for the case without static sensor nodes, i.e., pure capture mission. The solution is however greedy since each of the mobile sensors takes only into consideration what is best for it and a global optimum is not the objective, leading to a non-optimal use of resources.

When the target has already been located according to [1] we enter the Target Tracking category. Depending on the number of viewpoints necessary, there are two categories: Target Localization or Monitoring for multiple or single viewpoints, respectively.

Target Localization uses a team of mobile sensors to track targets with multiple points of view. This allows to get more information about the target and mainly aims at improving estimations on the target’s position. This field can be associated with positioning systems such as Global Positioning System (GPS), but there are applications in which such systems do not work. Finding the relative positions of interest and a multi-robot distributed cooperative solution are the main challenges behind this problem.

Monitoring is divided according to the number of targets an agent is responsible for. For single targets we are in the Following category and for multiple targets Observation. In [11] the problem of covering an area and simultaneously track a target is addressed using Voronoi partitions of the area. The authors use density functions to represent the points of the Voronoi partition in order to define the best positions of the sensors if they have targets to cover. While the targets position change so will the density function which will make the robot move in a trajectory similar to the target while not forgetting the remaining area of his Voronoi partition. The robots that do not have a target to track will stay in the centroid of their respective Voronoi partitions. The employment of the density function is therefore limited and could be used to define trajectories for the remaining robots. This approach was verified with real robot.

The existent approaches are mainly based in two types: planning and optimization. Planning simulates actions and their effects on the remaining environment in order to define a sequence of actions that reaches a goal. There is a variety of approaches able to solve planning problems, the main challenge being how to formulate them as a planning problem with respect to the area, actions and results for such actions. In optimization the main challenge is the formulation of the objective function, constraints and the representation of the world.

The Area Coverage mission can be associated with the Static Surveillance in the taxonomy presented by [1] since the ideas behind it can also be applicable to define an initial deployment.

Robin and Lacroix [1] defines Patrolling as finding a cyclic trajectory for the network. However, in our case, the Patrolling mission is associated with the Capture and Hunting areas because we want to
guarantee that every Anchor Point will be visited and therefore every target will be located in a well known region. However, there is lack of sensors and our algorithm should adapt to situations like faulty sensors.

Target Tracking and Escorting is associated with Following and Observation of [1]. Despite our objective is to find a solution that allows a sensor to follow an intruder, as a result of the developed approach, we are also able to follow more than one intruder.

1.3 Contributions

This thesis contributions can be summarized as follows:

- We formalize a novel extension for the $k$-means problem, that we term $k$-disks, which takes into consideration the distance to balls instead of the distance to points and that is used as background for solving three different missions, where the agents are not explicitly commanded to enter in different states: Area Coverage, Patrolling, and Target Tracking and Escorting.
- Using a probabilistic model for the sensors we define a non-symmetric Anchor Point placement strategy where every point gives information about the coverage probability of its surroundings.
- Using the Anchor Point definition and the novel $k$-disks approach we propose a solution that works in any dimension and takes into consideration both sensing range and velocity limitations.
- We formulate an approach that is able to efficiently perform tracking of a target while the remaining network performs Patrolling of the remaining ROI, without explicitly instructing which agent does what. Each agent’s role will be a consequence of an overall cost function.
- "$k$-disk: placement, patrolling and tracking for a team of mobile agents" in preparation, to be submitted to the IEEE Transactions on Control Systems and Technology

1.4 Thesis Outline

This thesis, in Chapter 2 gives a background of some important concepts such as convex optimization problems and clustering formulations from which we highlight the $k$-means problem. After the introduction of the basic concepts we formulate our working environment where we define the sensors model for detection, the ROI discretization and introduce our $k$-disks extension of the $k$-means problem, in Chapter 3.

Once the overall framework is defined we give the necessary formulation to solve our three missions: Area Coverage, Patrolling, and Target Tracking and Escorting in Chapters 4, 5 and 6, respectively. The three chapters start by giving a small formulation of the problems, where the small differences with respect to the common background of the $k$-disks formulation are presented. In all chapters the last section includes the presentation and discussion of the obtained results.

In Chapter 7 the achievements of the performed work are summarized, as well as some open possibilities for future work.
Chapter 2

Background

In this chapter a few necessary concepts for the remaining of the thesis are formulated. The chapter
starts by addressing optimization problems in Section 2.1, in which the general concepts of the problems,
a method to solve convex optimization problems (Fast Iterative Shrinkage-Thresholding Algorithm -
FISTA) and some other FISTA-related necessary concepts are given. In Section 2.2 the basic concepts
of k-means clustering are given, including the problem formulation, Lloyd’s solution and a proposed
improvement to the problem known as k-means ++.

2.1 Optimization and convex optimization problems

From [12] an optimization problem is a problem of the form

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \quad i = 1, ..., m,
\end{align*}
\]  

(2.1)

where the vector \( x = (x_1, ..., x_n) \) is the optimization variable, the function \( f_0 : \mathbb{R}^n \to \mathbb{R} \) is the objective
function sometimes also referred as potential function, the functions \( f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, ..., m \) are the
constraint functions and \( b_1, ..., b_m \) are the limits or bounds of the constraints. A vector \( x^* \) is called
optimal if it has the smallest value of the objective function in all domain of function \( f_0(x) \) that respect
the constraints.

Convex optimization problems are an important subclass which has the characteristic that both the
objective function and the constraint functions are convex, \( i.e. \)

\[
f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y),
\]  

(2.2)

for all \( x, y \in \mathbb{R}^n \) and all \( \alpha, \beta \in \mathbb{R}, \alpha \geq 0, \beta \geq 0 \) such that \( \alpha + \beta = 1 \).

The big interest behind convex optimization problems is that any found local minimum is always a
global minimum and therefore the optimal solution \( x^* \). Solving this type of problems as of today is easy,
however, the reformulation of a problem that at first sight is not convex into a convex problem is often
not obvious. Many algorithms have already been proposed to solve convex problems, one of which is the Fast Iterative Shrinkage-Thresholding Algorithm or FISTA that was used in this thesis. The choice of this algorithm was made taking into consideration its simplicity and fast convergence that outperforms methods like ISTA and MTWIST as shown in [13].

2.1.1 FISTA - Fast Iterative Shrinkage-Thresholding Algorithm

FISTA [13–16] is an algorithm that is used to solve convex minimization problems of the form

$$\text{minimize } f(x) = g(x) + h(x),$$

(2.3)

where $g(x)$ is a convex differentiable function with $\text{dom}(g) = \mathbb{R}^n$ and $h(x)$ is the constrain which is closed and convex with inexpensive proximal operator $\text{prox}_{\theta h}$, where $\text{prox}_{\theta h}$ is defined in next Section. In this formulation is not obvious that $h(x)$ stands for a constraint, however, this algorithm is interesting because it allows to find a solution for a optimization problem where $h(x)$ is a constraint like the indicator function of a closed convex set $C$ as given by

$$h(x) = \begin{cases} 0, & \text{if } x \in C, \\ +\infty & \text{otherwise.} \end{cases}$$

(2.4)

The method starts at any point $x^{(0)} = x^{(-1)}$ and for $k \geq 1$:

$$y = x^{(k)} + \frac{k}{k-1} \left( x^{(k-1)} - x^{(k-2)} \right)$$

$$x^{(k)} = \text{prox}_{\theta h}(y - t_k \nabla g(y))$$

(2.5)

where $t_k$ is the step, $k$ is the iteration number and $x^{(k)}$ is the solution found in the $k$ iteration.

An interesting property of the method is that it is not a descent method. However, it is ensured that the cost decreases as fast as $O(1/k^2)$ if the step is defined as $t_k = 1/L$ where $L$ is the Lipschitz constant (that will be defined in 2.1.3) of the gradient of function $g(x)$.

To use FISTA the gradient and the proximal operator have to be known. In the case where the Lipschitz constant is difficult to compute it is possible to use line search [13–15] to define the value of the step $t_k$.

2.1.2 Proximal operator

The definition of the proximal operator for a given constraint $h(u)$ of the minimization problem (2.3) is given by

$$\text{prox}_h(x) = \text{argmin} \left( h(u) + \frac{1}{2} \|u - x\|^2 \right).$$

(2.6)

For the solution of (2.6) to be unique it is necessary that

$$h(u) + \frac{1}{2} \|u - x\|^2$$

(2.7)
be convex.

It is known and easily shown that the sum of convex functions is also convex [12]. Dividing (2.7) in two functions, \( h(u) \) and \( \frac{1}{2} \| u - x \|^2 \) if both are convex than it is known that the prox operator has a unique solution. Since \( \frac{1}{2} \| u - x \|^2 \) is a squared norm multiplied by a constant factor it is straightforward to see that it is convex [12]. The only requirement is therefore to have \( h(u) \) convex.

In FISTA presented in Section 2.1.1 it is required that the constraint functions \( h \) be convex, therefore the solution of the prox operator is always unique. In FISTA the prox operator receives a value of \( y \) that might not be feasible and returns a feasible point \( x^{(k)} \).

It is easily seen that in the case where there are no defined constrains, i.e. \( h(u) = 0 \) the result of (2.6) is \( x = u \) and therefore by looking at (2.5) the solution is found using the gradient method. The problem is therefore simplified to an unconstrained one as can be seen by setting \( h(u) = 0 \) in (2.3).

Another interesting result is that if \( h(u) \) is the indicator function given by (2.4) of a convex set \( C \) then the prox operator is given by the projection on \( C \):

\[
\text{prox}_h(x) = \arg\min_{u \in C} \| u - x \|^2 = P_C(x),
\]

where \( P_C(x) \) is the projection of \( x \) into the convex set \( C \) [14].

### 2.1.3 Lipschitz Constant

An important parameter to ensure a fast convergence of FISTA is the definition of the step \( t_k \). One of the assumptions made in [13, 15] is that \( g : \mathbb{R}^n \to \mathbb{R} \) is a smooth convex function of the type \( C^{1,1} \), i.e. continuously differentiable with a known Lipschitz continuous gradient constant \( L \). A function \( f \) is said to be Lipschitz continuous with parameter \( L > 0 \) if:

\[
\| f(x) - f(y) \|_2 \leq L \| x - y \|_2, \quad \forall x, y \in \text{dom}(f).
\]

Knowing this value allows to find an upper bound on \( g \) which is an important step to prove that the algorithm decreases the cost as fast as \( \mathcal{O}(1/k^2) \) for a step \( t_k = 1/L \). The computation of \( L \) is therefore an important step to reliably use this algorithm.

It is important to notice that the value of the Lipschitz constant \( L \) is not unique. In fact if \( l \) is proved to be the Lipshitz constant of \( f \), then any \( L \geq l \) is also a possible Lipshitz constant. A general rule is that as the smaller the constant the larger the step size and therefore the faster the convergence.

### 2.2 \( k \)-means clustering

For Jain [17] the goal of clustering is to find a form of naturally grouping a set of patterns, objects or points of any type such that they are more similar between the members of the same cluster than any other. This techniques finds multiple applications, and according to Jain [17] is used mainly for three different purposes:
1. Underlying structure: to gain insight into data, generate hypotheses, detect anomalies and identify salient features.

2. Natural classification: to identify the degree of similarity among forms or organisms.

3. Compression: as a method for organizing the data and summarizing it through cluster prototypes.

One of the most popular formulations for clustering is $k$-means which has the objective of grouping a set of points of any dimension into $k$ clusters such that the distances between any point in a cluster and the centroid of the cluster is minimized. Considering a set of points $X$ that will be clustered with $x_i \in X$ and $\mathcal{S}$ the set of centroids such that $s_j \in \mathcal{S}$, the problem is formulated as

$$\min_{\mathcal{S}} \sum_{i=1}^{|X|} \min \left( \|x_i - s_1\|_2^2, \|x_i - s_2\|_2^2, \ldots, \|x_i - s_k\|_2^2 \right), \quad (2.10)$$

where $|X|$ stands for the cardinality of set $X$.

The standard $k$-means algorithm is usually non-convex for $k > 1$. To prove this statement the first step is to analyze the convexity of

$$f(x) = \|x_i - s_j\|. \quad (2.11)$$

Every norm is known to be non-decreasing and convex due to the triangular inequality. Since the only variables are the values $s_j$ by composition rule found in [12] it is known that (2.11) is convex in its domain $\mathbb{R}^+$. The same analysis is made to prove convexity of

$$g(x) = f(x)^2 = \|x_i - s_j\|^2. \quad (2.12)$$

Considering an arbitrary function $h(x) = x^2$ that is known to be convex and non-decreasing in $\mathbb{R}^+$ and knowing that $f(x)$ was already proved as convex and have domain $\mathbb{R}^+$ by the composition rule it is known that (2.12) is convex.

Problem (2.10) for $k > 1$ is a sum of the minimum between convex functions which is usually non-convex therefore the problem is also non-convex. In the trivial case of $k = 1$ the problem is convex and simplified to

$$\min_k \sum_{i=1}^{|X|} \|x_i - s\|_2^2, \quad (2.13)$$

which is the known problem of finding the centroid given by $s = \frac{1}{|X|} \sum_{s \in X} x$.

For $k > 1$, problem (2.10) is combinatorial and therefore NP-hard, as proven in the literature [18]. Even though usually non-convex and NP-hard there are a large variety of algorithms that give good approximations to solve the problem from which the most widely used is Lloyd’s algorithm described in the next Section.
2.2.1 Lloyd’s algorithm

In 1982, Stuart Lloyd advanced with an approach that is today known as Lloyd’s Algorithm [19]. By that time, the proposed algorithm was used to find at what voltage a noisy value corresponded, never imagining the world of opportunities created with this formulation.

The principle behind Lloyd’s algorithm is to first consider the initial deployment of the $k$ centroids, assign each of the points of $X$ to the closest center $s \in S$ to create clusters $c_j$ of points such that $\bigcup_j c_j = X$ and where each $c_j$ corresponds to one of the centroids $s_j \in S$. After the assignment the problem becomes simpler and (2.10) is transformed to $|S|$ problems of the form

$$\minimize_{s_j} \sum_{i=1}^{\mid c_j \mid} \|c_{ji} - s_j\|^2.$$ (2.14)

Just like the trivial problem for $k = 1$ (2.13), this problem has a trivial solution given by the centroid, $s_j = \frac{1}{\mid c_j \mid} \sum_{i=1}^{\mid c_j \mid} c_{ji}$. This process is repeated always considering the positions $S$ of the previous iteration until a fixed point is found, i.e., the set of positions of the centers $S$ does not change. This solution is seen as a two step partition of the problem that can also be posed as an alternating minimization and that greatly reduces the problem complexity.

A more synthetic way of writing the algorithm is:

1. Arbitrarily choose an initial position for the $k$ centers $S = \{s_1, s_2, ..., s_k\}$.
2. For each $j \in \{1, 2, ..., k\}$ set the cluster $c_j$ of points from $X$ such that any point of $c_j$ is closer to $s_j$, than $s_k$ where $j \neq k$.
3. For each $j \in \{1, 2, ..., k\}$ set $s_j$ to be the centroid of all points of $c_j$, i.e. $s_j = \frac{1}{\mid c_j \mid} \sum_{c \in c_j} c$.
4. Repeat steps 2 and 3 until $X$ does not change.

Today this is regarded as the standard algorithm to approximate the solution of $k$-means clustering, and is widely used.

2.2.2 $k$-means++

Although Lloyd’s algorithm gives an approximated solution to the problem it is not optimal. Initialization is a very important step and in Lloyd’s algorithm presented in Section 2.2.1 an arbitrary initial position is taken into consideration. An algorithm, proposed in [20], denominated $k$-means++ makes a careful initialization for Lloyd’s algorithm. The only requirement to apply it is to know the points locations and the number of $k$ centers in order to return a good initial deployment for the centers.

The algorithm is given by the following steps:

1. Take a center $s_1$, chosen uniformly at random from $X$.
2. Take a new center $s_j$, choosing $x_i \in X$ with probability $\frac{D(x_i)^2}{\sum_{x_i \in X} D(x_i)^2}$.
3. Repeat step 2 until the $k$ centers positions are defined.
4. Do the standard Lloyd’s $k$-means algorithm from steps 2-4 of Section 2.2.1.

In this algorithm $D(x_i)$ for an arbitrary center $s_j$ is the smallest distance between point $x_i$ and the already defined centers, i.e. $D(x_i) = \min(\|x_i - s_1\|, \|x_i - s_2\|, \ldots, \|x_i - s_{j-1}\|)$.

Arthur and Vassilvitskii [20] proved that the algorithm has an expected value for the objective function $\phi$ of $E[\phi] \leq 8 (\ln k + 2) \phi_{\text{optimal}}$ and their results usually finished almost twice as fast and obtained values of the objective function about 20% better than the standard Lloyd’s random initialization approach.

2.3 Voronoi diagram

Considering a set of discrete points $P$ a Voronoi diagram is a division of a known space taking into consideration the distances between those points. Each partition is called Voronoi polygon or Voronoi cell $V_i$ and every point of this polygon is closer to his generator point than any other point of the set $P$. The definition of the Voronoi polygon of point $p_i \in P$ is given by

$$V_i = \{x \in A : \|x - p_i\| \leq \|x - p_j\|, \forall p_j \in P \setminus \{p_i\}\}.$$  

This definition is extended to the high order Voronoi diagram [21] (also known as the $k$-order Voronoi diagram) where each partition has a generator set composed by $k$ points of $S$. A Voronoi polygon for a generator set $G_j$ with $|G_j| = k$ is defined as

$$V_j^k = \{x \in A : \|x - p_i\| \leq \|x - p_l\|, \forall p_i \in G_j, \forall p_j \in P \setminus G_j\}. $$

Even though the division is usually for the $\ell_2$ norm it can be used for any norm.

This representation of the space is usually used to represent the results of Lloyd’s $k$-means algorithm because the assignment of points is made according to the points that are closer to each center which can be seen as the points that are inside the Voronoi polygon of each of the centers. In the end of the algorithm every point will be inside the Voronoi cell of the center it was distributed to.
Chapter 3

Area discretization and problem transformation

In this chapter the formulation for the problem is given starting in Section 3.1 by the necessary subjects for the area discretization in points such as the model for the sensors. In Section 3.2 the \( k \)-means problem extension, \( k \)-disks, necessary to solve our missions is given.

3.1 Point Placement

In this thesis we want to solve our three missions (Area Coverage, Patrolling and Target Escorting and Tracking) using a modified \( k \)-means approach. The first step to solve our problems is to carefully define the points in which the area is going to be discretized. Using a sensor probabilistic model it is possible to define the points in a way that preserves information about the overall coverage of the area.

3.1.1 Sensor probabilistic model

There are usually two types of sensor models: a binary 0/1 model, where a point is considered to be covered if it lies inside a ball \[8\] and probabilistic models, that usually define a decreasing probability with the distance to the sensor. A binary model is simple but often not realistic, so we will use a probabilistic model that has been widely developed in [4, 6, 22, 23].

For the formulation of the model a set of sensors \( S \) is deployed in a region of interest ROI where \( s_j \in S \) is the \( j^{th} \) sensor center. A ball around sensor \( s_j \) is defined by

\[
B_j(x) = \{x \in \text{ROI} : \|x - s_j\| \leq R_j\},
\]

where \( R_j \) is the distance for which the coverage for sensor \( s_j \) is guaranteed.

The model we will use takes into consideration that there is a ball surrounding a sensor where all points are covered and outside that ball there is an exponential decrease which varies with the distance from each point to the ball. The probability of detection of an event in a point \( p_i \) depends on the distance
between \( p_i \) and \( B_j \). The model is explicitly defined by

\[
p(d_{ij}) = e^{-k_j d_{ij}},
\]  

(3.2)

where \( k_j \) is a positive constant that varies with the type of sensors considered.

The advantage of this model is that it has probabilities decreasing with distance, making it more realistic, and since it is stochastic takes into consideration unknown environmental issues such as obstacles. The value of parameter \( k_j \) gives the model the freedom to adapt to any environment and any type of sensors employed in a real situation.

3.1.2 \( \varepsilon \)-full area coverage

From [4] the definition of \( \varepsilon \)-full area coverage is adopted.

**Definition 1 (\( \varepsilon \)-full area coverage):** Sensor set \( S \) provides \( \varepsilon \)-full area coverage to the ROI if coverage of any point in ROI is no less than \( \varepsilon \), i.e.,

\[
\forall p_i \in \text{ROI}, \quad 1 - \prod_{j \in S} (1 - p(d_{ij})) \geq \varepsilon. \tag{3.3}
\]

One way of achieving \( \varepsilon \)-full area coverage is by ensuring that all the ROI is at a distance smaller than \( R_{j\varepsilon} \) of one of the sensors, where \( R_{j\varepsilon} \) is given by

\[
R_{j\varepsilon} = -\frac{\log(\varepsilon)}{k_j} + R_j. \tag{3.4}
\]

The value of \( \varepsilon \) should be chosen according to the specific application and always takes values between \( 0 \leq \varepsilon \leq 1 \).

3.1.3 Anchor Point \( \mathcal{A} \) location

The problem tackled in [4] is to determine which sensors should be turned on or off at each time step in order to achieve that \( \varepsilon \)-full area coverage. Their approach however does not take into consideration movable sensors and therefore is not of interest for cases where the area is not sensor filled.

In our case the goal is to obtain the positions for a known number of sensors such that \( \varepsilon \)-full area coverage is obtained or the distances between every \( B_j \) and the set of Anchor Points \( \mathcal{A} \) is minimized. As one can imagine such a problem is quite complex and to tackle it the ROI should be divided in a grid or a set of points. Some references found in the literature [6, 7, 22, 23] have used point discretization; however, the division is made without a defined criterion. In [4] a different approach is used. Using model (3.2) it is possible to define a set of Anchor Points for which if an Anchor Point is covered with some probability \( \tau > \varepsilon \) then a ball with radius \( b \) can be imagined around that point such that every point inside the ball has covering probability greater than \( \varepsilon \). The value of \( b \) is obtained by

\[
\tau e^{-k_j b} = \varepsilon \iff b = -\frac{1}{k_j} \log \left( \frac{\varepsilon}{\tau} \right). \tag{3.5}
\]
After obtaining the distance $b$, the set of anchor points $A$ can be defined in such way that all points of $A$ have neighbour points at distance smaller than $2b$ and the union of all balls around each point $a_k \in A$ contains the ROI. This can be written as minimization problem

$$\begin{align*}
\text{minimize} & \quad |A| \\
\text{subject to} & \quad \|a_k - a_l\| \leq 2b, \quad \forall a_k \in A, \quad a_j \in \mathcal{N}(a_k) \\
\text{ROI} & \subset \bigcup_{a_k \in A} B_{a_k}(x)
\end{align*}$$

(3.6)

where $|A|$ is the number of points of $A$ and $\mathcal{N}(a_k)$ is the set of first neighbours of $a_k$. Solving this set cardinality problem is very difficult.

It is a necessary condition for this point distribution that $\tau > \varepsilon$, otherwise $b = 0$ and no Anchor Points will be found according to (3.5). It is also of interest to note that, as the value of $\frac{\tau}{\varepsilon}$ decreases, the value of $b$ increases, thus simplifying the problem because the number of necessary anchor points $|A|$ decreases.

As stated in [4] this approach transforms an area coverage problem into a point coverage problem where by covering a certain point with probability $\tau$ we can assure that a ball of radius $b$ around that point is covered with probability $\varepsilon$.

Considering a rectangular ROI the most obvious way to solve problem (3.6) is by defining points in a grid such that each vertex of the grid is an Anchor Point. The grid will be composed of squares with diagonal $2b$ in such a way that the side of each square will be given by $l = \sqrt{2}b$. To do this, the algorithm starts by placing the first point at position $(0,0)$ and places all others considering that this point is a vertex of one of the squares. After all points are placed they are re-centered so that the distance from the points to the boundaries is equal. As an example, the result of the algorithm is shown in Figure 3.1 for a ROI of dimensions $25\text{m} \times 25\text{m}$ and sensors of equal parameters $\varepsilon = 0.8$, $\tau = 1$, $k_j = 0.05$ and $R_j = 5\text{m}$.

It is straightforward to see that the definition for the grid presented in Figure 3.1 is symmetric, which will be a problem for Patrolling missions. This placement is a solution of problem (3.6), there are infinitely many other solutions that are not symmetric.

A different, non-symmetric, placement of the Anchor Points is used. We know that the distances between vertices of the symmetric case are all equal, so instead of defining all distances equal to $l$ we can use random number generation to define different distances smaller than $l$ between consecutive Anchor Points so that the placement is non-symmetric but still feasible according to (3.6). The solution is applied to a ROI of dimension height $\times$ width and starts by defining a tolerance $\gamma$ in the interval $0 < \gamma \leq l$ and the parameters of maximum positions $x_{\text{max}}$ and $y_{\text{max}}$. The value of $x_{\text{max}}$ will be defined with respect to the point with smallest value of $x$ in the previous column $x_s$ and the point of smallest $x$ in the first column $x_1$ such that $x_{\text{max}} = \min(x_s + l, \text{width} - x_1)$. $y_{\text{max}}$ will be defined by the $y$ position of the previously defined point $y_p$ such that $y_{\text{max}} = y_p + l$. The point definition will be done column by column on the grid starting with $x_{\text{max}} = 0$ and $y_{\text{max}} = 0$. When a column is defined the value of $y_{\text{max}}$ will be reset to $y_{\text{max}} = 0$ and $x_{\text{max}}$ is kept constant for each column. Considering the definitions a new point position is always given by $(x_{\text{max}} - U(0,\gamma), y_{\text{max}} - U(0,\gamma))$ where $U(0,\gamma)$ represents a random number uniformly
Figure 3.1: Example of point placement solution to problem (3.6) that is symmetric where we can see that all area is covered by circles of radius $r = b$ centered in the Anchor Points. This point placement guarantees that if all Anchor Points are covered than all area is covered with some known probability $\varepsilon$.

This point definition is just an example among an infinite number of possibilities. This algorithm can easily be extended to arbitrary $N$-dimensions where the only difference will be that, besides $x_{\text{max}}$ and $y_{\text{max}}$, there would also be $N - 2$ parameters that would be updated with the same principle as $x_{\text{max}}$.

Using this algorithm in the case where the value of the tolerance is $\gamma = 0$ we return to the symmetric situation of Figure 3.1.

It is interesting to note that this algorithm can also be used for any other ROI. In that situation, one could consider the longest line inside the ROI as $a = \max\{||x - y||\}, \forall x, y \in \text{ROI}$, and use Algorithm 1 considering an area $a \times a$ to get a set of Anchor Points from which only those inside the ROI are considered. The only requirement is to carefully deal with the borders in order to get a feasible distribution.

### 3.1.4 Number of anchor points

Since the choice of the parameter of tolerance $\gamma$ is a compromise between symmetries and the number of points created it is interesting to formulate an upper bound on the number of points $N$.

**Proposition 3.1.1.** The number of points $N$ created in a ROI of dimension $D$ with volume $\prod_{i=1}^{D} \text{width}_i$ for a grid side $l$ and a tolerance $\gamma$ is upper bounded by

$$N \leq \frac{\text{volume}}{(l - \gamma)^D}. \quad (3.7)$$
Algorithm 1: Point placement for a rectangular ROI.

**input**: Area size $w \times h$, Tolerance $\gamma$, Square size $l$.

**output**: Set with points distribution $points$.

```
1 begin
2  $x_{max} \leftarrow 0$;
3  $y_{max} \leftarrow 0$;
4  first $\leftarrow 1$;
5  first $\leftarrow 0$;
6  while $x_{max} - first \cdot x \leq w$ do
7    column $\leftarrow \emptyset$;
8    aux $w \leftarrow x_{max} - U(0,\gamma)$ ;
9    aux $h \leftarrow y_{max} - U(0,\gamma)$ ;
10   first $y \leftarrow aux_h$;
11   while first $y - aux_h \leq h$ do
12      Concatenate $[aux_w,aux_h]$ to column;
13      y$_{max}$ $\leftarrow aux_h + l$ ;
14      aux $w \leftarrow x_{max} - U(0,\gamma)$ ;
15      aux $h \leftarrow y_{max} - U(0,\gamma)$ ;
16  end
17  center column with $h$ ;
18  y$_{max}$ $\leftarrow l/2$;
19  if first $== 1$ then
20    first $x \leftarrow$ minimum $x$ of column;
21    first $\leftarrow 0$ ;
22  end
23  x$_{max}$ $\leftarrow$ min(first $x + l$, $w - first_y$) ;
24  add column to points;
25  center points with $w$
26 end
```

**Proof.** If we consider that the grid has an arbitrary side size $a$ we know that the number of sides that can fit into an arbitrary width of the ROI is given by $\lceil \frac{\text{width}}{a} \rceil$. Since we are interested in the number of vertices and we know that the number of vertices $v$ is one more than the number of sides it is easy to see that $v$ is given

$$v = \left\lfloor \frac{\text{width}}{a} \right\rfloor + 1.$$  \hfill (3.8)

The worst case scenario where the number of points is higher happens for the smallest value of $a$. Since in Algorithm 1 the side of the grid is given by $l - U(0,\gamma)$ we know that the smallest possible $a$ happens for $a$ given by

$$a = l - \gamma.$$ \hfill (3.9)

Combining (3.8) with (3.9) and generalizing (3.9) for $D$ dimensions we arrive at the final result (3.7).

\[\square\]

Proposition 3.1.1 gives an upper bound that can be tightened.

An approximation to the number of points $N$ created in a ROI of dimension $D$ with volume $\text{height} \times \prod_{i=1}^{D-1} \text{width}_i$
for a parameter of side of grid $l$ and a tolerance $\gamma$ is approximately given by

$$N \approx \left\lceil \frac{\text{height}}{l - \gamma/2} \right\rceil \prod_{i=1}^{D-1} \left\lceil \frac{\text{width}_i}{l - \gamma} \right\rceil. \quad (3.10)$$

To find the reasoning behind this approximation we use the same definitions that were used in the proof of 3.1.1 with the exception of the volume that for this case is $\text{height} \times \prod_{i=1}^{D-1} \text{width}_i$.

In Algorithm 1 the Anchor Points were defined column by column. After a column is created the maximum position for the next column is given taking into consideration the minimum position of the current column. This last process is made for $D - 1$ dimensions.

The difference between the definition of points for the first dimension and the remaining $D - 1$ makes it possible to differentiate the number of points between the height dimension and the remaining width$_i$ dimensions.

For the width$_i$ dimensions we cannot make any reliable prediction on the size of the grid side because we are defining it considering a minimum between random numbers, so just like in the proof of 3.1.1 we will consider that the number of vertices $v_i$ is given by

$$v_i = \left\lceil \frac{\text{width}_i}{l - \gamma} \right\rceil. \quad (3.11)$$

For the height dimension we can work with the expected value of the uniform distribution, which is given by

$$E[U(0, \gamma)] = \frac{\gamma}{2}. \quad (3.12)$$
The number of vertices for this dimension, $v_h$, is therefore given by

$$v_h = \left\lceil \frac{\text{height}}{l - \frac{1}{2}} \right\rceil.$$

(3.13)

Generalizing (3.11) for $D - 1$ dimensions and combining with (3.13) we find the final result (3.10).

### 3.2 $k$-disks: $k$-means problem extension

The standard $k$-means formulation (2.10) has the objective of placing a group of center points such that the distances between a set of points and centers is minimized.

In our case, the centers will be the agents of sensors $s_j \in S$ and the points will be the Anchor Points $a_i \in A$ as defined in Section 3.1. Since the sensors have a known ball for which an event is assured to be detected with probability $\tau$ it is of interest to take this into consideration to find the optimal position for the set of sensors $S$. Therefore, instead of considering a point problem as in the standard $k$-means, a ball is considered. The problem is transformed to

$$\min_{s_j \in S} \sum_{i=1}^{\left| A \right|} \frac{w_i}{2} \min \left( \left( \| a_i - s_1 \| - R_{1\tau} \right)_+, \left( \| a_i - s_2 \| - R_{2\tau} \right)_+, \ldots, \left( \| a_i - s_k \| - R_{k\tau} \right)_+ \right),$$

(3.14)

where $R_{j\tau}$ is the distance for which an event has detection probability $\tau$ by sensor $s_j$ and $w_i$ is a mass that is constant to each point during the resolution of the problem and whose values will depend on the mission considered. As a general rule a high mass $w_i$ stands for a point that is more important to be covered.

Standard $k$-means is usually non-convex as shown in Section 2.2 and the same conclusion is also true for this extension of the problem. Just like the standard $k$-means we are minimizing a sum of the minimum of convex functions which for $k > 1$ is usually non-convex. Problem (3.14) is also combinatorial so very difficult to solve.

Although not obvious another way of writing problem (3.14) is to consider a distance to a ball centered in $a_i$ with radius $R_{j\tau}$. Considering that a ball around Anchor Point $a_i$ is given by (3.1) and that a distance to ball $B_i(x)$ is given by $d_{B_i}(x)$ the problem is written as

$$\min_{s_j \in S} \sum_{i=1}^{\left| A \right|} \frac{w_i}{2} \left( \min \left( d_{B_i}(s_1)^2, d_{B_i}(s_2)^2, \ldots, d_{B_i}(s_k)^2 \right) \right),$$

(3.15)

This corresponds to the extension from points to balls that allows to take into consideration the range of the sensor, where, obviously, $d_{B_i} = (\| a_i - s_j \| - R_{j\tau})_+$.

#### 3.2.1 $k$-means++ and Lloyd’s algorithm to the problem extension

In Section 2.2.2 $k$-means++ is presented as an initialization of Lloyd’s algorithm which increases its efficiency. This algorithm only requires knowledge of the set of points and the number of $k$ centers. In our case the set of points is $A$ and the number of centers is the number of sensors $k = |S|$. The formulation
presented for this extension does not require any change for this algorithm and its implementation is straightforward.

The principles behind Lloyd’s algorithm to solve \(k\)-disks, are the same used to solve the standard problem. The first step to solve the standard algorithm as seen in Section 2.2.1 is to divide the set of Anchor Points \(A\) according to the distances to the sensors of set \(S\). However, since we want to take into consideration the different ranges of the sensors we will consider the distance to the balls centered in \(S\) with radius \(R_{j\tau}\). This step divides the set \(A\) in \(|S|\) clusters \(c_j \in C\) where sensors with higher \(R_{j\tau}\) will also have larger areas to cover.

Once the points are grouped in clusters we determine the sensors position. In the standard \(k\)-means the position of the sensors are the centroids of the clusters. However, in this case we are taking into consideration the extension, so the standard problem (2.14) is transformed to

\[
\min_{s_j} \sum_{i=1}^{|c_j|} \frac{w_i}{2} \left( d_{B_i}(s_j) \right)^2. 
\]  

Problem (3.16) is convex because it is a sum of a convex function multiplied by a positive constant and, therefore, easy to solve.

To proof this statement we have to prove convexity of \(d_{B_i}^2(s_j)\). Since \(d_{B_i} = (\|a_i - s_j\| - R_{j\tau})_+\) we first need to analyze \(\|a_i - s_j\|\) that is just a different form of writing (2.11) and is therefore convex. The difference between this value and a constant \(R_{j\tau}\) is also convex however the range is \([-R_{j\tau}, +\infty]\) because it is just an offset of \(-R_{j\tau}\) relatively to the range of the norm \(\mathbb{R}^+\).

In [12] it is proved that the maximum between convex functions is also convex. Therefore since \((x)_+ = \max\{0, x\}\) returns the maximum between a constant and its argument the only necessary condition to prove convexity is that the argument is convex. In our specific case the argument is convex as already shown so the function is also convex. Another interesting characteristic for our problem is that the function takes an argument that has domain \([-R_{j\tau}, +\infty]\) and returns a value with range \(\mathbb{R}^+\).

Lastly, \(h(x) = x^2\) is proved to be convex and for \(x \in \mathbb{R}^+\) non-decreasing. \((\|a_i - s_j\| - R_{j\tau})_+\) is convex with range \(\mathbb{R}^+\) therefore \((\|a_i - s_j\| - R_{j\tau})^2_+\) is also convex using the composition rules of [12].

From the available algorithms to solve convex optimization problems we chose to use use FISTA (the method described in Section 2.1.1) due to its simplicity and performance comparative to other methods. To use it, we need to compute the gradient of the objective function \(f(s_j)\) of as given by

\[
f(s_j) = \sum_{i=1}^{|c_j|} \frac{w_i}{2} \left( d_{B_i}(s_j) \right)^2. 
\]  

From [24, Chapter X, Proposition 3.2.2 and Theorem 3.2.3] we know that

\[
\nabla \left( \frac{1}{2} d_S^2(x) \right) = x - P_S(x),
\]  

where \(S\) is a convex set and \(d_S(x)\) is the shortest distance between set \(S\) and point \(x\).
A ball is known to be a convex set so the gradient of $f(s_j)$ is given by

$$
\nabla f(s_j) = \nabla \left( \sum_{i=1}^{\left| s_j \right|} w_i \left( \frac{1}{2} (d_{B_i}(s_j))^2 \right) \right) 
= \sum_{i=1}^{\left| s_j \right|} w_i \nabla \left( \frac{1}{2} (d_{B_i}(s_j))^2 \right) 
= \sum_{i=1}^{\left| s_j \right|} w_i (s_j - P_{B_i}(s_j)) \tag{3.19}
$$

The gradient includes the projection onto a ball which, for a generic ball $B_c$ centered at $c$ with radius $R$, is given by

$$
P_{B_c}(x) = \begin{cases} 
  x, & \text{if } \|x - c\| \leq R \\
  c + R \frac{x - c}{\|x - c\|}, & \text{otherwise}
\end{cases} \tag{3.20}
$$

FISTA also requires the Lipschitz constant $L$ of the gradient of the objective function. In our case $L$ depends on the mission taken into consideration because each of the missions, even though using the same minimization problem background, have a different mass $w_i$ definition.

### 3.3 Results on the point distribution

For the results shown in this section the simulations will all be performed with parameters $\tau = 1$ and $\varepsilon = 0.8$ with identical sensors parameterized by $R_j = 5$ and $k_j = 0.05$.

#### 3.3.1 Analysis of the point distribution

One of the parameters of Algorithm 1 is the value of $\gamma$. It is theoretically expected that, as its value increases, the point distribution progressively departs from the symmetric configuration. In Figure 3.3 we show the result of the point placement in a region of area $50m \times 50m$ for four different values of $\gamma$.

As expected, as the value of $\gamma$ increases, so does the density of Anchor Points in the ROI. It is also possible to conclude that all the ROI is covered by at least one of the circles centered in the Anchor Points with radius $b$, showing that Algorithm 1 finds a feasible solution to problem (3.6).

To illustrate the performance of the point placement algorithm in areas with other shapes, we present Figure 3.4 where a circular area of radius $25m$ centered at $(25, 25)m$ and tolerance $\gamma = \frac{1}{4}l$ is considered. The point placement was obtained considering a square area of side $50m$ from which only the points inside the circle are considered and then the boundaries are carefully dealt.

Figure 3.4.a shows the results directly from Algorithm 1 after considering only the points inside the ROI. In this case, a few areas of the boundary are not covered by any ball surrounding an Anchor Point and therefore the placement is not a solution to (3.6). In Figure 3.4.b we add 5 points in the regions not covered by any balls so that it is a solution to (3.6).

Since the placement for this case uses the same algorithm as Figure 3.3 the same conclusions regarding the number and density of Anchor Points are also applicable.
Figure 3.3: The higher the value of the tolerance parameter $\gamma$ the higher the number of points and their density. All of them are solution of problem (3.6) since all area is covered by a circle of radius $b$ surrounding the anchor points. The area is $50\text{m} \times 50\text{m}$, the asterisks represent Anchor Points and the circles have radius $b$ and are centered in the Anchor Points.

### 3.3.2 Examples on three dimensions

As stated before the algorithm is easily extended to any dimension. A 3D example is given in Figure 3.5 where four different values of tolerance $\gamma$ are shown in a volume $50\text{m} \times 50\text{m} \times 50\text{m}$.

Just like in the 2D case, by looking at Figure 3.5, we can see that the larger the tolerance $\gamma$ adopted, the more spread the distribution is and as a general rule the more points are placed. Even though not represented, for all the different tolerances, the ROI is contained by balls centered in the Anchor Points with radius $R_j\tau$, therefore, the placements are all solution of (3.6). In Figure 3.5.b we can still see different symmetric planes, however, as the tolerance increases those symmetric planes disappear and in Figure 3.5.d the distribution appears to be chaotic.
Figure 3.4: The algorithm is also applicable to ROI with different shapes, as long as the boundaries are carefully dealt. Figure (a) shows the result of considering the Anchor Points of a raw distribution of Algorithm 1 that are inside the ROI, showing that some zones of the boundary are not covered by a ball. In Figure (b) points were added to the boundaries that are not covered in (a) so the placement is a solution of (3.6).

3.3.3 Analysis of the number of points

One of the critical points when choosing the parameter $\gamma$ is to find how the number of necessary points will increase. In Section 3.1.4 we stated Proposition 3.1.1 and approximation (3.10) that give an upper bound and a prediction on the value, respectively.

In this section, an experimental validation of these results is presented for different ROI of different shapes and different values of $\gamma$. Algorithm 1, that gives an Anchor Point placement, was performed 1000 times, from which the maximum and minimum number of Anchor Points obtained are presented. With chosen parameters for the sensors, we have a value of $l = 6.3 \text{m}$.

The first simulations are for tolerance $\gamma = 0$ and are summarized in Table 3.1 for squared and rectangular areas.

Table 3.1: In the case without tolerance ($\gamma = 0$) the two theoretical predictions yield the same result which is also obtained in the simulations.

<table>
<thead>
<tr>
<th>Height (m)</th>
<th>Width (m)</th>
<th># points</th>
<th>Upper bound</th>
<th>Predicted #</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>6400</td>
<td>6400</td>
<td>6400</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>25281</td>
<td>25281</td>
<td>25281</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>25</td>
<td>50</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>250</td>
<td>100</td>
<td>640</td>
<td>640</td>
<td>640</td>
</tr>
<tr>
<td>100</td>
<td>250</td>
<td>640</td>
<td>640</td>
<td>640</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
<td>1280</td>
<td>1280</td>
<td>1280</td>
</tr>
<tr>
<td>100</td>
<td>500</td>
<td>1280</td>
<td>1280</td>
<td>1280</td>
</tr>
</tbody>
</table>
Figure 3.5: The adaptation of the algorithm to 3D is straightforward. As in 2D higher values of $\gamma$ yield point distributions with more points and higher density.

For this case we can verify that our theoretical results are accurate and predict the number of anchor points defined. In fact by looking at the definitions of both Proposition 3.1.1 and 3.10 it is easy to see that the specific case of $\gamma = 0$ returns the same value for both. The value of $\gamma = 0$ means that there is no uncertainty in the point definition, therefore it is completely deterministic as can also be found in Table 3.1.

Table 3.2 shows the result for tolerance $\gamma = \frac{1}{4}l$ in squared areas.

Table 3.2: In squared areas the maximum number of created points never exceeds the upper bound and the predicted value is always closer to the obtained values than the upper bound. The presented results are obtained for tolerance $\gamma = \frac{1}{4}l$.

<table>
<thead>
<tr>
<th>Height (m)</th>
<th>Width (m)</th>
<th>Minimum #</th>
<th>Maximum #</th>
<th>Upper bound</th>
<th>Predicted #</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>36</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>91</td>
<td>110</td>
<td>121</td>
<td>110</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>385</td>
<td>397</td>
<td>441</td>
<td>399</td>
</tr>
<tr>
<td>250</td>
<td>250</td>
<td>2368</td>
<td>2441</td>
<td>2809</td>
<td>2438</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>9532</td>
<td>9660</td>
<td>11236</td>
<td>9646</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>38261</td>
<td>38373</td>
<td>44944</td>
<td>38584</td>
</tr>
</tbody>
</table>
It is seen that in all 1000 iterations, as expected, a higher number than the upper bound was never obtained. The predicted value is also accurate and most of the times is within the interval of the number of points obtained. In the last case, where an area $1000\times1000$ is considered, despite not being inside the interval, the predicted value has a relative error of only 0.8% to the minimum obtained.

If we consider rectangular areas as our ROI the obtained limits of the number of Anchor Points are given in Table 3.3 for a tolerance $\gamma = \frac{1}{4}l$.

Table 3.3: In rectangular areas the upper bound is also never exceeded and the relative errors of the predicted value never exceed 5% of the obtained minimum. As a general rule, if the larger dimension is the height the expected number of points is smaller. The results are obtained for $\gamma = \frac{1}{4}l$.

<table>
<thead>
<tr>
<th>Height (m)</th>
<th>Width (m)</th>
<th>Minimum #</th>
<th>Maximum #</th>
<th>Upper bound</th>
<th>Predicted #</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>25</td>
<td>45</td>
<td>50</td>
<td>66</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>50</td>
<td>50</td>
<td>56</td>
<td>66</td>
<td>55</td>
</tr>
<tr>
<td>250</td>
<td>100</td>
<td>953</td>
<td>971</td>
<td>1166</td>
<td>1012</td>
</tr>
<tr>
<td>100</td>
<td>250</td>
<td>956</td>
<td>990</td>
<td>1166</td>
<td>1007</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
<td>1900</td>
<td>1927</td>
<td>2332</td>
<td>2002</td>
</tr>
<tr>
<td>100</td>
<td>500</td>
<td>1921</td>
<td>1963</td>
<td>2332</td>
<td>2014</td>
</tr>
</tbody>
</table>

Once again, the upper bound is never exceeded by the maximum number of points in 1000 iterations. The predicted value is less accurate, however the relative errors for width and height of orders higher than two is 5% with respect to the minimum number of points obtained. It is also interesting to note that as a general rule if we consider the higher dimension as the height in Algorithm 1 it is expected to have smaller number of points than if it was the width, therefore reducing the complexity of the problem. We have to always take into consideration that this is not always true due to the geometry of the ROI.

In a 3 dimensional point placement the obtained results are expressed in Table 3.4.

Table 3.4: In 3D the upper bound is always respected and the prediction of the number of Anchor Points has a larger error. When $\text{width}_2$ is the larger dimension the number of points is larger in comparison with ROI with the same volume. The results are obtained for $\gamma = \frac{1}{4}l$.

<table>
<thead>
<tr>
<th>Height(m)</th>
<th>Width$_1$(m)</th>
<th>Width$_2$(m)</th>
<th>Minimum #</th>
<th>Maximum #</th>
<th>Upper bound</th>
<th>Predicted #</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td>1046</td>
<td>1167</td>
<td>1331</td>
<td>1210</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>8175</td>
<td>8256</td>
<td>10648</td>
<td>9196</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>25</td>
<td>232</td>
<td>253</td>
<td>396</td>
<td>360</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>25</td>
<td>249</td>
<td>265</td>
<td>396</td>
<td>330</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>50</td>
<td>250</td>
<td>290</td>
<td>396</td>
<td>330</td>
</tr>
<tr>
<td>500</td>
<td>100</td>
<td>100</td>
<td>40094</td>
<td>40201</td>
<td>51304</td>
<td>44044</td>
</tr>
<tr>
<td>100</td>
<td>500</td>
<td>100</td>
<td>40613</td>
<td>40763</td>
<td>51304</td>
<td>44308</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>500</td>
<td>41368</td>
<td>41535</td>
<td>51304</td>
<td>44308</td>
</tr>
</tbody>
</table>

It is possible to see from Table 3.4 that the upper bound is never exceeded. In 3D the baseline error when predicting the number of points increases due to the added dimension. It is an already expected result because the only dimension for which the prediction is expected to give a similar value to the algorithm is for the height dimension. It is also interesting to note that the cases where $\text{width}_2$ is higher, are also the cases where the number of points is higher. This is an expected result because a plane is defined relatively to the point with smaller value on the $\text{width}_2$ dimension of the previous plane. Since a plane has more points than a line it is expected that there is also a smaller value, therefore possibly increasing the number of necessary planes and in consequence the number of necessary points.
Chapter 4

Area Coverage

This chapter addresses the Area Coverage mission. The approach to the problem is given in Section 4.1 where we define the mass of the balls used in Point Placement and the Lipschitz constant for the optimal procedure. In Section 4.2 we present the full algorithm and the chapter ends in Section 4.3 with simulation results.

4.1 Approach

In Section 1.1 the global objective for this mission was presented without further explanation. In Chapter 3 we formulated the global problem we are addressing and the basic concepts that will allow us to solve the Area Coverage mission. We already know that the optimization problem we want to solve is (3.16) and the necessary steps to solve it. However, there are two remaining concepts to define: the mass value of the balls \( w_i \) and the Lipschitz constant.

4.1.1 Mass of the balls

The concept of mass of the balls is used to define which balls are more important to cover. The idea is to change its value according to the importance to cover each of the balls such those with more mass are more likely to be at a distance smaller than \( R_j \) from a sensor. The sensors lean to positions of higher mass to reduce the cost associated with that mass, while also taking into consideration the remaining balls.

For the Area Coverage mission we will consider that every point is equally important by setting the values of \( w_i = 1, \forall i \). The problem for this case is simplified to

\[
\text{minimize}_{s_j} \sum_{i=1}^{\left|\mathcal{S}_j\right|} \frac{1}{2} d_B^2(s_j).
\]

(4.1)

It is possible to consider from the outset the existence of balls that are more important to cover than others by considering higher values for the masses. However, this is not the main goal.
4.1.2 Lipschitz constant

To define the step for FISTA we need to find the Lipschitz constant for the gradient of the objective function of (4.1). In (4.1) the minimization variable is the position of the \( j \)th sensors, \( s_j \). Writing \( s_j \) or any arbitrary \( x \) is exactly the same, so the problem can also be written as

\[
\min_x \sum_{i=1}^{\left| c_j \right|} \frac{1}{2} d_{B_i}^2(x). \tag{4.2}
\]

The objective function of (4.2) has a known gradient given by (3.18): \( \nabla \left( \sum_{i=1}^{\left| c_j \right|} \frac{1}{2} d_{B_i}^2(x) \right) = \sum_{i=1}^{\left| c_j \right|} (x - P_{B_i}(x)) \).

In order to find a step value for which FISTA converges optimally we need to find a Lipschitz constant for the gradient of the objective function of problem (4.2). To find \( L \) we start by writing the problem as

\[
\left\| \sum_{i=1}^{\left| c_j \right|} (x - P_{B_i}(x)) - \sum_{i=1}^{\left| c_j \right|} (y - P_{B_i}(y)) \right\| = \left\| \sum_{i=1}^{\left| c_j \right|} (x - y) + \sum_{i=1}^{\left| c_j \right|} (P_{B_i}(y) - P_{B_i}(x)) \right\|, \tag{4.3}
\]

where \( x \) and \( y \) are any arbitrary points in the domain of the objective function. Using the triangle inequality we get

\[
\left\| \sum_{i=1}^{\left| c_j \right|} (x - y) + \sum_{i=1}^{\left| c_j \right|} (P_{B_i}(y) - P_{B_i}(x)) \right\| \leq \left\| \sum_{i=1}^{\left| c_j \right|} (x - y) \right\| + \left\| \sum_{i=1}^{\left| c_j \right|} (P_{B_i}(y) - P_{B_i}(x)) \right\|. \tag{4.4}
\]

For any set of \( N \) function \( f_i(x) \) we have \( \sum_{i=1}^{N} f_i(x) \leq \sum_{i=1}^{N} f_i(x) \). Using this principle we can write (4.4) as

\[
\left\| \sum_{i=1}^{\left| c_j \right|} (x - y) \right\| + \left\| \sum_{i=1}^{\left| c_j \right|} (P_{B_i}(y) - P_{B_i}(x)) \right\| \leq \sum_{i=1}^{\left| c_j \right|} \left\| x - y \right\| + \sum_{i=1}^{\left| c_j \right|} \left\| P_{B_i}(y) - P_{B_i}(x) \right\|. \tag{4.5}
\]

It is known that for every closed convex set \( S \) and for any points \( x, y \in \mathbb{R}^N \) we have \( \| P_S(x) - P_S(y) \| \leq \| x - y \| \) because there are only a few possibilities:

- Points \( x, y \in S \) and therefore \( P_S(x) = x \) and \( P_S(y) = y \) so that \( \| P_S(x) - P_S(y) \| = \| x - y \| \).
- Point \( x \in S \) and \( y \notin S \) so that \( \| P_S(x) - P_S(y) \| < \| x - y \| \).
- Points \( x, y \notin S \) which results in \( \| P_S(x) - P_S(y) \| \leq \| x - y \| \), as given in [24].

Thus, it is said that the projection is a contraction operator. Using this property and since a ball is a closed convex set, we get

\[
\sum_{i=1}^{\left| c_j \right|} \left\| x - y \right\| + \sum_{i=1}^{\left| c_j \right|} \left\| P_{B_i}(y) - P_{B_i}(x) \right\| \leq \sum_{i=1}^{\left| c_j \right|} \left\| x - y \right\| + \sum_{i=1}^{\left| c_j \right|} \left\| y - x \right\|, \tag{4.6}
\]

and with some algebraic manipulation we arrive at

\[
\sum_{i=1}^{\left| c_j \right|} \left\| x - y \right\| + \sum_{i=1}^{\left| c_j \right|} \left\| y - x \right\| = 2 \sum_{i=1}^{\left| c_j \right|} \left\| x - y \right\| = 2 |c_j| \| x - y \|, \tag{4.7}
\]
which yields the final value of $L = 2|c_j|$ as given by

$$
\|\nabla \sum_{i=1}^{|c_j|} \frac{1}{2} \left( d_{B_{j_i}}(x) \right)^2 - \nabla \sum_{i=1}^{|c_j|} \frac{1}{2} \left( d_{B_{j_i}}(y) \right)^2 \| \leq 2|c_j||x - y|.
$$

(4.8)

### 4.2 Algorithm for Area Coverage mission

We will initialize our algorithm with the modified version of $k$-means++ formulated in Section 3.2.1. $k$-means++ is a stochastic algorithm, so in order to find a placement as close to optimal as possible the algorithm is ran several times and the one with the smallest cost is chosen. The overall algorithm for Area Coverage is Algorithm 2.

**Algorithm 2: Area Coverage.**

```plaintext
input : Anchor Point set points, Number of sensors $n_{\text{sensors}}$, Number of attempts attempts.
output: Initial sensor placement sensors.
1 begin
2 possibility ← $k$-means++ with ($n_{\text{sensors}}$, points);
3 calculate cost for possibility;
4 min ← cost;
5 sensors ← possibility;
6 for i = 2 to attempts do
7 possibility ← $k$-means++ with ($n_{\text{sensors}}$, points);
8 calculate cost for possibility;
9 if cost < min then
10 min ← cost;
11 sensors ← possibility;
12 end
13 end
14 end
```

Since $k$-means is an optimization problem there is an unknown minimum that we want to approximate. The higher the number of times we run the algorithm the more likely it is to find that minimum. According to the problem formulation a smaller cost guarantees that the sum of distances to every point is smaller, thus guaranteeing better coverage with the same resources. We could also consider the distribution that has more covered area. However, in situations where large ROI with small sensor density and no overlap between the sensors coverage exists, that value would be equal in all of the possibilities. By using the objective function we avoid this problem and we have more useful information about the real coverage of each individual placement.

The fact that we distribute the Anchor Points according the range of the sensors $R_{j\tau}$ allows to have the sensors with larger range with higher number of Anchor Points, therefore maximizing the sensors resources. Most of the solutions found in the literature do not account for each sensor’s range; in our formulation we are able to consider it and also consider sensors with different ranges.
4.3 Results and discussion for Area Coverage mission

4.3.1 Area Coverage with high density of sensors

When an area has enough sensors to achieve $\varepsilon$-full area coverage as defined in Section 3.1.2 the modified $k$-means++ Algorithm 2 does not always find a $\varepsilon$-full area coverage solution. However, repeating the algorithm mitigates the issue. To illustrate the algorithm performance we used a set of 8 sensors with equal parameters $R_j = 5m$ and $k_j = 0.05$ to distribute in a ROI of $25m \times 25m$. The parameter $\varepsilon$ is defined as $\varepsilon = 0.8$, $\tau = 1$ and $\gamma = 1/4$ such that the point placement is the same as that represented in Figure 3.2. The values of $\tau$ and $\varepsilon$ will be the same for every simulation. A distribution that achieves $\varepsilon$-full area coverage is presented in Figure 4.1.

![Diagram](image)

Figure 4.1: The proposed approach is able to find a solution that achieves $\varepsilon$-full area coverage. Since all Anchor Points are at distance inferior to $R_j \tau$, as seen in (a), we can ensure that $\varepsilon$-full area coverage is achieved, as it is seen in (b).

From Figure 4.1.b we can see that 8 sensors are able to achieve $\varepsilon$-full area coverage since all the ROI is at a distance less than $R_j \varepsilon$ from at least one sensor. The same distribution is plotted in Figure 4.1.a where all Anchor Points are covered with probability of $\tau$ which ensures that all the ROI is covered with probability of $\varepsilon$ as stated in Section 3.1.3. We highlight the fact that the coverage of all Anchor Points with probability $\tau$ is a sufficient but not necessary condition to obtain $\varepsilon$-full area coverage. Figure 4.2 shows an example where this condition is not met but $\varepsilon$-full area coverage is attained anyway, under the same conditions of Figure 4.1.

As expected, the fact that not all points are covered is reflected in the objective function value. In Figure 4.1.a all Anchor Points were covered with probability $\tau$, so the objective function value is 0. In Figure 4.2.a not all Anchor Points are at distance inferior to $R_j \tau$ which results in an objective function value of 3.72. There are also situations where the algorithm performs worse than the shown results, with exactly the same characteristics. Figure 4.3 shows one such example in the same conditions where $\varepsilon$-full area coverage is not achieved.
Figure 4.2: The coverage of all Anchor Points with probability $\tau$ is not a necessary condition to have $\varepsilon$-full area coverage. In (a) we can see that not all Anchor Points are covered, however in (b) we can see that $\varepsilon$-full area coverage is achieved.

Figure 4.3: Algorithm 2 may sometimes find bad solutions where $\varepsilon$-full area coverage is not achieved. In this example a lot of Anchor Points are not covered, as seen in (a), and in (b) the top corners are not covered with probability $\varepsilon$ therefore $\varepsilon$-full area coverage is not achieved.

In Figure 4.3.a we can see that there are Anchor Points at distances above $R_{j\tau}$ so we cannot guarantee $\varepsilon$-full area coverage. Figure 4.3.b shows that for this situation in fact $\varepsilon$-full area coverage is not achieved. For this case the objective function value is 50.32.

The three examples were obtained under the same conditions and illustrate the importance of running Algorithm 2 several times. It is important to notice that, as expected, the smaller the objective function value the better distributed the Anchor Points are between the sensors, and the more likely it is to achieve $\varepsilon$-full area coverage. For this case the ROI has a high sensors density and if we used the covered area as a parameter to choose between the different results yield by $k$-means++ we might have chosen a better overall distribution, however using the objective function value we found a distribution for which we can ensure that $\varepsilon$-full area coverage is achieved.
4.3.2 Area Coverage with low density of sensors

When there are not enough sensors to achieve $\varepsilon$-full area coverage the smaller the objective function the better distribution we obtain. As an example, we will use a ROI of dimension $100m \times 100m$ with 10 sensors equal to the ones used in the previous simulations. For this case two instances of the algorithm are shown in Figure 4.4.

Figure 4.4: The repetition of the algorithm yields very different results for different runs of the $k$-means++. For these simulations the covered area is exactly the same, but in (b) there are larger uncovered areas than in (a). The asterisks represent the Anchor Points, the crosses the sensors, the solid circles $R_{j\tau}$ and the dashed circles $R_{j\varepsilon}$.

In Figure 4.4 besides the sensors, and ranges $R_{j\tau}$ and $R_{j\varepsilon}$ we also plotted the Voronoi diagram for the sensors, and the Anchor Points that each sensor is in charge of covering in different colors. Our first analysis is that every Anchor Point that is inside the same Voronoi cell also has the same color, a result that is expected since the algorithm distributes the Anchor Points to the sensors according to the distances to each sensor ball.

It is easy to note that most sensors of Figure 4.4.b have larger uncovered areas than the sensors of Figure 4.4.a, which means that it is a better distribution. This is also visible in the objective function values where, we obtained $2.92 \times 10^4$ and $3.59 \times 10^4$ for Figures 4.4.a and 4.4.b, respectively. These are exactly the cases for which using the covered area as a parameter to choose which distribution is better would yield the same exact values, even though the distributions are very different.

4.3.3 Area Coverage with different sensors

For the simulations with different types of sensors we will consider a ROI of $100m \times 100m$ and 10 sensors. Five of the sensors will be equal to those already used, i.e., $R_j = 5m$ and $k_j = 0.05$. The remaining sensors will have $R_j = 10m$ and $k_j = 0.02$. An instance of heterogeneous area coverage is shown in Figure 4.5.
Figure 4.5: In (a) we can see that the sensors with larger $R_{j\tau}$ are also the sensors with larger number of points to cover. In (b) we represent an instance where instead of considering the distance to the balls we consider the distance to each point. The Voronoi cells are always drawn taking in consideration the centers of each sensor so in (a) we can see that there are Anchor Points belonging to a sensor with higher $R_{j\tau}$ that are in a different Voronoi cell and in (b) all Anchor Points are in the respective Voronoi cell. The asterisks represent the Anchor Points where the different colors represent different sensors, the crosses represent the sensors center, the solid lines $R_{j\tau}$ and the dashed lines $R_{j\varepsilon}$.

The two different sensors are identifiable in Figure 4.5. In Figure 4.5.a we can see that the sensors with larger $R_{j\tau}$ are also those that have larger number of points to cover therefore reducing the objective function. In Figure 4.5.b instead of considering the distance to the balls we consider the distance to each point so that there is not a big difference in the size of the partitions of each type of sensors.

Figure 4.5.b is a worst option since the sensors resources are not used to their full potential. In fact, for Figure 4.5.a an objective function of $1.2 \times 10^4$ was obtained versus $1.8 \times 10^4$ for Figure 4.5.b, showing once again that the lower the value of the objective function the better overall coverage is achieved.

### 4.3.4 Area Coverage in the 3 dimensional case

To give a final example of the correct behaviour of the developed approach in any dimension we present results for the Area Coverage mission in a 3D ROI of volume $50\text{m} \times 50\text{m} \times 50\text{m}$ with 10 sensors equal to those used in the simulations of Sections 4.3.1 and 4.3.2 for a point placement of tolerance $\gamma = \frac{1}{4}$. In Figure 4.6 the solution is presented with the ranges $R_{j\tau}$.

Since the application of the algorithm is not different from the 2D case, the same conclusions regarding $\varepsilon$-full area coverage, the better initial placement between every run and the better placement when different sensors are considered are also true for any other dimension. For these cases the only difference is the point placement (already addressed in Section 3.3.2), which, if done carefully, yields good results for the mission.
4.3.5 Monte Carlo Experiment

Monte Carlo experiments are usually conducted in scenarios where there is an environment in which there is uncertainty regarding the outcome of a defined experiment. A set of free parameters with different values yields different results. Parsing those results provides information about the expected outcome of conducting an equal experiment under the similar environment conditions.

In this particular case we want to study the objective function value for different point placement (randomly obtained from Algorithm 1) with the same tolerance $\gamma = \frac{\epsilon}{4}$ in an area of dimensions $100\text{m} \times 100\text{m}$ with 10 sensors with the same characteristics that were used in Section 4.3.2, as well as $\epsilon = 0.8$ and $\tau = 1$. The values where obtained for 30 $k$-means++ runs from which the best Area Coverage solution is considered.

We conducted 300 experiments with different Anchor Point distributions where we studied the mean objective function value normalized to the number of Anchor Points $O(S, A)$ as given by

$$O(S, A) = \frac{1}{|A|} \sum_{j=1}^{|S|} \sum_{i=1}^{c_{ij}} \frac{1}{2} \left( d_{B_{ij}}(s_j) \right)^2,$$

(4.9)

where, as before, the notation $|A|$ stands for the cardinality of the set $A$ and $c_{ij}$ are the partition of $A$ according to the distances to the sensors in $S$. The variation of $O(S, A)$ with the increasing number of iterations is represented in Figure 4.7.
The mean of the normalized objective function for a ROI with the aforementioned characteristics is 75.85 with a variance of 0.58. Since the variance is low we can say that the results obtained for one Anchor Point placement are a good sample of the values we might get for other different placements.

If a normalized mean were not taken into consideration, \( i.e. \), for

\[
O(S, A) = \sum_{j=1}^{\left|S\right|} \sum_{i=1}^{\left|c_j\right|} \frac{1}{2} \left(d_{B_i}(s_j)\right)^2,
\]

we would get \( 2.97 \times 10^4 \) with variance \( 1.57 \times 10^5 \). In this case the high variance is a result of the different number of Anchor Points. In fact, the covariance between the number of Anchor Points and the objective function value is 638.4. This is an expected result since the objective function is a sum over all distances from each Anchor Point to a ball centered in the closest sensor with radius \( R_{j,r} \), therefore, the larger the number of points the more distances are taken into consideration, resulting in higher objective function values.
Chapter 5

Patrolling

This chapter addresses the Patrolling problem by defining our approach in Section 5.1, where some necessary concepts are exposed, and then enunciating the algorithm in Section 5.2. The chapter ends in Section 5.3 with the presentation of some results including time until the ROI is covered at least once.

5.1 Approach

In the Patrolling mission our objective is to visit every ball approximately the same number of times without forgetting any of them. In our case we want to start from the Area Coverage distribution and find the best path for each of the sensors in a collaborative way such that the limitations on the velocity and covering range are taken into consideration.

Even though the general formulation of the optimization problem for the three missions is the same, there are a few necessary changes. The first is the definition of the mass of the balls from which specific missions may be solved. For the patrolling mission the masses are defined in Section 5.1.1 and the new definition changes the steps to determine the Lipschitz constant, even though its value turns out to be the same as the one found in the Area Coverage mission.

The optimization problem for this case will have the same cost function as (3.16) but with a constraint on the movement as given by

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{\left|e_{ij}\right|} \frac{w_{ij}}{2} (d_{B_i}(s_{ij}(t)))^2 \\
\text{subject to} & \quad \|s_{ij}(t) - s_{ij}(t-1)\| \leq d_j
\end{align*}
\]

(5.1)

where the added constrain connects two consecutive time steps fixing a limit on the movement of the sensor \(d_j\) for the time difference considered.

It was already shown that the objective function for this problem is convex, so in order for the problem to be convex it is necessary that \(\|s_{ij}(t) - s_{ij}(t-1)\|\) be convex. Since it is a norm between a variable and a constant, by the norm definition, it is a convex constraint.
5.1.1 Mass of the balls

The first concern for the definition of the masses is that no balls should be forgotten. Defining $\Delta_i$ as the difference between the current time step and the last time the ball $i$ was covered, the masses have to take into consideration a temporal exploration such that the largest the value of $\Delta_i$, the larger the $w_i$ and the more likely this point is to be covered in the next iteration.

In the Area Coverage mission the masses were considered to be all equal to one so the optimal sensor placement was found taking into consideration only a minimization on the distances as in (4.1). If the mass is defined as

$$w_i = \frac{\Delta_i}{\sum_{k=1}^{|c_j|} \Delta_k},$$

(5.2)

every point that is not being covered would have exactly the same value of $w_i = w$ in the first iteration.

Having masses all equal to any number $w_i = w > 0$ produces the same optimal point as having all masses equal to one because the only difference between every point is the distance, so it is possible to say that the problems

$$\min_{s_j} \sum_{i=1}^{|c_j|} \frac{1}{2} (d_{B_i}(s_j))^2 \Leftrightarrow \min_{s_j} w \sum_{i=1}^{|c_j|} \frac{1}{2} (d_{B_i}(s_j))^2,$$

(5.3)

return the same minimizer.

For this to be true we have to be careful with the points that are covered in the first iteration because those points will have $w = 0$. However, in the left formulation of (5.3), the distance between the sensor position $s_j$ and the balls will also be zero, making the problems completely equivalent.

Since the points not covered will always increase $\Delta_i$ in the same amount, leading to $w_i = w$ at all time steps and the points that are covered will always have distance zero, then, with this definition the Patrolling problem will always be equivalent to the Area Coverage problem, which means they will always produce the same optimal position for the sensor $s_j$.

Defining the masses as (5.2) is not a good solution for the Patrolling mission because the sensors have to move to scan every point. The sum of distances between Anchor Points and sensors will be different as long as there are no symmetries. The masses will be defined as

$$w_i(t) = \alpha \frac{\Delta_i^m}{\sum_{k=1}^{|c_j|} \Delta_k^m} + (1 - \alpha) \frac{\max \{0, \| s_j(t-1) - c_{ji} \| - R_{j\tau} \}^n}{\sum_{k=1}^{max} \max \{0, \| s_j(t-1) - c_{jk} \| - R_{j\tau} \}^n}.$$  

(5.4)

The value of $\alpha$ is in the interval $0 \leq \alpha \leq 1$ and should be chosen taking into consideration a trade-off between time and spatial exploration. A few comments should be made:

- The weights $w_i(t)$ are normalized to 1 for each of the clusters of points $c_j$.
- The value of $m > 0$ is a factor that determines the importance of an increment of the time factor.
- The value of $n > 0$ is a factor that determines the importance of an increment of the spatial factor.
• If $\alpha = 0$ only the spatial factor will be considered, which does not guarantee that every point will be visited.

• If $\alpha = 1$ only the time factor is considered, so we will return to the impractical case of (5.2).

In an area where there is lack of resources there are balls that will not be covered by the initial deployment of the Area Coverage mission. To minimize the cost in a symmetric situation the solution leans to the centroid of the points, thus they are all at the same distance. One of the most trivial situations is the case of a single sensor covering 4 symmetric points as in Figure 5.1.

![Figure 5.1](image-url): In symmetric situations where there are uncovered Anchor Points the sensors will be placed at the centroid of the points in order to be at the same distance from all of them. If this happens the spatial and temporal factors will all be equal for all of the points and therefore we are in a static situation. The Anchor Points are represented by asterisks, the crosses are the sensors, solid lines are $R_{j\tau}$ and dashed lines are $R_{j\varepsilon}$.

In the Area Coverage mission for a small enough $R_{j\tau}$ (as in Figure 5.1) the sensor will be placed at the centroid of the points and by looking at the mass definition (5.4) it is easy to see that the spatial factor will be equal for all points. Since the temporal factor is also equal for all points the masses will always have the same value, and just like in the situation of equation (5.2), the problem will be equivalent to the Area Coverage mission, resulting in a constant optimal position for the sensor. Non-symmetric point placement is therefore needed and is obtained via Algorithm 1.

### 5.1.2 Lipschitz constant

To obtain a Lipschitz constant of the gradient we have to return to problem (5.1), where it is known that the values $w_i$ are constant for solving the problem. The gradient of the objective function is therefore the same found for the Area Coverage mission multiplied by a known constant $w_i$ as given by

$$
\nabla \left( \sum_{i=1}^{|\sigma|} \frac{w_i}{2} \left( d_{B_i}(x) \right)^2 \right) = \sum_{i=1}^{|\sigma|} w_i \left( x - P_{B_i}(x) \right). \quad (5.5)
$$
Using the Lipschitz constant definition (2.9) the Lipschitz constant of the gradient is found by

\[ \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - P_{B_{j_i}}(x)) - \sum_{i=1}^{\lfloor c_j \rfloor} w_i (y - P_{B_{j_i}}(y)) \| = \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - y) + \sum_{i=1}^{\lfloor c_j \rfloor} w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \|. \]  

(5.6)

Using the triangular inequality we arrive at

\[ \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - y) + \sum_{i=1}^{\lfloor c_j \rfloor} w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \| \leq \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - y) \| + \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \|. \]  

(5.7)

It was shown that for any set of \( N \) functions \( f_i(x) \) we have \( \| \sum_{i=1}^{N} f_i(x) \| \leq \sum_{i=1}^{N} \| f_i(x) \| \) and applying this result to this case we get

\[ \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - y) \| + \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \| \leq \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (x - y) \| + \| \sum_{i=1}^{\lfloor c_j \rfloor} w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \|. \]  

(5.8)

In Section 5.1.1 we defined the masses \( w_i \) and one of its fundamental aspects is that they are normalized to 1. Therefore we will always have \( w_i \leq 1 \). This allows us to write

\[ \sum_{i=1}^{\lfloor c_j \rfloor} \| w_i (x - y) \| + \sum_{i=1}^{\lfloor c_j \rfloor} \| w_i (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \| \leq \sum_{i=1}^{\lfloor c_j \rfloor} \| (x - y) \| + \sum_{i=1}^{\lfloor c_j \rfloor} \| (P_{B_{j_i}}(y) - P_{B_{j_i}}(x)) \|. \]  

(5.9)

When we get (5.9) we arrive at exactly the same result as in (4.5) so, from this point, the demonstration is exactly the same as for the Area Coverage mission of Section 4.1.2. The Lipschitz constant is exactly the same \( L = 2|c_j| \), as given by

\[ \| \nabla \sum_{i=1}^{\lfloor c_j \rfloor} \frac{w_i}{2} (d_{B_{j_i}}(x))^2 - \nabla \sum_{i=1}^{\lfloor c_j \rfloor} \frac{w_i}{2} (d_{B_{j_i}}(y))^2 \| \leq 2|c_j||x - y|. \]  

(5.10)

### 5.1.3 Scanning path

Doing a continuous calculation of the optimal positions is unrealistic, so a time discretization is needed. The computation of the masses for the balls requires knowledge about the last time a ball was actually scanned. A ball is considered to be scanned if its center is scanned. This comes from the definition of the Anchor Points made in Section 3.1. If the center is scanned a ball around that point can be imagined in which we can ensure that any point at a distance smaller than \( b \) is covered with a probability larger than \( \varepsilon \).

The discretization comes with the disadvantage of having points scanned in the path made from consecutive optimal positions of each time step that have to be taken into account. Therefore we update \( \Delta_t \) taking into consideration that path.

To calculate the distance of an arbitrary point \( p \) to a line defined by two consecutive sensor positions \( s_1 \) and \( s_2 \) we first refer to Figure 5.2, where we represented points \( s_1, s_2 \) and \( p \). The orthogonal projection of \( p \) onto the line is \( p_2 \).
Figure 5.2: Since we need to make a time discretization the path made between consecutive iterations has to be taken into account. A point \( p \) is said to be scanned if its distance to the line formed by two consecutive sensor positions \( s_1 \) and \( s_2 \) is smaller than \( R_j \).

Finding the position of \( p_2 \) can be written as an optimization problem. We know that the line formed by \( s_1 \) and \( s_2 \) is written as \( s(t) = s_1 + t(s_2 - s_1) \) where an arbitrary point \( s(t) \) is between points \( s_1 \) and \( s_2 \) if \( t \in [0, 1] \). The problem is therefore written as

\[
P_l(p) = \arg\min_{y \in \mathcal{L}} \| p - y \|_2,
\]

(5.11)

where \( \mathcal{L} \) is the line formed by \( s_1 \) and \( s_2 \) which is an affine set and therefore convex.

Problem (5.11) can also be written as

\[
\min_t \| p - s_1 - t(s_2 - s_1) \|^2,
\]

(5.12)

where the optimization function is a squared norm between a convex set and a known point, so it is also convex and therefore the problem has a unique solution. \( s_1 + t(s_2 - s_1) \) is the definition of line \( l \) between points \( s_1 \) and \( s_2 \) parameterized by \( t \).

To find the value of \( t \) that minimizes this problem we first compute the gradient of the optimization function (5.12) as

\[
\nabla \| p - s_1 - t(s_2 - s_1) \|^2 = \nabla \sum_i (p_i - s_{1,i} - t(s_{2,i} - s_{1,i}))^2
\]

\[
= 2 \sum_i (s_{2,i} - s_{1,i}) (p_i - s_{1,i} - t(s_{2,i} - s_{1,i}))
\]

\[
= 2 \sum_i ((p_i - s_{1,i}) (s_{2,i} - s_{1,i}) - t(s_{2,i} - s_{1,i})^2, \quad (5.13)
\]

\[
= 2 \sum_i (p_i - s_{1,i}) (s_{2,i} - s_{1,i}) - 2t \sum_i ((s_{2,i} - s_{1,i})^2
\]

\[
= 2 (p - s_1) \cdot (s_2 - s_1) - 2t\|s_2 - s_1\|^2
\]
where we used the definition of norm and dot product for an arbitrary dimension $i$. 

The value that minimizes (5.12) is found when the gradient of the objective function is zero, so we arrive at the final result

$$\nabla \|p - s_1 - t(s_2 - s_1)\|^2 = 0$$

$$2(p - s_1) \cdot (s_2 - s_1) - 2t\|s_2 - s_1\|^2 = 0$$

$$t = \frac{(p - s_1) \cdot (s_2 - s_1)}{\|s_2 - s_1\|^2}.$$  \hspace{1cm} (5.14)

Therefore we know that $p_2$ of Figure 5.2 is given by $p_2 = s(t)$ where $t$ is given by (5.14) and the distance to the line is given by $\|p_2 - p\|$.

The necessary conditions for a point to be scanned in the path between two consecutive iterations are therefore to have $\|p_2 - p\| < R_j \tau$ and $t \in [0, 1]$ or $\|s_1 - p\| < R_j \tau$ or $\|s_2 - p\| < R_j \tau$.

This result could also be found using the gradient of the projection into a convex set given in (3.18).

5.2 Algorithm for Patrolling mission

The pseudo-code for the algorithm that solves the Patrolling mission is quite simple. Algorithm 3 combines the principles already presented.

\begin{verbatim}
Algorithm 3: Patrolling.
\end{verbatim}

\begin{verbatim}
  input : Point distribution points, Initial sensor placement sensors, Time of last scan $T$, Mass update parameters $\alpha$, $m$, $n$.
  output: Sensor position for next iteration sensors.
  1 do indefinitely
  2 Update $T$ with scanned path;
  3 Update $W$ with (5.4);
  4 do Lloyd’s $k$-means extension algorithm with (points, sensors, $W$) obtain sensors;
  5 end
\end{verbatim}

In Algorithm 3, we define $W$ as the set of masses composed by all $w_i$ and $T$ is the set with the time difference between the current iteration and the last iteration when the points were covered, i.e., composed by all $\Delta_i$.

5.3 Results and discussion of Patrolling mission

5.3.1 Analysis of the mass update parameters

The mass of the balls is updated according to (5.4) where there are three different free parameters. Each of those has its own significance and leads to different sensor placement at each iteration and different final coverage results, making their analysis mandatory.
Analysis of $m$

The value of $m$ is set according to the importance of the time factor between two different balls. The bigger the $m$ the more important it is in comparison with the remaining balls.

This analysis was performed using the initial deployment represented in Figure 4.4.a with the same sensor characteristics, $n = 50$, $\alpha = 0.6$ and $d_j = 10\text{m}$. The results of the number of iterations until all balls are scanned at least once are shown in Figure 5.3 for different values of $m$.

![Figure 5.3: The higher the value of the parameter $m$ the bigger the importance of the difference between values of the time the point was last scanned and therefore the faster the patrolling mission is.](image)

As the value of $m$ increases the number of iterations until all balls are visited at least once is decreasing. The best results are obtained for $m = 150$ and $m = 75$. Since the difference between the results is not significant we will consider for the remaining tests $m = 75$. Higher $m$ also leads to better distributed paths and less points being covered more than once. One might think that the involved values for $m$ are quite large, however, the differences between different $\Delta_i$ are small, therefore to evidence them we need larger $m$.

These results are validated by the histograms of Figure 5.4 that represent the number of points that were visited and how many times, where bins of size 12 are used.

In Figure 5.4.a we can see that there are points visited as many as 200 times before all points are covered, however in Figures 5.4.c and 5.4.d the maximum number of times that a point is visited is only on the order of 70, showing that these are indeed better choices for $m$.

Analysis of $n$

The parameter $n$ is used to define the relative importance between differences in the distances between the sensors and the balls, therefore accounting a spatial exploration.

The tests to analyse this parameter are performed with the same initial position of Figure 4.4.a and with the same sensor characteristics. They will also all use $\alpha = 0.6$, $m = 75$ and $d_j = 10\text{m}$. The number of iterations until all balls were covered at least once for different $n$ are shown in Figure 5.5 where the line represents a situation where an increasing $n$ given by $n = \Delta_i$ is considered.
Figure 5.4: Increasing the value of $m$ leads to a better scanning distribution since there are fewer points being covered more times. In (a) there are points that were scanned as many as 200 times, however in (c) and (d) the points are only scanned as many as 70 times. The histograms are plotted until all points are scanned at least once.

Figure 5.5: Varying $n$ changes the importance between differences in the spatial factor. As can be seen the parameter does not directly effect the time until all points are covered at least once. The line stands for a model that takes into consideration the time exploration given by $n = \Delta_i$ for the $i$ ball.
This parameter does not directly effect the time until all points are covered so choosing it has to be done carefully. The mission objective is not to always try to reach the points that are furthest away from the sensors but to cover all balls approximately the same number of times without forgetting any of them which is exactly the reason why the Updating method was formulated. This was defined to consider a time exploration while at the same time performing the spatial exploration this method did however performed poorly in comparison with constant definition of the value $n$ as it is shown in Figure 5.5.

For constant $n$ there is no objective conclusion as to what values give a better performance. However, a value of $n = 50$ takes fewer iterations to scan all balls of the ROI and therefore will be the adopted value.

The paths performed for different $n$ show that all the ROI is covered at least by the path of one of the sensors, but all distributions are worse than for $n = 50$. The histograms for the values are represented in Figure 5.6 and for $n = 50$ is the same as represented in Figure 5.4.c.

![Figure 5.6: The number of times an Anchor Point is visited for different $n$ parameter shows that this parameter does not directly effect the time until all points are covered, however, the best distribution is obtained for $n = 50$ for which the points are visited as many as 70 times. For the remaining values of $n$ the number of visits exceeds 90.](image)

It is interesting to notice that the value of $n = 50$ outperforms all the remaining ones not only in the
number of iterations until all balls are scanned but also in the maximum number of times that a point is covered. \( n = 50 \) obtains a maximum of approximately 70, whereas all values of Figure 5.6 achieve approximately 100 scans.

**Analysis of \( \alpha \)**

The last parameter of the mass update that requires analysis is the parameter \( \alpha \) which sets the relative importance between spatial and time exploration. A higher value gives more importance to time exploration and less to spatial exploration.

The results for this analysis were obtained with the initial deployment of Figure 4.4.a with the same sensor characteristics, \( m = 75 \), \( n = 50 \) and \( d_j = 10 \text{m} \). The number of iterations until all balls are scanned at least once is represented in Figure 5.7.

![Figure 5.7: Changing \( \alpha \) sets the relative importance between the spatial and temporal factors. High values lead to high spatial consideration, therefore more iterations are needed to cover the area. Low values lead to slow motion and a high number of iterations. The best results are between 0.6 and 0.7, where a favorable compromise between the two explorations is reached.](image)

A value of \( \alpha = 0 \) will lead to taking into consideration only the spatial exploration, so no guarantees can be given that all balls will be covered. On the other hand, a value of \( \alpha = 1 \) takes only in consideration the spatial factor, which leads to the static situation already addressed in Section 5.1.1. Choosing the values of \( \alpha \) has to take into consideration these two limits, therefore we chose to test values between 0.5 and 0.8. For \( \alpha = 0.8 \) we are close to the static situation, thus having the disadvantage of some of the sensors being static for a small number of iterations. The best values happen for values of \( \alpha = 0.6 \) and \( \alpha = 0.7 \). As for \( \alpha = 0.5 \), the sensors are often in a situation where they consecutively move between the same two positions.

The histograms for the different \( \alpha \) values are represented in Figure 5.8.

In Figure 5.8.a the two peaks at approximately 75 and 160 represent the time intervals where the sensors were consecutively moving between the same positions. In Figure 5.8.d there are three peaks that were scanned more times than the remaining ones, which coincide with the balls that were scanned when
Figure 5.8: The effect of the alteration of $\alpha$ show that if a low value is considered, like in (a), we achieve situations where there is a movement between the same two points, as it seen by the two peaks at about 75 and 160. If a high value is considered like (d) the sensors are static, so some points are covered many times. For values around 0.6 and 0.7 like (b) and (c) we achieve a compromise and therefore a better distribution.

The sensors were static. Since the smallest number of iterations until all points are covered is reached for $\alpha = 0.7$ this is the adopted value.

5.3.2 Analysis of parameter $d_j$

The impact of the allowed movement per iteration $d_j$ was studied taking into consideration the same point distribution and initial deployment of Figure 4.4.a and the same considered sensors with parameter $\alpha = 0.7$, $m = 75$ and $n = 50$. In Figure 5.9 we present the number of iterations until all Anchor Points are covered at least once for different values of $d_j$.

From the values of Figure 5.9 it is easy to conclude that the higher the value of $d_j$ the faster the algorithm scans every Anchor Point. This is already expected because smaller movement at each time step requires more time to go through a given area. However, there is another effect that has to be taken into consideration. Every sensor is only able to find an optimal position for the current iteration and
never takes into consideration the ensuing positions. A bigger area results in a higher number of balls, and a small \( d_j \) will sometimes lead to situations of balance where a sensor will move only between two positions. These happen because the masses will increase for balls that are far from the sensor’s position. Since it cannot reach them it will not reset the masses, resulting in a movement between positions where it tries to reach those high mass balls consecutively. This undesirable equilibrium is usually broken when another sensor is close enough to change the balls he has to cover.

The bigger the value of \( d_j \) the better distributed the paths are. The histograms that represent the number of times the Anchor Points were scanned until all of them are scanned at least once are given in Figure 5.10, where bins of size 12 are used. The histogram for the case with \( d_j = 10 \text{m} \) is the same as Figure 5.4.c

From these histograms we can see that as \( d_j \) increases, the number of points that were covered more times showing that larger \( d_j \) is preferable. It is interesting to note that the case were \( d_j = 50 \text{m} \) requires the same number of iterations as having no restriction, \( i.e. \ d_j = +\infty \). It is not guaranteed that the movement without restrictions guarantees better results, however, as a general rule, the number of iterations decreases with higher \( d_j \).

### 5.3.3 Monte Carlo experiments

To verify that we are able to replicate the obtained results with the chosen parameters we conducted Monte Carlo experiments where we will analyze the number of iterations until all points are covered at least once for different point placements and initial deployments. The simulations were performed under the same conditions of Section 5.3.2 with \( d_j = 10 \text{m} \).

The variation of the mean number of iterations in the 600 performed simulations is presented in Figure 5.11, where we obtained an average number of 108 iterations with a variance of \( 1.9 \times 10^3 \). The high variance is not a good indicator, however, of the 600 simulations only 26 took more than 150
iterations to scan every point. The high variance is therefore a result of the outlier cases where a high number of iterations is obtained.

The maximum obtained in the 600 simulations is of 559 iterations and the 26 values higher than 150 have a contribution to the variance of about 85%. In fact, if the 26 simulations that took more than 150 iterations were discarded, the variance would be only 290.

Another interesting result is that for the Patrolling mission the number of iterations until all Anchor Points are scanned is not related to the number of Anchor Points since the covariance of the two values is only 1.92, a value that is very close to 0. This variance shows that the number of Anchor Points does not affect the number of iterations needed to scan a ROI. We can say that, apart from the outlier cases where the number of iterations is large, one Anchor Point distributions is enough to have an overall perspective of the number of iterations needed to scan a ROI.
Figure 5.11: The 600 iteration allowed to achieve a near constant value for the mean number of iterations until all points are scanned. The final value is of 108 iterations, a value that is obtained with a variance of $1.9 \times 10^3$.

Analysis of intruder identification

To evaluate the efficiency of the Patrolling mission we will consider the same ROI parameters and the same sensors characteristics to identify randomly placed targets in the ROI. A target $p_t$ is considered to be identified by sensor $s_j$ if

$$p(d_{tj}) > U(0, 1),$$

where $p(d_{tj})$ comes from the definition of the sensors model made in (3.2).

We will consider 20 different Anchor Point placements and initial positions in a ROI. The Patrolling mission will be done during 500 time steps and a target will be randomly placed in the ROI with a probability of 70%. During the simulations 7064 targets were created from which 68% are identified in the first iteration.

This results allow us to conclude that for the considered ROI the intruder identification is done in the first iteration in approximately 68% of the cases. We can however guarantee that every static intruder in the ROI will be identified during the mission due to the masses definition that guarantee that every part of the ROI will be visited.
Chapter 6

Target Tracking and Escorting

The last mission we are interested in is Target Tracking and Escorting, which is addressed in this Chapter. We formalize the three different types of trajectories that will be considered for the intruders in Section 6.1. The approach to the mission is given in Section 6.2 where we define the masses for the balls and the Lipschitz constant for the problem. Finally, some results are presented for the different considered trajectories and for more than one intruder in Section 6.4.

6.1 Intruder trajectories

For the Target Tracking and Escorting mission it is necessary to first define the trajectories of the sensors for the simulations of the algorithm. In this thesis three different trajectories will be taken into consideration: linear, circular with increasing radius and bounded random.

6.1.1 Linear trajectory

To define the linear trajectory it is only necessary to know the displacement per unit of time $\vec{v}$, and the initial position $p_i$.

The position of the intruder at time $t$ is given by $p(t) = p(t-1) + \vec{v}$ and initiated at $p(1) = p_i$. An example is shown in Figure 6.1.

6.1.2 Outward spiral

For this trajectory it is necessary to define a center position $p_c$, the travelled distance $v$, the initial radius $r_i$, the radius increase $\Delta r$ and the initial angular position $\theta_i$.

The position at time $t$ is given by $p(t) = p_c + [r \cos(\theta(t)) \quad r \sin(\theta(t))]$ where $\theta(t) = \theta(t-1) + \frac{v}{r}$ and $r(t) = r(t-1) + \Delta r$ with $\theta(1) = \theta_i$ and $r(1) = r_i$.

The increment on $\theta$ comes from the fundamental relation for the angular velocity $\omega$ given by $\vec{w} = \frac{r \times \vec{v}}{|\vec{v}|^2}$ and is used to guarantee that the intruder travels approximately always the same distance.

An example with $v = 2m$, $r_i = 5m$, $\Delta r = 0.2$, $p_i = (30,30)$ and 100 time steps is given in Figure 6.2.
6.1.3 Bounded random trajectory

The last trajectory uses a random number generator with a uniform distribution to find the next position. It is necessary to define the initial position $p_i$ and the limits of the random numbers $x_{\text{lim}-}$, $x_{\text{lim}+}$, $y_{\text{lim}-}$ and $y_{\text{lim}+}$.

The positions are updated according to $p(t) = p(t - 1) + [U(x_{\text{lim}-}, x_{\text{lim}+}) \ U(y_{\text{lim}-}, y_{\text{lim}+})]$. An example is shown in Figure 6.3, with $p_i = (25, 25)$, $x_{\text{lim}-} = y_{\text{lim}-} = -2.5$ and $x_{\text{lim}+} = y_{\text{lim}+} = 2.5$. 

Figure 6.1: Intruder linear trajectory, where the cross represents the initial position.

Figure 6.2: Outward spiral trajectory, where the cross represents the initial position.
6.2 Approach

For this mission we are also taking into consideration a time discretization so the optimization problem will be the same as the Patrolling mission (5.1) which was already proved to be convex. The difference to the Patrolling mission is the definition of the mass, which will also result in a different Lipschitz constant. The mission objective is to keep Patrolling the ROI while at the same time being able to escort one or more intruders in the same ROI.

6.2.1 Mass of the balls

In this mission a sensor should be closer to an intruder than $R_{j\tau}$, but Patrolling of the remaining area is still required.

Two sets will be considered, the already defined set of Anchor Points $\mathcal{A}$ and the intruder set $\mathcal{I}$. The algorithm will do the division of the set of balls $\mathcal{A} \cup \mathcal{I}$ into the sets $c_j$ of problem (5.1).

The division of the sets in Anchor Points and intruders is important to define the masses which will be given by

$$w_i(t) = \begin{cases} \text{(5.4)}, & \text{if } c_{ji} \in \mathcal{A} \\ N, & \text{if } c_{ji} \in \mathcal{I} \end{cases}$$  (6.1)

The principle behind (6.1) is to use the same mass definition for the anchor points that was used in Patrolling, so that the sensors are able to continuously perform the Patrolling mission. The mass for the intruders will be a constant given by $N$. It is known that the masses should reflect the importance of the respective points, so the intruders need a mass higher than the anchor points. Since the Anchor Points masses are always normalized to unit sum within the same $c_j$, to have a higher value for the intruders it is enough to choose $N \gg 1.$
6.2.2 Lipschitz constant

For the computation of the Lipschitz constant we divide the cluster set \( c_j \) between the points that are intruders \( I \) and the Anchor Points \( A \) in such way that \( c_j = A \cup I \). The sets \( I \) and \( A \) are represented as (6.2) and (6.3), respectively.

\[
I = \{ p : p \in c_j, \ p \in I \} \tag{6.2}
\]

\[
A = \{ p : p \in c_j, \ p \in A \} \tag{6.3}
\]

The optimization problem can therefore be decomposed as

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{n} w_i \left( \frac{d_{B_A}(s_i(t))}{2} \right)^2 + \sum_{i=1}^{m} w_i \left( \frac{d_{B_I}(s_i(t))}{2} \right)^2 \\
\text{subject to} & \quad \|s_j(t) - s_j(t-1)\| \leq d_j.
\end{align*}
\tag{6.4}
\]

The gradient of the objective function of problem (6.4) considering an arbitrary \( x \) will be given by

\[
\nabla \left( \sum_{i=1}^{n} w_i \left( \frac{d_{B_A}(x)}{2} \right)^2 + \sum_{i=1}^{m} w_i \left( \frac{d_{B_I}(x)}{2} \right)^2 \right) = \sum_{i=1}^{n} w_i \left( x - P_{B_A}(x) \right) + \sum_{i=1}^{m} w_i \left( x - P_{B_I}(x) \right). \tag{6.5}
\]

To arrive at the Lipschitz constant we start by writing

\[
\| \sum_{i=1}^{n} w_i \left( x - P_{B_A}(x) \right) + \sum_{i=1}^{m} w_i \left( x - P_{B_I}(x) \right) - \sum_{i=1}^{n} w_i \left( y - P_{B_A}(y) \right) - \sum_{i=1}^{m} w_i \left( y - P_{B_I}(y) \right) \| =
\| \sum_{i=1}^{n} w_i \left( x - y \right) + \sum_{i=1}^{m} w_i \left( x - y \right) + \sum_{i=1}^{n} w_i \left( P_{B_A}(y) - P_{B_A}(x) \right) + \sum_{i=1}^{m} w_i \left( P_{B_I}(y) - P_{B_I}(x) \right) \|, \tag{6.6}
\]

and using the triangle inequality we get

\[
\| \sum_{i=1}^{n} w_i \left( x - y \right) \| + \| \sum_{i=1}^{m} w_i \left( x - y \right) \| + \| \sum_{i=1}^{n} w_i \left( P_{B_A}(y) - P_{B_A}(x) \right) \| + \| \sum_{i=1}^{m} w_i \left( P_{B_I}(y) - P_{B_I}(x) \right) \|. \tag{6.7}
\]

From (6.7) we can separate the demonstration in two different problems, one for the set \( A \) and the other of set \( I \). For the set \( A \) we know that the \( w_i \) are normalized to unit sum and using the fact that for any set of \( N \) function \( f_i(x) \) we have \( \| \sum_{i=1}^{N} f_i(x) \| \leq \sum_{i=1}^{N} \| f_i(x) \| \), we can easily see that

\[
\| \sum_{i=1}^{n} w_i \left( x - y \right) \| + \| \sum_{i=1}^{m} w_i \left( P_{B_A}(y) - P_{B_A}(x) \right) \| \leq \| \sum_{i=1}^{n} \left( x - y \right) \| + \| \sum_{i=1}^{m} \left( P_{B_A}(y) - P_{B_A}(x) \right) \|. \tag{6.8}
\]

From this point we are in exactly the same situation as (4.5) of the Area Coverage mission so we know...
that the Lipschitz constant for the sub-problem in the anchor points $A$ is given by $2|A|$.

For the sub-problem in $I$ we know that the masses are all given by $w_i = N$ so we can write

$$\left\| \sum_{i=1}^{I} w_i (x - y) \right\| + \left\| \sum_{i=1}^{I} w_i (P_{B_i}(y) - P_{B_i}(x)) \right\| \leq \sum_{i=1}^{I} N \left\| x - y \right\| + \sum_{i=1}^{I} N \left\| P_{B_i}(y) - P_{B_i}(x) \right\|. \quad (6.9)$$

In Section 4.1.2 we saw that for any closed convex set $S$ we have $\|P_S(x) - P_S(y)\| \leq \|x - y\|$ and applying this rule to (6.9) we get

$$\sum_{i=1}^{I} N \left\| x - y \right\| + \sum_{i=1}^{I} N \left\| P_{B_i}(y) - P_{B_i}(x) \right\| \leq \sum_{i=1}^{I} N \left\| x - y \right\| + \sum_{i=1}^{I} N \left\| y - x \right\| = 2N|I|\|x - y\|. \quad (6.10)$$

Combining the results to the two sub-problems we arrive at the final value for the Lipschitz constant $L = 2|A| + 2N|I|$ as

$$\left\| \nabla \left( \sum_{i=1}^{A} \frac{w_i}{2} (d_{B_i}(x))^2 \right) + \sum_{i=1}^{I} \frac{w_i}{2} (d_{B_i}(x))^2 \right\| + \nabla \left( \sum_{i=1}^{A} \frac{w_i}{2} (d_{B_i}(y))^2 \right) + \sum_{i=1}^{I} \frac{w_i}{2} (d_{B_i}(y))^2 \right\| = (2|A| + 2N|I|)\|(x - y)\|. \quad (6.11)$$

### 6.3 Algorithm for Target Tracking and Escorting mission

The pseudo-code for the algorithm that solves the Target Tracking and Escorting mission is almost the same as the used in the Patrolling mission. The pseudo-code is given by Algorithm 4.

<table>
<thead>
<tr>
<th>Algorithm 4: Target Tracking and Escorting.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong>: Point distribution points, Initial sensor placement sensors, Time of last scan $T$, Mass update parameters $\alpha, m, n, N$.</td>
</tr>
<tr>
<td><strong>output</strong>: Sensor position for next iteration sensors.</td>
</tr>
<tr>
<td>1 do indefinitely</td>
</tr>
<tr>
<td>2 Update $T$ with scanned path;</td>
</tr>
<tr>
<td>3 Update $W$ with (6.1);</td>
</tr>
<tr>
<td>4 do Lloyd’s $k$-means extension algorithm with (points, sensors, $W$) obtain sensors;</td>
</tr>
<tr>
<td>5 end</td>
</tr>
</tbody>
</table>

In Algorithm 4, just like in Algorithm 3, $W$ is defined as the set of weights composed by all $w_i$ and $T$ is the set with the time difference between the current iteration and the last iteration when the points were scanned, i.e. $\Delta t$. The overall concept between the algorithms of both missions is the same, however, in this case the update of the masses is given by (6.1) and the Lipshitz constant is given by (6.11).
6.4 Results and discussion for Target Tracking and Escorting

The results presented in this Section will take into consideration the initial deployment of Figure 4.4.a with the same Anchor Point placement and the same sensor characteristics. After the analysis performed in Section 5.3, the adopted values for the parameters that update the masses are $m = 75$, $n = 50$ and $\alpha = 0.7$. The following analysis will be made under these conditions for the trajectories given in Section 6.1 with $N = 1000$.

6.4.1 Analysis of linear trajectory

For the linear trajectory tracking simulation we used $\vec{v} = (0.75, 0.75)\text{m/step}$ and $p_i = (2, 2)\text{m}$. The trajectory was performed with one sensor tracking the whole intruder’s path with 120 iterations considered. The paths for the target and the sensor are represented in Figure 6.4.

![Figure 6.4](image.png)

Figure 6.4: A sensor is able to perform the Tracking and Escorting mission of an intruder following a linear trajectory path while at the same time leaning to positions that do the area Patrolling. The crosses represent the initial positions of the sensor and the intruder.

At the initial iteration the sensor is no yet following the target (see Figure 4.4.a) but in the remaining iterations the tracking mission is performed successfully.

To shown that there is an adaptation of the network to the sensors in charge of the intruder we represent Figure 6.5, where the sensor positions, Anchor Points, sensing ranges, Voronoi cells and intruder are represented for four different steps.

From Figure 6.5.a to Figure 6.5.b we can see that in step 35 there is a large uncovered area in the lower left corner because the sensor that is in charge of covering it is tracking an intruder. As the target gets further inside the ROI the sensor that was previously in charge of that corner is no longer close to it, so the remaining sensors adapt to cover that corner.

In Figure 6.5.c and Figure 6.5.d it is possible to see that there is a continuous adaptation between the network agents to the sensor that is in charge of tracking the target.
While an intruder performs a linear trajectory, the network is able to collectively adapt to the path followed by the sensor that is tracking an intruder while not compromising the patrolling of the remaining area. This figure represents the evolution at different time steps where the target is marked as a red square. Sensors and ranges are represented as in Figure 4.4.

While one sensor was in charge of following the target the remaining ones were able to scan the anchor points in the 120 iterations, as shown in the histogram of Figure 6.6 where the number of times the anchor points were scanned is represented.

6.4.2 Analysis of outward spiral trajectory

For the outward spiral a initial position of (50, 50) m, $v = 2$ m/step, $r = 5$ m and $\Delta r = 0.25$ m were considered. The paths performed by the sensor in charge of covering it and the intruder are represented in Figure 6.7 for 150 steps.

From Figure 6.7 it is possible to see that the sensor is able to perform the tracking mission for the represented target trajectory. A few representations of the whole network at different time steps is shown.
In all time steps represented in Figure 6.8 the network adapts to the movement of the sensor that is following the target. The adaptation is not a surprise since the Anchor Points that each of the sensors is responsible for change in each time step automatically. While there is a sensor performing Target Tracking and Escorting, the remaining ones are able to perform Patrolling of the area where they only take 89 steps to scan all the ROI at least once. The histogram representing the number of times a point was scanned is represented in Figure 6.9.

6.4.3 Analysis of bounded random trajectory

The bounded random trajectory was performed with initial position $p_i = (50, 50) \text{m}$ and with limits $x_{\text{lim}} = y_{\text{lim}} = -5$ and $x_{\text{lim}} = y_{\text{lim}} = 5$. The representation of the sensor performing the tracking of
Figure 6.8: The alteration of the intruders while performing an outward spiral trajectory does not affect the networks ability to adapt to the tracking of an intruder. The remaining sensors are therefore able to achieve Patrolling of the area.

From Figure 6.10 it is possible to see that the sensor is able to perform tracking of the intruder at all times. While one sensor performs the Target Tracking and Escorting the remaining ones were able to perform Patrolling of the ROI, taking only 112 steps to scan every Anchor Point of the region. Just like in the previous situations the network adapts to the path of the sensor that is tracking the intruder. The histogram representing the number of times the anchor points were scanned is shown in Figure 6.11.

For the trajectories represented, the histograms of Figures 6.6, 6.9 and 6.11, show that there is no significant difference with respect to the case where there is no sensor performing a Target Tracking mission. In fact, this result is also shown by the number of iterations until all Anchor Points are scanned at least once, where the largest difference is of only 26 iterations and happens for the linear trajectory.
Figure 6.9: In the 150 iterations considered for tracking an intruder performing an outward spiral, the Patrolling mission is successful since all Anchor Points are covered at least once. As seen by the histogram most of the points are covered less than 50 times.

Figure 6.10: If a bounded random trajectory is considered the sensor is also able to perform the tracking mission without difficulty. The crosses represent the initial positions of the sensor and the intruder.

6.4.4 Analysis for target with larger velocity than the sensors

Another interesting case to consider is when the intruder moves faster than the sensors. For this simulation we will use the bounded random trajectory with the same initial position as Section 6.4.3 but with limits $x_{\text{lim}, 1} = y_{\text{lim}, 1} = -15$ and $x_{\text{lim}, 2} = y_{\text{lim}, 2} = 15$. Two situations where there is a change in the sensor that is in charge of the intruder are represented in Figure 6.12.

In Figures 6.12.a and 6.12.b we represent two consecutive time steps in which the intruder is being covered by two different sensors. It is easy to note that the transfer of responsibility does not affect the network behaviour since there is a straightforward adaptation of any sensor to the new conditions. In fact this example shows that changing between the Patrolling and Target Tracking is effortless.

For this situation it took only 75 steps to scan every anchor point of the ROI showing once again that
Figure 6.11: In 150 iterations the network is able to track an intruder with bounded random trajectory and all points were scanned at least once, where most of the points are scanned less than 50 times, showing that there is a good distribution in the Patrolling mission. For the histogram only the Anchor Points are considered.

Figure 6.12: When an intruder has a velocity higher than the sensors the network is able to adapt and most of the times it is able to change the sensor in charge of tracking. An example of the network behaviour at consecutive time steps while tracking a target marked as a square performing a bounded random trajectory with high velocity is presented.

the fact that there is an intruder inside the ROI does not affect the overall performance of the network for the Patrolling mission.

6.4.5 Analysis with two targets

The simulations given in Sections 6.4.1, 6.4.2, 6.4.3 and 6.4.4 were obtained considering only one target deployed in the ROI. Changing to a situation where two or more targets are present is straightforward; however, some examples for two targets will be given in this Section.
Outward spiral and bounded random trajectories

In the first simulation we will consider two different trajectories, the outward spiral that will be the used in Section 6.4.2 and the bounded random trajectory that will use the same limits used in Section 6.4.3 but with initial position (25, 75)m. The paths taken by the intruders and the sensors are represented in Figure 6.13.

![Figure 6.13](image)

In Figure 6.13 we can see that the sensors are able to track the intruders, escorting their paths. Just like in every situation presented so far, the network is able to adapt to these two intruders. In this simulation 109 iterations were necessary to scan all the anchor points.

Example with one sensor following two targets

Even though not being the ideal situation the sensors are able to follow two targets at the same time. An example of such situation is given in Figure 6.14.

In Figure 6.14.a the intruders are being covered by two different sensors, however, as the circular trajectory gets closer to the bounded random trajectory a single sensor covers both of them at the same time (Figure 6.14.b). The continuous movement of the intruders makes them diverge again. The sensor is no longer able to cover both at the same time, so it resets to an equidistant position (Figure 6.14.c). As the distance between them increases arises the need the need arises for two sensors to cover the intruders, which is represented in Figure 6.14.d.

6.4.6 Monte Carlo experiments

To evaluate the performance of the Target Tracking and Escorting mission we conducted Monte Carlo experiments with different intruder trajectories are considered. We will analyze the time it took the
network to cover the area, if the target is ever lost, \textit{i.e.}, not covered at same time step, and the number of times it is lost for different defined trajectories.

For all tests 300 different Anchor Point placement and trajectories will be considered, where the sensors and ROI used the same parameters of Section 4.3.5. The mission is considered to be successful if in 150 steps the target is never lost. If all Anchor Points are not scanned in 150 steps tracking will be extended as long as needed.

\textbf{Linear trajectory}

The considered trajectory for these tests always starts with $p_i = (2, 2)\text{m}$ and the velocity is given by $\vec{v} = (U(0, 5), U(0, 5))$, where $U(0, 5)$ is a random number with uniform distribution between 0 and 5. Once the position of the target hits a boundary, a new velocity vector that allows to get away from the
boundary is chosen according to \( \vec{v} = (U(-5, 5), U(-5, 5)) \).

The mission for this case was successful in 81% of the cases. The unsuccessful cases are coincident with the initial iterations. The fact that the initial position is in the corner of the ROI difficult the initial identification of the intruder in the area. Nevertheless, once the intruder is identified the tracking mission is successfully accomplished.

For this trajectory the average time it took to scan every Anchor Point of the ROI was of 95 time steps with a variance 246. This is not a high value, however, it is associated with the randomness imposed by the definition of the new velocity once a sensor hits a boundary. The mean value obtained for the time until all Anchor Points are scanned at least once is almost the same obtained for the case where there is no target. It is interesting to note that this result is better than that obtained in Section 6.4.1 which is a result of a higher velocity considered for this case.

**Bounded random trajectory**

For the bounded random trajectory we will consider an initial position \( p_i = (50, 50) \text{m} \) and limits \( x_{\text{lim}^-} = y_{\text{lim}^-} = -5 \) and \( x_{\text{lim}^+} = y_{\text{lim}^+} = 5 \).

For the bounded random trajectory, the mission was successful in 99% of the simulations, which means that in only 3 occasions the tracking was not accomplished. In these 3 instances detection fails at the beginning of the algorithm, which means that once the target is detected the sensors are able to track it and never lose it. Compared to the previous linear trajectory, the target starts in a much more central position and is easier to detect.

The mean time it took to scan every Anchor Point for this case is 118 time steps with a variance of 410 time steps. The variance is high, however we have to take into consideration that there is a random trajectory that continuously change the performed path in every iteration therefore changing the needed network adaptation. Nevertheless, the average time is not significantly higher when compared to the case where there is no target to track.
Chapter 7

Discussion and Perspectives

In this Chapter we summarize the main Achievements of the work in Section 7.1 and give some ideas for Future Work that could improve the algorithm performance in Section 7.2.

7.1 Achievements

Target management is a field with a wide variety of challenges. Each different formulation brings different possibilities and difficulties, which makes it difficult to find a unified framework that can tackle these missions efficiently. Our main achievement is the novel $k$-disks formulation together with the developed point distribution technique, which is able to solve three different types of missions using a common background.

Even though we are not the pioneers, to the best of our knowledge we are the first to use the ROI discretization into Anchor Points alongside $k$-means to define Area Coverage, Patrolling or Target Tracking policies. This discretization allows us to solve our missions and also makes each individual Anchor Point able to give information about its surroundings. Most solutions that aim at covering a discrete set of points are able to do it and ours is no exception; however, our Anchor Points are judiciously selected such that the discrete problem acts as a proxy for the full continuous problem. In this sense the framework is more powerful than that of other discrete approaches that don’t offer such guarantees.

In the Area Coverage mission, which considers static sensors, we formulated an initial placement solution for a known set of sensors. Most point coverage solutions are not able to consider limited range sensors, however, our solution has that ability. It is also capable of considering sensors with different ranges and different model characteristics in such a way that it defines which sensor should be placed to which Anchor Point partition to attain the best solution. Our formulation is also able to find the positions that achieve $\varepsilon$-full area coverage if it exists. The Area coverage problem for 3D volumes and areas with different shapes is also addressed and some examples are given in Section 4.3. The conducted Monte Carlo experiments also show that the initial placement found for one sample is able to characterize any ROI with the same considered sensors and dimensions.
The Patrolling mission is addressed in Chapter 5 where we use for the first time the concept of masses. Their update has a set of free parameters for which we performed an extensive analysis that reveals how they impact the time until all Anchor Points are scanned. The proposed solution for this mission has the same background as the Area Coverage mission, so it is able to accomplish its goal in any ROI with any shape, dimension or sensors characteristics.

Target Tracking and Escorting mission is presented in Chapter 6 where three different intruder trajectories are considered. The algorithm is then tested for those trajectories and for cases where multiple intruders are considered inside the ROI. The simulations show that the algorithm is able to adapt to the different trajectories and that in cases where the intruders have a velocity higher than the sensors, the network is able to cooperatively guarantee that the target will be covered at all times. The algorithm is also able to perform the mission when more than one intruder is considered, and a single sensor is able to track more than one intruder at a given time. The proposed algorithm also has the ability to give each of the intruders a mass according to their relevance, which allows to prioritize the different targets. The simulation results for this mission show that the presence of an intruder inside the ROI does not compromise the Patrolling mission of the remaining sensors since the time until all points are covered is similar.

The algorithm is able to run autonomously since the only difference between the missions is the Lipschitz constant used in the FISTA algorithm. However, its value for Target Tracking and Escorting (which is the most restrictive because it is higher) is also feasible for the remaining missions.

The robustness of the algorithm is also proved by the different simulations where the network was able to overcome situations like multiple intruders, intruders with high velocity, or sensors tracking more than one intruder.

The results presented for the Patrolling, and Target Tracking and Escorting mission can all be found in https://github.com/p-carrasqueira/k-disks, where it is possible to find videos of the simulations, histograms showing the number of points visited and the developed code.

7.2 Future Work

Even though the algorithm is currently robust and autonomous, there are still developments that can be foreseen.

At the moment the algorithm is centralized, i.e., executed at a central location based on all available data. However, there are solutions that can be implemented in order to find a distributed solution. One possibility is to consider that each individual sensor has a limited range of communication with the remaining sensors that allows them to change information such as their position or the Anchor Points they have scanned. Under these conditions each individual sensor would be able to perform its own independent $k$-means taking into consideration only the sensors that are within communication range.

The placement of the non-symmetric Anchor Points is a feasible solution, however it is not optimal. Even though the optimal solution is very hard to obtain there is room for improvement of the attained placement. Such improvement would increase the algorithm performance since it will reduce the number
of necessary Anchor Points.

The mass update parameters were tuned taking into consideration the attain results for different values. A more systematic way of tuning these parameters would be interesting for real-world employment. Another open possibility is that the presented model might not be the best to achieve the most optimal covering strategy. The development of new models should be taken into consideration.

Incorporating a localization algorithm for the target that allows to predict the intruder’s next position might also be interesting. That algorithm alongside with a solution that takes into consideration the future steps, would also be of interest.

Currently, the algorithm resorts to discrete time iterations and achieves better results for large time steps, making it difficult to find a continuous-like control law. The smoothness of the approach should therefore be worked on.

Finally, the results presented in this thesis are all computational simulations, therefore it would be interesting to test this approach in sizable real-world environments. Although, we feel confident that the algorithm would perform well in a such situations.
Bibliography


