TRABAJO FIN DE MÁSTER



BIO-INSPIRED COMPLETE COVERAGE PATH PLANNING FOR SARS-COV-2 ULTRAVIOLET DISINFECTION ROBOTS

TRABAJO FIN DE MÁSTER PARA LA OBTENCIÓN DEL TÍTULO DE **MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL**

FEBRERO **2021**

Daniel Vicente Rodrigo Muñoz

DIRECTORES DEL TRABAJO FIN DE MÁSTER:

Matilde Santos Peñas

Jesús Enrique Sierra García



UNED





Facultad de Informática

Universidad Complutense de Madrid
(Conjunto con Universidad Nacional de Educación a Distancia)

Bio-inspired complete coverage path planning for SARS-CoV-2 ultraviolet disinfection robots

Trabajo Fin de Máster

Tipo–A: Proyecto específico propuesto por un profesor

Máster en Ingeniería de Sistemas y de Control

AUTOR: Daniel Vicente Rodrigo Muñoz
DIRECTOR/A: Matilde Santos Peñas
Jesús Enrique Sierra García

2021

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Daniel Vicente Rodrigo Muñoz

Firma del alumno



"An algorithm must be seen to be believed."
Donald Knuth

Agradecimientos

No podía finalizar sin aprovechar la ocasión de dedicaros unas palabras. Con este trabajo pongo fin a una etapa, la cuál comencé en un lugar de La Mancha, de cuyo nombre si quiero acordarme, y siempre lo haré.

Quiero agredecer a todos aquellos, que tanto en la universidad, como fuera de ella, me han formado como persona. Gracias por valorar mi creatividad, y contrubuir a mi desarrollo profesional.

Gracias a Matilde y Enrique, por ofrecerme la oportunidad de trabajar con vosotros. Vuestra guía y consejo han sido fundamentales a lo largo de todo el proyecto.

Gracias a mis padres, por ofrecerme su apoyo incondicional. Vosotros me dais la fortaleza necesaria para afrontar todos los retos que surgen en mi vida.

Gracias a mi hermana, simplemente por serlo. Puede que nuestros caminos vayan en distintas direcciones, pero siempre nos quedará nuestro hogar.

Gracias a Esther, por estar en mi vida. Tu apoyo y ayuda son inestimables para lograr todas mis metas.

Abstract

The *Bio-inspired complete coverage path planning for SARS-CoV-2 ultraviolet disinfection robots* project has been developed for the inter-university Master's Degree in Systems and Control Engineering, in the Departments of Computer Science and Automation of UNED (Universidad Nacional de Educación a Distancia) and Computer Architecture and Automation of the UCM (Complutense University of Madrid).

This document explores the disinfection of environments through UV radiation using mobile robotics. Once the pandemic (COVID-19) started the previous year, many companies and research groups made an effort to find the most effective solution to disinfect contaminated spaces, guaranteeing the safety of people. The use of ultraviolet light for the disinfection of buildings is combined with the usage of autonomous robots. In this approach, the main problem is to find the path which covers the entire environment and guarantees its complete disinfection.

This document pretends to solve the need to generate a path for disinfecting the affected environment. Bio-inspired algorithms are used based on the behaviour of neuronal networks. These techniques have previously been used in many other applications. However, an improved algorithm is proposed to perform the complete coverage path planning focused on disinfection through ultraviolet radiation.

An approximate model of disinfection has been proposed to verify the generated path behaviour, considering the features of the lamps on the market. Although numerous pathogens can be disinfected using this radiation, specific values have been selected for the effective disinfection of SARS-CoV-2. These models provide a perception of the radiation dose received in the workspace once the robot covers the entire path.

A speed controller has been designed to adjust the dose received by the environment. The purpose is to maximise disinfection without harming the material elements in the surroundings. The radiation model used and the information obtained through the complete coverage path planning algorithm are fundamental for this task.

A simulation environment has implemented to verify and validate the performance of the entire implemented system. Several tests have been done, varying the complexity of

the map increasingly using this environment. The results obtained have made it possible to compare the two different motions of the robot (spiral motion and boustrophedon[1] motion). Moreover, in each of the tests, the path generated for each of the maps can be analysed, as well as the degree of disinfection achieved in the environment.

In conclusion, a system capable of generating paths for the complete disinfection of the workspace using bio-inspired techniques has been implemented, as well as an approximation of the dose that the environment would receive following this route.

Index Terms: complete coverage path planning (CCPP), Glasius bio-inspired neural network (GBNN), dose, disinfection process, robot, ultraviolet germicidal irradiation (UVGI).

[1]Movement consisting of alternately performing one vertical movement from bottom to top and the next one from top to bottom or vice versa.

Resumen

El proyecto *Planificación de rutas de cobertura completa bio-inspiradas para robots de desinfección ultravioleta de SARS-CoV-2* ha sido desarrollado para el Máster interuniversitario de los departamentos de Informática y Automática de la UNED (Universidad Nacional de Educación a Distancia) y de Arquitectura de Computadores y Automática de la UCM (Universidad Complutense de Madrid)

El presente documento explora la desinfección de espacios a través de radiación UV, empleando robótica móvil. Una vez iniciada la pandemia (COVID-19) el año anteior, muchas empresas y grupos de investigación se esforzaron por encontrar la solución más eficaz para desinfectar los espacios contaminados, garantizando la seguridad de las personas. El uso de luz ultravioleta para la desinfección de edificios se combina con la utilización de robots autónomos. Mediante este enfoque, el principal problema que aparece es encontrar el camino que cubra todo el entorno y garantice su completa desinfección.

Este trabajo pretende resolver la necesidad de generar una ruta de desinfección que cubra todo el entorno afectado. Se emplean algoritmos bio-inspirados basados en el comportamiento presente en las redes neuronales. Estas técnicas se han utilizado previamente en muchas otras aplicaciones. Sin embargo, se propone un algoritmo mejorado para realizar la planificación de la ruta que cubra todo el espacio de trabajo, enfocado en la desinfección mediante radiación ultravioleta.

Se ha propuesto un modelo aproximado de desinfección para verificar el comportamiento de la ruta generada, teniendo en cuenta las características de las lámparas de desinfección disponibles en el mercado. Aunque se pueden desinfectar numerosos patógenos con esta radiación, se han seleccionado valores específicos para la desinfección eficaz del SARS-CoV-2. Estos modelos proporcionan una percepción de la dosis de radiación recibida en el espacio de trabajo una vez que el robot finaliza su recorrido. Alcanzar la dosis adecuada garantiza la eliminación del patógeno a tratar.

Se ha diseñado un control de velocidad que permita ajustar la dosis de radiación que recibe el entorno. El objetivo principal es maximizar la desinfección, sin dañar los elementos materiales. El modelo de radiación utilizado, así como la información obtenida

mediante el algoritmo que genera la ruta que debe seguir el robot para cubrir todo el espacio de trabajo, son fundamentales para esta labor.

Se ha implementado un entorno de simulación para verificar y validar el rendimiento de todo el sistema. Se han realizado varias pruebas, variando la complejidad del mapa de manera incremental, empleando este entorno de simulación. Los resultados obtenidos han permitido comparar dos técnicas de movimiento diferentes para el robot (movimiento en espiral y movimiento boustrófedon[2]). Además, en cada una de las pruebas se puede analizar la ruta generada para cada mapa, así como el grado de desinfección alcanzado en el entorno.

En conclusión, se ha implementado un sistema capaz de generar rutas para la completa desinfección del espacio de trabajo utilizando técnicas bio-inspiradas, así como una aproximación de la dosis recibida en el entorno a partir de la ruta generada para garantizar dicha desinfección.

Palabras Clave: complete coverage path planning (CCPP), Glasius bio-inspired neural network (GBNN), dosis, proceso de desinfección, robot, irradiación germicida ultravioleta (UVGI).

[2]Movimiento que consiste en realizar alternativamente un desplazamiento vertical de abajo a arriba y el siguiente de arriba a abajo o viceversa.

Index of contents

Chapters **Page**

1 Introduction **1**
1.1 General framework of the project . . . 2
1.2 Motivation . . . 3
1.3 Objectives . . . 4
1.4 Structure of the document . . . 5

2 State of the art **6**
2.1 Complete Coverage Path Planning . . . 7
2.1.1 Definition . . . 7
2.1.2 Historical background . . . 7
2.1.3 Classification . . . 8
2.1.3.1 Environment decomposition approaches . . . 9
2.1.3.2 Sequence Optimization techniques . . . 16
2.1.3.3 Path Smoothing . . . 16
2.1.4 Applications . . . 16
2.2 Ultraviolet radiation for disinfection . . . 17
2.2.1 Definition . . . 17
2.2.1.1 Ultraviolet light . . . 17
2.2.1.2 Ultraviolet germicidal irradiation . . . 18
2.2.2 Historical background . . . 20
2.2.3 Classification . . . 21
2.2.4 Applications . . . 21
2.2.4.1 Common uses . . . 21
2.2.4.2 Recommendations . . . 22
2.3 Development tools . . . 23
2.3.1 MATLAB . . . 23
2.3.1.1 Robotic system toolbox . . . 23
2.3.1.2 Parallel computing toolbox . . . 23
2.3.1.3 Bioinformatics toolbox . . . 23

3 Problem analysis and modelling **24**
3.1 Problem modelling . . . 25
3.1.1 Problem context . . . 25
3.1.2 Definitions . . . 25
3.1.3 Variables . . . 26
3.1.4 Parameters . . . 27
3.1.5 Cost function . . . 27
3.2 Kinematic model of a differential drive robot . . . 29
3.2.1 Robot features . . . 29
3.2.2 Robot model . . . 30
3.2.3 Kinematic equations . . . 32
3.2.4 Robot model implementation . . . 33
3.3 Ultraviolet distribution model . . . 35
3.3.1 UV model . . . 35
3.3.2 Ultraviolet germicidal irradiation equations . . . 37
3.3.3 UV model implementation . . . 37

4 Design and development **39**
4.1 System architecture . . . 40
4.1.1 Initialization . . . 40
4.1.2 Complete coverage path planning . . . 41
4.1.3 Path planning . . . 41
4.1.4 Logger . . . 42
4.2 Map pre-processing . . . 43
4.3 Motion strategy . . . 44
4.4 Complete coverage path planning . . . 46
4.4.1 Neural network algorithm . . . 46
4.4.1.1 Environment representation . . . 46
4.4.1.2 Bio-inspired neural network . . . 46
4.4.1.3 Glasius Bio-inspired neural network . . . 49
4.4.1.4 Algorithm implementation . . . 50
4.4.2 Neural activity . . . 52
4.4.3 Path selection strategy . . . 54
4.4.4 Disinfection approximation . . . 57
4.4.4.1 UV estimation . . . 57
4.4.4.2 External input approach . . . 58
4.4.5 Deadlock detector . . . 61
4.4.5.1 Approach . . . 61
4.4.5.2 Dijkstra algorithm . . . 62

4.5 Simulation environment . . . 66
4.5.1 Simulation model . . . 66
4.5.2 Goals generator . . . 66
4.5.3 Speed controller . . . 67
4.5.4 Pure pursuit controller . . . 69
4.5.5 Disinfection simulator . . . 71
4.6 Log manager . . . 73

5 Testing and analysis of results 74
5.1 Metrics . . . 75
5.2 Test 1: Obstacle-free map . . . 78
5.2.1 General . . . 78
5.2.2 GBNN algorithm analysis . . . 78
5.2.3 Simulation analysis . . . 80
5.2.4 UV disinfection analysis . . . 81
5.3 Test 2: Narrow passage map . . . 84
5.3.1 General . . . 84
5.3.2 GBNN algorithm analysis . . . 84
5.3.3 Simulation analysis . . . 86
5.3.4 UV disinfection analysis . . . 87
5.4 Test 3: Simple office map . . . 90
5.4.1 General . . . 90
5.4.2 GBNN algorithm analysis . . . 90
5.4.3 Simulation analysis . . . 92
5.4.4 UV disinfection analysis . . . 93
5.5 Test 4: Complex office map . . . 96
5.5.1 General . . . 96
5.5.2 GBNN algorithm analysis . . . 96
5.5.3 Simulation analysis . . . 98
5.5.4 UV disinfection analysis . . . 99
5.6 Test 5: Sorting facility warehouse map . . . 102
5.6.1 General . . . 102
5.6.2 GBNN algorithm analysis . . . 102
5.6.3 Simulation analysis . . . 104
5.6.4 UV disinfection analysis . . . 105
5.7 Summary . . . 108

6 Final considerations 109
6.1 Conclusions . . . 110
6.2 Further research . . . 112

Bibliography **118**

A Environments used in simulation **120**

B Templates **123**

Index of figures

2.1 Complete Coverage Path Planning Motion Strategies 8
2.2 Random Coverage Path Planning [11] 9
2.3 Cell-based decomposition techniques 10
2.4 Morse decomposition is an exact cellular decomposition whose cells are connected components of $\mathcal{FCS}\backslash l$ [12] 10
2.5 Wavefront distance transform for the start cell (S) to the goal cell (G) [13] 11
2.6 Coverage path generated using the wavefront distance transform [13] . . . 11
2.7 Grid approximation of a given workspace [14] 12
2.8 An execution example of Spiral-STC algorithm [14] 12
2.9 Bio-inspired neural network for map building [15] 13
2.10 Rapidly-exploring random trees example [16] 13
2.11 Graph Planning for Environmental Coverage [17] 14
2.12 3D planning for environmental coverage [18] 15
2.13 3D planning for urban structure Coverage [19] 15
2.14 Ultraviolet portion of the electromagnetic spectrum[21] 17
2.15 Effect of UV on DNA structure [20] 18
2.16 Spectral energy distribution curve for germicidal action [22] 19

3.1 ZenZoe Robot [29] 29
3.2 Robot scheme on a Cartesian coordinate system 30
3.3 Robot's position and orientation 32
3.4 Differential drive kinematics model [25] 33
3.5 Operating principle for common fluorescent lamps [31] 35
3.6 Relative radiation profiles of some of the available UV-LEDs [33] 36
3.7 Propagation of light according to the inverse square law 36
3.8 UV radiation profile approximation ($550\,\mu J/cm^2$) 37

4.1 System architecture diagram 40
4.2 Binary occupancy grid example 43
4.3 Path planning with different motion strategies 44
4.4 Example of obstacle avoidance with boustrophedon motion 44

4.5 Grid region covered by the robot [37] . . . 46
4.6 Architecture of a 2D neural network with three layers [15]. . . . 48
4.7 Transfer function $g(a)$ with $\beta = 0.6$ (4.7) . . . 52
4.8 Weight connection calculation w_{ij} with $\alpha = 2$ (4.8) . . . 52
4.9 Neural activity landscape when the robot reaches $p_c(7, 2)$ . . . 53
4.10 Movement scheme for (4.10) . . . 54
4.11 Optimized template of robot obstacle avoidance example . . . 55
4.12 Optimized template of robot obstacle avoidance example . . . 56
4.13 Updating the auxiliary obstacle map while the robot is moving . . . 56
4.14 UV radiation profile approximation . . . 57
4.15 Dose estimation using different motion strategies . . . 58
4.16 example of the use of the external input . . . 58
4.17 External input landscape when robot reaches $p_c(7, 2)$ . . . 59
4.18 An alternative external input distribution . . . 60
4.19 External input comparison . . . 60
4.20 Deadlock situation with a neural approach . . . 61
4.21 Deadlock situation with Dijkstra approach . . . 62
4.22 The execution of Dijkstra's algorithm [44] . . . 63
4.23 Deadlock situations solved by neural approach (Iterations = 475) . . . 63
4.24 Deadlock situations solved by Dijkstra's algorithm (Iterations = 475) . . . 64
4.25 Path generated using Dijkstra's algorithm . . . 64
4.26 Update of neuronal activity(x) and external input(I) with Dijkstra path . 65
4.27 Model in Simulink to analyse the behaviour of the robot during workspace disinfection (model step = 0.1 s) . . . 66
4.28 Goals generated using the path achieved with the GBNN algorithm . . . 67
4.29 Example of a peak in the estimated dose . . . 68
4.30 Example of complete UV disinfection with different speed conditions . . . 69
4.31 Pure pursuit controller explanation . . . 69
4.32 Look ahead distance effects . . . 70
4.33 Look ahead distance comparison ($v = 0.1\,m/s$, $w = 3.14\,rad/s$) . . . 71
4.34 UV radiation profile approximation ($550\,\mu J/cm^2$) . . . 72

5.1 Path for Test 1, generated by complete coverage path planning . . . 78
5.2 UV % estimated during the complete coverage path planning of Test 1 . . 79
5.3 Robot path and speed for Test 1 . . . 80
5.4 D_{UV} received with spiral motion for Test 1 . . . 81
5.5 D_{UV} received with boustrophedon motion for Test 1 . . . 82
5.6 D_{UV} received categorised into thresholds for Test 1 . . . 83
5.7 Path for Test 2, generated by complete coverage path planning . . . 84

5.8 UV % estimated during the complete coverage path planning of Test 2 . . 85
5.9 Robot path and speed for Test 2 86
5.10 D_{UV} received with spiral motion for Test 2 87
5.11 D_{UV} received with boustrophedon motion for Test 2 88
5.12 D_{UV} received categorised into thresholds for Test 2 89
5.13 Path for Test 3, generated by complete coverage path planning 90
5.14 UV % estimated during the complete coverage path planning of Test 3 . . 91
5.15 Robot path and speed for Test 3 92
5.16 D_{UV} received with spiral motion for Test 3 93
5.17 D_{UV} received with boustrophedon motion for Test 3 94
5.18 D_{UV} received categorised into thresholds for Test 3 95
5.19 Path for Test 4, generated by complete coverage path planning 96
5.20 UV % estimated during the complete coverage path planning of Test 4 . . 97
5.21 Robot path and speed for Test 4 98
5.22 D_{UV} received with spiral motion for Test 4 99
5.23 D_{UV} received with boustrophedon motion for Test 4 100
5.24 D_{UV} received categorised into thresholds for Test 4 101
5.25 Path for Test 5, generated by complete coverage path planning 102
5.26 UV % estimated during the complete coverage path planning of Test 5 . . 103
5.27 Robot path and speed for Test 5 104
5.28 D_{UV} received with spiral motion for Test 5 105
5.29 D_{UV} received with boustrophedon motion for Test 5 106
5.30 D_{UV} received categorised into thresholds for Test 5 107

A.1 An empty map used in Test 1 [27 x 26] m 120
A.2 Map from [15] used in Test 2 [32 x 32] m 121
A.3 Map with a simple wall distribution used in Test 3 [27 x 26] m 121
A.4 Map with complex wall distribution used in Test 4 [52 x 41] m 122
A.5 A typical warehouse of a distribution facility used in Test 5 [60 x 60] m . . 122

B.1 Boustrophedon motion templates 123
B.2 Diagonal movement without obstacles 123
B.3 Diagonal movement templates 124
B.4 Narrowing situation that requires diagonal movement 124
B.5 Obstacle templates 125

Index of tables

2.1 UV deactivation 19
2.2 Summary of UV irradiation on Coronaviruses [23]. 20

3.1 Notation. 26
3.2 Parameters. 27
3.3 Specifications ZenZoe [29]. 30

5.1 Parameters for the analysis of the GBNN algorithm. 75
5.2 Parameters for the analysis of the simulation. 76
5.3 Parameters for the analysis of the disinfection coverage. 76
5.4 Problem Parameters 77
5.5 Time measurement for Test 1. 78
5.6 GBNN algorithm metrics for Test 1 with spiral motion. 79
5.7 Robot metrics for Test 1. 80
5.8 Disinfection metrics for Test 1. 83
5.9 Time measurement for Test 2. 84
5.10 GBNN algorithm metrics for Test 2 with spiral motion. 85
5.11 Robot metrics for Test 2. 86
5.12 Disinfection metrics for Test 2. 89
5.13 Time measurement for Test 3. 90
5.14 GBNN algorithm metrics for Test 3 with spiral motion. 91
5.15 Robot metrics for Test 3. 92
5.16 Disinfection metrics for Test 3. 95
5.17 Time measurement for Test 4. 96
5.18 GBNN algorithm metrics for Test 4 with spiral motion. 97
5.19 Robot metrics for Test 4. 98
5.20 Disinfection metrics for Test 4. 101
5.21 Time measurement for Test 5. 102
5.22 GBNN algorithm metrics for Test 5 with spiral motion. 103
5.23 Robot metrics for Test 5. 104
5.24 Disinfection metrics for Test 5. 107

Chapter 1

Introduction

1.1 General framework of the project

The project realized belongs to the Master's Degree in Systems and Control Engineering. This Master's Degree is an inter-university belonging to the Departments of Computer Science and Automation of UNED (Universidad Nacional de Educación a Distancia) and Computer Architecture and Automation of the UCM (Universidad Complutense de Madrid).

After the pandemic outbreak (COVID-19), there has been a race against time that has caused companies and health centres to adopt different disinfection mechanisms in which innovation and technology are the main characters. One of them is disinfection through the use of ultraviolet light.

UV light effectiveness for disinfection was proven in the 19th century. This technology is under development, considering the current situation. The main reason is that using UV light for room disinfection is safer than conventional chemical products.

This disinfection is achieved using robots, guaranteeing the safety of people. Many research fields influence the design of these robots. The creation of paths for the disinfection process is one of the most important.

This aspect is explored in this document. It uses strategies for coverage path planning strategies previously used in cleaning robots, agriculture, exploration, and many other applications. The proposed algorithm is based on bio-inspired techniques, considering the environment as a neural network, allowing to obtain a path that covers the entire workspace.

1.2 Motivation

The use of UV light for room disinfection is not a new phenomenon. The innovation is in doing so with autonomous robots. In this way, this combination of technologies makes it possible to increase the efficiency of disinfection.

One of the main problems when using these disinfection robots is choosing the path that covers the whole environment and guarantees its complete disinfection. The problem of finding the best path for a robot has existed since the first autonomous robots appeared. Therefore, there is a great variety of algorithms that can find the solution.

Although, once the path problem has been solved, it is necessary to guarantee the environment's disinfection. As mentioned above, this problem has been solved less often than others. It is the first time that lamps are incorporated into these autonomous vehicles. Therefore, there is not so much documentation about it. However, this problem can be approximated to the mission that cleaning robots have nowadays.

This document deals with the implementation of a complete coverage path planning algorithm focused on finding a path to disinfect all areas of a previously known map. The algorithm chosen to obtain this path has been developed using bio-inspired techniques, based on bio-inspired neural networks.

The algorithm incorporates modifications to adopt this bio-inspired approach in the application developed in this project. Approximate models are used to simulate the behaviour of UV radiation. These models provide an analysis of the radiation dose received in the workspace once the robot covers the entire path. Also, a speed controller is implemented to assure that, for the most part, the radiation applied is appropriate.

The mission of the system is to perform the complete coverage path planning for SARS-CoV-2 ultraviolet disinfection. The novelty of this work is the integration of ultraviolet disinfection lights with autonomous robots. This approach takes advantage of the background knowledge in their respective fields and goes one step further by combining them to solve the autonomous ultraviolet disinfection problem.

1.3 Objectives

The essential aim of the project is the design, development, and implementation of a system able to find the best possible path when disinfecting a known environment. The main objectives are:

- Complete coverage path algorithm that provides the best disinfection path, based on the use of neural networks.
- Speed control that can adjust the dose received in the workspace.
- Simulation environment to check the disinfection process once the robot path has been generated.

The following considerations are assumed to develop and implement the algorithms that compose the system:

- Only 2D environments are employed to generate the path for the complete coverage path planning.
- The model for UV disinfection also works in a 2D environment. The main reason for this decision is the computational increment that a 3D model would require.
- The model adopted for disinfection is an approximation. Hence, the disinfection is based purely on the inverse square law of the light. Therefore, it does not consider the shadows that the obstacles on the map could generate.
- All map elements are considered with the same absorption coefficient. Therefore, only the distance from the source of radiation influence the evaluation of the dose received.

1.4 Structure of the document

Chapter 1 introduces the project, its contextualisation, as well as the objectives that have been set to define its success.

Chapter 2 details the state of the art of the technology and techniques applied to the project. It comments on the origins and historical evolution they have experienced, the different classifications used, and the most common applications for their use.

Chapter 3 exposes the analysis and modelling of the problem. It is the base of the developed algorithms and suppose the principal base of the whole implemented system.

Chapter 4 explains the design followed and the functioning of the algorithms designed. Furthermore, it details their integration into a single system capable of satisfying the proposed objectives.

Chapter 5 analyses the tests realised to understand the results achieved and draw relevant information.

Chapter 6 concludes the project with a series of ideas that summarise the relevant aspects and investigates the possible lines of action based on the results obtained, which would give rise to new projects.

Appendix A presents all the maps used for the development of all the tests described in Chapter 6.

Appendix B details the main features of the templates used. These patterns are essential during the complete coverage path planning algorithm to generate the path.

Chapter 2

State of the art

2.1 Complete Coverage Path Planning

2.1.1 Definition

A fundamental problem of determining a collision-free path for a robot is the coverage path planning [1].

The Complete Coverage Path Planning (CCPP) approach requires the vehicle route to pass through all map areas, avoiding obstacles. This problem is fundamental in the field of robotics.

In [2], the requirements that a CCPP algorithm must meet to be successful in the coverage mission and achieve an optimal path are defined as follow:

- The vehicle must move through all the points, covering it completely.
- The vehicle must visit the area without overlapping paths.
- The vehicle must avoid any repetition of paths.
- The vehicle must evade all obstacles.
- The vehicle should preferably follow simple motion trajectories.

Nevertheless, it is not always possible to satisfy all these conditions in challenging environments.

2.1.2 Historical background

Research on path planning algorithms during the period between $1980 - 1989$ shows that best-first search algorithms (A^* search algorithm) have been used on numerous occasions to determine the optimal path. From 1990 to 1998, the proposed algorithms were based on probabilistic techniques, such as artificial potential field, road map, modified A^* search and, genetic algorithms [3].

In the year 2000, researches on the Physical A^* algorithm (PHA) took place. These researches intended to solve the problem of finding the shortest path between two points in an unknown environment. The difference with the traditional A^* algorithm is that its complexity is based and measured by travelling moving agents [4].

In 2005, a planning solution with obstacle avoidance is provided by van den Berg. In this implementation, he uses a road map considering approximately a time-optimal trajectory. Van den Berg reaches the goal configuration without colliding with any obstacles. To find this trajectory, he makes a two-level search using a depth-first search at a local level and the A^* search at the global level [5].

From 2010 onwards, there are numerous contributions to Complete Coverage Path Planning. The CCPP problem is related to the covering salesman problem, a variant of the well-known travelling salesman problem. In contrast, instead of visiting each city, an agent must visit a neighbourhood of each town.

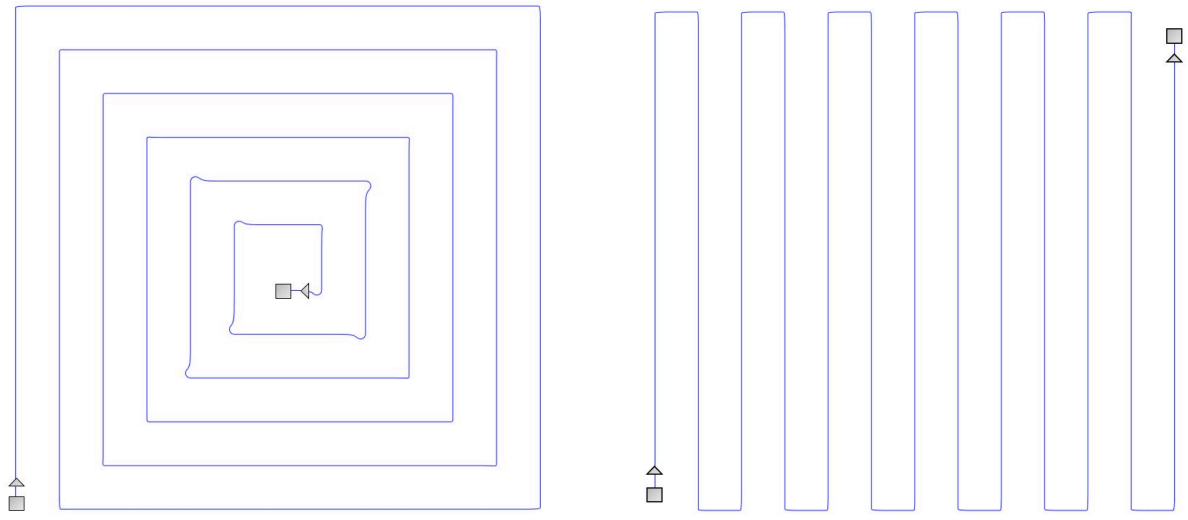
In agriculture, work has started with trapezoidal decomposition for optimal coverage path planning for arable farming [6]. Since then, all kinds of tests were carried out using different methods, such as those mentioned in Section 2.1.3. In particular, emphasizing the use of genetic algorithms [7], Morse decomposition [8] y el A^* search algorithm and all its variants [9], in a wide range of applications such as mining, disinfection or demining task.

Even though the problem of planning the coverage path has been analysed by many researchers, as reflected in [1]. Today, the existing state of the art in this field must evolve significantly in terms of robustness, precision, efficiency, and optimization.

2.1.3 Classification

Typically, this algorithm is categorized as offline and online methods. Offline algorithms use well-known information about the workspace. While the online approach uses real time measurements to take decisions [10].

There are two principal basic motion strategies used to perform the complete coverage, the square spiral motion and the boustrophedon[3] motion (Figure 2.1).



(a) Spiral motion

(b) Boustrophedon motion

Figure 2.1: Complete Coverage Path Planning Motion Strategies

[3]Movement consisting of alternately performing one vertical movement from bottom to top and the next one from top to bottom or vice versa.

This achievement could be performed using a single robot or a multi-robot tactic according to the environment's size. However, both ways have common aspects when performing the algorithm.

- The environment decomposition method determines the strategy to divide the workspace into cells.
- The sweep direction influences the path generated in terms of performance.
- The optimal backtracking mechanism ensures that no area is left unvisited, reaching the highest possible efficiency.

2.1.3.1 Environment decomposition approaches

The workspace could be cluttered with obstacles. A fundamental part of this algorithm is to divide the environment into obstacles and free space.

Random coverage path planning This methodology is widely used in the field of cleaning robots. It is based on the idea of moving the robot arbitrarily in a straight line until it finds an obstacle, then the robot turns with a random angle. This process is repeated until the entire environment is covered (Figure 2.2).

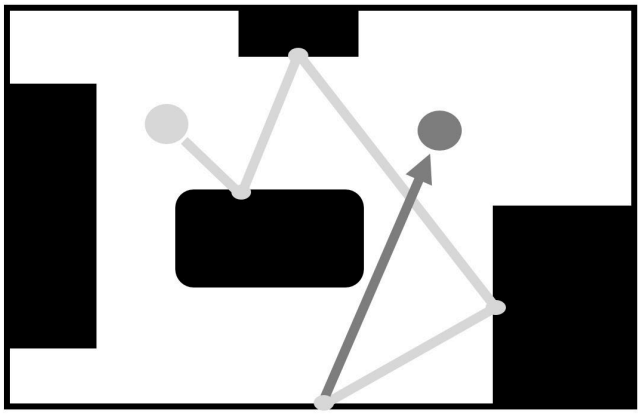
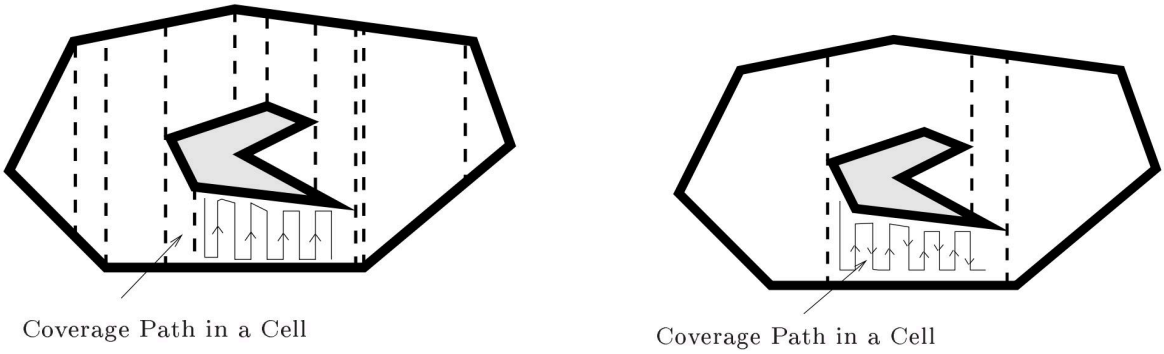


Figure 2.2: Random Coverage Path Planning [11]

Cell-based decomposition techniques The technique of decomposing the workspace into cells is one of the first approaches used. It consists of decomposing the workspace into an adjacency graph, where the cells represent nodes and the edges represent links between cells. The decomposition in cells can be catalogued in:

- Trapezoidal cell decomposition (Figure 2.3).
- Boustrophedon cell decomposition (Figure 2.3).
- Morse cell decomposition (Figure 2.4).

The trapezoidal method has limitations as it generates numerous small regions. This fact can be corrected with the boustrophedon technique, which reduces the number of areas created and allows coverage to be carried out in environments with different obstacle sizes.



(a) Trapezoidal decomposition [12]

(b) Boustrophedon decomposition [12]

Figure 2.3: Cell-based decomposition techniques

Lastly, Morse decomposition generalizes previous techniques. Also, It allows the handling of non-polygonal obstacles together with the polygonal ones.

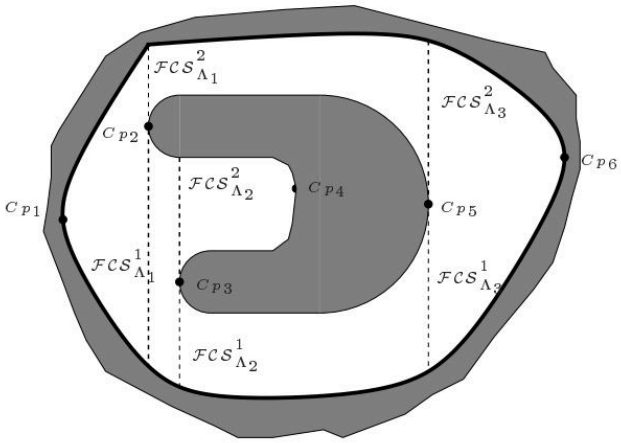


Figure 2.4: Morse decomposition is an exact cellular decomposition whose cells are connected components of $\mathcal{FCS}\backslash l$ [12]

Grid-based decomposition techniques The grid-based method represents the workspace in the form of uniform grid cells. This representation facilitates the localization and mapping operations during the coverage task. The limitation of this type of algorithm resides mainly in memory, which grows exponentially as the map size increases. Furthermore, obstacles are rendered inaccurately.

Grid-based coverage using wavefront algorithm: This method, based on grid representation, requires a start cell and destination cell. A specific value is assigned to the rest of the grid elements, depending on their proximity to the origin.

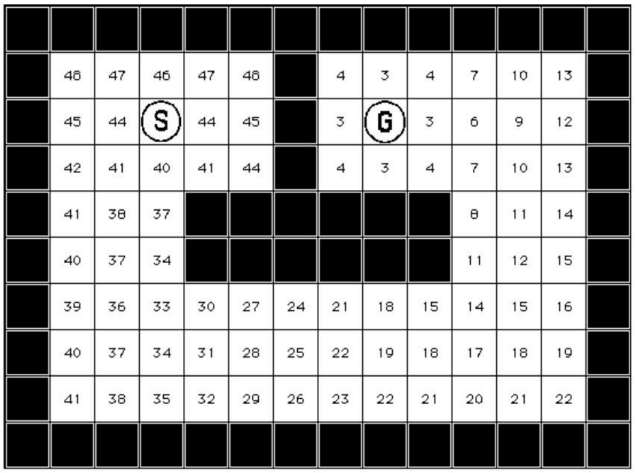


Figure 2.5: Wavefront distance transform for the start cell (S) to the goal cell (G) [13]

Once the weights of each cell are obtained, the shortest path is searched, choosing those cells with a smaller value, which allows reaching the desired goal.

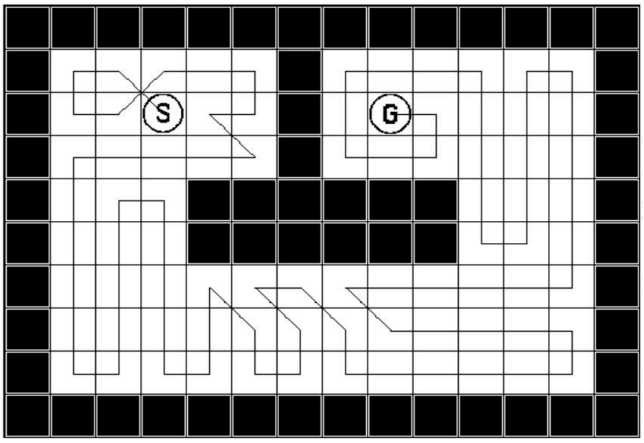


Figure 2.6: Coverage path generated using the wavefront distance transform [13]

Grid-based coverage using spanning trees algorithm: The Spiral-STC (Spanning Tree Coverage) algorithm consists of subdividing the environment into a grid map and following a spiral path. These goals are obtained by following a spanning tree of the partial grid that the algorithm incrementally builds. As is shown in Figure 2.7, the workspace is divided into mega cells. These cells are made up of small cells that match the size of the robot.

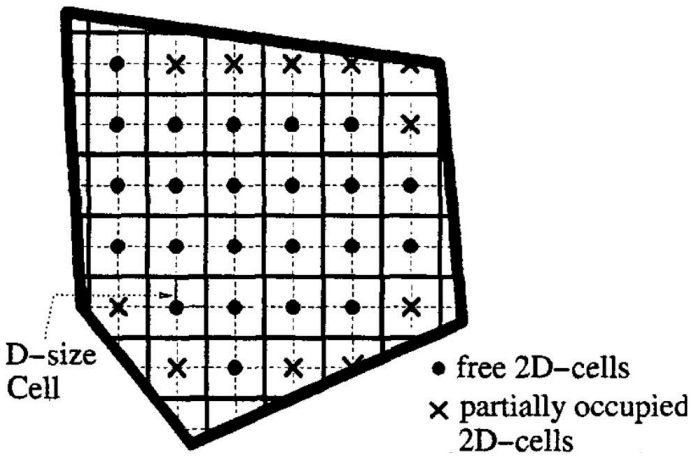


Figure 2.7: Grid approximation of a given workspace [14]

The robot must choose a movement selecting the first new mega cell in the free-space in an anti-clockwise direction to perform coverage. Then, a new spanning tree edge is grown from the current mega cell to the new one. The robot moves along one side of the spanning tree and stops when the current position has no new neighbours. Every mega cell is considered as visited if at least one of its smaller cells is covered.

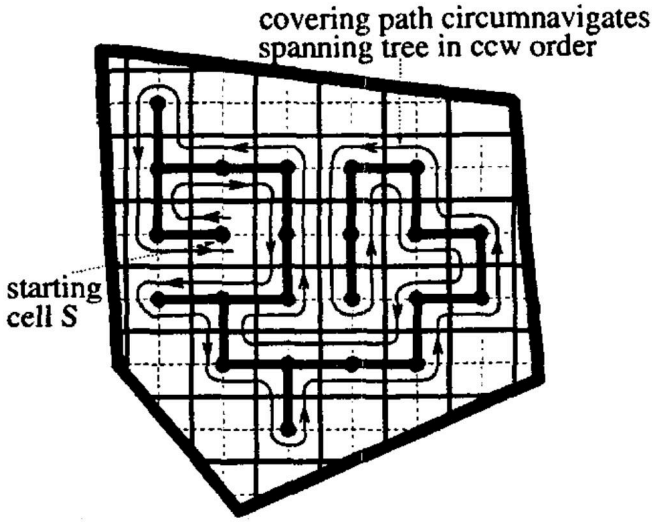


Figure 2.8: An execution example of Spiral-STC algorithm [14]

NN-based coverage on grid maps: This methodology proposes to use a neural network to discretize a space on a grid map. The size of each cell (neuron) is adjusted to the dimension of the robot. All neurons are associated with every cell in the grid. Each element has connections to its 8 immediate neighbours (Figure 2.9)

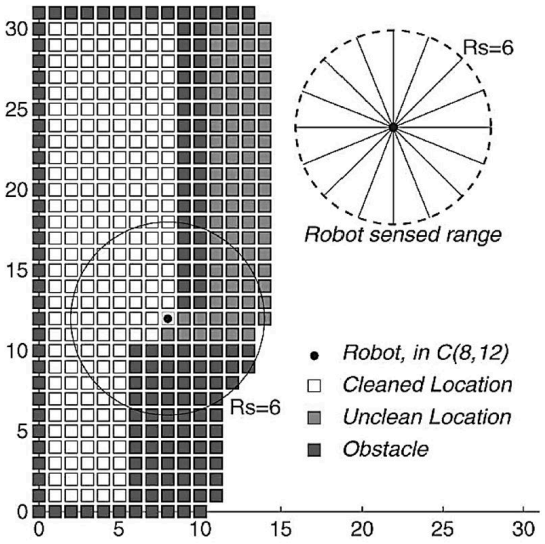
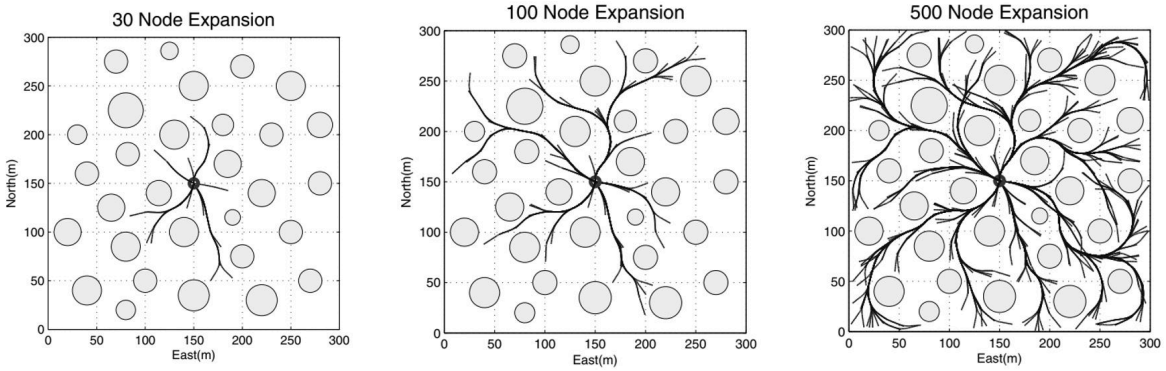


Figure 2.9: Bio-inspired neural network for map building [15]

The algorithm can be tuned in a wide range of options during the network design phase, but no learning process is used during coverage. A shunting equation determines each neuron dynamics, attracting the robot to unvisited areas, avoiding already detected obstacles, and visited regions.

Rapidly-Exploring Random Trees: This algorithm grows a tree rooted at the initial position using random samples from the search space for efficiently searching non-convex high-dimensional spaces. The tree becomes bigger in each iteration, reducing the expected distance of a random point on the tree.



(a) Paths (Nodes = 30) (b) Paths (Nodes = 100) (c) Paths (Nodes = 500)

Figure 2.10: Rapidly-exploring random trees example [16]

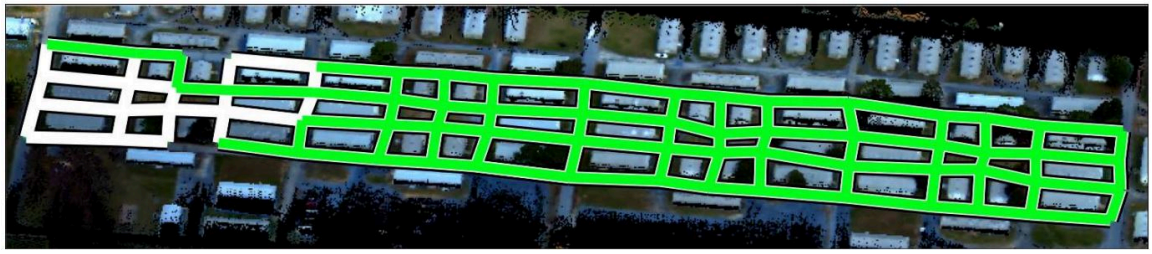
This approach is recommended for complex path planning problems that involve obstacles. For this reason, it is considered a valuable tool to use along with path planning algorithms.

Graph-based coverage techniques The graph search method is used for complicated coverage tasks, for example, environments such as a street or road network. It takes into account the following issues:

- The information on the map as a graph might be incomplete.
- The workspace presents some constraints following traffic regulations.
- It provides strategies to rethink the path when modifications are detected during real-time execution.



(a) Urban environment

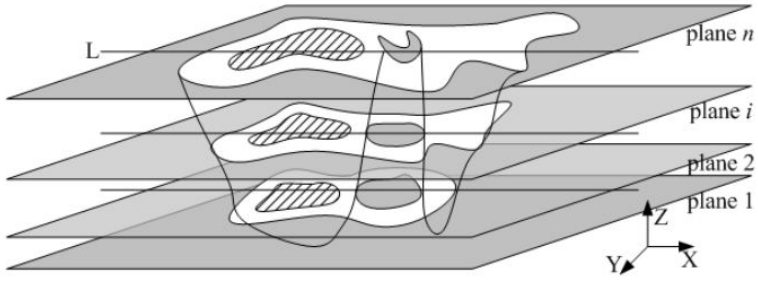


(b) Algorithm execution (green = visted / white = unvisited)

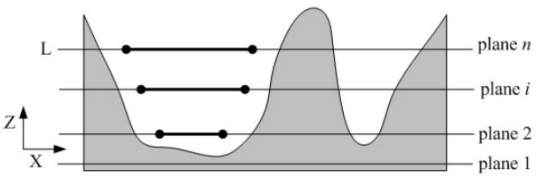
Figure 2.11: Graph Planning for Environmental Coverage [17]

3D Coverage Planning methods typically work with flat surfaces, which is perfectly valid as long as the end application allows it. However, on certain occasions, more than two dimensions may be required.

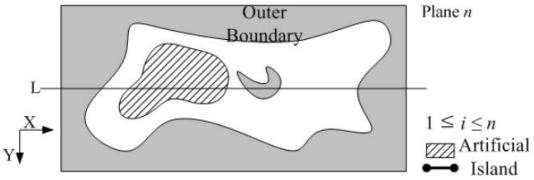
A planar coverage algorithm in successive horizontal planes can be used to perform the 3D coverage. The technique considers that the three-dimensional workspace consists of various planes at different highs. This algorithm has the advantage of decrease the total path length and the time cost for the robot to accomplish the mission.



(a) Three dimension view of the environment



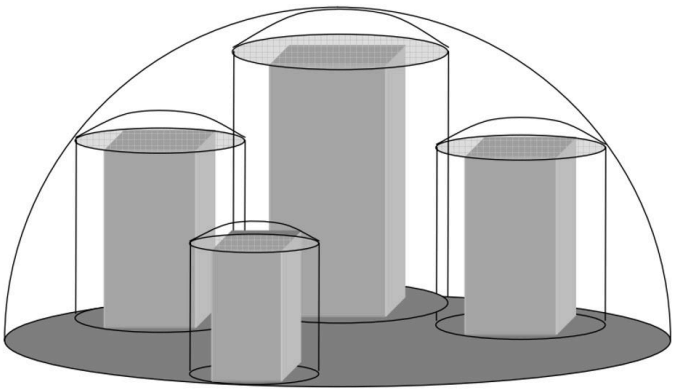
(b) Vertical sectional view



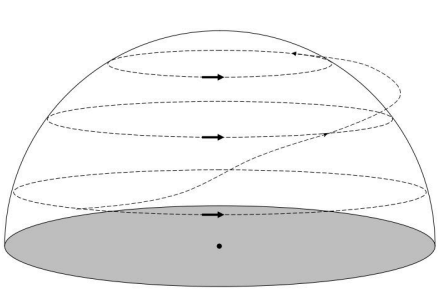
(c) Cross sectional view

Figure 2.12: 3D planning for environmental coverage [18]

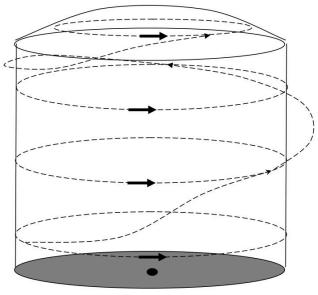
Another approach used in urban environments consists of simplifying the buildings to be covered into cylinders and spheres, so the vehicle must cover simplified flat surfaces, following a cylindrical trajectory, as shown in Figure 2.13.



(a) Simplified coverage model for an urban environment



(b) Spherical coverage model



(c) Cylindrical coverage model

Figure 2.13: 3D planning for urban structure Coverage [19]

2.1.3.2 Sequence Optimization techniques

- Greedy Algorithms
  - Depth First Search (DFS)
  - Dijkstra's Algorithm
  - A* Algorithm
  - D* Algorithm
  - Theta* Algorithm
- Dynamic Programming
- Evolutionary algorithms
  - Genetic algorithm
  - Ant Colony Optimization (ACO)
  - Particle Swarm Optimization
  - Neural Networks

2.1.3.3 Path Smoothing

- Graphical Methods
- Function-Based Methods

2.1.4 Applications

This algorithm is essential for many applications such as cleaning, painting, underwater operations, mine sweeping, lawn mowing, agriculture, tracking, and rescue missions.

- Agriculture Field Coverage
- Cleaning Task Coverage
- Humanitarian Demining
- Seabed Coverage
- Mining Application
- Arial Coverage
- Underwater Mining

2.2 Ultraviolet radiation for disinfection

2.2.1 Definition

2.2.1.1 Ultraviolet light

Ultraviolet light is a type of electromagnetic radiation, invisible to humans. The electromagnetic spectrum includes many types of radiation, each defined by a range according to wavelength (λ) or frequency (f). The wavelength for UV radiation varies from 100 nm to 400 nm, and this range of energy can be divided into 4 types as shown in 2.14 [20].

- UV-A ($320 - 400$ nm): Long-wavelength, which constitutes 98% of the ultraviolet radiation contained in solar radiation. It is the closest to the visible spectrum. Also, it is capable of crossing the ozone layer and reaching the earth's surface.
- UV-B ($280 - 320$ nm): Medium-wavelength. It is usually absorbed by the ozone present in the ozonosphere. For this reason, very little reaches the surface of the planet.
- UV-C ($200-280$ nm): Short-wavelength. It interacts with DNA and RNA producing irreparable damage. It is the germicidal spectrum, which can inactivate different types of microorganisms. They are artificially generated since they cannot penetrate the stratosphere (The range of $250 - 280$ nm is considered germicide).
- UV-V ($100 - 200$ nm): It is called vacuum ultraviolet because it can only be transmitted in a space where a vacuum exists, otherwise it is absorbed by the oxygen in the air, producing ozone.

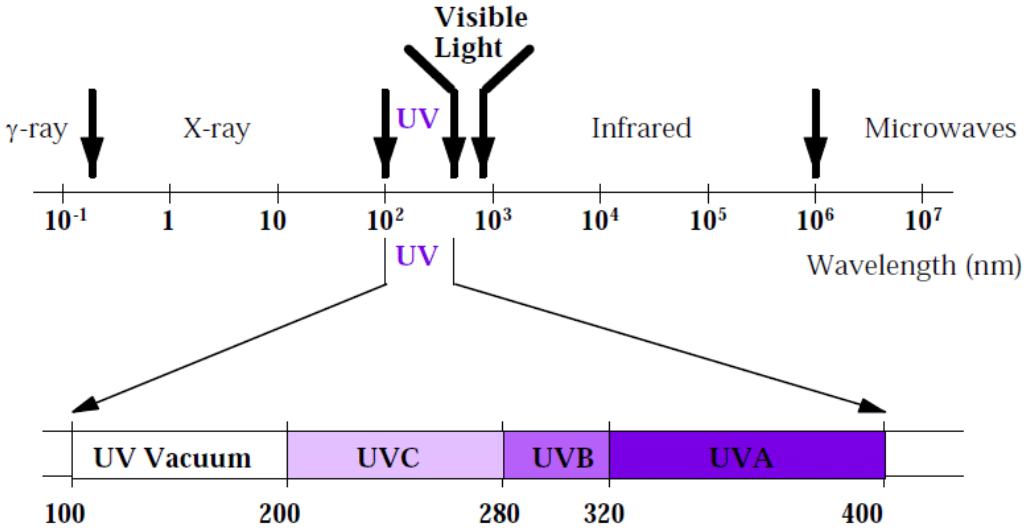


Figure 2.14: Ultraviolet portion of the electromagnetic spectrum[21]

2.2.1.2 Ultraviolet germicidal irradiation

The term UVGI refers to using UV radiation energy to inactivate fungi, bacteria, or viruses, applied at a given location. The UV light provides a fast and efficient inactivation of microorganisms through a physical process.

The UV-C radiation penetrates the cell wall of the microorganisms, being absorbed by the genetic material (DNA or RNA). This absorption generates damage and impedes microorganisms from surviving (inability to replicate or cell death).

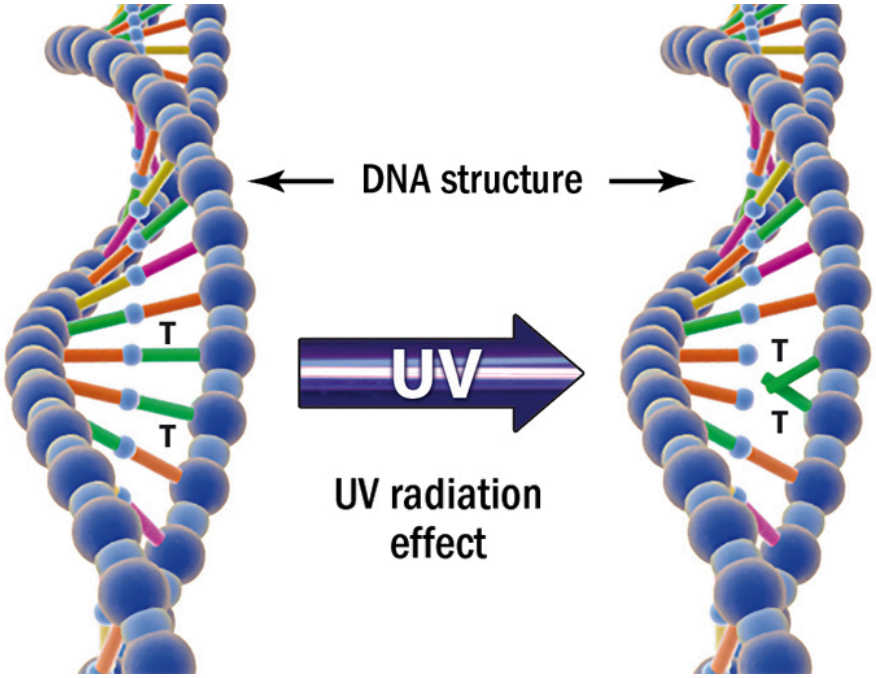


Figure 2.15: Effect of UV on DNA structure [20]

The mechanism involves the absorption of a UV photon by pyrimidine bases (main thymine), where two pyrimidine bases are together in the DNA chain. A dimer is produced that binds the two bases, causing the DNA chain to break so that when cell division occurs, DNA replication is inhibited. In the case of the virus, this is an RNA chain. So, the base affected by the photon would be the uracil, where the dimer in question would be formed (Figure 2.15) [20].

This process is due to the selective absorption of the wavelengths by particular biological molecules. DNA and RNA absorb the radiation at $260 - 265\ nm$, due to the double bond between positions 4 and 5 of the purine and pyrimidine bases [20].

The wavelength of $265\,nm$ can be considered as the optimum to achieve the disinfection of the highest number of microorganisms, since it is the peak of the DNA absorption curve, as shown in Figure 2.16. However, the disinfection occurs in a wide range of different lengths, depending on the microorganism involved [20].

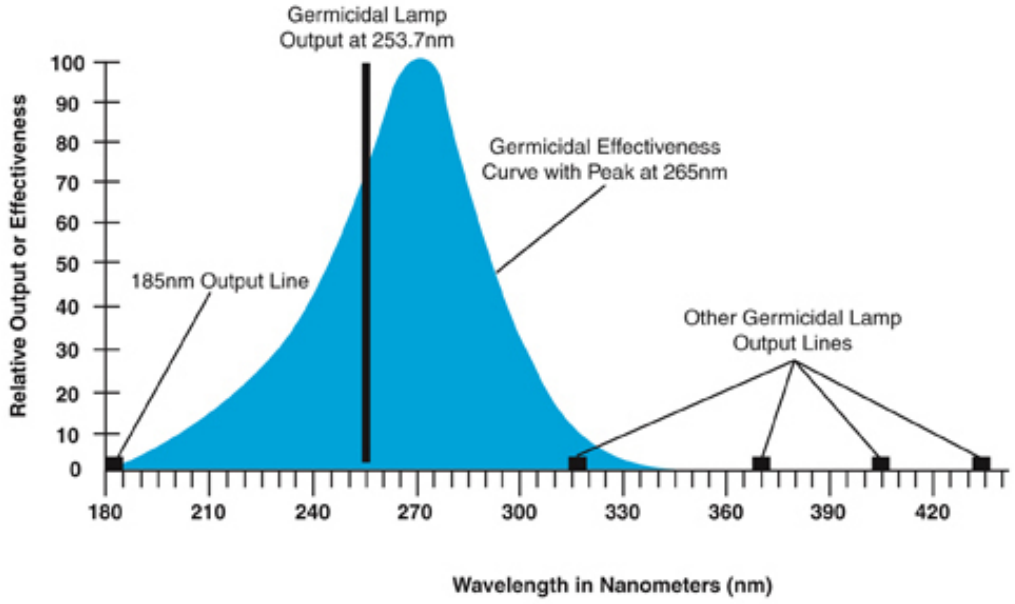


Figure 2.16: Spectral energy distribution curve for germicidal action [22]

After the microorganisms have been exposed to UV-C radiation, they suffer a reduction in their number, as shown in Table 2.1.

Table 2.1: UV deactivation

Log reduction	% **Microorganisms eliminated**	% **Reduction**
1	900.000	90%
2	990.000	99%
3	999.000	99.9%
4	999.990	99.99%
5	999.999	99.999%

The magnitude used to indicate the radiation to which microorganisms have been exposed is the ultraviolet radiation dose. The term dose describes the total amount of energy that needs to be absorbed by a microorganism. This dose is necessary to eliminate and disable its reproduction. Section 3.3 details all the calculations needed to determine this value.

Table 2.2 summarizes the results of studies that have been carried out on Coronaviruses under exposure to ultraviolet light [23]. The D_{90} value indicates the UV dose for 90% inactivation. The range of D_{90} values for coronaviruses is $7 - 241\ J/m^2$, which average is $67\ J/m^2$, should describe the ultraviolet susceptibility of the SARS-CoV-2 virus.

Table 2.2: Summary of UV irradiation on Coronaviruses [23].

Microbe	**D_{90} (J/m^2)**	**UV (km^2/J)**	**Base Pairs kb**
Coronavirus	7	0.35120	30741
Berne virus (Coronaviridae)	7	0.32100	28480
Murine Coronavirus (MHV)	15	0.15351	31335
Canine Coronavirus (CCV)	29	0.08079	29278
Murine Coronavirus (MHV)	29	0.08079	31355
SARS Coronavirus CoV-P9	40	0.05750	29829
Murine Coronavirus (MHV)	103	0.02240	31335
SARS Coronavirus (Hanoi)	134	0.01720	29751
SARS Coronavirus (Urbani)	241	0.00955	29751

2.2.2 Historical background

As reflected in [24], it was in 1877 that Downes and Blunt discovered the power of sunlight to prevent microbial growth. The light can inactivate microorganisms depending on the dose, wavelength of radiation, and the sensitivity of the specific type of microorganism.

It was in 1901 when the mercury-vapour lamp was developed as a source of artificial UV light. The practical application of UV radiation began in 1905. It was essential to work with quartz as the ideal material for its construction. In 1910, experiments began to use it as a water disinfectant in France [20].

In the United States, it was used between 1916 and 1926 to provide drinking water for ships. However, due to the low cost of chlorine disinfection and the fact that the first lamps produced were not entirely reliable, their use began to decline [20].

In 1930, Gates published the first analytical spectrum of germicidal action. Gates found maximum effectiveness at 265 nm, quite close to the wavelength used in germicidal lamps of the time (254 nm). During that decade, Wells presented the concept of infection through "droplet nuclei" evaporated droplets containing infectious organisms that can remain suspended in the air. He demonstrated the germicidal efficacy of the ultraviolet irradiation, putting it into practice by preventing large measles epidemics in day schools in suburban Philadelphia [24].

From 1940 to 1950, several attempts were made to replicate the previous success in more schoolchildren, but this was not accomplished, and the method was no longer used for air disinfection [24].

Between 1956 and 1962, Riley experimented with guinea pigs, transmitting infected air to different groups, irradiating with ultraviolet radiation only the air duct of some of them. In the end, only guinea pigs whose air duct had not been irradiated became infected. Subsequent studies allowed him and his colleagues to confirm that ultraviolet radiation effects are less effective in high humidity conditions. Finally, in 1974, Riley was able to demonstrate the potential of UVGI to reduce tuberculosis infections.

The COVID-19 pandemic situation caused by SARS-CoV-2 has invoked widespread interest in effectiveness and reliable disinfection methods to help combat the virus, including ultraviolet germicidal inactivation (UVGI). Of particular note is the development of germicidal robots that use ultraviolet light to disinfect high-risk locations. UV-C radiation energy is usually emitted by large vertical mercury or xenon lamps. Although it is not easy to estimate the emitted dose, it could cover a large part of the room in a relatively short time. Moreover, by moving autonomously around the unoccupied space, it can reach surfaces unreachable by other technology types.

2.2.3 Classification

The current types of UV disinfection technology could be summarized as follows:

- **UV-C wall luminaires**: Located on the ceiling, they act on the air located in the upper part of the rooms due to the effect of convection and flow turbulence caused by the air-conditioning and ventilation system.
- **UV-C lamps inside the ducts or air-conditioning equipment**: Due to the exposure time, the air that passes through the air-conditioning equipment is disinfected and then pushed inside the air-conditioned rooms.
- **Mobile UV-C robots**: Mostly used in hospitals and health facilities where there are rooms that can be closed for some time. These robots can be used to disinfect surfaces with UV-C in all directions.

2.2.4 Applications

2.2.4.1 Common uses

The most frequent uses of ultraviolet radiation are

- Industry.
  - Non-destructive testing.
  - Curing of adhesive substances, paints or dyes.
  - Pest control.

  - Mining.
  - Document authentication
  - Disinfection
- Medicine
  - Ultraviolet light therapies
  - Forensic science
  - Biology
  - Cosmetics
- Research
  - Laboratories

2.2.4.2 Recommendations

Caution should be taken when treating with UV-C energy because it is harmful to the skin and eyes, and overexposure of the materials to this radiation will accelerate their ageing.

Erythema or "sunburn" is a reddening of the skin, which usually appears four to eight hours after exposure to UV radiation and gradually disappears after a few days.

In the case of the eyes, the maximum sensitivity occurs in the germicidal spectrum. This radiation is strongly absorbed by the cornea and the conjunctiva. This absorption can cause keratoconjunctivitis in the case of prolonged over-extension.

When interacting with materials, non-ionising radiation is not dangerous. It does not leave any trace of residue or remaining energy. However, the degradation of the materials occurs when UV radiation penetrates. The radiation can affect the colour and surface of some organic materials, especially plastics.

2.3 Development tools

2.3.1 MATLAB

MATLAB (MATrix LABoratory) is a mathematical software that offers an integrated development environment (IDE) and its programming language. It is available for a multitude of platforms. It allows the manipulation of matrices, data representation, creation of user interfaces, and communicating with hardware devices with different program languages. Furthermore, it has a powerful graphic tools that allow obtaining good results, both in 2D and 3D.

MATLAB has additional features that expand its capabilities, namely Simulink (Simulation and Model-Based Design) and GUIDE (Graphical User Interface Development Environment). Additionally, MATLAB incorporates different toolboxes depending on the research field since it is widely used in universities and research centres.

2.3.1.1 Robotic system toolbox

Robotic System Toolbox provides tools and algorithms for designing, simulating, and testing different types of robots. This toolbox allows software development by combining the kinematic and dynamic models provided [25].

It will be fundamental to model the behaviour of the differential drive kinematic robot proposed in Section 3.2.4. Also is essential for the pure pursuit controller implemented in 4.5.4, in charge of calculating the angular velocity command that moves the robot from its current position to reach some goal.

2.3.1.2 Parallel computing toolbox

Parallel Computing Toolbox provides tools to solve problems using multicore processors, GPUs, and computer clusters. Parallel for-loops, special array types and parallelized numerical algorithms enable the software to parallelize MATLAB applications [26].

Specifically, it will be used to speed up Simulink simulations, which can be run in parallel, distributing across multicore CPUs, to reduce overall simulation time.

2.3.1.3 Bioinformatics toolbox

Bioinformatics Toolbox offers algorithms for Next Generation Sequencing (NGS), microarray analysis, mass spectrometry, and gene ontology. This tool also provides statistical techniques for detecting peaks, processing missing data, and selecting features [27].

It will be used indirectly by using the tools offered in [28] to integrate the Dijkstra's algorithm in case of deadlock situations when performing the coverage.

Chapter 3

Problem analysis and modelling

3.1 Problem modelling

3.1.1 Problem context

The main problem covered in this project is to perform the correct disinfection of the environment in the shortest possible time. First, an algorithm is created to perform the complete coverage path planning to obtain the best path. The robot must follow this route to disinfect the environment. Secondly, a speed controller is created to adjust the dose received in the environment, providing a final dose in a proper range of values. The next considerations are assumed for the problem:

- Only 2D environments are used.
- The cell under the robot is not considered as disinfected in this iteration.
- The UV radiation model is an approximation based on circular geometry.
  - The model is based only on the inverse square law of the light.
  - The model used considers that radiation can pass through walls.
- All map elements (cells) have the same absorption coefficient.

3.1.2 Definitions

Some definitions are given for a better understanding of the behaviour of the algorithms. For simplicity, the neuron at position (k, l) is also called the neuron $N(k, l)$.

Definition 1 (Complete Coverage): Given the 2D Cartesian workspace $\mathcal{W}$ and a neuron $N(k, l)$ at time t, the complete coverage at a total time T is defined as the situation in which all neurons without obstacles have been disinfected and the current neuron $N(k, l)$ has the following characteristics:

- $I(k, l) := 0$.
- $f(k, l) := 1$.

Definition 2 (Neighbourhood): The neighbourhood of a neuron $N(k, l)$ consists of some neighbouring neurons ($N(m, n)$) that enclose the last one. The neighbourhood of a neuron $N(k, l)$ in a shunting neural network is defined in (3.1), where r is the number of circles that enclose the neuron (Figure 4.6).

$$\mathcal{N}_r(k, l) \quad = \quad N(m, n) | max|m - k|, |n - l| \leq r, 1 \leq m \leq N_x, 1 \leq n \leq N_y \qquad (3.1)$$

The neighbouring neuron $N(m, n)$ close to $N(k, l)$ has the following property:

$$m \in \{k - 1, k, k + 1\} \text{ and } n \in \{l - 1, l, l + 1\}$$

Definition 3 (Deadlock): Given the 2D Cartesian workspace $\mathcal{W}$ and a neuron $N(k,l)$, a deadlock situation occurs when the neighbour neuron has been previously visited, is an obstacle, or has less neural activity. In this case, the following property remains:

$$if\ \exists(m,n),\ \forall m\ \in\ \{k-1,k,k+1\}\ y\ n\ \in\ \{l-1,l,l+1\},$$
$$m \neq k_c, n \neq l_c,\ 1 \leq m \leq N_x, 1 \leq n \leq N_y,$$
$$\text{s.t. } ((f(m,n) == 1\ or\ 2)\ or\ x(m,n) < x(k_c,l_c))\ then\ f(m,n) := 3$$

3.1.3 Variables

The variables that define the problem are listed in Table 3.1.

Table 3.1: Notation.

Variable	Definition
(k_c, l_c)	Current position, (k_c, l_c).
(k_p, l_p)	Previous position, (k_p, l_p).
(k_n, l_n)	Next position (k_n, l_n) where activity is maximal.
$N(k,l)$	Non-visited position, $(k,l), 1 \leq k \leq N_x, 1 \leq l \leq N_y$.
$\bar{N}(k,l)$	Visited position, $(k,l), 1 \leq k \leq N_x, 1 \leq l \leq N_y$.
$\hat{N}(k,l)$	Non-obstacle position, $(k,l), 1 \leq k \leq N_x, 1 \leq l \leq N_y$.
$\check{N}(k,l)$	Obstacle position, $(k,l), 1 \leq k \leq N_x, 1 \leq l \leq N_y$.
$N_r(k,l)$	Neighbouring neurons within the receptive field r_0.
$N_n(k,l)$	Eight neighbouring neurons of the central neuron $N(k,l)$.
$x(k,l)$	Neural activity at position, (k,l).
$I(k,l)$	External input to neuron $N(k,l)$.
$f(k,l)$	State flag in neuron $N(k,l)$.
n_{GBNN}	Iterations used in algorithm for complete coverage path planning
n_{goals}	Total number of goals.
d_e	The estimated distance that the robot will cover
$UV_{\%}(k,l)$	Approximate disinfection percentage in neuron $N(k,l)$.
t_{robot}	Time used by the robot to finish the disinfection coverage.
d_{robot}	The total distance that the robot will cover
v_{robot}	Linear velocity
w_{robot}	Angular velocity
p_{robot}	Robot position $p\{x\, y\, \theta\}$ expressed in Cartesians coordinates
$D_{\hat{N}}$	Dose received in the cells without obstacles
D_N	$0.9D_{SARS-CoV-2} < D_{total} < 1.1D_{SARS-CoV-2}$

3.1.4 Parameters

The parameters that determine the problem are listed in Table 3.2.

Table 3.2: Parameters.

Parameter	Definition
$\mathcal{N}(k,l)$	Set of discretized workspace, $(k,l), 1 \leq k \leq N_x, 1 \leq l \leq N_y$.
N_x	Map size on the x-axis
N_y	Map size on the y-axis
$r0$	Receptive field radius of a 2D neural network
α	Positive constant for the weight connection calculation w_{ij}
β	Positive constant for the transfer function $g(a)$
c	Positive constant for the path selection strategy
dt	Simulation environment time step
ρ	Speed controller fitting parameter
$D_{SARS-CoV2}$	SARS-CoV-2 dose
D_{UV}	Lamp radiation
d_{LA}	Look-ahead distance for pure pursuit controller

3.1.5 Cost function

The function (3.2) seeks to minimize the number of iterations performed in the GBNN algorithm. A lower number of iterations means a lower number of visited cells. Consequently, the distance covered by the robot will also be reduced. Function 3.2 is s.t to (3.3) and (3.4):

$$f_1 \quad = \quad MIN(n_{GBNN}) \tag{3.2}$$

The constraint (3.3) implies that the robot is not allowed to visit any cell that is considered an obstacle.

$$if\, \bar{N}(k,l) = 1,\, then\, f(k,l) \quad \neq \quad 2 \;\; \forall\, k,l \in \mathcal{N} \tag{3.3}$$

The constraint (3.4) implies that only 10% of the total number of cells considered free space can be left without "disinfecting". Non-disinfected, according to the external input model I represented in Figure 4.16. This constraint maximizes the number of cells that receive the maximum dose.

$$\sum(I(k,l) > 0) \quad \leq \quad 0.1 \sum(\hat{N}(k,l)) \;\; \forall\, k,l \in \mathcal{N} \tag{3.4}$$

Once the path has been generated, with the different goals to where the robot must go, the main purpose of the speed controller is detailed in the functions (3.5) and (3.6).

The function (3.5) seeks to minimize the time needed by the robot to complete the disinfection process. Besides saving time, it also means saving energy.

$$f_2 = MIN(t_{robot}) \tag{3.5}$$

The function (3.6) seeks to maximize the percentage of cells with a ultraviolet radiation dose in the expected range. It should be noted that if it is possible to maximize the areas that are within the desired range:

- The zones that exceed the threshold are minimized, avoiding the degradation of the environment's materials and unnecessary energy consumption.
- The zones that do not reach the expected threshold are minimized, ensuring a more trustworthy environment disinfection.

$$f_3 = MAX(D_N) \tag{3.6}$$

3.2 Kinematic model of a differential drive robot

3.2.1 Robot features

ZenZoe is the disinfection robot selected for this project. Its characteristics are used in the model that has been used to test the developed algorithms. This technology promises an efficient solution in terms of disinfection based on the proven principle of Ultraviolet Germicidal Irradiation (UVGI), being able to deactivate most germs. Specifically, it has a disinfection efficiency of 99.99% in the case of SARS-CoV-2.

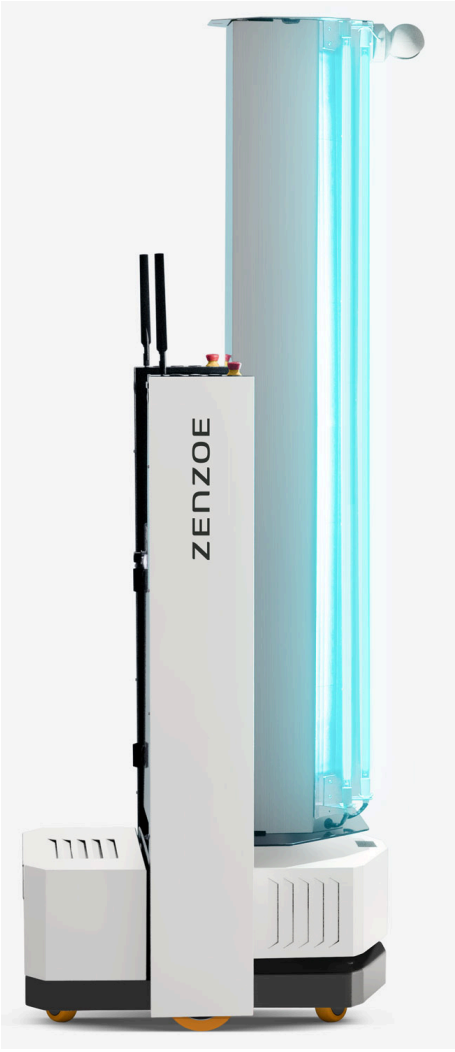


Figure 3.1: ZenZoe Robot [29]

All relevant information about this robot can be found on the manufacturer's website. Some of its characteristics, collected in Table 3.3, are used to define the robot model in charge of testing the implemented software.

Table 3.3: Specifications ZenZoe [29].

Configuration	4 UV-C Lamps
UV-C at 100h	192 W
Dimension(LxWxH)	$740 \times 662 \times 1.970$ mm
Wheel Radius	≈ 50 mm
Track Width	≈ 590 mm
Working Voltage	24 $V(dc)$
Environment	Indoor applications
Safety	Emergency stop and safety sensors
Linear Speed	$v_{max} \approx 1,2$ m/s
Angular Speed	$\omega_{max} \approx 3.1$ m/s
Battery Charging Time	60 minutes
Disinfection Time	$6-8$ minutes / 25 m^2 room
Autonomy	$2,5 \sim 2,7$ hours
Navigation	SLAM / Free navigation
Connectivity	4G or Wi-Fi
Configuration	8 UV-C Lamps

3.2.2 Robot model

According to Figure 3.2, given that the position $p\{x\, y\, \theta\}$ is expressed in Cartesians coordinates of the inertial frame $\{X_I, Y_I\}$, the relationship between the inertial frame $\dot{\varepsilon}_I$ and the robot $\dot{\varepsilon}_R$ is defined in equation (3.7).

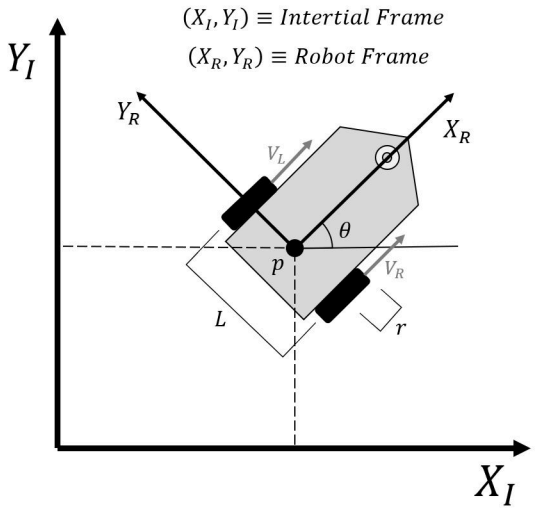


Figure 3.2: Robot scheme on a Cartesian coordinate system

$$\dot{\varepsilon}_R = R(\theta)\dot{\varepsilon}_I = R(\theta)\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} \tag{3.7}$$

Where $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ are the velocity component and $R(\theta)$ is the rotation matrix, defined in the expression (3.8).

$$R(\theta) = \begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3.8}$$

Each wheel of the differential drive robot is driven independently, so navigation is achieved by speed changes. The third wheel is a caster wheel needed for the stability of the mobile robot. All wheels participate in the robot's motion imposing constraints [30]. A series of hypotheses must be taken into account to simplify the behaviour of the robot:

- The robot works on a flat surface, ideally without friction.
- The axles of the wheels are not perpendicular to the ground where it moves.
- It moves only by force exerted by the rotating movement of the wheels.
- The wheels do not slide, expressed by the non-holonomic constraint defined in (3.9).

$$\dot{x}_r \sin\theta - \dot{y}_r \cos\theta = 0 \tag{3.9}$$

- It is a solid rigid mechanism without flexible parts.

The position and orientation of the robot, considering the global system, are the state variables that the robot wants to reach. For this reason, it must have exact knowledge about the speed on each wheel, as the linear speed of the robot can be defined as the average speed of the wheels. According to Figure 3.2 the following equations collect these constraints:

$$V_R = rW_R \tag{3.10}$$

$$V_L = rW_L \tag{3.11}$$

$$V = \frac{(V_R + V_L)}{2} \tag{3.12}$$

Where V_R and W_R are the linear and angular velocity, respectively, for the right-hand wheel. While the terms V_L and W_L represent the same but concerning the left wheel. The r value corresponds to the radius of the robot's wheel.

On the other hand, the robot angular velocity will be the speed difference of each wheel. As described in the expression (3.13) it is taking into account the length between the wheels.

$$W = \frac{(V_R - V_L)}{L} \tag{3.13}$$

3.2.3 Kinematic equations

A differential drive system is represented in Figure 3.3 to understand the mechanical behaviour of the robot. The kinematic model has been defined at some arbitrary position p_c some distance away from the goal position p_n both defined in the global inertial frame.

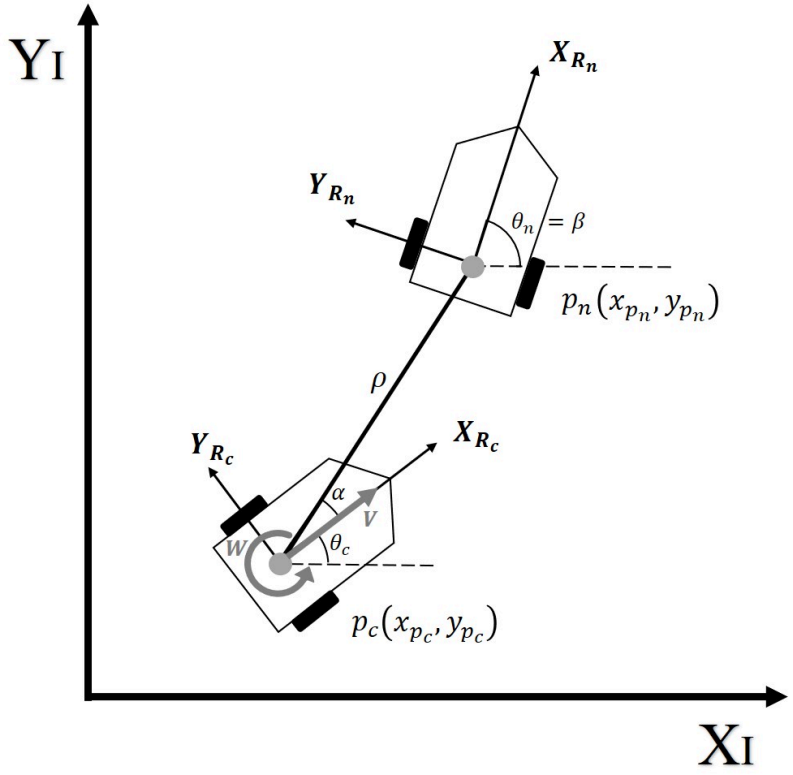


Figure 3.3: Robot's position and orientation

The combined action of linear velocity V and angular velocity W governs the coordinate system of the robot. The kinematic model is defined by the following formulas:

$$\dot{x} = V \cos\theta_c \tag{3.14}$$

$$\dot{y} = V \sin\theta_c \tag{3.15}$$

$$\dot{\theta} = W \tag{3.16}$$

Where the components of V along X-Y axes are represented by (3.14) and (3.15), respectively. The value θ is measured considering the inertial reference frame. Also, the robot position can be represented in terms of polar coordinates. Considering error distance $\rho \neq 0$:

$$\dot{\rho} = -V \cos \alpha \tag{3.17}$$

$$\dot{\beta} = \frac{V \sin \alpha}{\rho} \tag{3.18}$$

$$\dot{\theta} = W \tag{3.19}$$

Where the distance $\rho = \sqrt{(x_{p_n} - x_{p_c})^2 + (y_{p_n} - y_{p_c})^2}$ and the angle $\alpha = \beta - \theta$, measured between robot axes $[X_{R_c}, Y_{R_c}]$. Finally, conclude into the following equations:

$$\dot{\rho} = -V \cos \alpha \tag{3.20}$$

$$\dot{\alpha} = -W + \frac{V \sin \alpha}{\rho} \tag{3.21}$$

$$\dot{\beta} = \frac{V \sin \alpha}{\rho} \tag{3.22}$$

3.2.4 Robot model implementation

Thanks to the Robotic System Toolbox [25] it is possible to model a differential traction vehicle to simulate the simplified dynamics of the robot. This model can be approximated by a vehicle with a single fixed axle and wheels separated by a specific track width.

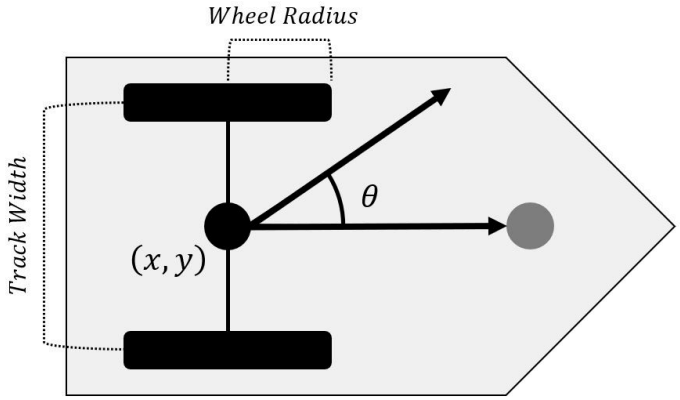


Figure 3.4: Differential drive kinematics model [25]

It is important to note the following aspects.

- Wheels are independent.
- Speed and heading are defined from the centre of the axis.
- Vehicle state is defined as a three-element vector, $[x(m), y(m), \theta(rad)]$.

To compute the vehicle motion using this differential drive kinematic model offered by the toolbox, the following parameters, obtained by Table 3.3, are needed.

- Vehicle inputs:
  - Vehicle speed (m/s).
  - Heading angular velocity (rad/s).
- Parameters:
  - Wheel radius (m): The wheel radius of the vehicle.
  - Wheel speed range (m/s): The vehicle speed range provides the minimum and maximum vehicle speeds.
  - Track width (m): The vehicle track width refers to the distance between the wheels, or the axle length.
  - Initial state (x_0, y_0, θ_0): The initial position of the robot.
- Vehicle outputs:
  - Robot state (m): Current position and orientation of the vehicle, specified as a $[x\ y\ \theta]$ vector.
  - Derivatives of state $(m/s); (rad/s)$: current linear and angular velocities of the vehicle specified as a $[\dot{x}\ \dot{y}\ \dot{\theta}]$ vector.

3.3 Ultraviolet distribution model

3.3.1 UV model

To measure the disinfection dose provided by the robot during the mission, a UV distribution model is needed. The germicidal low-pressure UV lamp is quite similar to the ordinary fluorescent lamp (Figure 3.5), but the tube does not contain fluorescent phosphorus and is made of quartz glass, instead of using borosilicate glass, to make the UV light come out of the lamp.

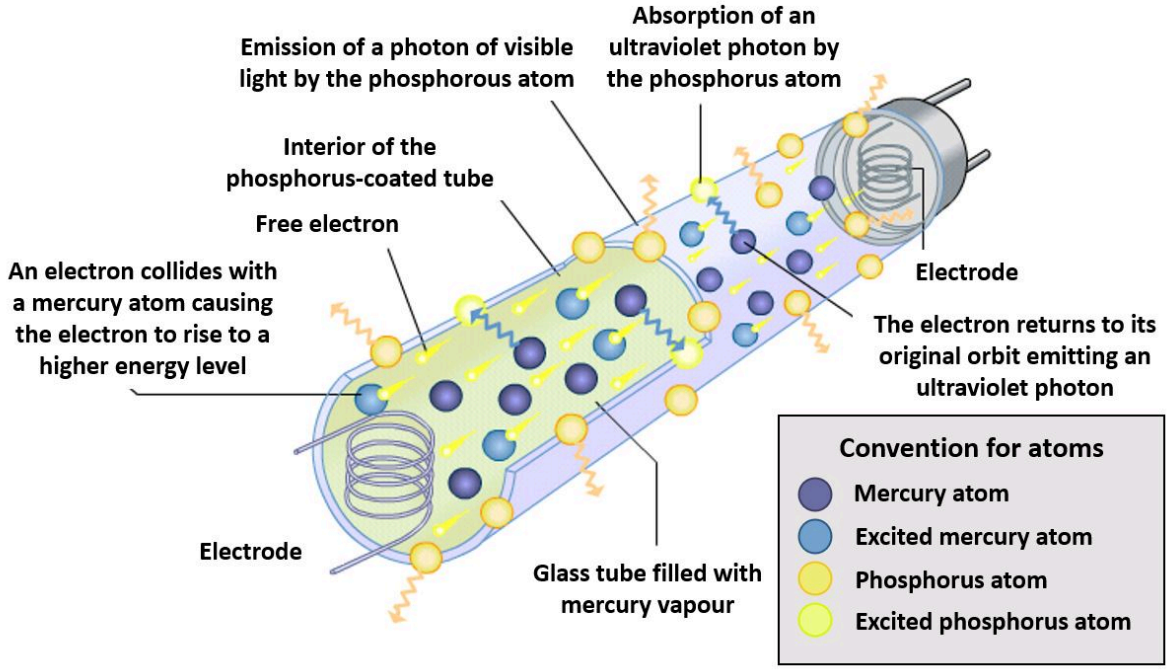


Figure 3.5: Operating principle for common fluorescent lamps [31]

The UV source is a fused silica quartz tube, where the inert gas with which the tube is filled provides the primary discharge and action required to excite and vaporise the minuscule amounts of mercury inside [32].

An alternative to mercury-based UV lamps is Light Emitting Diodes (LEDs). These devices are semiconductor devices that can emit radiation of different wavelengths, including the germicidal range.

UV-LEDs are typically characterised by their radiation profile, their radiant power and their peak wavelength. In contrast, the radiation profile of low-pressure mercury lamps is axisymmetric. The spectrum of the emitted radiation is constant, so the lamp power can be calculated by measuring the irradiation from a single point at a specific distance from the lamp in a steady-state using an appropriate formula. This procedure is not directly applicable to determining the power of LED-UV [33].

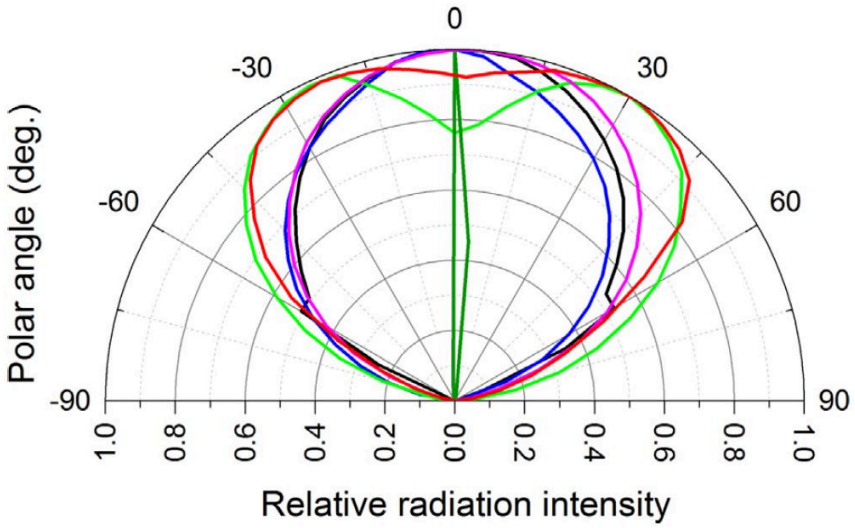


Figure 3.6: Relative radiation profiles of some of the available UV-LEDs [33]

The different radiation patterns of UV-LEDs are described in Figure 3.6. The irradiation at the centreline and a specific distance from a UV-LED will differ, depending on its radiation profile.

The factors that have the highest impact on disinfection are humidity and distance from the source of radiation. The germicidal dose is the product of time and intensity. High intensities over a short period and low intensities over a long period are fundamentally the same in their lethal action on bacteria. The law of the inverse square applies to germicidal ultraviolet light. Therefore, lethal power decreases as increases the lamp distance (Figure 2.16).

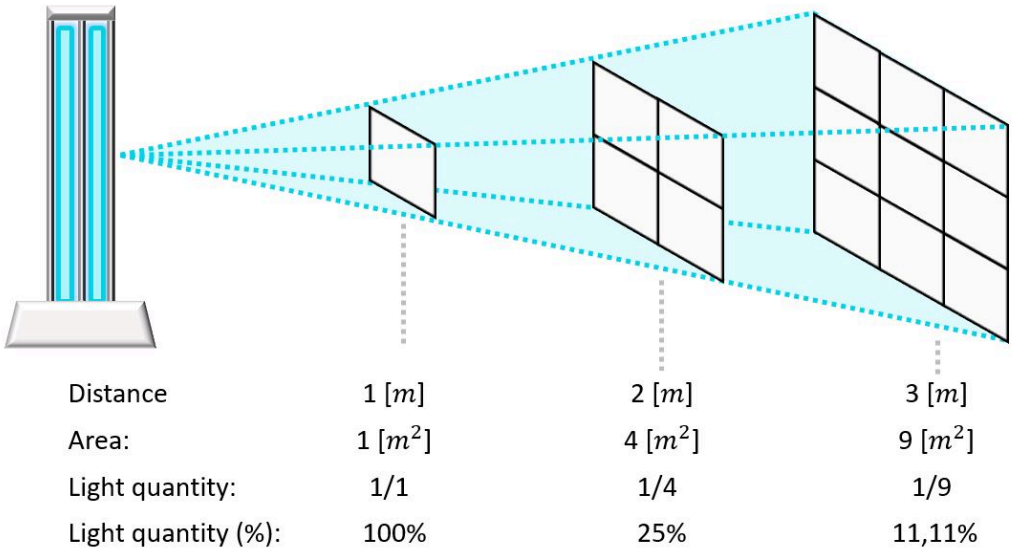


Figure 3.7: Propagation of light according to the inverse square law

3.3.2 Ultraviolet germicidal irradiation equations

When a microorganism is exposed to UV-C radiation, cell nuclei are modified due to photolytic processes. As a result, cell division and reproduction are prevented. The relationship between dose and destruction of a target microorganism can be summarized as follows [32]:

$$\frac{N(t)}{N_0} = e^{-kIt} = e^{-kD} \tag{3.23}$$

Where:

- N = Number of microorganisms
- k = Constant value
- I = UV intensity
- t = time
- D_{UV} = dose

The UV dose (energy per unit area) is the product of UV intensity (power per unit area) and exposure time (expressed in seconds) [34]:

$$D_{UV}\left(\frac{mWs}{cm^2}; \frac{mJ}{cm^2}; \frac{Ws}{m^2}; \frac{J}{m^2}\right) = I(mW/cm^2) \cdot t(s) \tag{3.24}$$

3.3.3 UV model implementation

Section 3.3.1 is assumed to be a reference to model this behaviour. From this information, as simple an approximation as possible has been chosen.

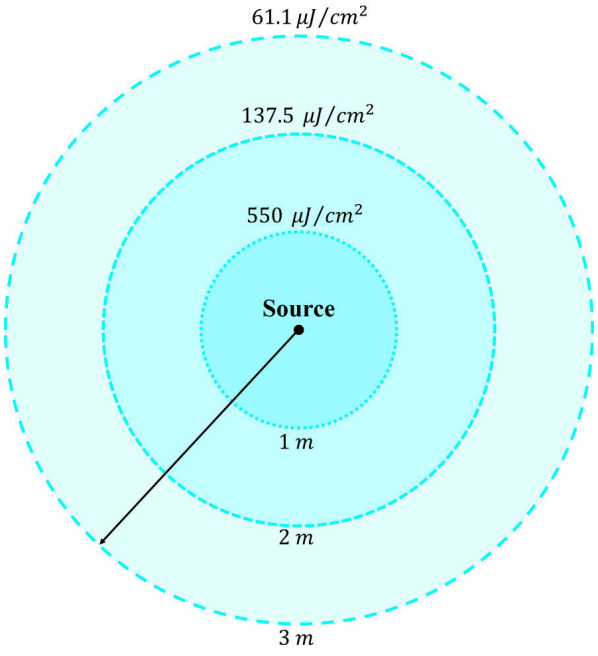


Figure 3.8: UV radiation profile approximation ($550\,\mu J/cm^2$)

Therefore, without taking into account the 8 UV-C lamps configuration of the robot, with a specific radiation profile, and considering that work is being done in a 2D environment, a circular radiation profile has been determined that only follows the inverse square law described in Section 3.7.

The power of each lamp that composes this hypothetical circular configuration is approximately $0.55\,mW/cm^2$. The lamp used emits a radiation of $550\,\mu J/cm^2$ at a distance of 1 meter in one second [35], and it allows generating a radiation profile as shown in Figure 3.8. It must be taken into account that the area under the robot is not receiving radiation, and therefore its disinfection is not considered.

Although the value extracted from [35] is considered to be representative, the data comes from the tests performed in [36]. During these tests, UV-C type radiation was used $(260 - 285\,nm)$, with irradiation of $0.55\,mW/cm^2$ at the point of exposure $(0.5\,m)$.

It is essential to consider that anything between the light source and the target will block UV-C rays, resulting in shaded areas. Similarly, in a room, there may be specks of dust that cause these shadows, which may mean that not all microorganisms are receiving the expected dose. However, this effect is not considered in the proposed model, and it is assumed that radiation can pass through walls.

Chapter 4

Design and development

4.1 System architecture

The system proposed to generate the path that satisfies the complete coverage path planning in terms of UV disinfection, as well as the simulation environment that provides a way to check this disinfection is detailed in Figure 4.1.

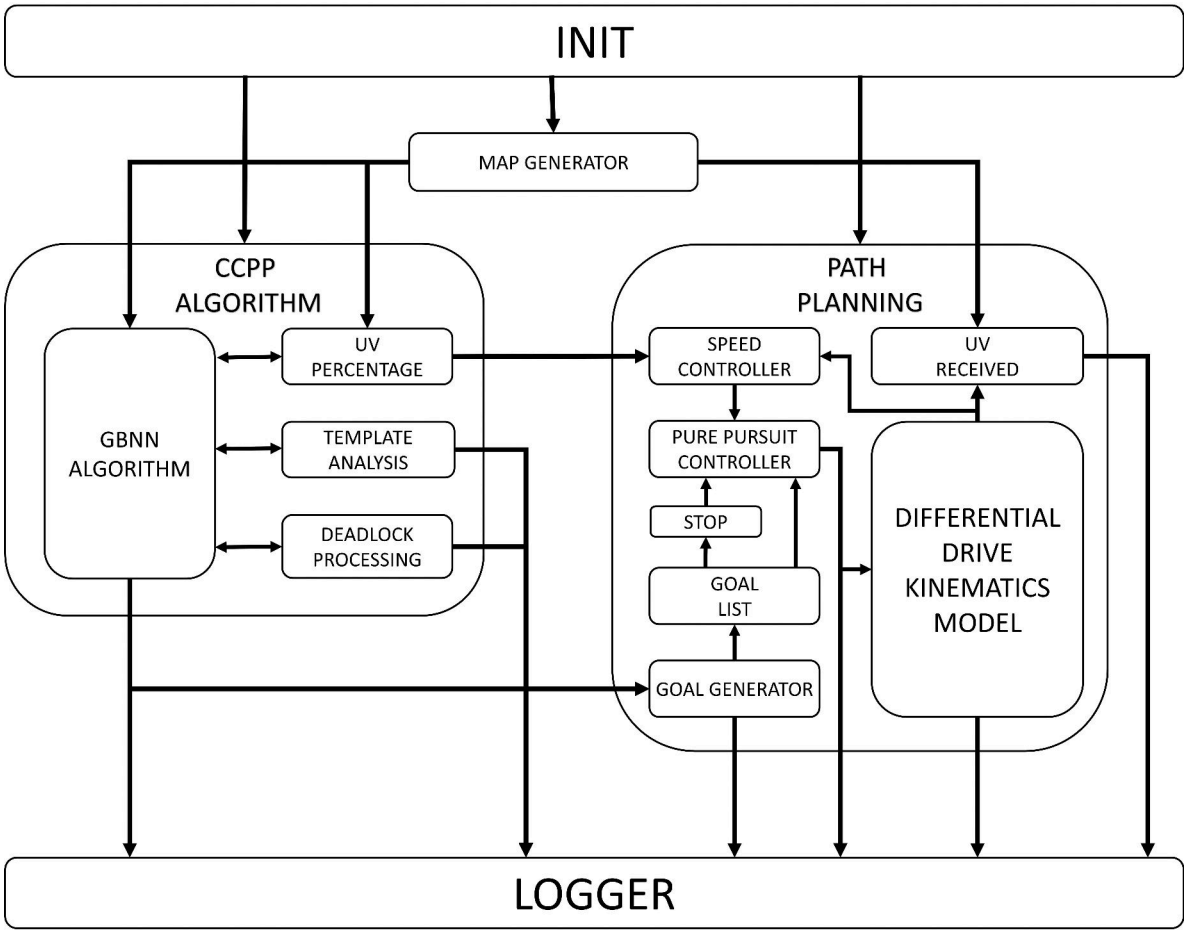


Figure 4.1: System architecture diagram

4.1.1 Initialization

The first block of system **INIT** is in charge of initialising all the parameters and variables that affect each of the possible algorithms. Moreover, the initial position of the robot is determined. During the tests, the initial position will be position $p(2, 2, \frac{\pi}{2})$.

As part of this initialization, the motion strategy to be used throughout the algorithm is also chosen. Only the techniques described in Section 4.3 are addressed. Furthermore, inside the **MAP GENERATOR** block, the steps described in section 1 are made to define the map to be disinfected.

4.1.2 Complete coverage path planning

The previous information is relevant to the main modules of the system. The first module of this system can be found in the **CCPP ALGORITHM** block. Inside this function, the complete coverage path planning algorithm is performed. The behaviour of this algorithm is described in Section 4.4.

The Glasius Bio-inspired neural network algorithm is performed inside this block, based on the knowledge developed in Section 4.4. Specifically, the algorithm of Section 4.4.1.3 is implemented in block **GBNN ALGORITHM** with a series of modifications to apply it to disinfection using ultraviolet radiation.

In Section 4.4.2, the function that determines the neuronal activity of each map cell is defined. This activity is vital in determining the different weights between each of the neuronal connections. These weights will influence the route the robot will take. As well as the neuronal information, the orientation of the robot will be considered. Those cells that require a minor orientation changes are weighted more (Section 4.4.3).

A series of templates are used to increase the algorithm's efficiency. The **TEMPLATE ANALYSIS** block is responsible for deciding if any specific extra movements are required, depending on the current situation of the robot. These movement strategies are described in the Section 4.4.3 and in the Appendix B.

As part of the algorithm, a percentage estimate of the dose that the environment will receive is obtained. This information is produced in the **UV PERCENTAGE** block. To calculate this percentage, the approximation described in section 4.4.4.1 is used. However, this disinfection approximation is not used to update the neuronal activity in the GBNN algorithm. For this task, the criteria described in Section 4.4.4.2 are used.

Lastly, to guarantee the efficiency of the complete coverage path planning, the **DEADLOCK PROCESSING** block is incorporated. This function allows the algorithm to detect if the robot will get stuck, and through Dijkstra's algorithm, it establishes an escape route to new areas without disinfecting.

4.1.3 Path planning

The second module of the system is called **PATH PLANNING**. It consists of a simulation environment capable of verifying the robot behavior when it follows the route proposed in the previous module. Its main components are detailed in the Section 4.5.

The first step is to generate the goal list from the generated path using complete coverage path planning. This algorithm generates a list of the cells to visit, but in **GOAL GENERATOR** block, the most representative cells of that list (goals) are extracted.

This extraction is explained in the Section 4.5.2. With this list, inside the **GOAL LIST** block, it is checked when the robot reaches these positions. Once it has reached some goal, the next point is commanded so that the controller can act accordingly. Once all the goals have been visited, the **STOP** block will be in charge of stopping the simulation.

The **SPEED CONTROLLER** block receives the information generated in the **UV PERCENTAGE** block to determine the speed that the robot should have. Section 4.5.3 details how the speed increases or decreases, depending on the percentage estimate calculated in the Section 4.4.4.1.

Once the robot speed is known, the **PURE PURSUIT CONTROLLER** block is in charge of calculating the robot's angular speed to reach the previously defined goals. In Section 4.5.4 the controller behaviour can be analysed.

The robot model used is based on the equations described in Section 3.2 , considering the properties described in Table 3.3. This model is created in the **DIFFERENTIAL DRIVE KINEMATICS MODEL** block based on what is explained in Section 3.2.4.

The **UV RECEIVED** block updates the dose received in the workspace with each robot movement. As for the model used in Section 4.4.4.1, in Section 4.5.5, the approximate model used to analyse the dose received in the environment is defined, including all its limitations.

4.1.4 Logger

To conclude, the **LOGGER** block is used to record the different events that occur during the algorithm execution. The main messages used are detailed in Section 4.6.

4.2 Map pre-processing

In the proposed algorithm, the working environment is considered to be known, so the exact location of all obstacles on the map is known. All the proposed maps have the following characteristics:

- The workspace is previously known and will not experiment with any modifications during the execution of the algorithm.
- The workspace is defined as a 2D environment (cells 1x1 [m]).
- The limits of the workspace are approximated by a wall, to consider a closed space.
- The starting point is always $[2, 2]$.

To start working it is necessary to carry out a pretreatment on the map:

1. Load the map.
2. Invert logical values.
3. Consider the boundaries of the workspace as obstacles.
4. Obtain the binary occupancy.
5. Rotate the map $-90°$.
6. Set the expected minimum UV radiation limit.

There are different types of maps to test the algorithm. When none is selected, an empty closed space is created. All the environments used are listed in Appendix A.

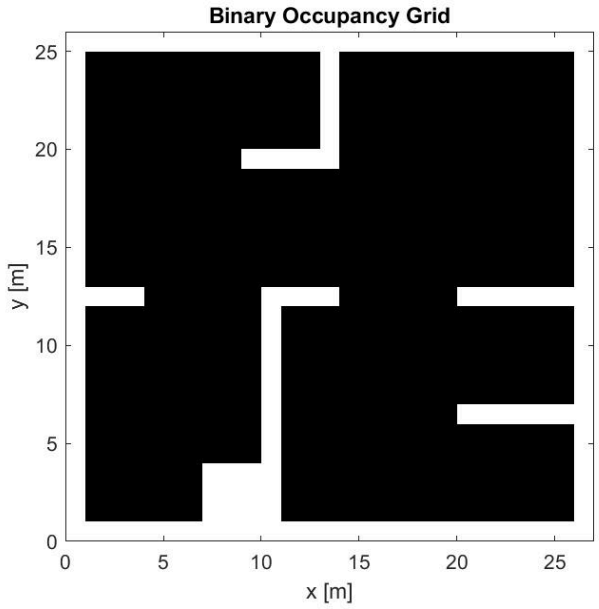
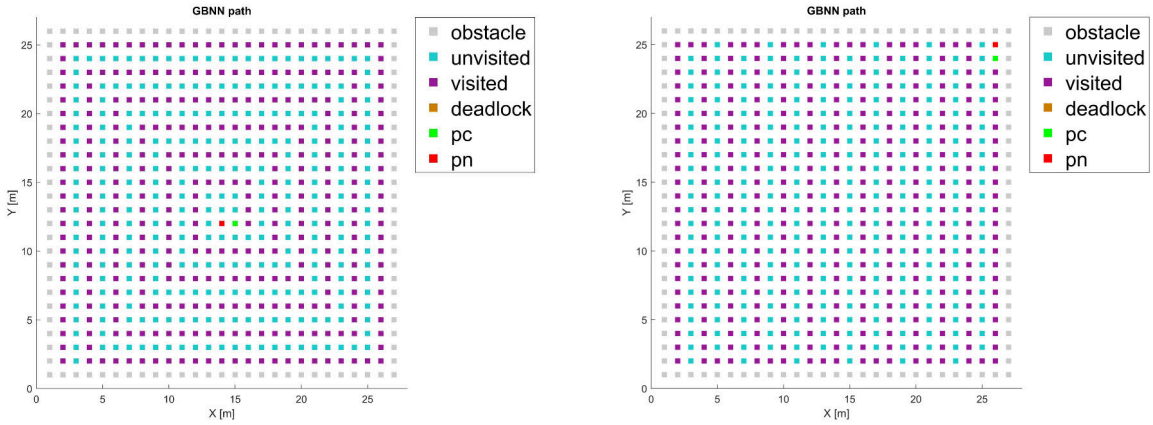


Figure 4.2: Binary occupancy grid example

4.3 Motion strategy

The generated path will have some characteristics or others according to the motion strategy used to perform the complete coverage navigation. The two described in Figure 2.1 are taking as a reference.



(a) Spiral motion

(b) Boustrophedon motion

Figure 4.3: Path planning with different motion strategies

One of the most used movement patterns in the literature is the boustrophedon movements, also known as the "snake" pattern [1]. However, complete coverage is not always achieved, and there are overlaps in previously covered cells. Therefore, as described in Section 4.4.3, when implementing templates, the boustrophedon technique will also avoid obstacles, mimicking the spiral movement before continuing with its zig-zag.

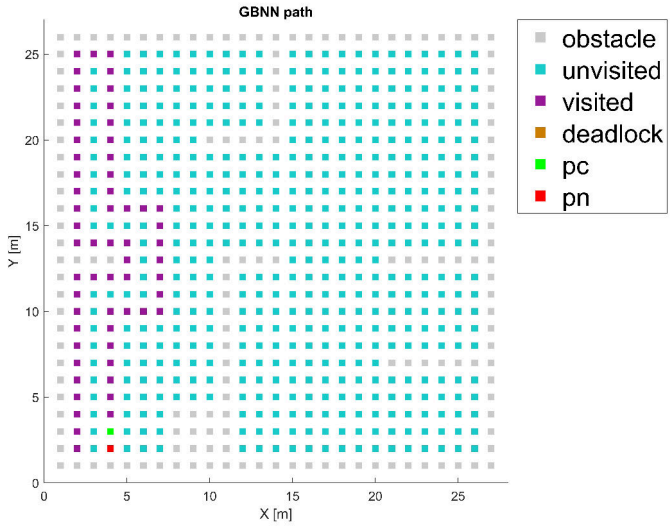


Figure 4.4: Example of obstacle avoidance with boustrophedon motion

In most cases, when the environment is full of obstacles, it is impossible to achieve a single pass coverage. Further techniques are required to complete the disinfection of the workspace. These improvements over the main algorithm will be described in sections 4.4.3 and 4.4.5.

The fact of leaving the cells without visiting occurs with both movement strategies and is a characteristic feature of the proposed algorithm. However, a distinction between visited cells and disinfected cells must be made, as it will be explained in more detail in Section 4.4.4.

Visited cells are those through which the robot will pass to carry out complete coverage. However, as disinfection is performed with ultraviolet radiation, some cells will be disinfected without being visited.

4.4 Complete coverage path planning

4.4.1 Neural network algorithm

4.4.1.1 Environment representation

In general, this methodology uses a 2D Cartesian workspace discretized into square or rectangle cells with the following characteristics (Figure 4.5):

- The diagonal length of each cell should be equal to the robot sweeping radius.
- The robot sweeping radius is the size of the robot footprint.
- Each cell has a height h and a width l (if the cell is square: $h = l$).

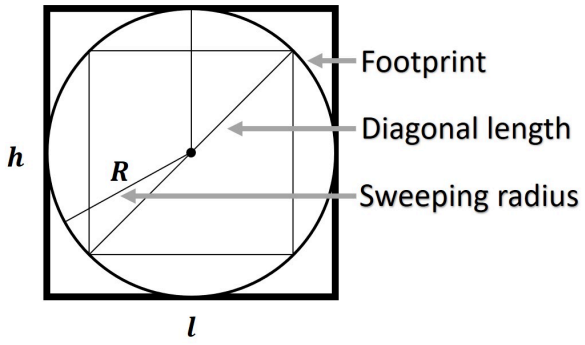


Figure 4.5: Grid region covered by the robot [37]

In this case, according to the Section 3.3, the cells have a square shape with $h = l = 1$ m, regardless of the size of the robot (whose radius must be ≤ 50 cm). The main reason for taking these values is to simulate the decrease in radiation intensity. However, the decision to have a resolution in meters instead of a smaller unit is because, as will be seen in Section 4.4.1.2, the computational complexity increases with the number of cells.

4.4.1.2 Bio-inspired neural network

The first model of a biological neural system, using electrical elements, was proposed by Hodgkin and Huxley [38]. This approach considers the dynamics voltage over the membrane V_m, which is represented by equation (4.1).

$$C_m \frac{dV_m}{dt} = -(V_r + V_m)g_r + (V_e - V_m)g_e - (V_s + V_m)g_s \tag{4.1}$$

Considering:

- C_m: membrane capacitance.
- V_s, V_e, V_r: saturation potentials.
- g_s, g_e, g_r: synaptic and resting conductances.

If the $C_m = 1$ is fixed and considers $x_i = V_r + V_m$, $A = g_r$, $B = V_e + V_r$, $D = V_s - V_r$, ${S_i}^e = g_e$, y ${S_i}^i = g_s$ the expression (4.1) will be as in (4.2) proposed by [38].

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i){S_i}^e - (D + x_i){S_i}^i \tag{4.2}$$

The value x_i represents the neural activity (membrane activity) of neuron i in the neural network. The constants A, B y D are not negatives. They define the passive decay rate and the lower and upper limits of the neural activity of x_i. Also, ${S_i}^e$ and ${S_i}^i$ describe the excitatory and inhibitory inputs of the neuron.

The collision-free robot trajectory is generated from dynamic activity in the neural network and the robot's previous position, ensuring that all regions are visited during a smooth and continuous trip, avoiding turns.

The proposed topology is represented in a 2-D Cartesian workspace $\mathcal{W}$. Neuron position i in the state space S of the neural network is defined by the vector $q_i \in R^2$. The excitatory input comes from unvisited zones. On the other hand, the inhibitory input results in visited areas and obstacles.

Each neuron has a local lateral connection to its neighbouring neurons, which make up a subset R_i in S. This subset is defined as a receptive field of the neuron i. This neuron responds exclusively to the stimulus in that field. Therefore, the neural dynamics of i is represented by the shunting equation (4.3).

$$\frac{dx_i}{dt} = -Ax_i + \left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+ \right) - (D + x_i){I_i}^- \tag{4.3}$$

The index k corresponds to the number of neural connections of neuron i to its neighbouring, in the receptive field R_i. The external input I_i is defined as:

- $I_i = +E$, if the cell is not disinfected.
- $I_i = 0$, if the cell is disinfected.
- $I_i = -E$, if the cell is an obstacle.

It considers $E \gg B$ as a high positive constant to ensure that non-disinfected regions can reach the peak. Meanwhile, obstacles can remain in the valley in the neural activity landscape.

$[I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+$ and $[I_i]^-$ correspond to the excitatory and inhibitory inputs ${S_i}^e$ and ${S_i}^i$ in the expression (4.2). Considering $[a]^+$ and $[a]^-$ as a linear-above threshold functions described by $[a]^+ = \max[a, 0]$ or $[a]^- = \max[-a, 0]$, respectively. The connection weight $w_{ij} = f(|q_i - q_j|)$, between neuron i and j, is defined as the Euclidean distance between vectors q_i and q_j in the state space.

As described in [15], the function $f(a)$, defined in (4.4), is monotonically decreasing. The constants μ and r_0 are positive, so each neuron has only local lateral connections in a small area $[0, r_0]$.

$$f(a) = \begin{cases} \mu/a, & 0 \leq a < r_0 \\ 0, & a \geq r_0 \end{cases} \tag{4.4}$$

The weights $w_i j$ are symmetric, so $w_{ij} = w_{ji}$, as described in Figure 4.6. The receptive field of the current neuron i is denoted by a circle with a radius of r_0.

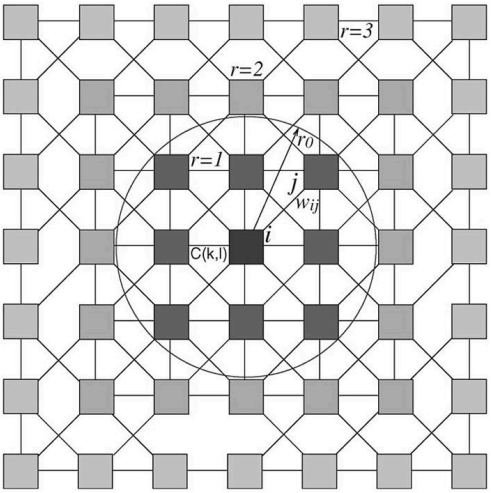


Figure 4.6: Architecture of a 2D neural network with three layers [15].

The shunting equation proposed in (4.3) ensures that the positive neural activity can propagate to the entire state space, remaining the negative activity locally. As a result, regions without disinfection attract the robot, while obstacles push the robot away to avoid collisions. It considers that neural activity changes dynamically due to the varying external inputs and the internal activity propagation between neurons.

The proposed neural network system is stable, with neural activity x_i limited by the interval $[-D, B]$ [39]. The convergence and stability of the model can be proved with the Lyapunov stability theory [40], substituting some terms in (4.2 - 4.3) the result is described in (4.5). The rigorous resolution is shown in [41], concluding that neural network dynamics ensures the convergence to an equilibrium state of the system.

$$\frac{dzi}{dt} = a_i(z_i)\left(b_i(z_i) - \sum_{j=1}^{N} c_{ij} d_j(z_j)\right) \tag{4.5}$$

The shunting function (4.3) is combined with the previous robot position to create the path that guarantees complete coverage of the workspace. At first, when $t = 1$, neural activity for all neurons is 0. After that, the activity is modified based on the influence of external inputs. The algorithm ends when the neural network reaches a steady state.

4.4.1.3 Glasius Bio-inspired neural network

The bio-inspired neural network algorithm (BINN) is improved with the new equations proposed in [42] to reduce the amount of computation and enhance the algorithm's operation speed. These changes provide benefits from adaptive learning, increasing the efficiency of planning the complete coverage path.

GBNN is a Hopfield-type discrete-time neural network algorithm, whose main difference with the BINN algorithm lies in the shunting equation 4.3 used to obtain the neural activity. In this case, the dynamic characteristics of each neuron activity are described by the resulting law (4.6) defined in [42].

$$x_i(t+1) = g\left(\sum_{j=1}^{k} w_{ij}[x_j(t)]^+ + I_i\right) \tag{4.6}$$

The function $g(a)$ is chosen as:

$$g(a) = \begin{cases} -1, & a < 0 \\ \beta a, & 0 \leq a < 1,\ \beta > 0 \\ 1, & a \geq 1 \end{cases} \tag{4.7}$$

Where $x_i(t+1)$ is the neural activity of the neuron i at $t+1$, x_j is the neural activity of the neighbour neuron j at t, which is laterally connected with the neuron i.

The index k correspond to the neural connections of neuron i to its neighbouring, in the receptive field R_i. Considering $[a]^+$ as a linear-above threshold functions described by $[a]^+ = \max[a, 0]$, because only positive neural activities can influence other neurons and propagate globally, while negative neural activities only have local effect. The connection weight w_{ij} between neuron i and j, is defined in (4.8), where $|q_i - q_j|$ is the Euclidean distance between vectors q_i and q_j in the state space and α is a positive constant.

$$w_{ij} = \begin{cases} e^{-\alpha|q_i - q_j|^2}, & 0 < |q_i - q_j| \leq R_i \\ 0, & |q_i - q_j| > R_i \end{cases} \tag{4.8}$$

The external input I_i is defined as:

- $I_i = +E$, if the cell is not disinfected.
- $I_i = 0$, if the cell is disinfected.
- $I_i = -E$, if the cell is an obstacle.

Accordingly, for the non-disinfected regions ($I_i = +E$), where $\sum_{j=1}^{k} w_{ij}[x_j(t)]^+ + E \gg 1$, the neural activity is 1. Therefore, for the obstacle regions ($I_i = -E$), where $\sum_{j=1}^{k} w_{ij}[x_j(t)]^+ - E < 0$, guarantee that the neural activity for an obstacle is -1. When the neuron is covered, its external input I_i is equal to 0.

4.4.1.4 Algorithm implementation

This section details the behaviour of the algorithm to develop the map performing to complete coverage. This work is based on the models detailed in Section 4.4.1.2 and 4.4.1.3. With a clear influence of the work proposed in [15] and [42]. With this influence, an algorithm has been created to disinfect the entire environment using UV radiation.

The main task of the robot is to guarantee the disinfection of all regions while dynamically is building a path. The algorithm starts from an initialization phase, followed by the construction of the path for complete coverage.

A series of flags $f(k,l)$ are defined for the neural position (k,l), indicating the current state of the cell grid:

$$f(k,l) \;=\; \begin{cases} 0, & \text{si } if \text{ } it \text{ } is \text{ } unvisited \\ 1, & \text{si } if \text{ } it \text{ } is \text{ } visited \\ 2, & \text{si } if \text{ } it \text{ } is \text{ } obstacles \\ 3, & \text{si } if \text{ } it \text{ } is \text{ } deadlock \end{cases} \tag{4.9}$$

It is important to note that the flags refer to visited and unvisited cells, regardless of the disinfection status of the area. It will be later, through the neuronal activity of the environment, when this degree of disinfection is considered by the algorithm.

In the beginning, the robot has prior knowledge of all obstacles in the environment. The algorithm consists of two essential phases, a first initialization stage before the coverage process. The notation used during algorithms definition is shown in Table 3.1.

Algorithm 1: Initialization

1. set $k_c := k_0$; set $l_c := l_0$, where (k_0, l_0) is a initial point.
   // Set initial point to a current neuron.
2. set $k_p := k_0$; set $l_p := l_0$, where (k_0, l_0) is a initial point.
   // Set previous point to a current neuron.
3. set $\forall\ 1 \leq k \leq N_x, 1 \leq l \leq N_y$:
   $f(k,l) := 0, \forall\ N(k,l) \neq obstacle$;
   $f(k,l) := 2, \forall\ N(k,l) = obstacle$;
   // Initialize flag state of each neuron.
4. set $\forall\ 1 \leq k \leq N_x, 1 \leq l \leq N_y$:
   $I(k,l) := 100, \forall\ N(k,l) \neq obstacle$;
   $I(k,l) := -100, \forall\ N(k,l) = obstacle$;
   // Initialize external input of each neuron.
5. set $\forall\ 1 \leq k \leq N_x, 1 \leq l \leq N_y$:
   $x(k,l) := 1, \forall\ N(k,l) \neq obstacle$;
   $x(k,l) := -1, \forall\ N(k,l) = obstacle$;
   // Initialize neural activity of each neuron.

In the initialization phase, all the parameters and variables are loaded in their initial conditions, as shown in Algorithm 1.

Algorithm 2: Complete Coverage Path Planning

1. Compute neural activity by (4.6).
2. $\mathcal{N}(k,l) = N(m,n)|m \in (k-1,k,k+1)\ y\ n \in (l-1,l,l+1)$
   // Scan neighbouring neurons (two methods)
   $(k_n,l_n) \in \{\mathcal{N}_r|m \in (k-1,k,k+1)\ y\ n \in (l-1,l,l+1)$
   if Template == false $then$ $(k_n,l_n) = arg\max(x(m,n) + cy(m,n))$.
   // Find the next neighbouring neuron with maximal neural activity.
   //// Neural Motion
   $else$ $(k_n,l_n) = (k_{template}, l_{template})$.
   // Find the next neighbouring neuron selected by the template.
   //// Template Motion
3. Update position index
   - set $k_c := k_n$; $l_c := l_n$.
     // Set current neuron to neighbouring neuron
   - set $k_p := k_c$; $l_p := l_c$.
     // Set previous neuron to current neuron
4. $if\ \exists I(g,h) \in \mathcal{N}_n(m,n),\ \forall g\ \in\ \{k-1,k,k+1\}\ y\ h\ \in\ \{l-1,l,l+1\}$,
   s.t $f(g,h) = 1$
   - set $I(g,h) := 0$.
     // Set current neuron to neighbouring neuron
5. Update $UV_{\%}$ matrix
6. $if\ \exists(m,n),\ \forall m\ \in\ \{k-1,k,k+1\}\ y\ n\ \in\ \{l-1,l,l+1\}$,
   $m \neq k_c, n \neq l_c,\ 1 \leq m \leq N_x, 1 \leq n \leq N_y$, s.t.
   - $f(m,n) = 1,\ or$
   - $f(m,n) = 2,\ or$
   - $x(m,n) < x(k_c,l_c)$

   $then f(m,n) := 3$.
   // Search for deadlock situations:
7. if the deadlock situation remains or the robot is expected to get stuck
   - Apply Dijkstra's algorithm to find a way to escape.
   - Set $I(k_{Dijksta}, l_{Dijkstra}) := 100\ \forall$ points generated with Dijkstra's algorithm.
   - Set $x(k_{Dijksta}, l_{Dijkstra}) := 1\ \forall$ points generated with Dijkstra's algorithm.
8. Go to 2 (Until disinfection is complete).

In the coverage phase shown in Algorithm 2, the external inputs I_i are the source of information about the environment. The robot moves to non-disinfected areas, until all regions have been disinfected. Non-disinfected regions attract the robot, while impediments only have a local impact that prevents accidents.

The computational complexity depends linearly on the state space size of the neural network, which is proportional to the map size. The number of neurons used is $M = N_x \times N_y$. If the workspace is $N \times N$, there are N^2 neurons, and each neuron has 8 local neural connections. As a result, the total neural connections are $8 \times N^2$, so the complexity of the proposed algorithm is $O(N^2)$.

4.4.2 Neural activity

As described in Section 4.4.1.3, the equations proposed in [42] are used to improve the performance of the BINN algorithm. To obtain the neural activity $x_i(t+1)$, the dynamic characteristics are obtained from the expression (4.6).

The term $\left(\sum_{j=1}^{k} w_{ij}[x_j(t)]^+ + I_i\right)$ comes from the shunting equation defined in (4.3). However, g is used to saturate the activity, and its rules are expressed in (4.7).

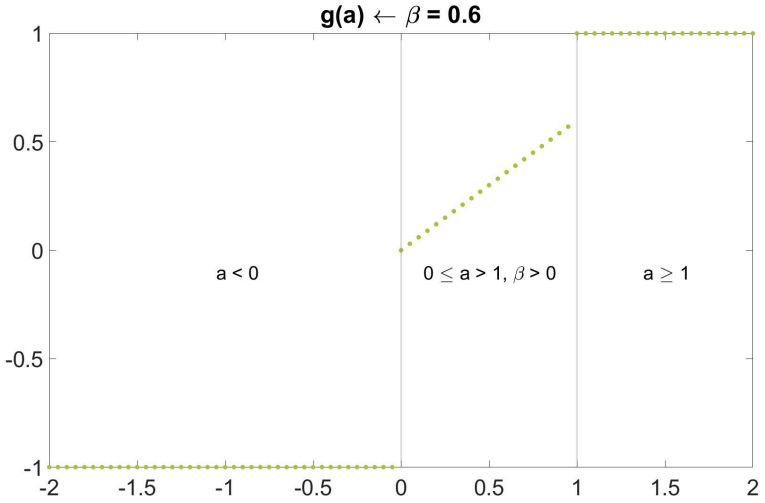


Figure 4.7: Transfer function $g(a)$ with $\beta = 0.6$ (4.7)

Likewise, to calculate the weight of the connection w_{ij} the expression (4.8) is used instead of the expression (4.4) defined in [7]. Therefore, the further the neighbouring neuron is from the central neuron, the less influence it will have on it.

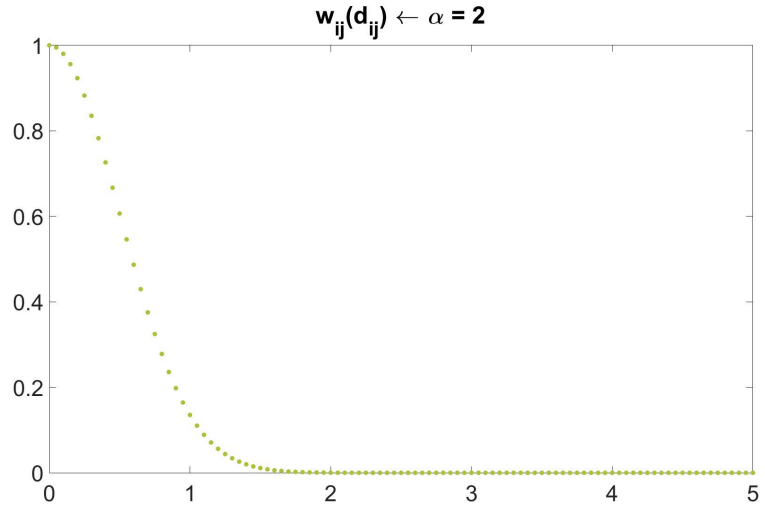
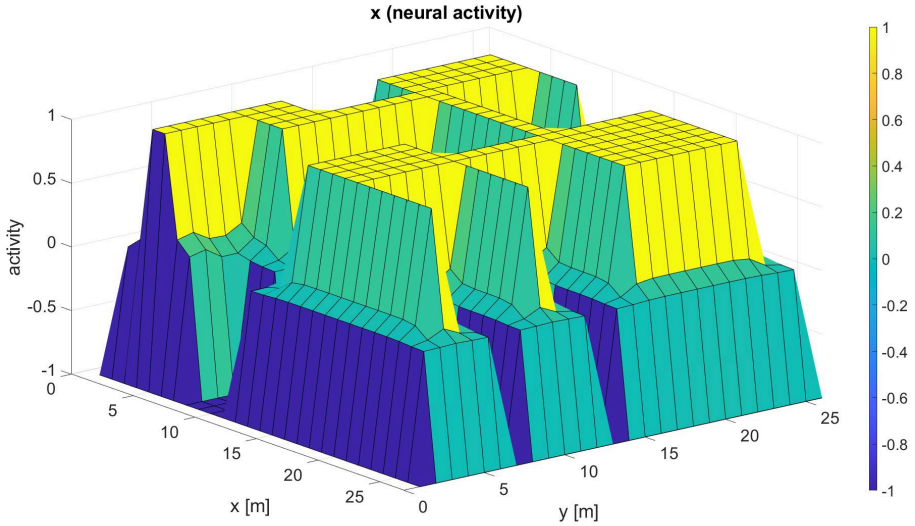


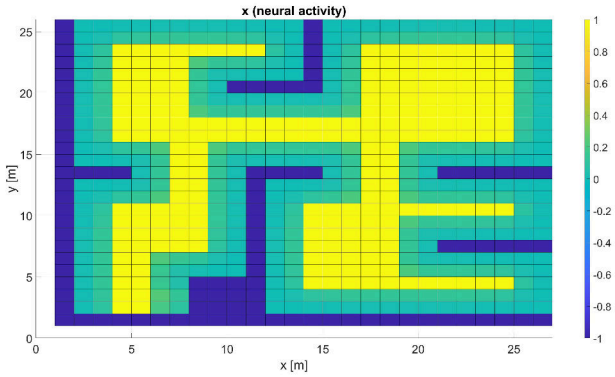
Figure 4.8: Weight connection calculation w_{ij} with $\alpha = 2$ (4.8)

In Figure 4.9 it is possible to see how the neural activity landscape evolves as the robot moves around obstacles in the workspace, the three basic neural states can be appreciated:

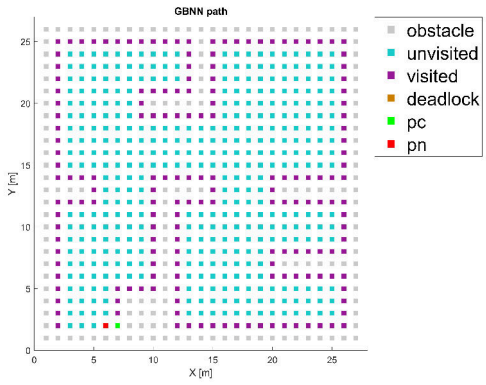
- Non-disinfected: $1 \leq x(k,l) < 0$.
- Disinfected: $x(k,l) = 0$.
- Obstacles: $0 < x(k,l) \geq -1$.



(a) Neural activity (3D view)



(b) Neural activity (2D view)



(c) Path covered by spiral motion

Figure 4.9: Neural activity landscape when the robot reaches $p_c(7,2)$

The robot always looks for zones of maximum activity to move, although as detailed in the equation 4.10, the necessary turn to reach the zone of maximum activity is also taken into account, choosing the cell that satisfies both criteria.

4.4.3 Path selection strategy

Time and energy efficiency is achieved when the robot travels the shortest path, avoiding disinfected areas and unnecessary turns. In the proposed strategy, the trajectory is generated from the dynamic activity landscape and the previous position. Working in this way can reduce direction changes during navigation. Thus, the next position denoted by p_n is calculated from the current position, p_c, as in (4.10).

$$x_{p_n} = \max\{x_j + cy_j, j = 1, 2, .., k\} \tag{4.10}$$

Where x_j is the neural activity of neuron j, c is a positive constant, and k is all the following possible positions. Function y_j (4.11) is monotonically increasing and is described as the angular difference with the current orientation to reach that position. For this problem a value $c = 0.1$ is considered.

$$y_j = 1 - \frac{\Delta\theta_j}{\pi} \tag{4.11}$$

The term $\Delta\theta_j \in [0, \pi]$ corresponds to the angle of rotation between the current and the possible next moving direction. The general expression that defines the above is shown in equation (4.12).

- If the robot goes straight: $\Delta\theta_j = 0$
- If the robot goes backward: $\Delta\theta_j = \pi$

$$\Delta\theta_j = \theta_j - \theta_c = |atan2(y_{p_j} - y_{p_c}, x_{p_j} - x_{p_c}) - atan2(y_{p_c} - y_{p_p}, x_{p_c} - x_{p_p})| \tag{4.12}$$

After reaching the target position, the values of the algorithm are updated. After that, the system continues its adaptive changes based on the varying environment.

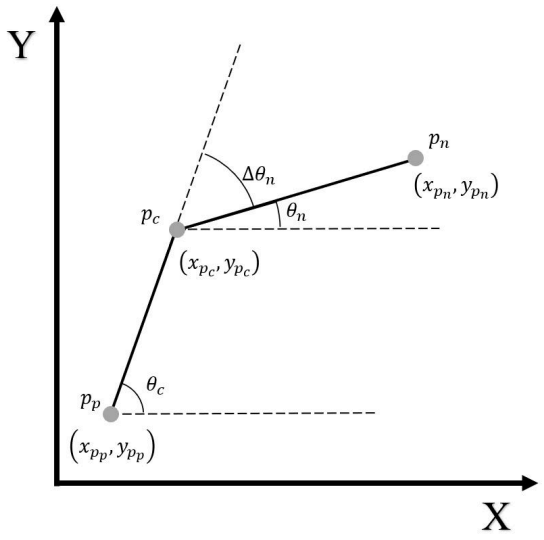


Figure 4.10: Movement scheme for (4.10)

However, to achieve an organized path, according to the selected motion strategy, diagonal movements are disabled, and their use is only allowed when these movements occur in deadlock situations. This situation occurs primarily because, with diagonal movements, it is easier for the robot to leave spaces without disinfecting while moving forward. Then, the robot has to return to these areas with the consequent waste of time and energy that would suppose.

To follow the motion technique used, be it spiral or boustrophedon, the overall coverage trend is considered to move from top to bottom and from left to right, thus optimizing the full coverage path, reducing repetition and clutter. When selecting the next point to move, it is important to consider the need to perform specific movements on certain occasions. These events are:

- The need for specific turns when following the boustrophedon strategy.
- The need to avoid corners without making a diagonal movement during a deadlock situation.
- The need to get around obstacles in a certain way.

For this, a series of templates are used to check the robot status considering the environment. These templates decide the sequence of movements necessary to continue on its way. Figure 4.11 shows one of the many possible cases where the robot has to abandon the general movement trend and go around an obstacle.

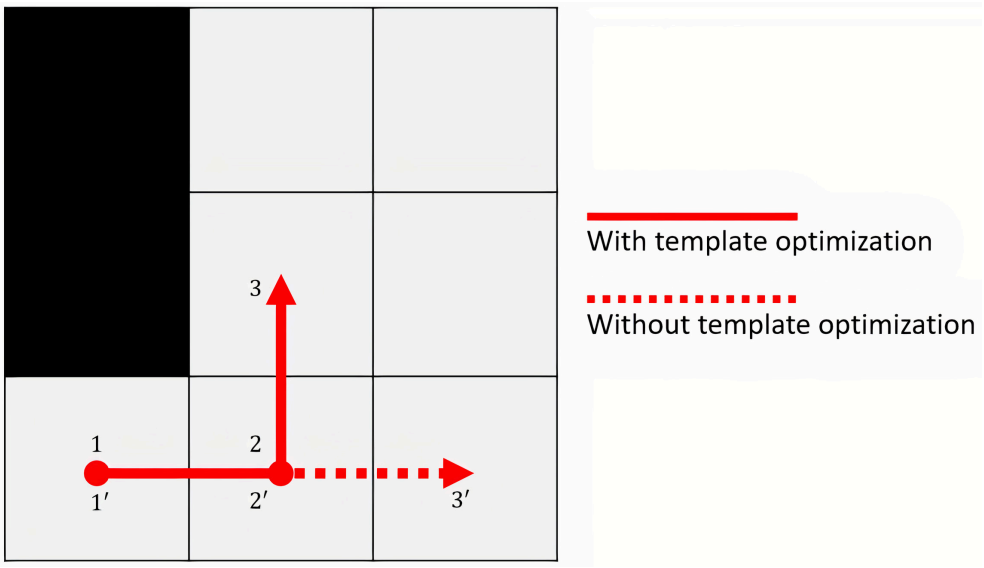


Figure 4.11: Optimized template of robot obstacle avoidance example

Something similar happens when generating the boustrophedon movement since it is not a turn that occurs naturally using neural network behaviour. It is the example described in Figure 4.12.

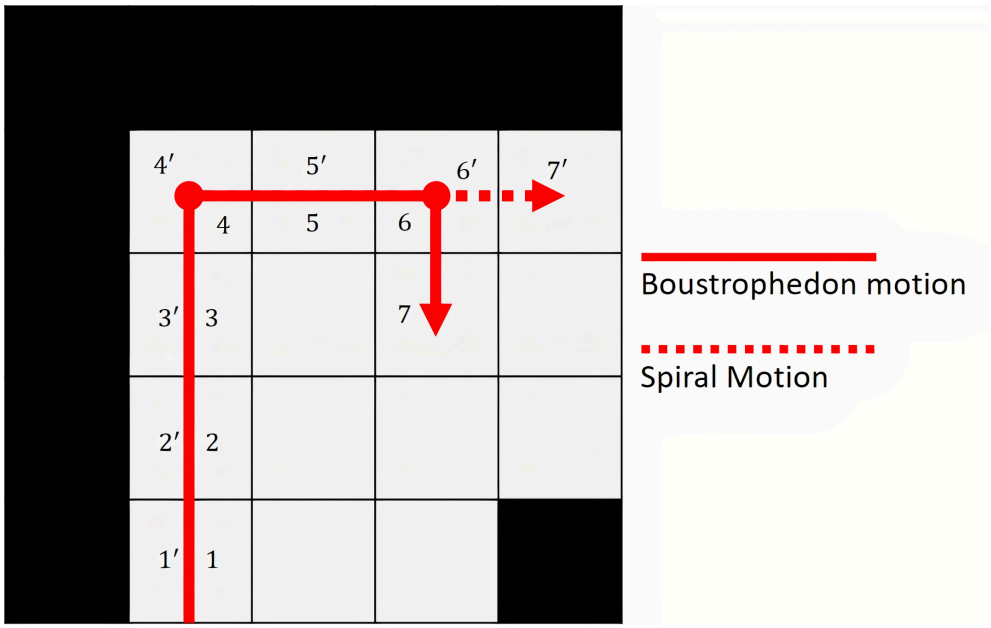
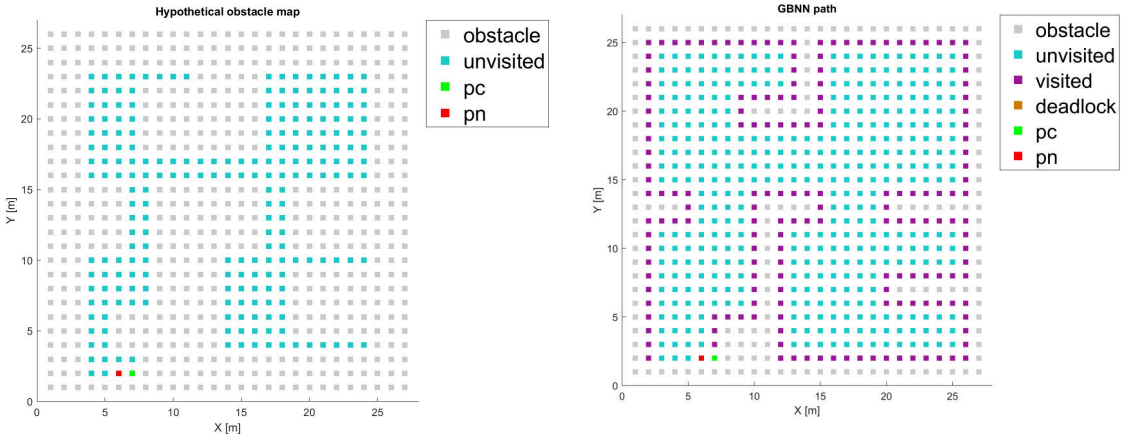


Figure 4.12: Optimized template of robot obstacle avoidance example

Another important feature is the auxiliary obstacle map. This map is not part of the neural network but assists the usage of the templates. In this map, the cells that have been disinfected are stored as if they were obstacles. A delay of 3 samples is used to analyse the map.



(a) Auxiliary obstacle map

(b) real robot trajectory

Figure 4.13: Updating the auxiliary obstacle map while the robot is moving

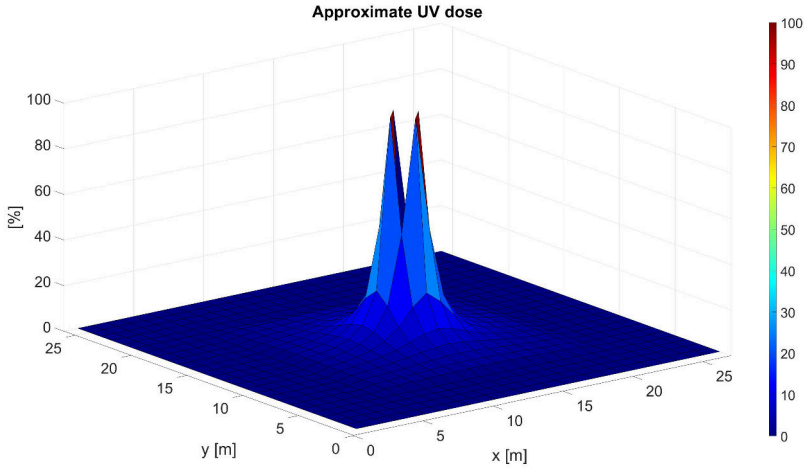
The obstacle map shown in Figure 4.13 is the one used to detect if any template should be used. For more information, Appendix B contains more details on the different categories of templates used and the movements described for each one.

4.4.4 Disinfection approximation

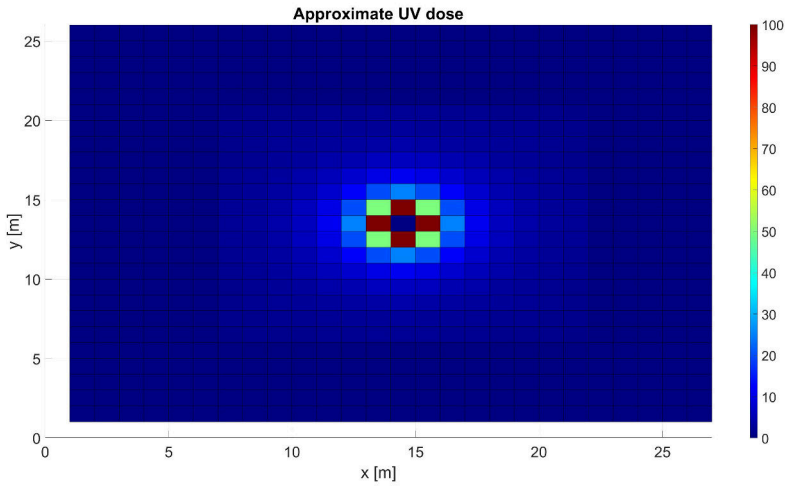
4.4.4.1 UV estimation

The radiation value that will exist in each cell is obtained approximately during the complete coverage path planning using the GBNN algorithm.

As described in Section 3.3.3, a grid map is created to store how the dose received changes with each robot move. On this occasion, work is done only with the distance between the cell and the radiation source. The lamp features are not considered. In this way, the cells (their centres) closest to the lamp will receive a higher dose when the robot passes by.



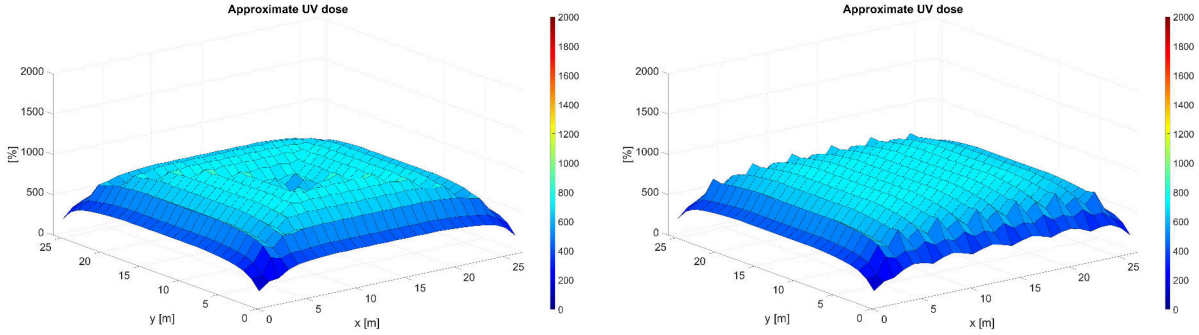
(a) 2D view



(b) 3D view

Figure 4.14: UV radiation profile approximation

The robot emits a radiation of $550\,\mu J/cm^2$ at a distance of 1 meter in one second. As shown in Figure 4.14, the area under the robot is not considered as disinfected, and therefore the cell associated with the current position remains unaffected.



(a) Spiral motion

(b) Boustrophedon motion

Figure 4.15: Dose estimation using different motion strategies

The value represented in each cell corresponds to the % of the dose each cell receives. At the end of the algorithm, there is a basic notion of which cells receive the most radiation. With this information, the speed is adjusted in the most appropriate way to avoid unnecessary peaks in the received dose.

4.4.4.2 External input approach

However, it should not be forgotten that the GBNN algorithm needs to receive this information as well. The algorithm only considers the eight neurons closest to the central neuron as disinfected to simplify its implementation.

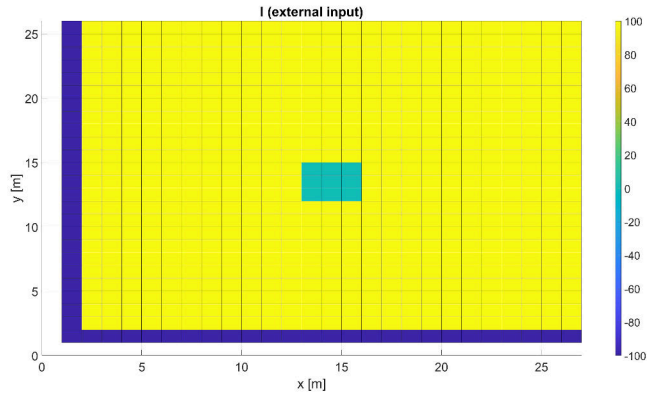
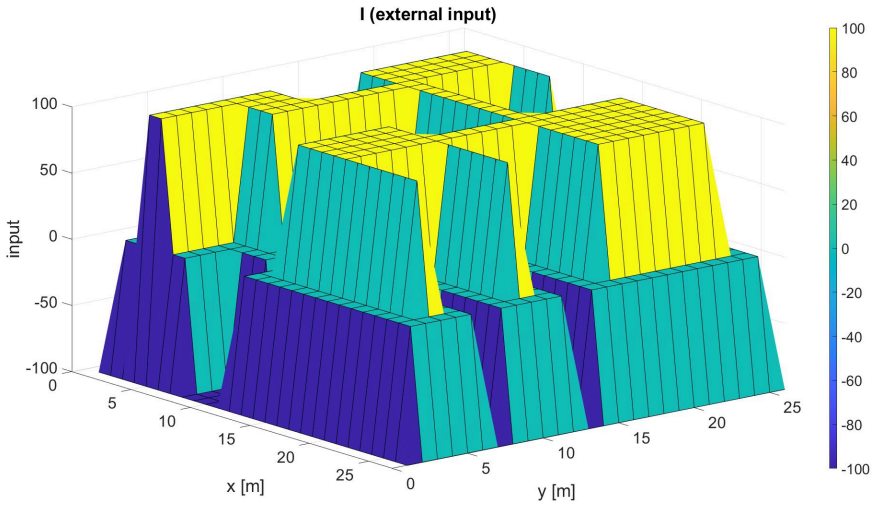


Figure 4.16: example of the use of the external input

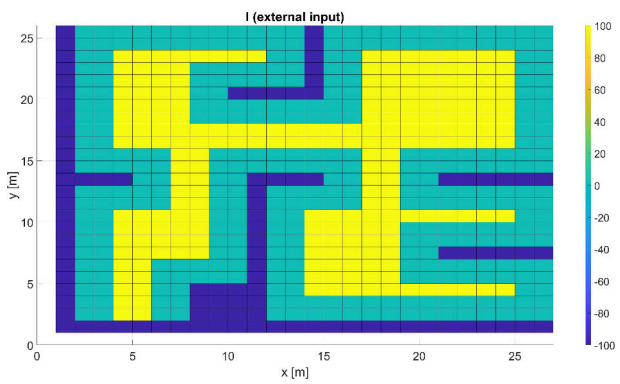
Only this range is considered because it is the zone where most radiation is received by proximity to the source of radiation. It also offers more regular trajectories than the previously mentioned UV distribution, which is influenced by the inverse square law.

In contrast to the matrix in Figure 4.14, in this case, the current position of the robot is also considered as disinfected. This information is used in equation (4.6), using the term I_i associated with the external input.

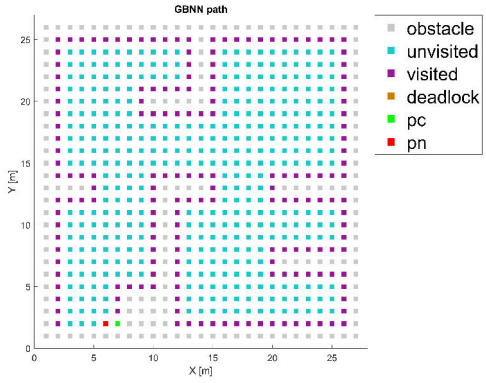
In fact, if 4.9 ,where the neural activity is represented, is compared with 4.17, the great similarity between both graphs can be verified. This situation is caused because the neural activity x is updated with the information received from the external input I_i as described in (4.6).



(a) External input (3D view)



(b) External input (2D view)



(c) Path covered by spiral motion

Figure 4.17: External input landscape when robot reaches $p_c(7, 2)$

Only this I distribution is considered because it is where most radiation is received. The dose received is higher since these cells are near to the source of disinfection. It also offers more regular paths than an equivalent distribution based on 4.14, which is influenced by the inverse square law. For example, if the I distribution is as detailed in Figure 4.18, it would be more probable that the robot would be lost in the search for the cells with the highest neuronal activity.

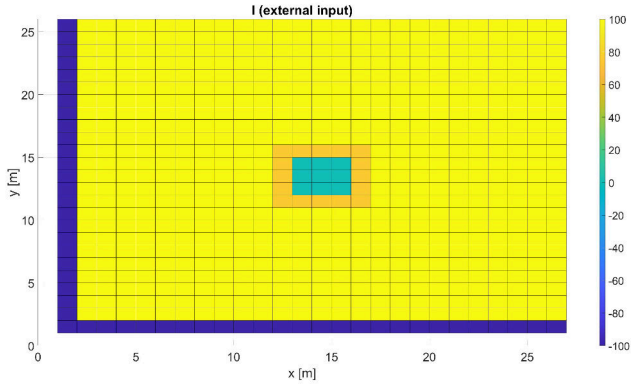
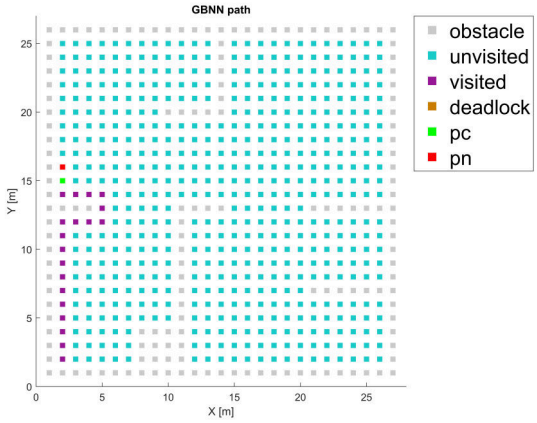
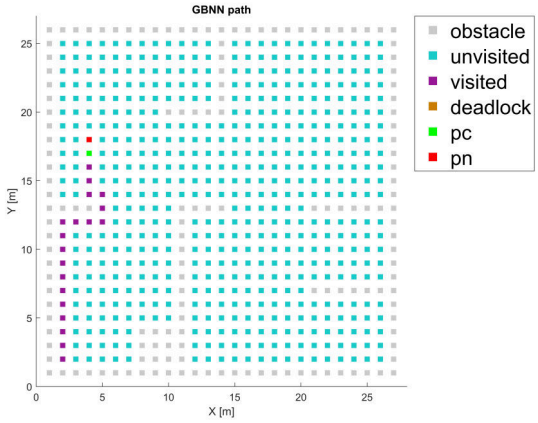


Figure 4.18: An alternative external input distribution

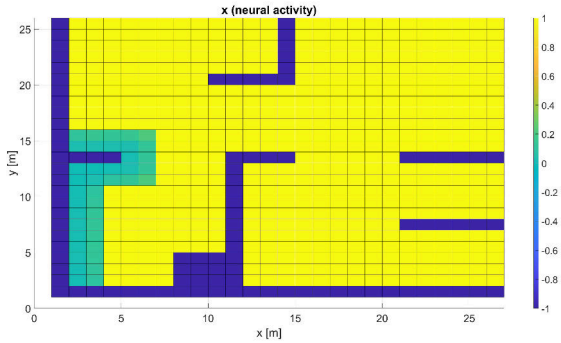
This situation is shown in Figure 4.19, where both alternatives are compared. It can be seen how the robot moves away from the expected path in the second case by going to areas of higher neuronal activity.



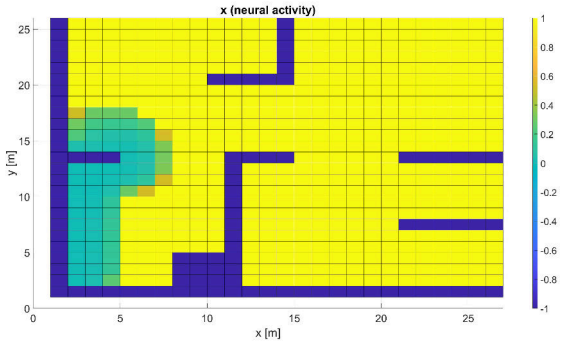
(a) Case 1: Path $\rightarrow I$(Figure 4.17)



(b) Case 2: Path $\rightarrow I$(Figure 4.18)



(c) Case 1: $x \rightarrow I$(Figure 4.17)



(d) Case 2: $x \rightarrow I$(Figure 4.18)

Figure 4.19: External input comparison

4.4.5 Deadlock detector

4.4.5.1 Approach

When the robot reaches a deadlock position, the neural network model and the proposed coverage algorithm can solve this situation. The robot trapped in the deadlock will wait until the neural activity propagates to its position, and then it will be able to get out of this situation and continue with the coverage.

The robot can escape from the deadlock, as defined in Section 4.4, definition 3. Using a dynamic neural neighbourhood analysis method and blocking detection algorithm to escape from deadlock.

Figure 4.20 shows a clear example of this aspect of the algorithm. It can be seen how the robot must go back before it finds a new path without disinfecting. The escape is achieved through the propagation of neuronal activity.

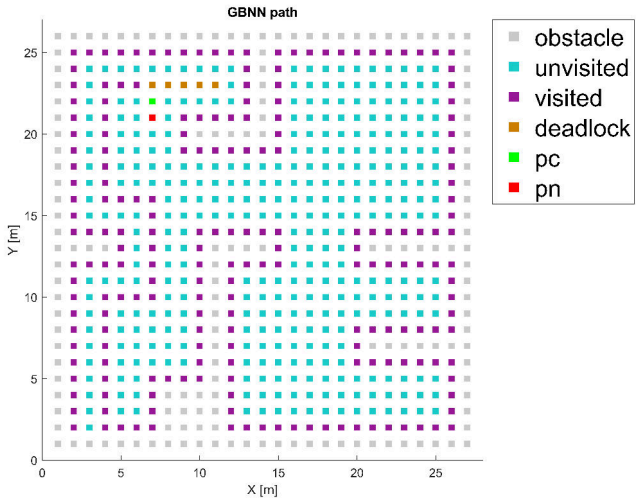


Figure 4.20: Deadlock situation with a neural approach

An alternative to the neuronal approach details above is proposed. This solution is based on Dijkstra's algorithm. This algorithm calculates the shortest path between two nodes within the same graph.

This technique allows the robot to escape from a deadlock situation as soon as possible. If the path in Figure 4.20 previously observed is compared with the path in Figure 4.21, a shorter route is achieved in the second case.

To be precise, 214 iterations are used in the neuronal approach to reach the same point (the cell with red colour), while for Dijkstra's algorithm, only 210 are needed.

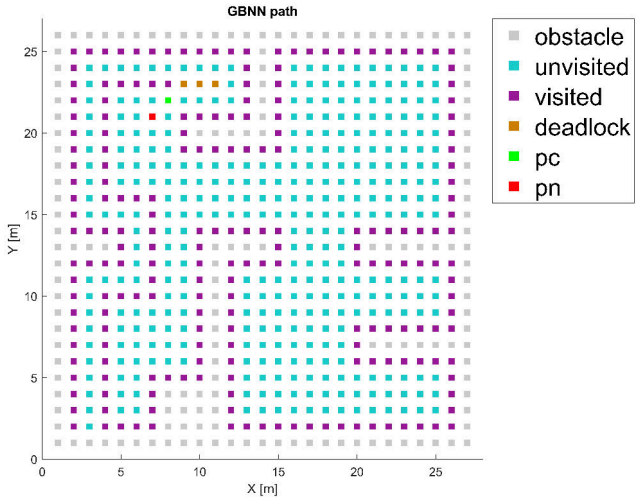


Figure 4.21: Deadlock situation with Dijkstra approach

4.4.5.2 Dijkstra algorithm

This approach requires a higher computational cost. But it is not only used to get the shortest path to escape from a deadlock situation. It is used because when the maps are more complex, the neuronal approach is incapable of preventing the robot from getting stuck in a deadlock situation.

Dijkstra's algorithm finds the shortest paths between a source to any node in a weighted graph $G = (V, E)$, where V are vertexes and E are edges. In the context of greedy algorithms, Dijkstra's method looks in every node (vertex) for the short path(edge) [43]. The algorithm work as follows:

Algorithm 3: Dijkstra's Shortest Path First

1. for each vertex $v \in G.V$
   - $v.d = \infty$;
   - $v.\pi = NIL \; \forall \, v \in V$;

   // Initialize d and π values
2. $s.d = 0$
3. set $S := 0$; // Initialize S to the empty set
4. set $Q := G.V$;

   // Initialize the min priority queue Q to contain all vertexes in V
5. $while \; Q \neq 0$
   - $u = EXTRACT - MIN(Q)$

     // Extracts a vertex u from $Q = V - S$
   - $S \cup \{u\}$ // Adds u to set S
   - for each vertex $v \in G.Adj[u]$
     - $if \; v.d > u.d + w(u, v)$
       - $v.d = u.d + w(u, v)$
       - $v.\pi = u$

     // Relax each edge (u, v) leaving u

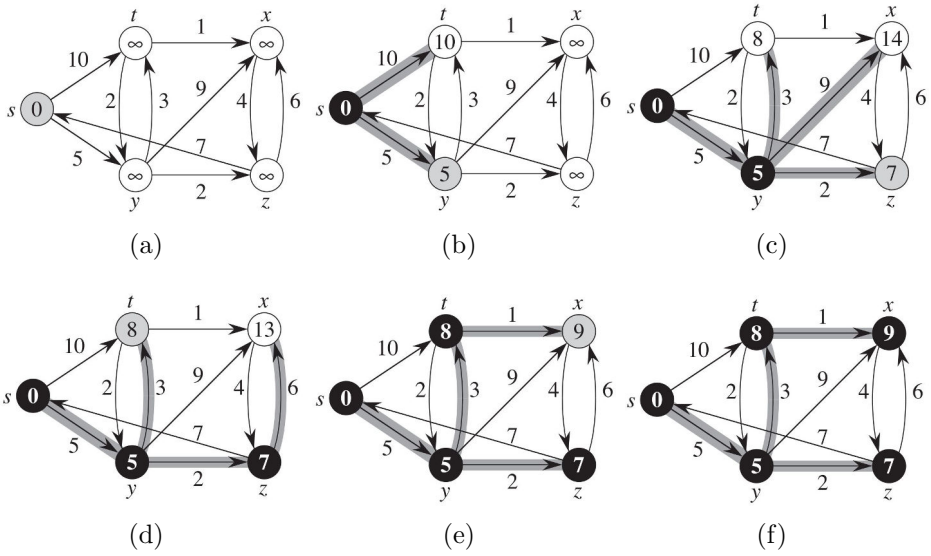
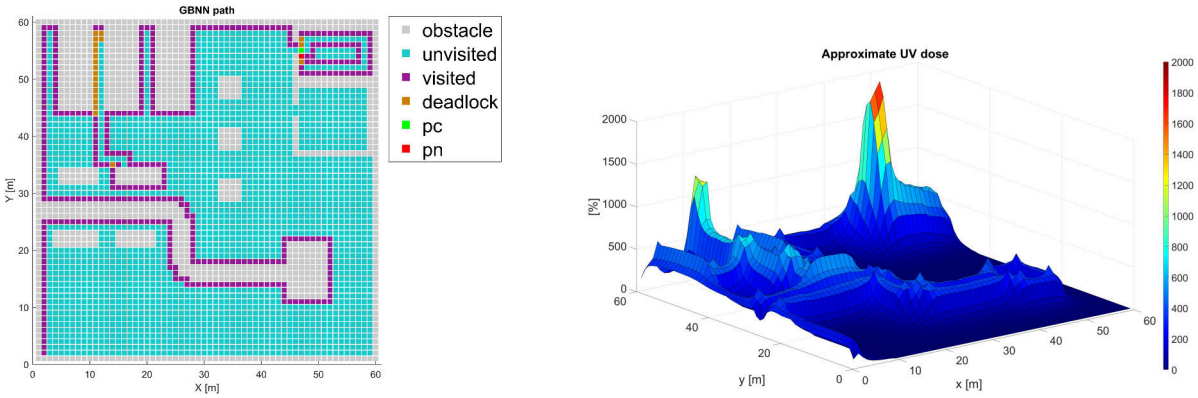


Figure 4.22: The execution of Dijkstra's algorithm [44]

Dijkstra's algorithm provides a set S of vertexes whose final shortest path weights from the source node s have already been determined. The algorithm repeatedly selects vertex $u \in V - S$ with the minimum shortest path estimation, adds u to S, and relaxes all edges leaving u. It is useful to use a minimum priority queue Q of nodes, which is adjusted by their d values [44].

According to Figure 4.22, (**a**) is set just before the first iteration of the while loop of lines 4-8. The shaded vertex has the minimum value of d and is chosen as vertex u on line 5. From (**b**) to (**f**), the situation is shown after each successive iteration of the while loop. The shaded vertex of each part is chosen as vertex u on line 5 of the next iteration. The d values and predecessors shown in (**f**) are the final values.

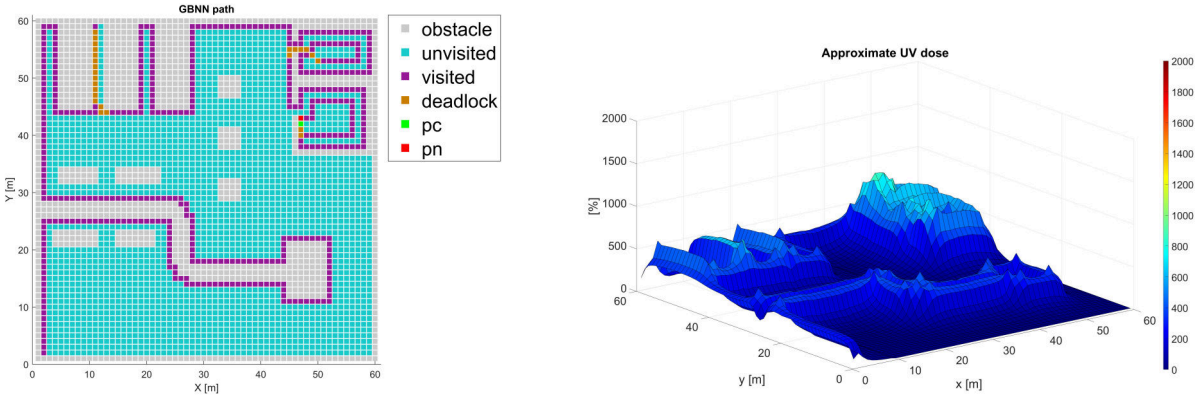


(a) Robot's path

(b) Estimated UV radiation graph

Figure 4.23: Deadlock situations solved by neural approach (Iterations = 475)

Figure 4.23 is a clear example of this problem. The neural approach can solve the first deadlock situation, although not in the best way. However, in the last situation, the robot gets stuck and is not able to go any further, as shown in the UV graph. Nevertheless, using Dijkstra's algorithm, it is always possible to escape from these deadlock situations in a better way than in the previous case (4.24).



(a) Robot's path

(b) Estimated UV radiation graph

Figure 4.24: Deadlock situations solved by Dijkstra's algorithm (Iterations = 475)

The Dijkstra algorithm used is based on that described above. The code has been generated from [28], where it implements a series of functions that make it possible to obtain the shortest path between two points in a graph. It is essential to generate a graph of the working environment to accomplish this. Then, the functions of [28], which use the Bioinformatics Toolbox, will ensure the optimal path between two given nodes is obtained.

Once the main characteristics of this technique have been clarified, it is important to know how it is integrated into the main GBNN algorithm. In an effort to improve the results, a deadlock detector has been implemented.

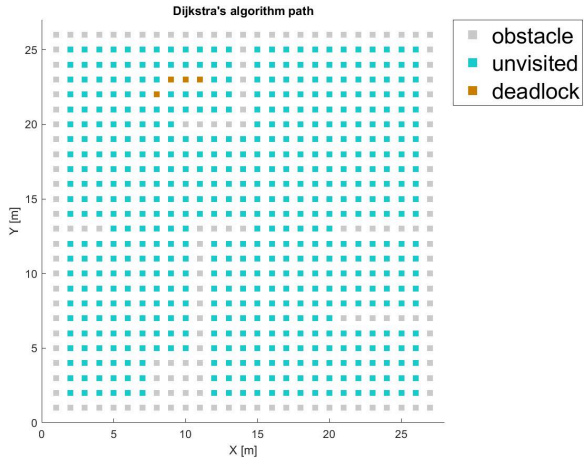


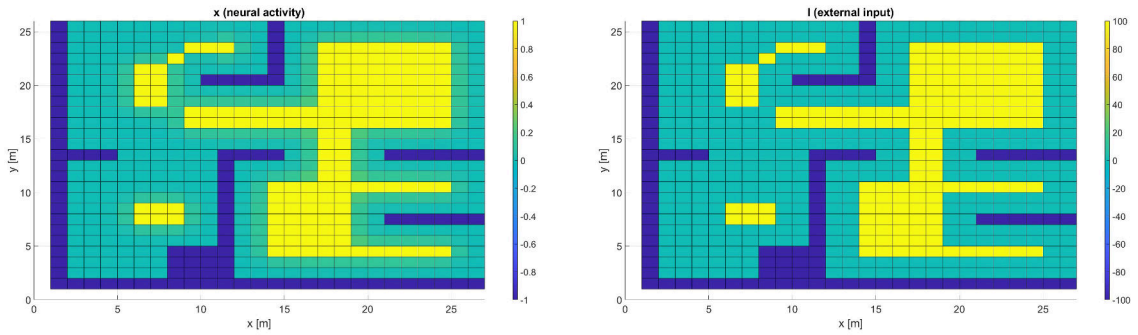
Figure 4.25: Path generated using Dijkstra's algorithm

Figure 4.25 shows the path generated by this algorithm for the deadlock situation detailed above in Figure 4.21. The path shown is the most appropriate solution to escape among all the available options.

The algorithm uses the I matrix to analyse if the cells close to the current position have already been disinfected. The algorithm can predict whether the robot will get stuck by using this information. In particular, the 7×7 matrix enclosing the current position is analysed. Once the imminent deadlock situation has been detected, a search is made, looking for the shortest path between the current robot position and the remaining non-disinfected cells. The next steps are followed to perform this search:

1. The Euclidean distance between the current position and the rest of the candidate cells is calculated. The existence of obstacles is not considered.
2. Candidates are classified according to their distance from the current position from lowest to highest.
3. Dijkstra's algorithm is applied to all previously ordered candidates.
   - 3.1. With this algorithm, the existence of obstacles is considered.
   - 3.2. The value of the target cell is updated, when the distance obtained by this algorithm is the smallest one achieved until now.
   - 3.3. In an attempt to avoid wasting CPU time, when the distance obtained using Dijkstra is $d_{Dijkstra} \approx |q_i - q_j| \pm 0.1\,m$, the search ends with that candidate as the destination to escape from the deadlock situation.

The neural network must be updated once the escape path is generated. The neuronal activity x and the external input I are re-establish with the information of the path to be followed (Figure 4.26). In this way, the affected cells are restored to their original state of non-disinfected, and the GBNN algorithm can continue its normal activity to escape the deadlock situation in a natural way.



(a) Neural activity (b) External input

Figure 4.26: Update of neuronal activity(x) and external input(I) with Dijkstra path

4.5 Simulation environment

4.5.1 Simulation model

It is necessary to implement a simulation environment to verify the advantages and disadvantages of each generated path.

The kinematic behaviour of the robot is modelled from Section 3.2.4 information. Also, the model of the lamp is detailed in Section 3.3.3. The Simulink model shown in Figure 4.27 is used to emulate the behaviour of the robot. Furthermore, it is possible to analyse the disinfection process that would be followed on the generated path.

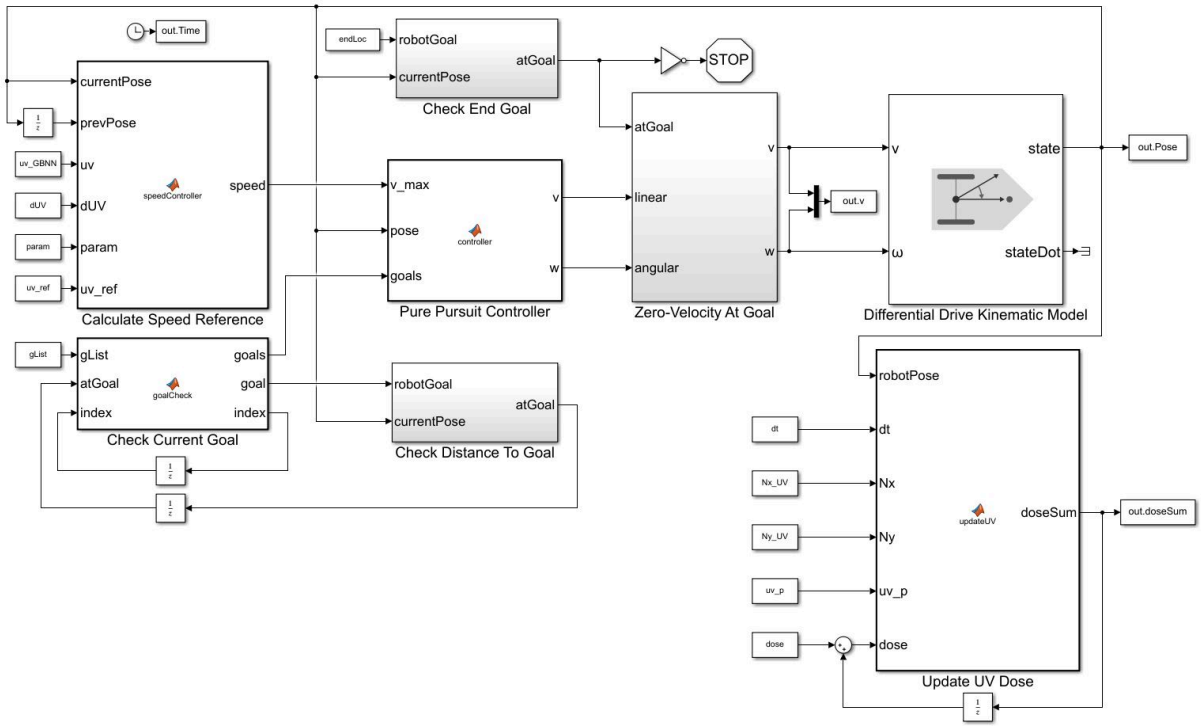
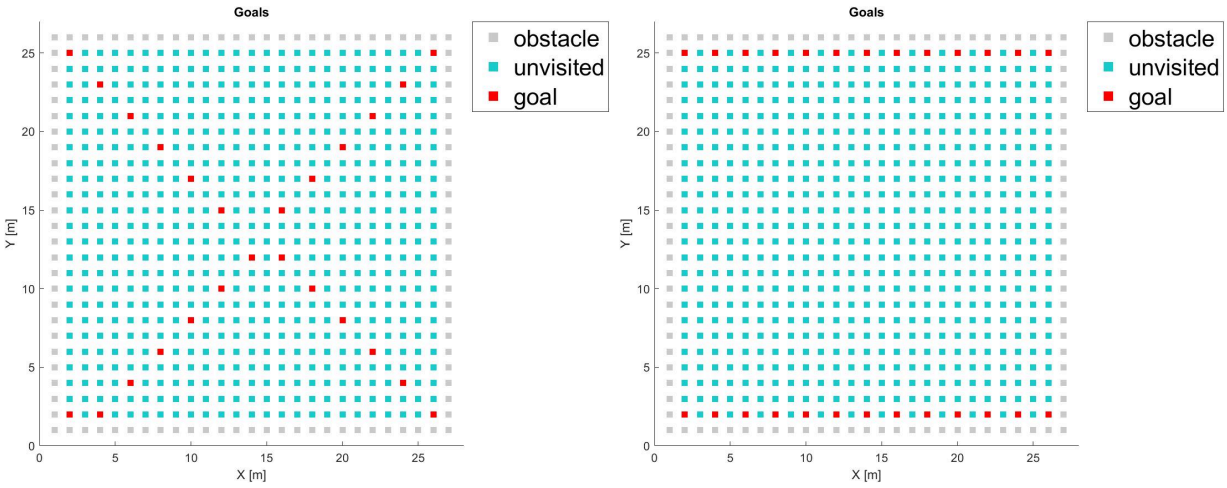


Figure 4.27: Model in Simulink to analyse the behaviour of the robot during workspace disinfection (model step = 0.1 s)

In this model, a calculation of the desired speed is made based on the information obtained in each cell during the GBNN algorithm. With this speed, a pure pursuit controller is used to reach the appropriate goal in the expected conditions.

4.5.2 Goals generator

Once the path is acquired using the GBNN algorithm, the goals are extracted and used for the robot's path control. Thus, from the paths determined in figure 4.3, the goals shown in Figure 4.28 are obtained. These targets will be the references that the controller will use to perform the disinfection coverage.



(a) Spiral motion

(b) Boustrophedon motion

Figure 4.28: Goals generated using the path achieved with the GBNN algorithm

4.5.3 Speed controller

The information provided by the proposed coverage algorithm is used to estimate the speed of the robot in each section associated with a cell (Section 4.4.4). Based on this percentage matrix, the lamp model, and the dose required for disinfection, a speed value is obtained using the expression (4.13).

$$v = \rho UV_{\%} \left(\frac{D_{UV}}{D_{SARS-CoV-2}} \right) \tag{4.13}$$

Where:

- v = Robot Speed (m/s)
- ρ = Fitting parameter
- $UV_{\%}$ = The estimated average percentage of the dose received, considering the position of the robot and its eight neighbouring cells (%)
- D_{UV} = Lamp radiation (mJ/cm^2)
- $D_{SARS-CoV-2}$ = SARS-CoV-2 dose (mJ/cm^2)

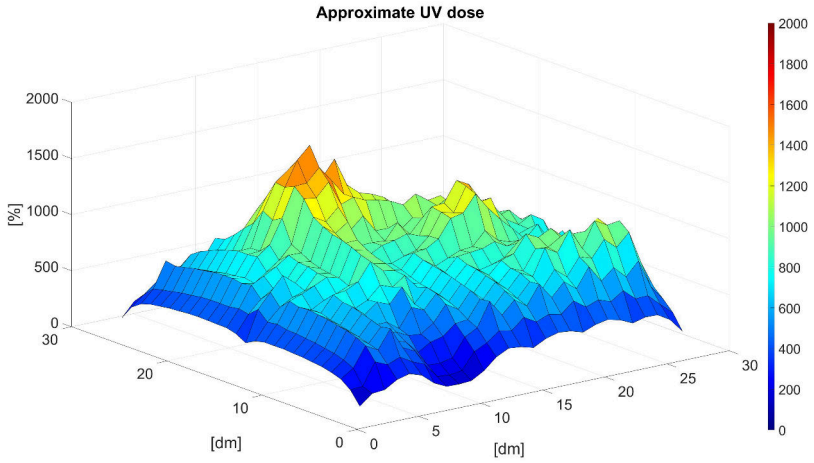
The D_{UV} value is based on the specifications in section 3.3.3, according to the lamp model used. Specifically, a value of 0.55 $mJ/cm2$ is used.

On the other hand, to achieve the inactivation of SARS-CoV-2, it would be prudent to apply a dose of 250 J/m^2 in the air and on surfaces. This value would guarantee a 90% reduction of the virus.

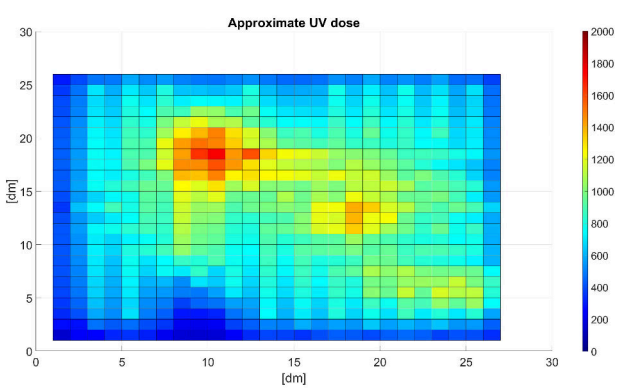
However, as described in [45], to achieve a 99% reduction, power must be doubled to increase the inactivation by one log. For all the above reasons, it is considered appropriate to apply the dose of 500 J/m^2 at the application point.

The parameter ρ is used to adjust the estimated percentage of $UV_{\%}$, so that the desired disinfection is obtained without unnecessarily surpassing the desired dose. For this application, a value of 1.25 has been used to maintain the final dose produced in the desired range.

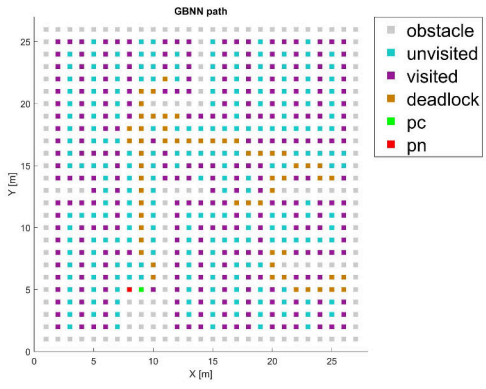
In Figure 4.29, a peak can be seen that highlights more than the rest. This peak means that one area will receive more radiation than the others if the robot travels at a constant speed.



(a) 3D View



(b) 2D View

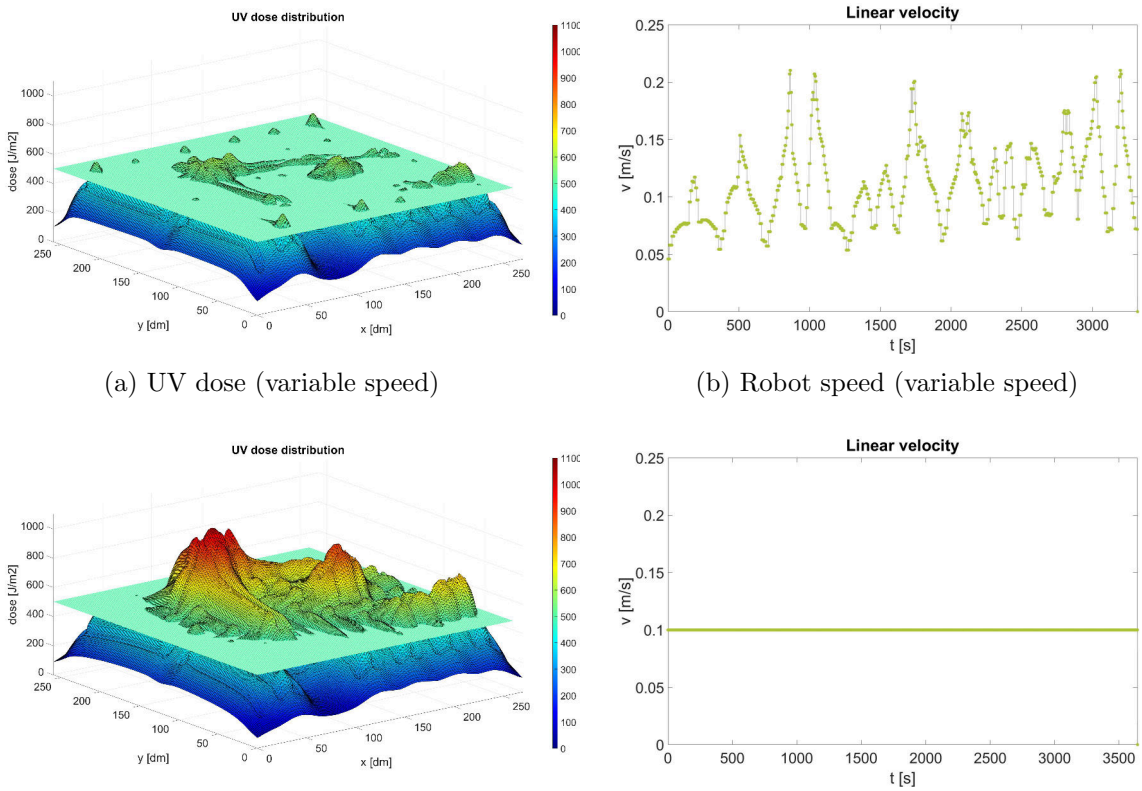


(c) Coverage path using bosutrophedon motion

Figure 4.29: Example of a peak in the estimated dose

Therefore, with the $UV_{\%}$ value, the robot speed can be adapted according to the map necessities. As a result, the dose distribution is the most equitable possible.

In Figure 4.30 , the significance of this controller can be analysed. Despite its simplicity, it can achieve quite interesting results in disinfecting all regions of the workspace uniformly.



(a) UV dose (variable speed)

(b) Robot speed (variable speed)

(c) UV dose (constant speed 0.1 m/s)

(d) Robot speed (constant speed 0.1 m/s)

Figure 4.30: Example of complete UV disinfection with different speed conditions

4.5.4 Pure pursuit controller

Pure pursuit is a path-tracking algorithm. It calculates the angular velocity command that moves the robot from its current position to reach some look-ahead point in front of the robot. The linear velocity comes from the speed control. This controller constantly pursues a point in front of the robot.

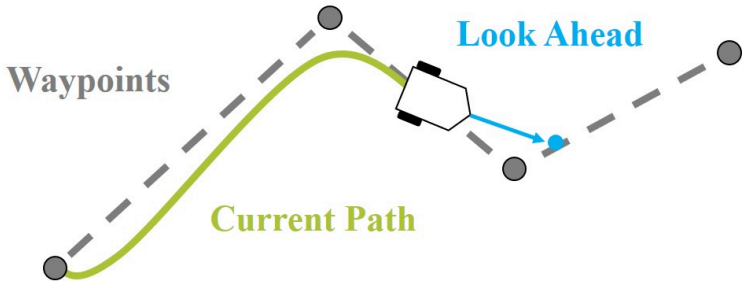
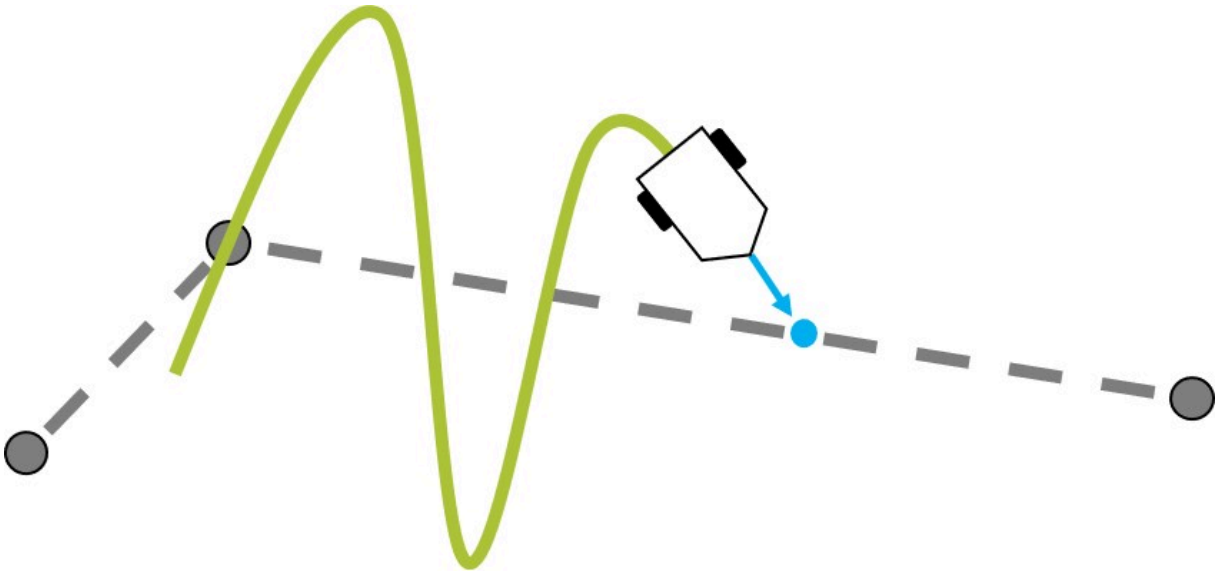
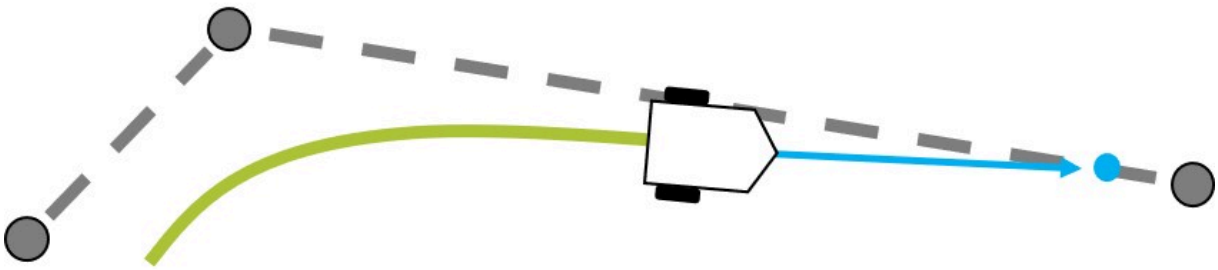


Figure 4.31: Pure pursuit controller explanation

As shown in Figure 4.32, waypoints are $[x\ y\ \theta]$ coordinates, which are used to compute the robot velocity commands. The look-ahead distance is how far along the path the robot should look from the current location to compute the angular speed.



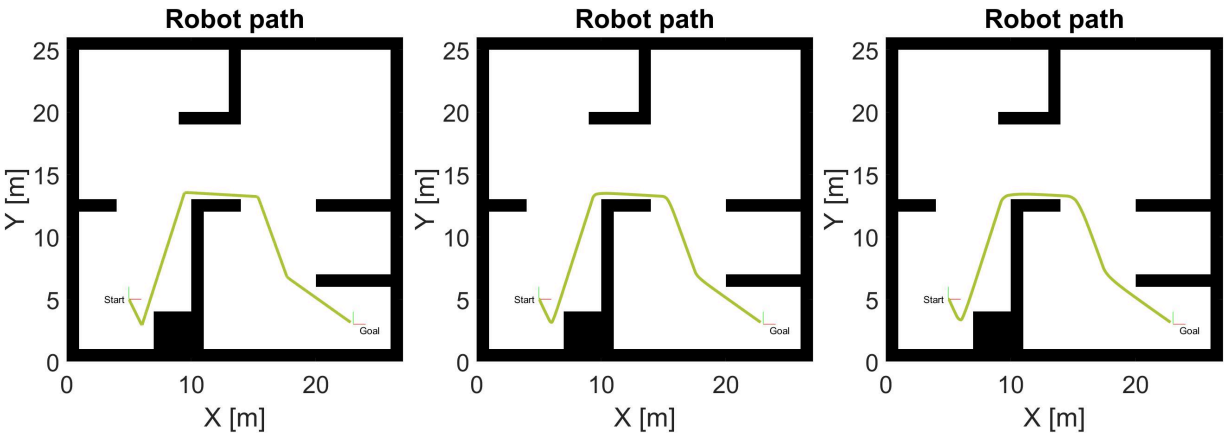
(a) Short look-ahead



(b) Large look-ahead

Figure 4.32: Look ahead distance effects

Figure 4.32 shows the effect of varying the look-ahead distance. A small look-ahead distance will cause the robot to move quickly towards the path, but the robot overshoots the track causing oscillations. On the other hand, a large look-ahead distance might result in larger curvature near the corners.



(a) $d_{LA} = 0.1\ m$ (b) $d_{LA} = 0.5\ m$ (c) $d_{LA} = 0.9\ m$

Figure 4.33: Look ahead distance comparison ($v = 0.1\,m/s$, $w = 3.14\,rad/s$)

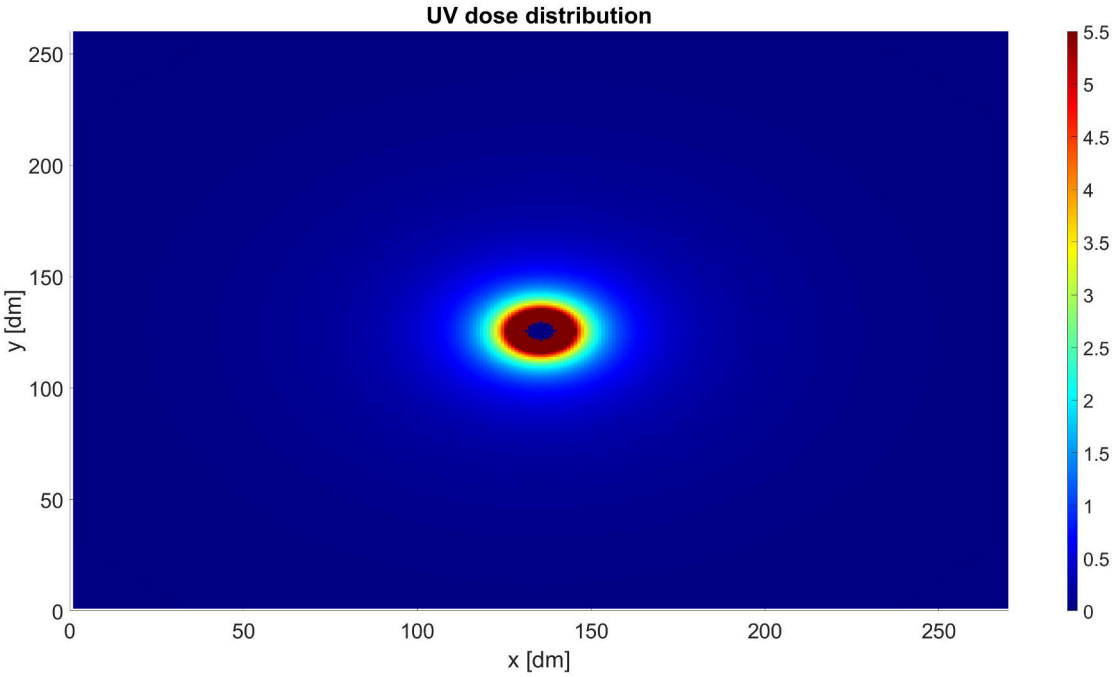
The effect on the curves as the look-ahead distance increases is shown in Figure 4.33. For the implemented application, a look-ahead distance of 0.2 m. With this value, it is possible to achieve a path similar to that determined by the complete coverage path planning algorithm. To avoid the undesired effects of overshoots, it works with small sections of the final route.

4.5.5 Disinfection simulator

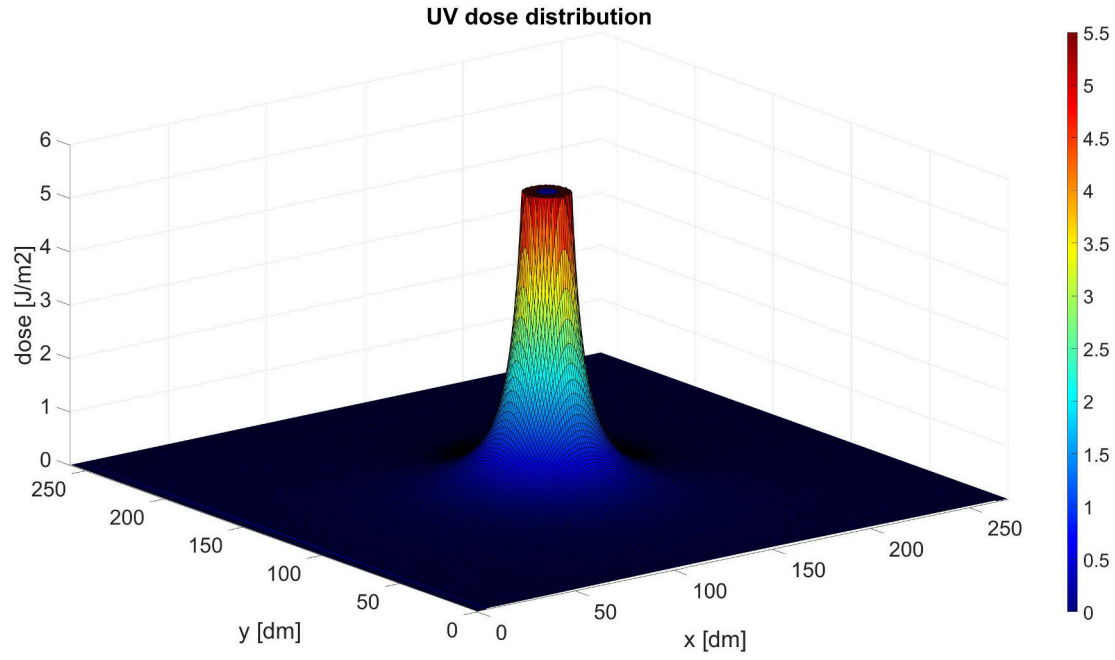
Based on the information detailed in Section 3.3.3 and considering what was explained in Section 4.4.4, the disinfection process is simulated using a cell-matrix. Each cell is $0.1 \times 0.1\,m$ to have more precision for the posterior analysis. It should be noted that this approach has certain limitations:

- The proposed radiation profile has no distance limitation.
- The intensity decays with the square of the distance, as reflected in Figure 3.8.
- The existence of obstacles is not considered. Therefore, through this model, the radiation crosses the existing walls in the different maps.
- The absorption coefficients of each material, whose value may influence the final dose received, are not contemplated.

Figure 4.34 shows the dose acquired when using the previously mentioned lamp model, after one second of exposure.



(a) 2D view



(b) 3D view

Figure 4.34: UV radiation profile approximation ($550\,\mu J/cm^2$)

4.6 Log manager

The logger allows the creation of objects that can be used to record different events during the execution of algorithms. The proposed logger class is based mainly on the work shared in [46], with some modifications in the format and type of events to be registered. The structure of a log message is "[EVENT] [HH:MM:SS] Message". The different events are:

- Information: [INFO] [19:18:58.577] This is an Information message example
- Warning: [WARN] [19:18:58.579] This is a Warning message example
- Error: [ERROR] [19:18:58.582] This is an Error message example
- Debug: [DEBUG] [19:18:58.586] This is a Debug message example

All logged events will be stored in a .txt file in the Logs folder. This file is named *run_yyyy_mm_dd_HH_MM_SS_x*, where x is a random integer number within the range $[1, 10000]$. Debug type can be enabled or disabled, but other events are always logged.

Chapter 5

Testing and analysis of results

5.1 Metrics

The CPU time (t_{CPU}) defines how long the code takes to run. The characteristics of the computer used for the different tests are:

- Type: Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz
- Clock: 2201 MHz
- Cache: 1536 KB
- Processors: 6
- OS type: Windows
- OS Version: Microsoft Windows 10 Education N

Table 5.1 shows the metrics used to analyse the behaviour of the GBNN algorithm used for complete coverage path planning. To compare, a lower number of iterations will imply a lower distance covered by the robot.

Table 5.1: Parameters for the analysis of the GBNN algorithm.

Symbol	**Criteria**	**Definition**
t_{GBNN} $[DD:MM:SS]$	*Algorithm time*	Time used in algorithm for complete coverage path planning
n_{GBNN} $[units]$	*Algorithm iterations*	Iterations used in algorithm for complete coverage path planning
$n_{Dijkstra}$ $[units]$	*Dijkstra's support*	Number of times Dijkstra's algorithm is required to get out of a deadlock situation
d_e $[m]$	*Estimated distance*	The estimated distance that the robot will cover
r_e $[rad]$	*Estimated radians*	The estimate of total radians that the robot will rotate

The values that affect the robot once the simulation of the disinfection process is finished are shown in Table 5.2.

Table 5.2: Parameters for the analysis of the simulation.

Symbol	Criteria	Definition
t_{sim} $[DD:MM:SS]$	*Simulation time*	Time required to perform the simulation.
t_{robot} $[DD:MM:SS]$	*Robot time*	Time used by the robot to finish the disinfection coverage.
n_{goals} $[units]$	*Goals*	Total number of goals, which define the path to be followed by the robot.
d_{robot} $[m]$	*Robot distance*	The total distance that the robot will cover
r_{robot} $[rad]$	*Robot radians*	The total radians that the robot will rotate
v_{max} $[m/s]$	*Maximum speed*	Maximum speed reached by the robot situation
v_{min} $[m/s]$	*Minimum speed*	Minimum speed reached by the robot
v_{av} $[m/s]$	*Average speed*	Average speed of the robot

The parameters in Table 5.3 are used to measure the dose received in the workspace according to thresholds around the desired dose. Graphs are used to complement this information.

Table 5.3: Parameters for the analysis of the disinfection coverage.

Symbol	Criteria	Definition
D_{max} $[J/m^2]$	*Maximum dose*	Maximum dose reached
D_{min} $[J/m^2]$	*Minimum dose*	Minimum dose reached
D_{VH} [%]	*Very high dose*	$D_{total} > 1.2D_{SARS-CoV-2}$
D_H [%]	*High dose*	$1.1D_{SARS-CoV-2} < D_{total} < 1.2D_{SARS-CoV-2}$
D_N [%]	*Normal dose*	$0.9D_{SARS-CoV-2} < D_{total} < 1.1D_{SARS-CoV-2}$
D_L [%]	*Low dose*	$0.8D_{SARS-CoV-2} < D_{total} < 0.9D_{SARS-CoV-2}$
D_{VL} [%]	*Very low dose*	$D_{UV} < 0.8D_{SARS-CoV-2}$

Table 5.4 details the value of the parameters used to perform the tests. These parameters come from Table 3.2. However, those values related to the map size are omitted since they depend on the test to be analysed. Appendix A provides more details on the map dimensions.

Table 5.4: Problem Parameters

Parameter	**Value**	**Section**	**Application**
$r0$	3	4.4	Receptive field
α	2	4.4.2	Weight connection calculation w_{ij}
β	0.6	4.4.2	Transfer function $g(a)$
c	0.1	4.4.3	Path selection strategy
dt	$0.1\,ms$	4.5.1	Time step
ρ	0.8	4.5.3	Speed controller
$D_{SARS-CoV2}$	$500\,J/m^2$	4.5.3	Speed controller
D_{UV}	$5.5\,J/m^2$	4.5.3	Speed controller & disinfection simulator
d_{LA}	0.2	4.5.4	Pure pursuit controller

5.2 Test 1: Obstacle-free map

5.2.1 General

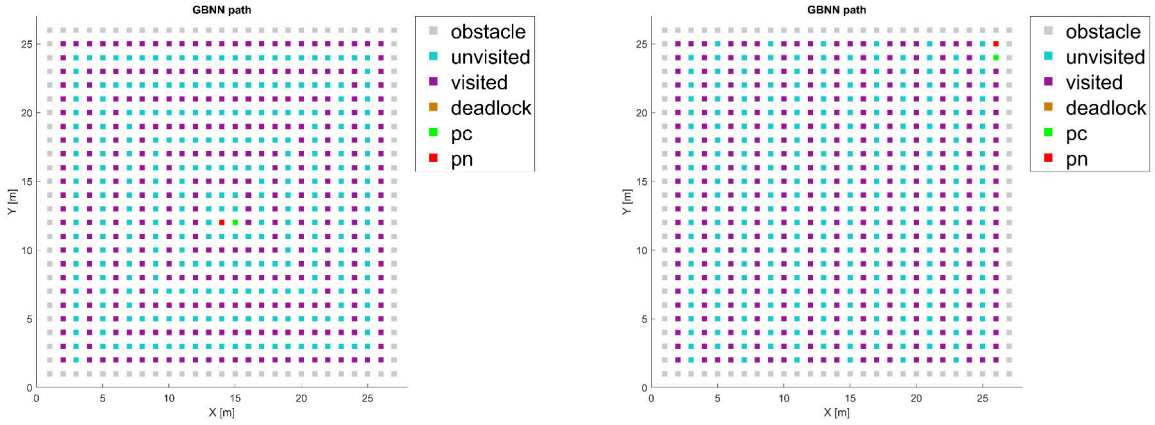
First, an empty map is used. This space is a working environment free of obstacles but surrounded by walls. In Table 5.5 similar times can be observed. Nevertheless, the robot working with spiral motion requires a longer time to complete the coverage.

Table 5.5: Time measurement for Test 1.

	$\mathbf{t_{CPU}}$	$\mathbf{t_{sim}}$	$\mathbf{t_{GBNN}}$	$\mathbf{t_{robot}}$
Spiral	00 : 02 : 13	00 : 01 : 19	00 : 00 : 05	01 : 02 : 44
Boustrophedon	00 : 02 : 06	00 : 01 : 12	00 : 00 : 05	01 : 02 : 18

5.2.2 GBNN algorithm analysis

In Figure 5.1, the paths involved are reasonably simple and without further complications. The robot moves from the outside to the inside using spiral motion until it reaches the centre of the map. On the other hand, with boustrophedon movements, the robot moves around the environment describing a snake movement.



(a) Spiral

(b) Boustrophedon

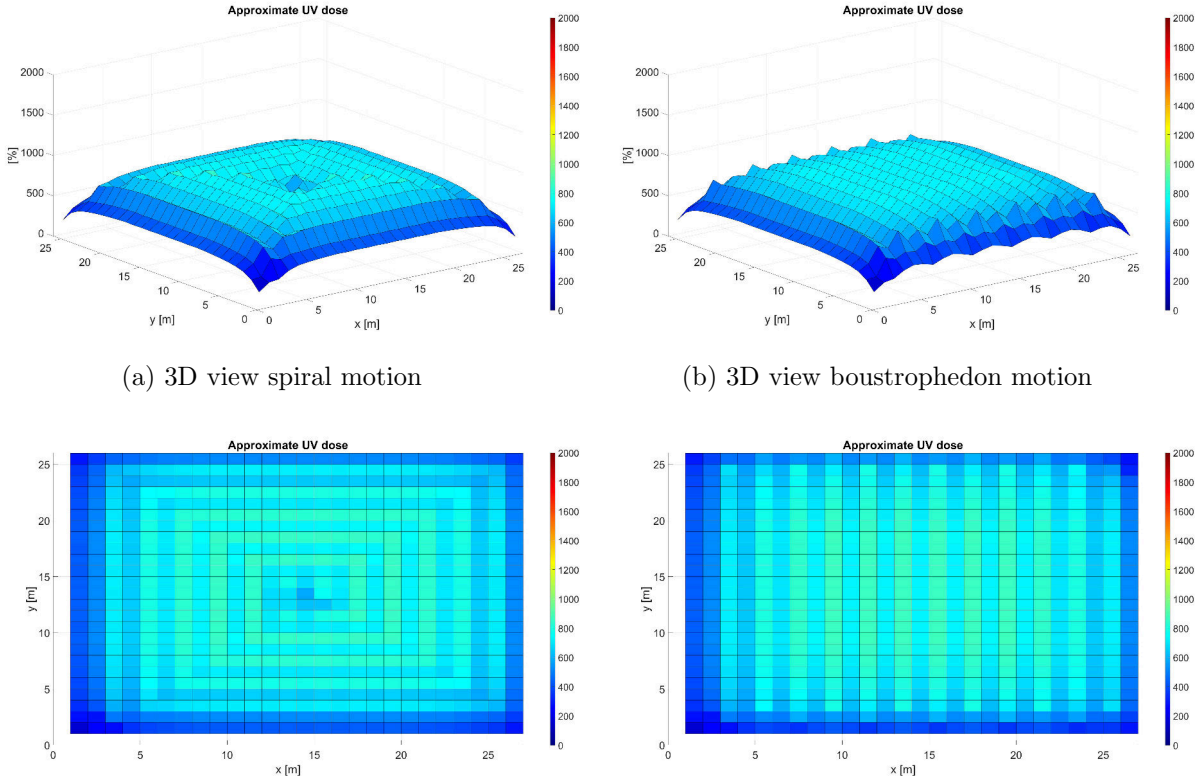
Figure 5.1: Path for Test 1, generated by complete coverage path planning

The number of cells visited is nearly the same. As can be seen in Table 5.6, both techniques have a similar number of turned radians at the end of the coverage.

Table 5.6: GBNN algorithm metrics for Test 1 with spiral motion.

	n_{GBNN}	$n_{Dijkstra}$	d_e	r_e
Spiral	322	0	322	36.13
Boustrophedon	323	0	323	37.69

In Figure 5.2, it can be noted how there is a higher dose coverage overlap in the mid zone, using boustrophedon motion. However, in the case of using spiral motion, the dose received in the map centre decreases in the same way as in the edges. This phenomenon is explained because this point is the last one to be visited, and therefore the robot will have to go slower in this area. Something similar happens at the last point visited with boustrophedon motion, but it is less visually evident.



(a) 3D view spiral motion

(b) 3D view boustrophedon motion

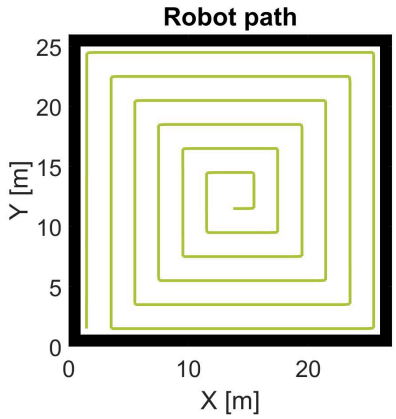
(c) 2D view spiral motion

(d) 2D view boustrophedon motion

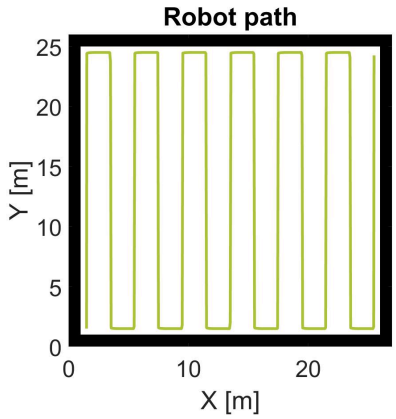
Figure 5.2: UV % estimated during the complete coverage path planning of Test 1

5.2.3 Simulation analysis

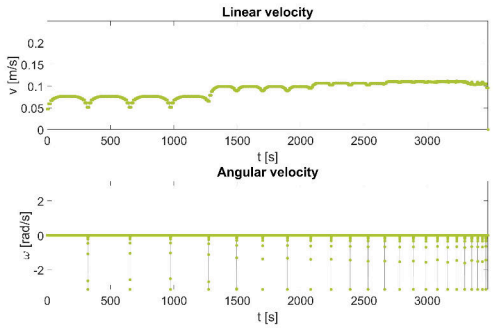
The paths followed by the robot in each situation are shown in Figure 5.3. They are two routes, which do not present any problems during the entire experiment. It is important to highlight the different speed settings used in each case. The robot that uses spiral motion has a gradual increase as it moves closer to the central point. Considering boustrophedon motion, the speed presents a more uniform profile during the path. There are times where the speed decreases in Figure 5.3 (d). This speed is associated with the areas where the robot is approaching a wall.



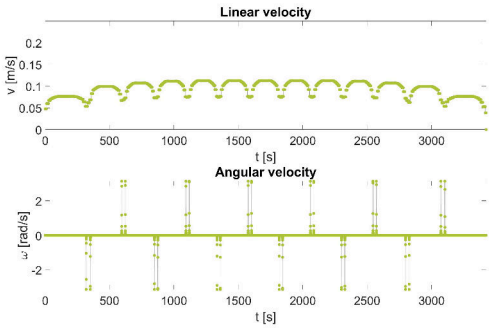
(a) Path with spiral motion



(b) Path with boustrophedon motion



(c) Speed with spiral motion



(d) Speed with boustrophedon motion

Figure 5.3: Robot path and speed for Test 1

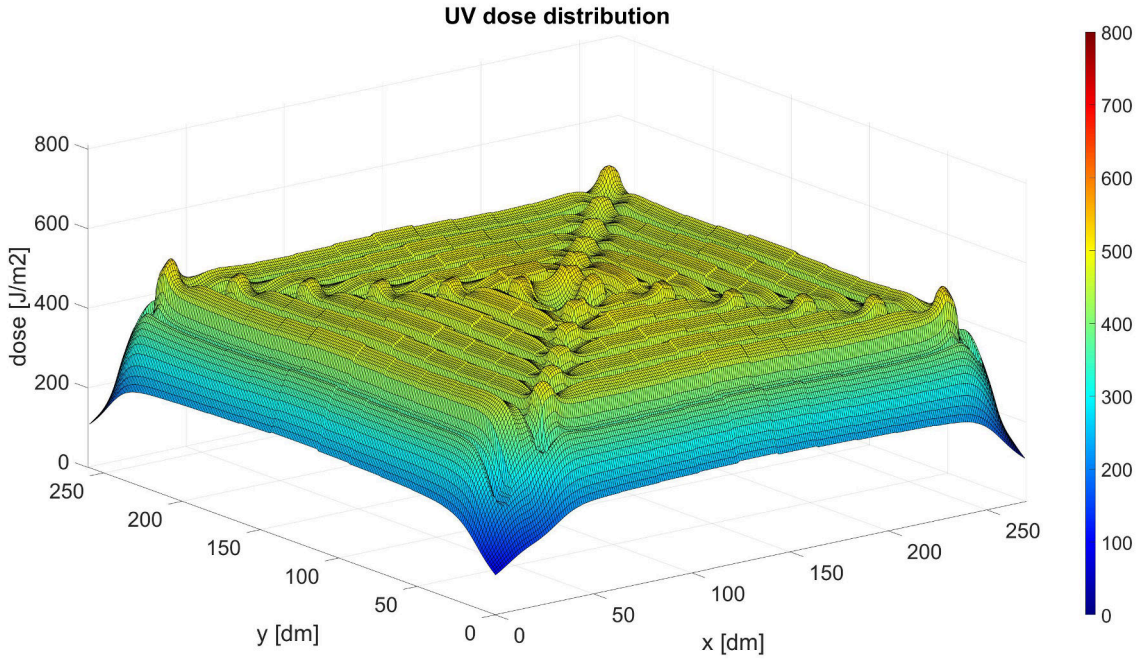
As can be seen in Table 5.7, the generated paths have a similar distance. In fact, the distance is smaller using spiral motion. However, due to the dose estimation obtained during the GBNN algorithm, it requires more time. In both cases, v_{av} is $0.1\,m/s$.

Table 5.7: Robot metrics for Test 1.

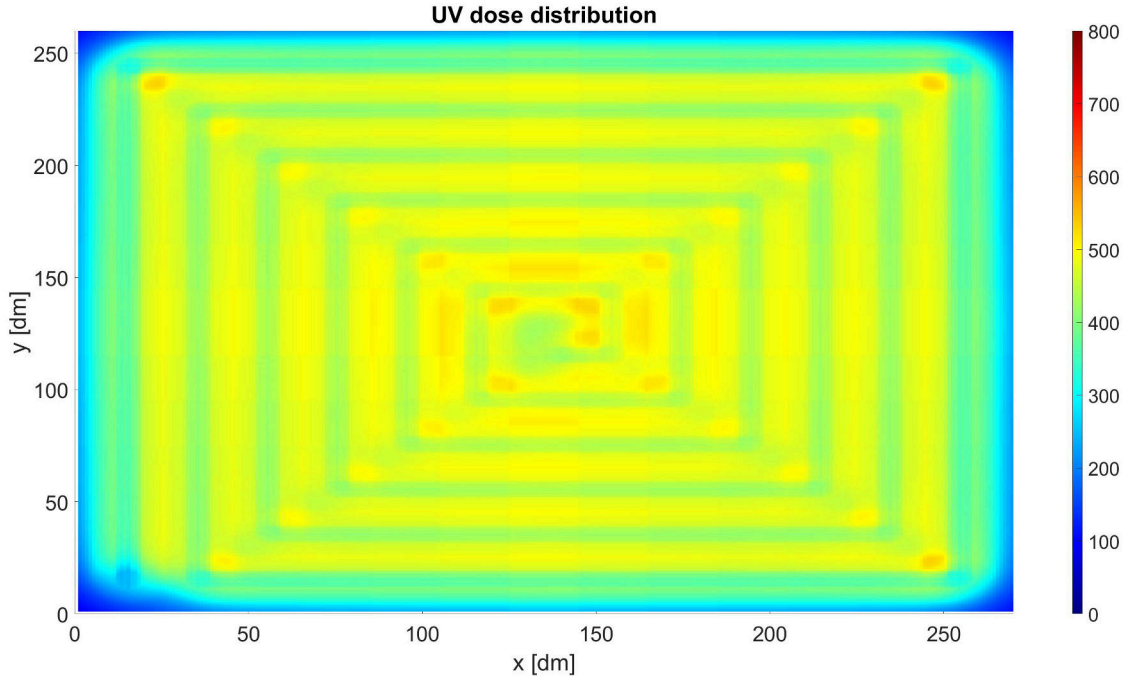
	$\mathbf{n_{goals}}$	$\mathbf{d_{robot}}$	$\mathbf{r_{robot}}$	$\mathbf{v_{max}}$	$\mathbf{v_{min}}$	$\mathbf{v_{av}}$
Spiral	25	320.22	36.13	0.10	0.04	0.10
Boustrophedon	27	321.03	37.69	0.10	0.03	0.10

5.2.4 UV disinfection analysis

Figure 5.4 shows a very uniform dose distribution. It is possible to see where the robot passed since the disinfection is lower, describing a kind of lane. The disinfection is lower closer to the walls, and the peaks that appear are caused when the robot turns.



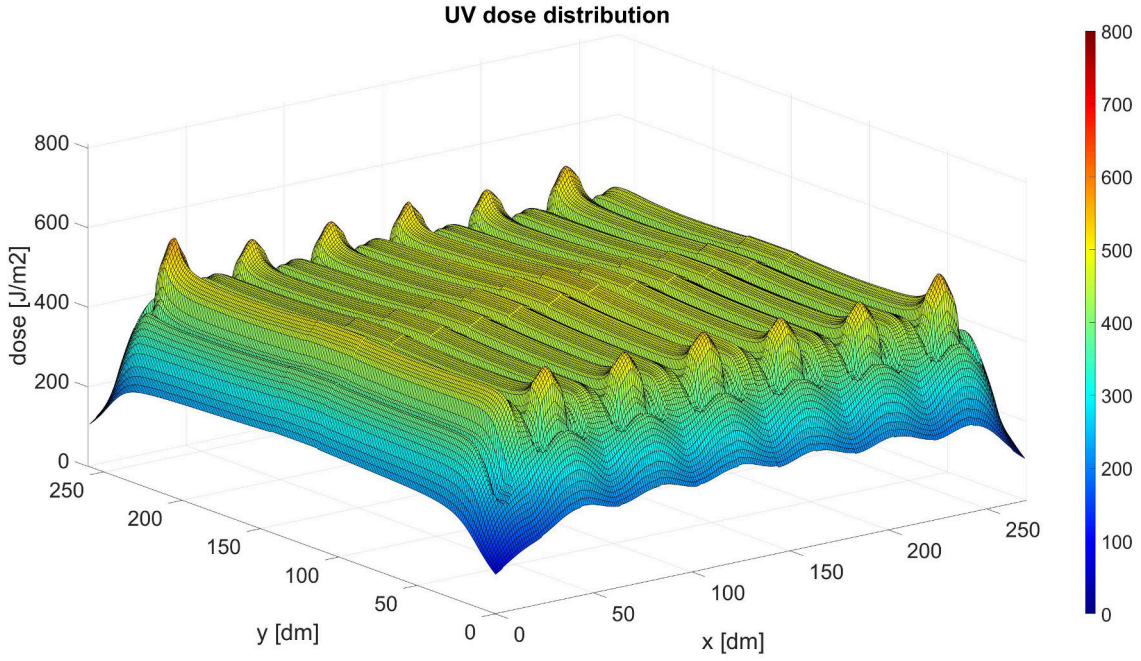
(a) 3D view



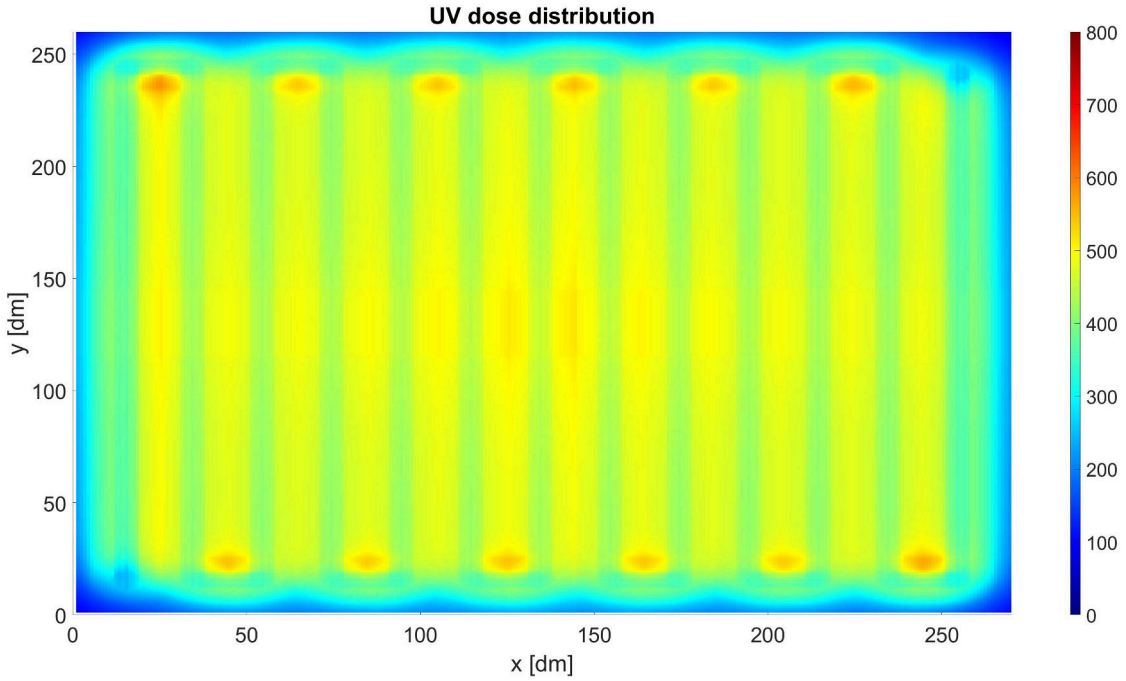
(b) 2D view

Figure 5.4: D_{UV} received with spiral motion for Test 1

Figure 5.5 shows a very uniform dose distribution. It is possible to see where the robot passed since the disinfection is lower, like in the previous case. The disinfection is lower, closer to the walls too. Moreover, the peaks that appear are higher than using spiral motion, thanks to the distinctive turn of this motion technique.



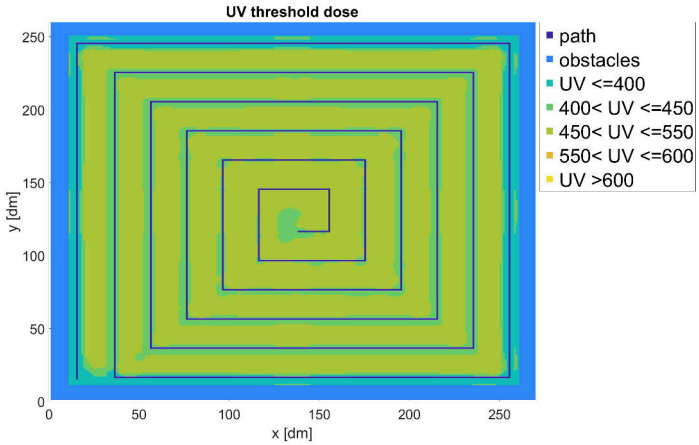
(a) 3D view



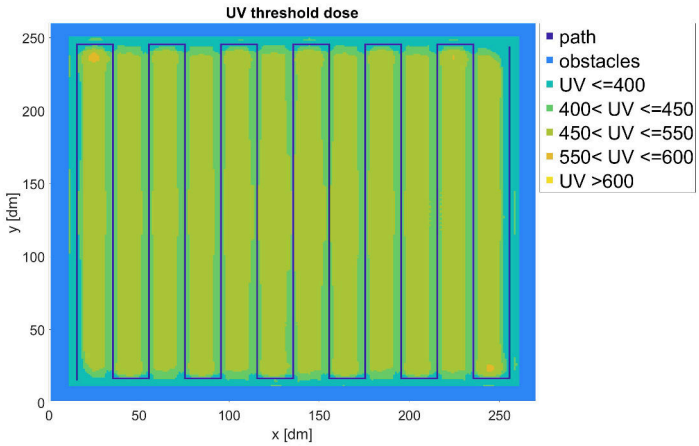
(b) 2D view

Figure 5.5: D_{UV} received with boustrophedon motion for Test 1

Lastly, in Figure 5.6, it is confirmed that the radiation received is mostly inside the expected range. It is evident where the robot has passed because the dose received is lower. Similarly, near the walls, the desired dose is not reached.



(a) Spiral motion



(b) Boustrophedon motion

Figure 5.6: D_{UV} received categorised into thresholds for Test 1

Table 5.8 depicts that the environment has been disinfected as expected. The areas that exceed the threshold are nearly non-existent. As mentioned above, it is near the walls where the dose is lower. Using spiral motion achieves a higher percentage within the expected range, although the result with boustrophedon motion is very similar. However, it should also be noted that the $D_{max} - D_{min}$ difference is less in the spiral path.

Table 5.8: Disinfection metrics for Test 1.

	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{D_{VL}}$	$\mathbf{D_{L}}$	$\mathbf{D_{N}}$	$\mathbf{D_{H}}$	$\mathbf{D_{VH}}$
Spiral	534.32	234.81	11.85%	23.83%	64.32%	0.00%	0.00%
Boustrophedon	582.11	183.77	11.62%	28.51%	59.78%	0.09%	0.00%

5.3 Test 2: Narrow passage map

5.3.1 General

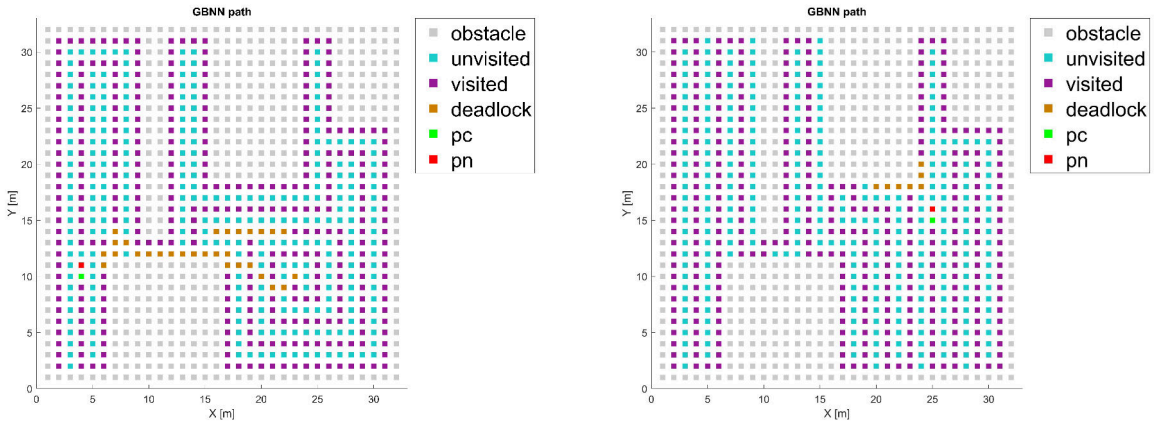
This map is available at [15]. In this workspace, the robot must access different corridors. This map makes the disinfection work harder if it is compared with the previous scenario. In Table 5.9 it is shown how similar are calculation times for both motion techniques.

Table 5.9: Time measurement for Test 2.

	$\mathbf{t_{CPU}}$	$\mathbf{t_{sim}}$	$\mathbf{t_{GBNN}}$	$\mathbf{t_{robot}}$
Spiral	00 : 02 : 57	00 : 01 : 54	00 : 00 : 08	01 : 14 : 50
Boustrophedon	00 : 02 : 55	00 : 01 : 51	00 : 00 : 08	01 : 13 : 30

5.3.2 GBNN algorithm analysis

In Figure 5.7, it can be seen how the robot that uses spiral motion presents more complications. This is due to the fact that it passes to a greater disinfection area leaving another area unvisited. This is not the case with the boustrophedon technique, because thanks to its distinctive movement, it can disinfect the entire workspace, minimizing the transit through already disinfected areas.



(a) Spiral

(b) Boustrophedon

Figure 5.7: Path for Test 2, generated by complete coverage path planning

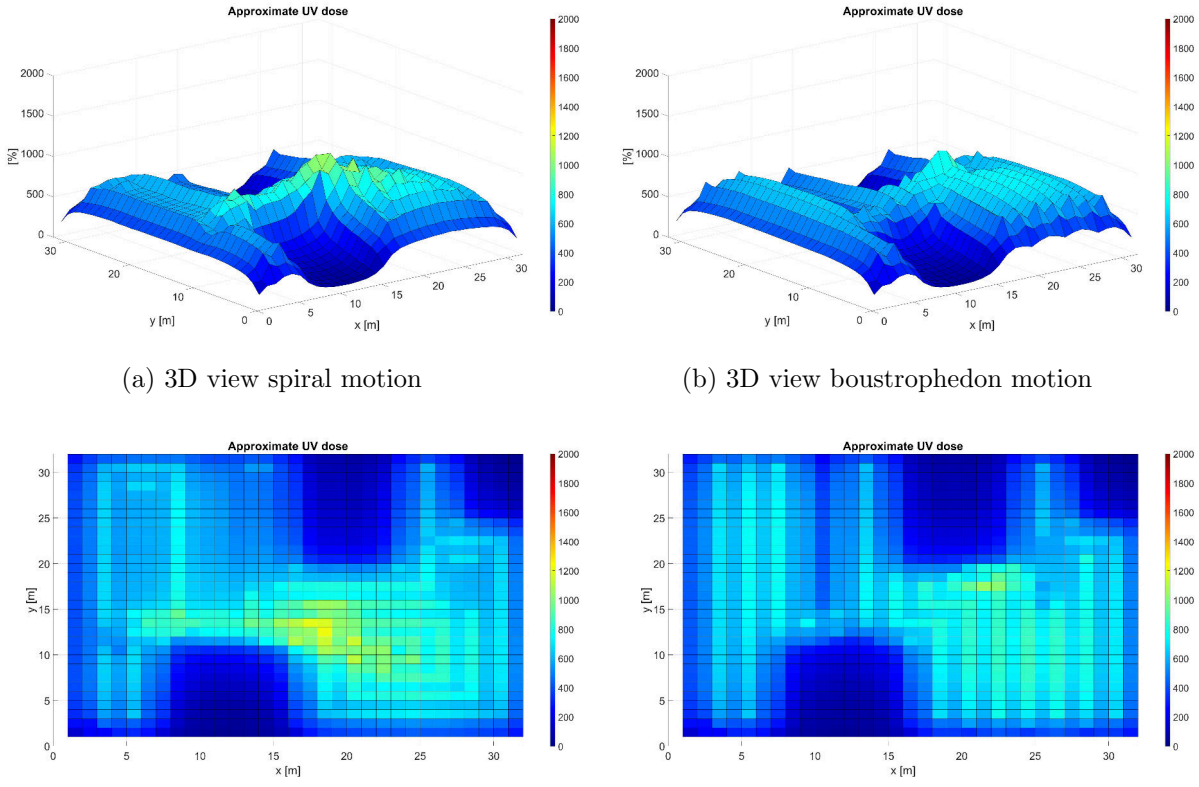
The above mentioned implies a higher number of iterations when using spiral motion, as shown in Table 5.10. Also, with this technique, it has been necessary to face deadlock situations on more occasions.

Table 5.10: GBNN algorithm metrics for Test 2 with spiral motion.

	n_{GBNN}	$n_{Dijkstra}$	d_e	r_e
Spiral	361	3	362.63	70.69
Boustrophedon	345	1	345	65.97

In Figure 5.8, it can be perceived that the path generated by the algorithm using boustrophedon motion obtains a more balanced dose distribution.

The peak that appears using spiral motion should be corrected by the speed control. This overshoot of radiation is caused because the robot needs to pass again through a previously visited area. This type of problem increases as the complexity of the maps grows.



(a) 3D view spiral motion

(b) 3D view boustrophedon motion

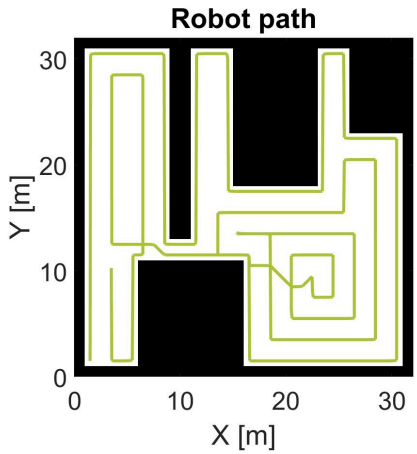
(c) 2D view spiral motion

(d) 2D view boustrophedon motion

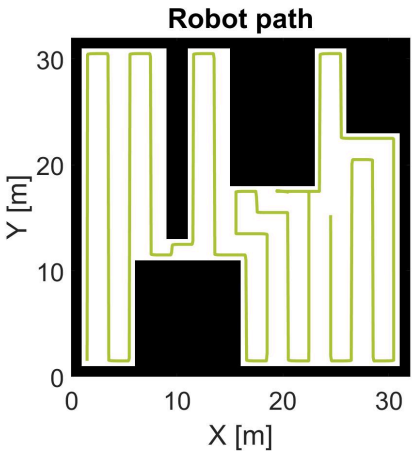
Figure 5.8: UV % estimated during the complete coverage path planning of Test 2

5.3.3 Simulation analysis

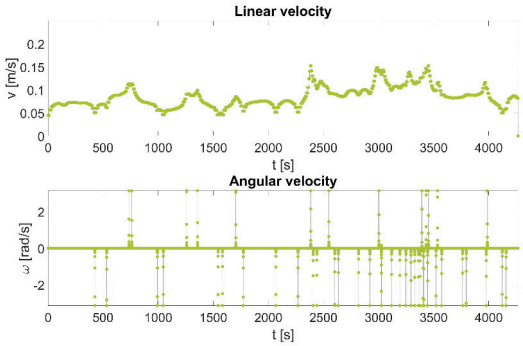
The real routes pursued by the robots are shown in Figure 5.9. Despite the simplicity of the map, the need to use Dijkstra's algorithm to escape from deadlock situations is evident. This fact can be seen, as said before, in the map generated with a spiral motion.



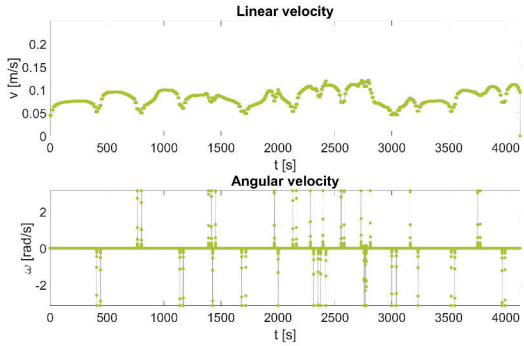
(a) Path with spiral motion



(b) Path with boustrophedon motion



(c) Speed with spiral motion



(d) Speed with boustrophedon motion

Figure 5.9: Robot path and speed for Test 2

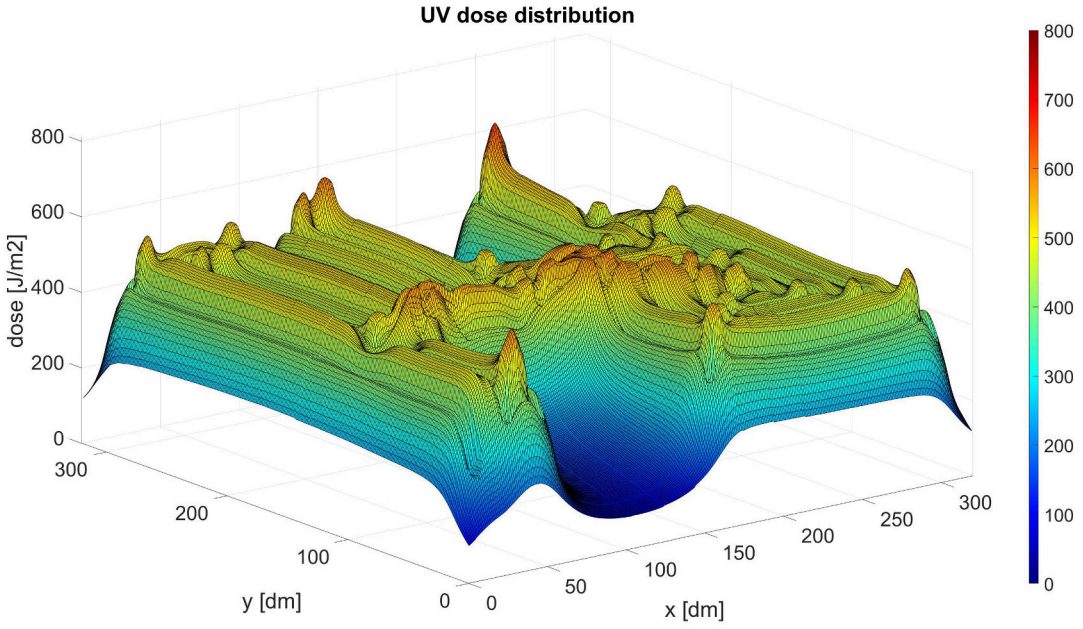
As can be seen in Table 5.11, the route generated by boustrophedon motion is shorter. As reflected in Table 5.9, this means that less time is required to complete the disinfection coverage. The average speed in both cases is the same. However, there are more peaks in spiral motion because of the narrowing between the two main areas of the map.

Table 5.11: Robot metrics for Test 2.

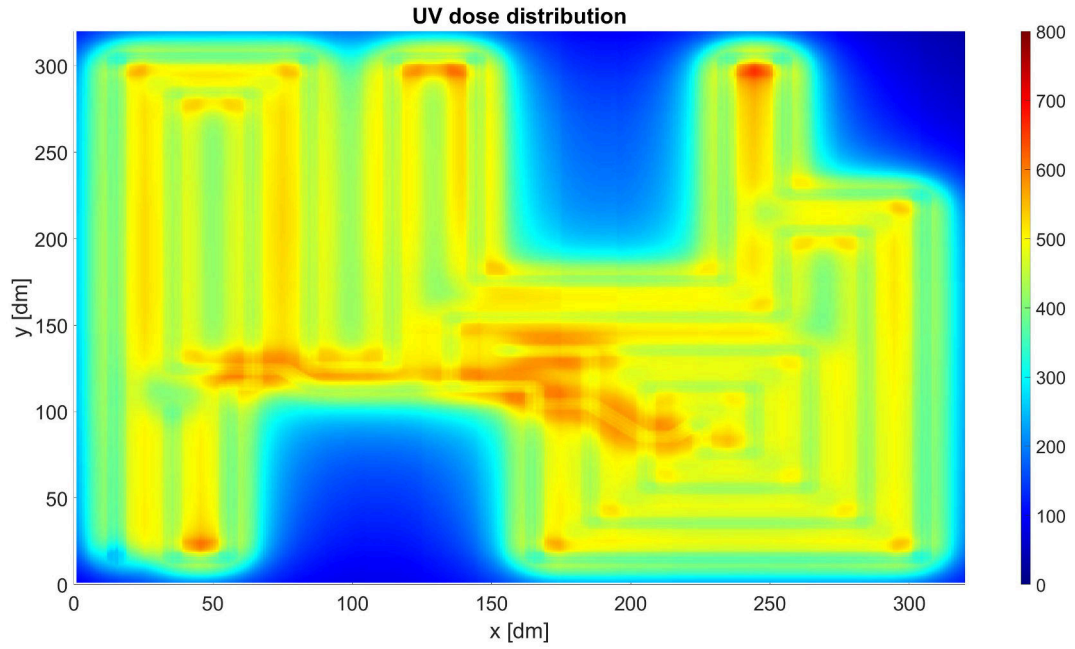
	n_{goals}	d_{robot}	r_{robot}	v_{max}	v_{min}	v_{av}
Spiral	49	359.96	71.22	0.16	0.04	0.08
Boustrophedon	42	342.46	67.42	0.13	0.04	0.08

5.3.4 UV disinfection analysis

Figure 5.10 details the final dose received in the workspace once the complete coverage path planning is completed. Although it has already been mentioned, a higher amount of the dose received appears in the narrowing zone. Even though the speed controller manages to reduce the final dose, there are still peaks in that area.



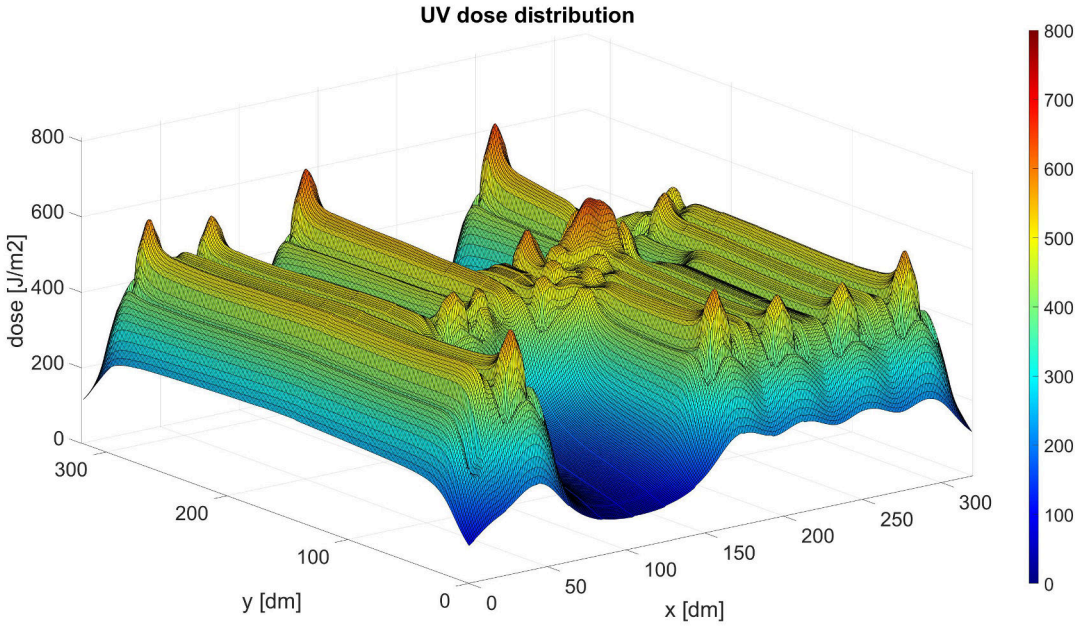
(a) 3D view



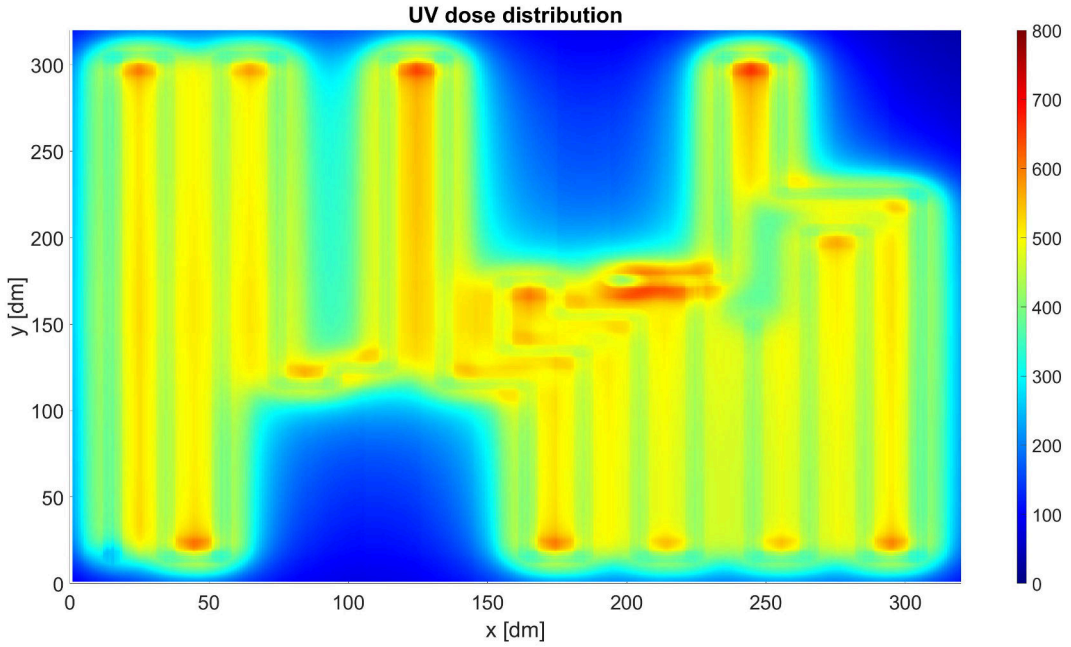
(b) 2D view

Figure 5.10: D_{UV} received with spiral motion for Test 2

For the path with boustrophedon motion (Figure 5.11), the dose received is more uniform, showing the typical peaks of each turn. As in the previous case, it is also worth mentioning the section where a deadlock appears whose received dose is higher than in the rest of the regions.



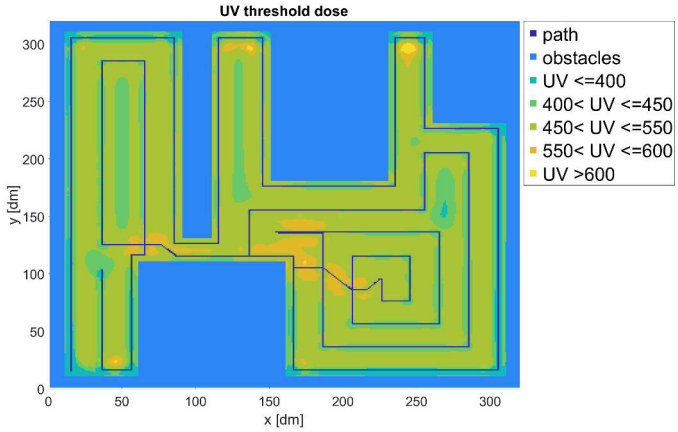
(a) 3D view



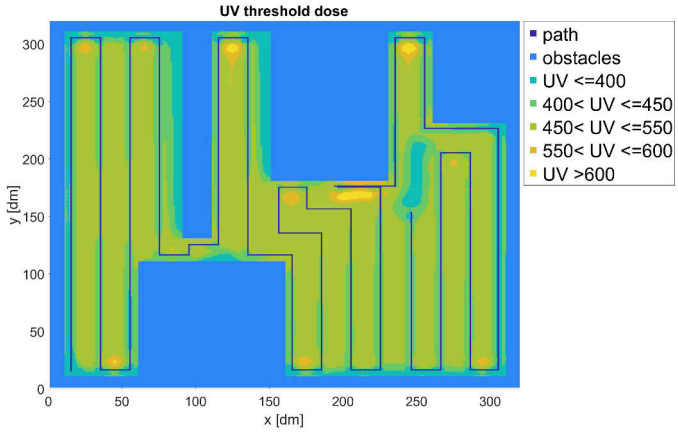
(b) 2D view

Figure 5.11: D_{UV} received with boustrophedon motion for Test 2

Lastly, in Figure 5.12, it can be seen that more areas exceed the threshold for the path generated by spiral motion (deadlock situation). In both cases, there are areas below the expected threshold. However, the area with a low dose received that appears using boustrophedon motion is remarkable. This situation occurs because the radiation overlap in these cells is not the same as in the others.



(a) Spiral motion



(b) Boustrophedon motion

Figure 5.12: D_{UV} received categorised into thresholds for Test 2

Table 5.12 summarises the comments above. Both options provide a reasonable solution for the disinfection process. Also, the use of spiral motion results in better disinfection, even though it requires more time. Besides, the maximum received dose is similar, but the minimum is lower when using boustrophedon motion.

Table 5.12: Disinfection metrics for Test 2.

	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{D_{VL}}$	$\mathbf{D_{L}}$	$\mathbf{D_{N}}$	$\mathbf{D_{H}}$	$\mathbf{D_{VH}}$
Spiral	659.89	243.38	6.45%	25.45%	64.30%	3.65%	0.15%
Boustrophedon	658.20	179.61	11.06%	28.00%	58.94%	1.57%	0.43%

5.4 Test 3: Simple office map

5.4.1 General

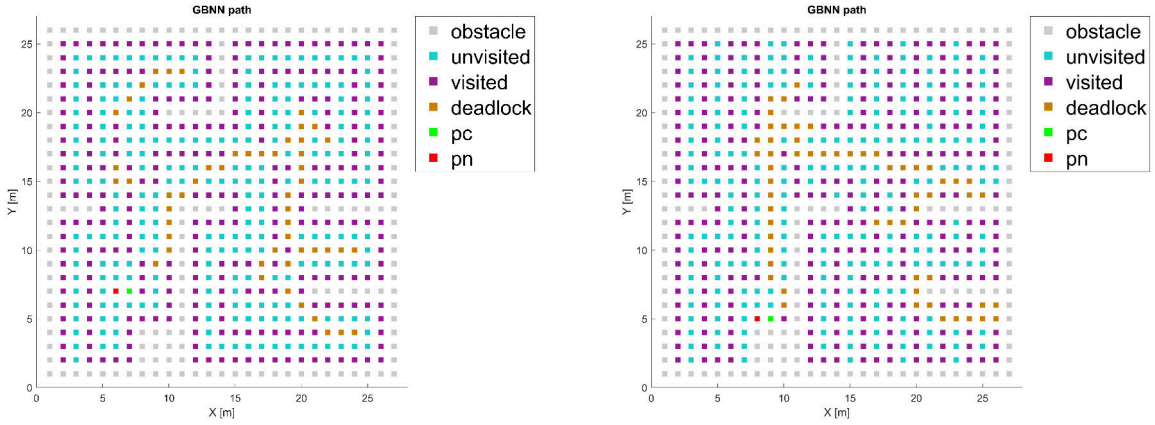
In Test 3, the workspace is used to simulate a realistic environment. This distribution could be found in homes or offices. As in the previous tests, the times are quite similar, but there is still a lower t_{robot} when using boustrophedon motion.

Table 5.13: Time measurement for Test 3.

	$\mathbf{t_{CPU}}$	$\mathbf{t_{sim}}$	$\mathbf{t_{GBNN}}$	$\mathbf{t_{robot}}$
Spiral	00 : 02 : 10	00 : 01 : 16	00 : 00 : 06	01 : 00 : 48
Boustrophedon	00 : 02 : 11	00 : 01 : 14	00 : 00 : 06	00 : 58 : 07

5.4.2 GBNN algorithm analysis

On this occasion, considering the distribution of obstacles on the map, it has been necessary to escape from deadlock situations, independently of the motion used. In Figure 5.13, it can be seen that both paths are very similar. The use of both movement techniques requires the use of Dijkstra's algorithm, covering previously visited areas.



(a) Spiral

(b) Boustrophedon

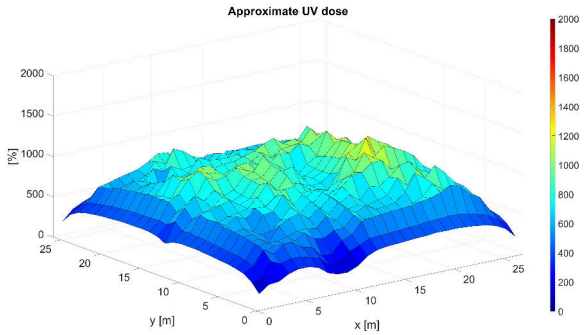
Figure 5.13: Path for Test 3, generated by complete coverage path planning

The robot using boustrophedon motion requires more iterations to perform the coverage. Nevertheless, this does not suppose a problem to spend less time in the route, as it was already seen in Table 5.13. Both robots must deal with deadlock situations, as reflected in Table 5.14. With spiral motion, the number of deadlock situations is higher than with boustrophedon motion. The paths affected by this condition are longer.

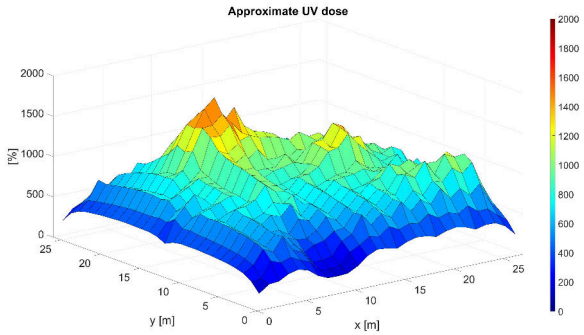
Table 5.14: GBNN algorithm metrics for Test 3 with spiral motion.

	$\mathbf{n_{GBNN}}$	$\mathbf{n_{Dijkstra}}$	$\mathbf{d_e}$	$\mathbf{r_e}$
Spiral	341	5	348.46	109.96
Boustrophedon	366	4	369.73	155.51

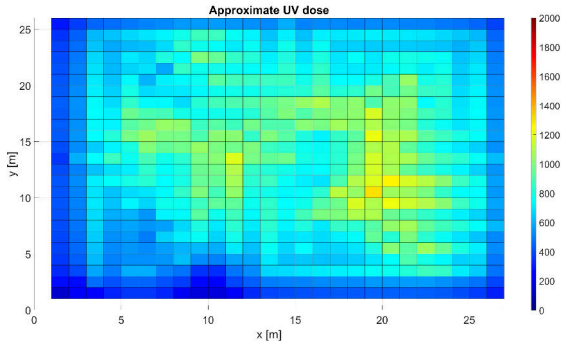
In Figure 5.14, when using spiral motion, excesses of radiation can be detected spread over the map. On the other hand, the robot that uses boustrophedon motion has a single zone where all the excess dose is located. Although the speed controller will attempt to reduce it, in Figure 5.17, it can be shown that this region has the highest concentration considering the rest of the map.



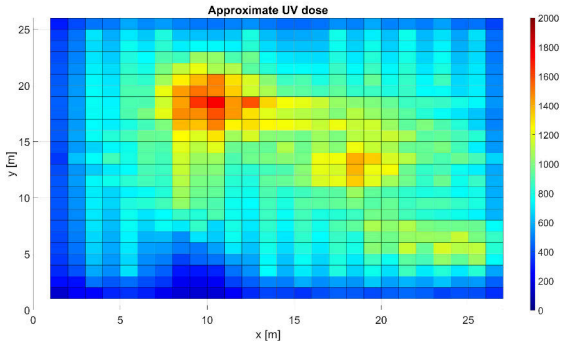
(a) 3D view spiral motion



(b) 3D view boustrophedon motion



(c) 2D view spiral motion

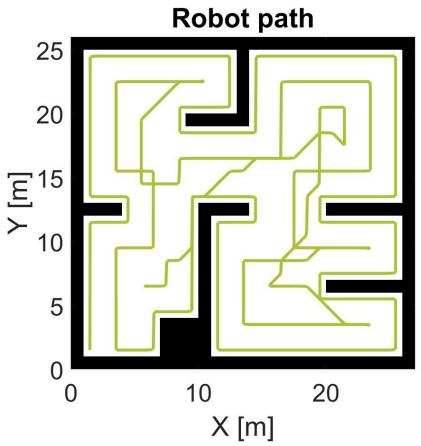


(d) 2D view boustrophedon motion

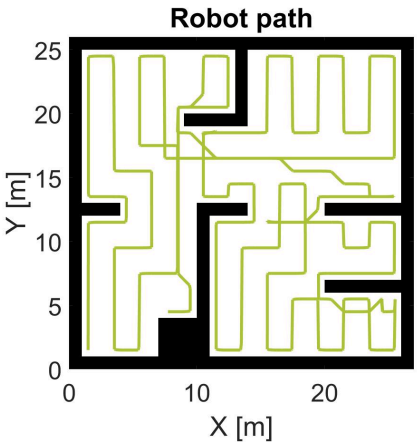
Figure 5.14: UV % estimated during the complete coverage path planning of Test 3

5.4.3 Simulation analysis

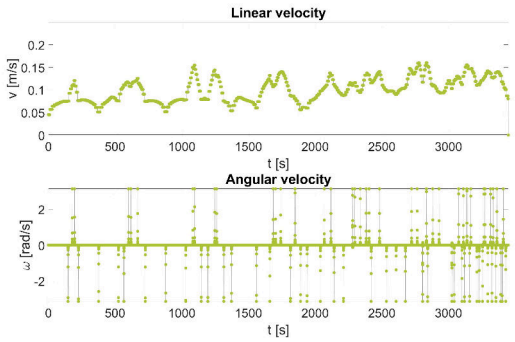
The paths followed by each robot are shown in Figure 5.15. Both routes, each with a different motion technique, achieve to cover the whole space. However, neither succeeds in covering the entire workspace without having to pass through previously visited areas. The map generated with boustrophedon movements has higher peaks in the speed signal. This circumstance is caused because the paths affected by the deadlocks have a greater distance than the existing ones with spiral motion.



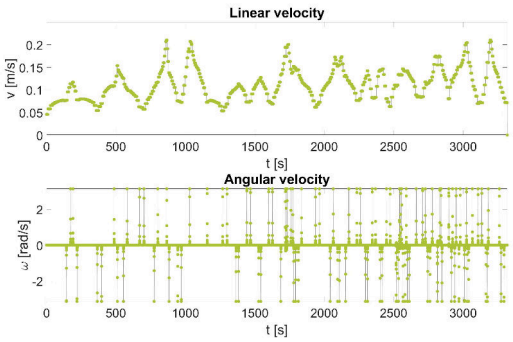
(a) Path with spiral motion



(b) Path with boustrophedon motion



(c) Speed with spiral motion



(d) Speed with boustrophedon motion

Figure 5.15: Robot path and speed for Test 3

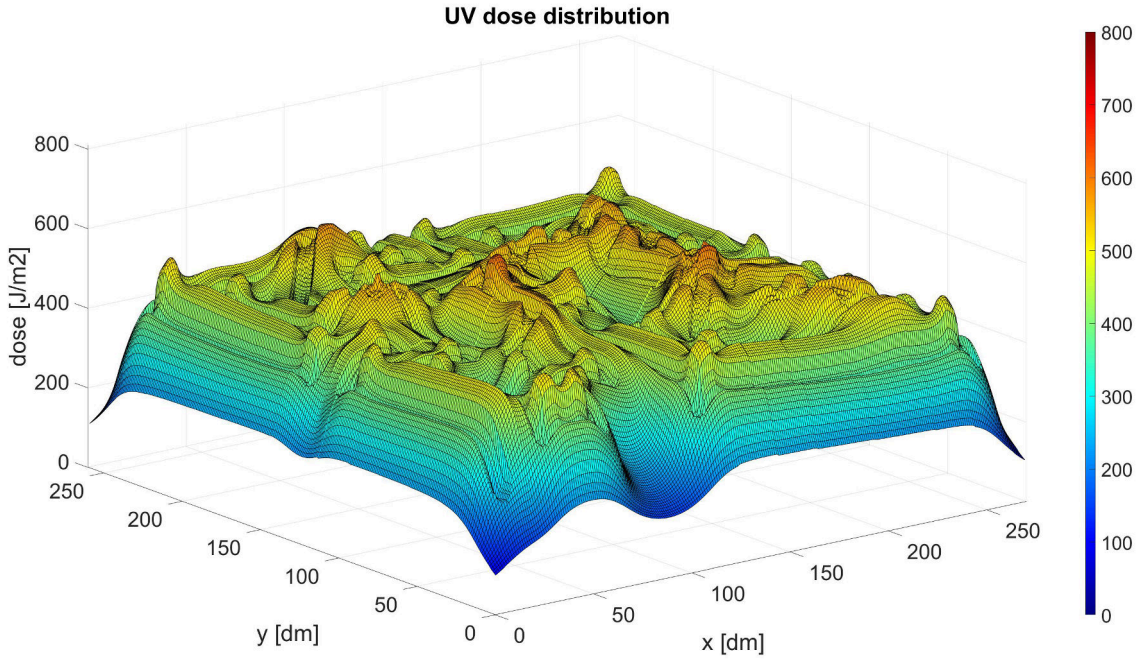
In Table 5.15, it can be seen that, with boustrophedon motion, a longer distance is covered. A significant number of turns can be seen, for both cases. Therefore, as in all other cases, the first robot to finish the route is the one with the highest speed.

Table 5.15: Robot metrics for Test 3.

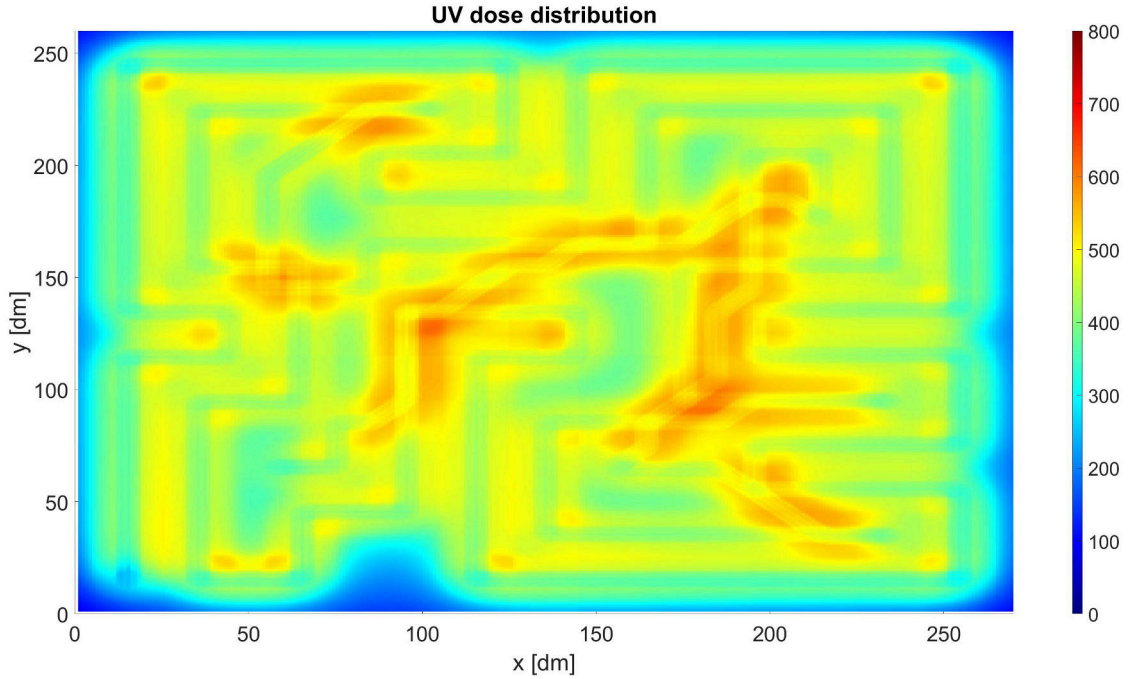
	$\mathbf{n_{goals}}$	$\mathbf{d_{robot}}$	$\mathbf{r_{robot}}$	$\mathbf{v_{max}}$	$\mathbf{v_{min}}$	$\mathbf{v_{av}}$
Spiral	78	344.34	111.58	0.18	0.04	0.09
Boustrophedon	101	365.04	160.51	0.23	0.04	0.10

5.4.4 UV disinfection analysis

In Figure 5.16, it can be appreciated that the areas which receive more radiation are those registered in Figure 5.14. The speed controller can smooth these regions but does not achieve the expected accuracy.



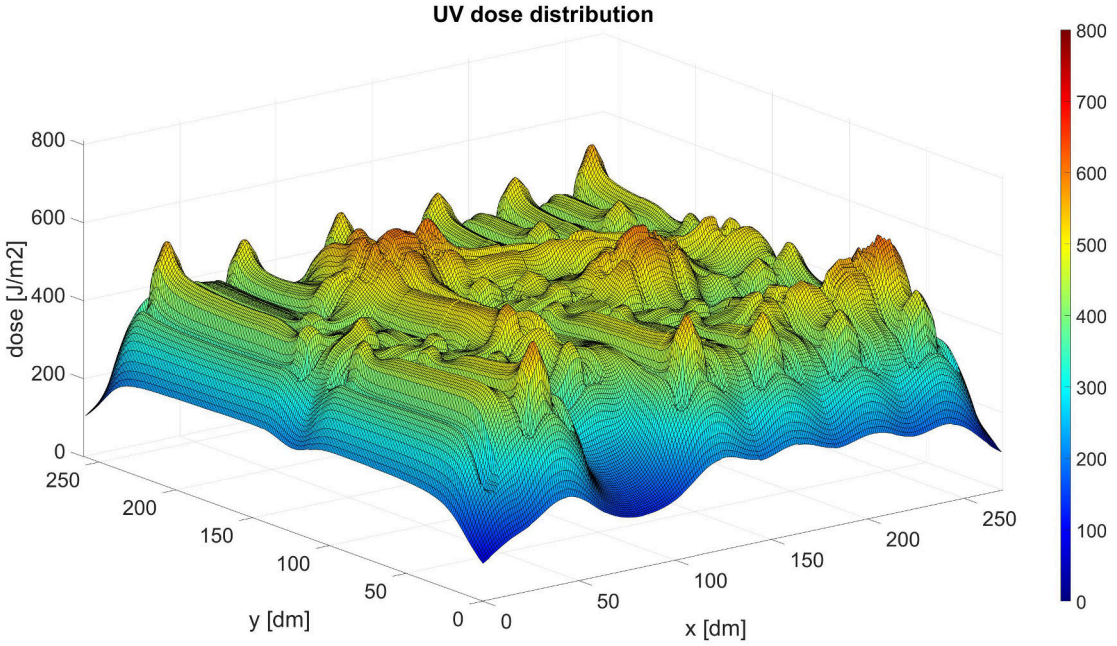
(a) 3D view



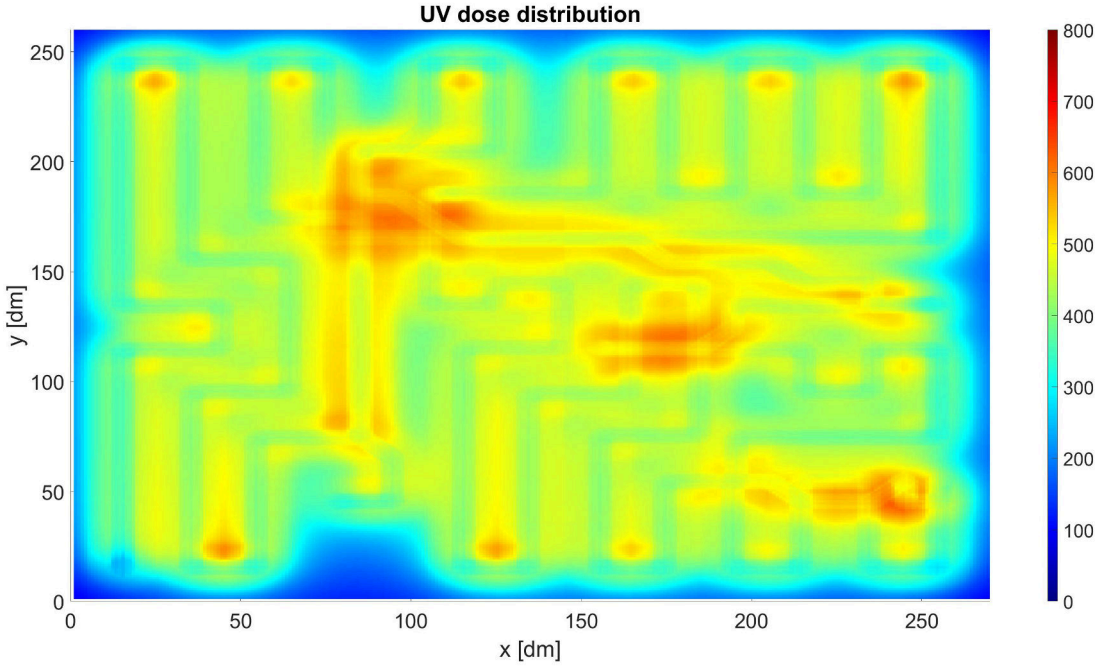
(b) 2D view

Figure 5.16: D_{UV} received with spiral motion for Test 3

Similarly, Figure 5.17 shows an equivalent result. Even though the excessive amount of radiation seen in 5.14 was reduced, the problem is the same as with spiral motion. The speed control reduces the dose received in that area, but it exceeds the desired quantity. This fact is enhanced as the map complexity grows, as will be seen in the next tests.



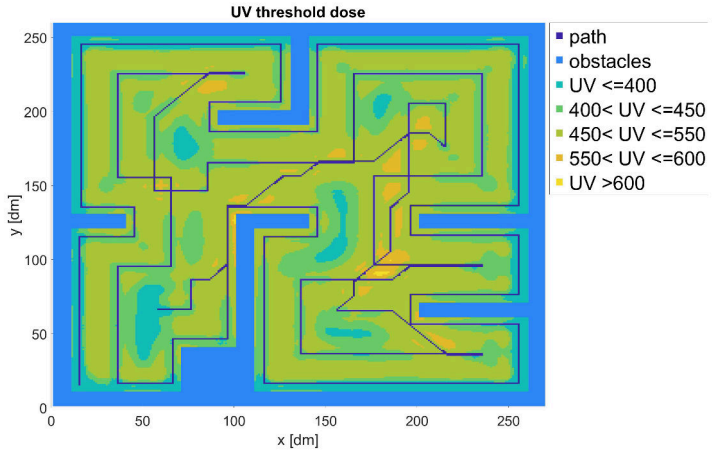
(a) 3D view



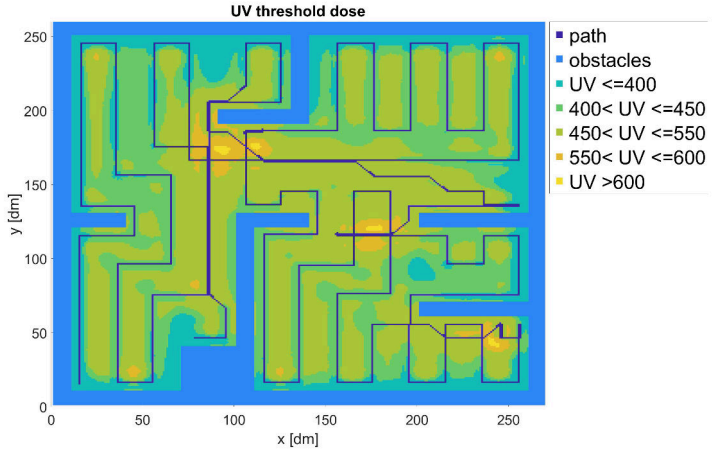
(b) 2D view

Figure 5.17: D_{UV} received with boustrophedon motion for Test 3

Analysing the Figure 5.18, in both cases the expected threshold is exceeded by a similar percentage. However, in the first case, it is possible to be closer to the expected dose in those areas near the room walls. As shown in Figure 5.13, the areas that do not reach this limit correspond to obstacles. However, a glance at the figures shows that the boustrophedon technique produces considerably more areas below the limit. Basically, there are also these areas using spiral motion, but a priori, it seems a better solution.



(a) Spiral motion



(b) Boustrophedon motion

Figure 5.18: D_{UV} received categorised into thresholds for Test 3

As summarised in Table 5.16, with spiral motion results are better. D_N exceeds 50%, and the difference $D_{max} - D_{min}$ is less than using boustrophedon motion.

Table 5.16: Disinfection metrics for Test 3.

	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{D_{VL}}$	$\mathbf{D_L}$	$\mathbf{D_N}$	$\mathbf{D_H}$	$\mathbf{D_{VH}}$
Spiral	612.29	242.06	14.72%	27.87%	55.08%	2.28%	0.05%
Boustrophedon	624.294	195.61	17.40%	33.93%	45.75%	2.72%	0.20%

5.5 Test 4: Complex office map

5.5.1 General

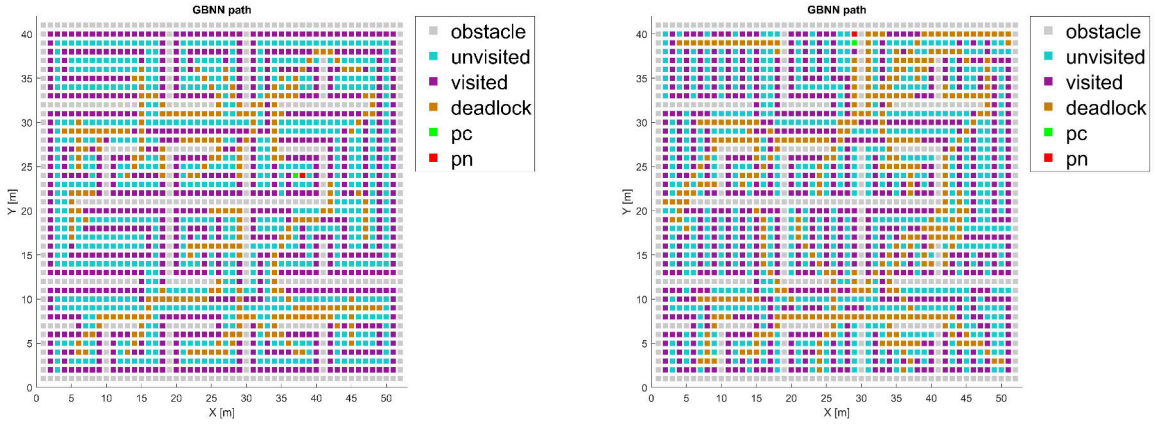
The map for Test 4 displays a similar distribution of obstacles as the previous one. However, the complexity in this test is higher. In Table 5.17, a significant time increase can be found comparing with previous maps. Due to the map complexity, in both cases, the robot would take around 3 hours to finish the disinfection process.

Table 5.17: Time measurement for Test 4.

	$\mathbf{t_{CPU}}$	$\mathbf{t_{sim}}$	$\mathbf{t_{GBNN}}$	$\mathbf{t_{robot}}$
Spiral	00 : 09 : 05	00 : 06 : 45	00 : 01 : 03	02 : 56 : 35
Boustrophedon	00 : 09 : 26	00 : 06 : 46	00 : 01 : 21	02 : 51 : 44

5.5.2 GBNN algorithm analysis

This map has a significant number of rooms. In Figure 5.19, the different paths followed by each robot appear. In both cases, it can be seen that the robot passes into a particular room and disinfects it. However, to leave and travel to another room, it needs to use Dijkstra's algorithm, as a deadlock situation occurs. In this environment, the true potential of the proposed algorithm is tested.



(a) Spiral

(b) Boustrophedon

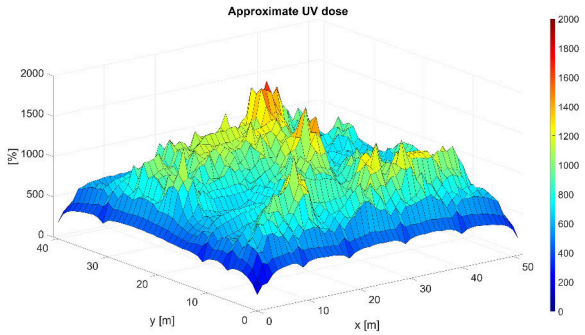
Figure 5.19: Path for Test 4, generated by complete coverage path planning

As shown in Table 5.18, just like in previous tests, the fact of having longer distances does not imply longer experiment duration when mean the robot uses boustrophedon motion. This circumstance is because the disinfection robot can go faster through previously disinfected areas.

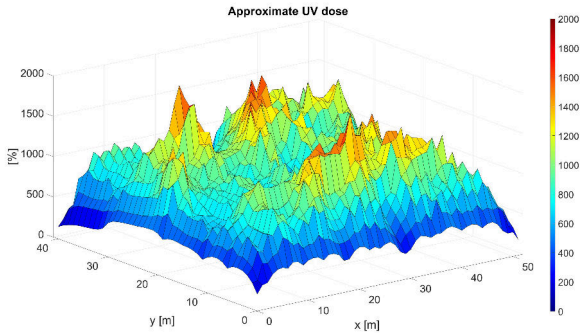
Table 5.18: GBNN algorithm metrics for Test 4 with spiral motion.

	$\mathbf{n_{GBNN}}$	$\mathbf{n_{Dijkstra}}$	$\mathbf{d_e}$	$\mathbf{r_e}$
Spiral	1175	25	1198.61	402.12
Boustrophedon	1288	28	1329.01	562.35

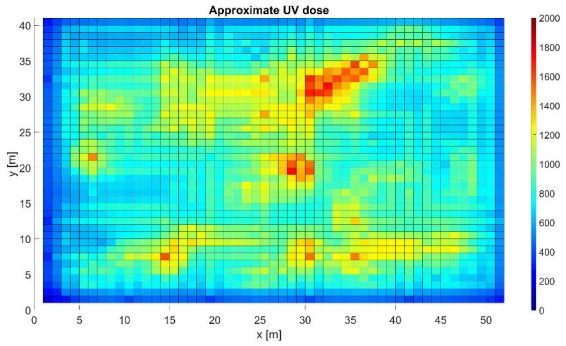
In Figure 5.20, it can be seen that in both situations there are a lot of unwanted peaks. This effect is clearly due to the increase in deadlock situations, besides the number of turns needed to move around the proposed environment. A fact to emphasize is that there are areas on the map in which both experiments present excess radiation. Although, the peaks are higher in the robot that uses boustrophedon motion.



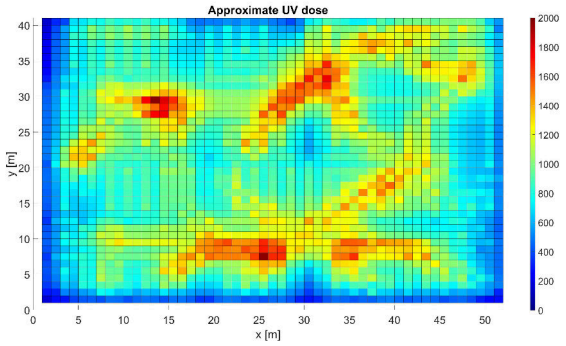
(a) 3D view spiral motion



(b) 3D view boustrophedon motion



(c) 2D view spiral motion

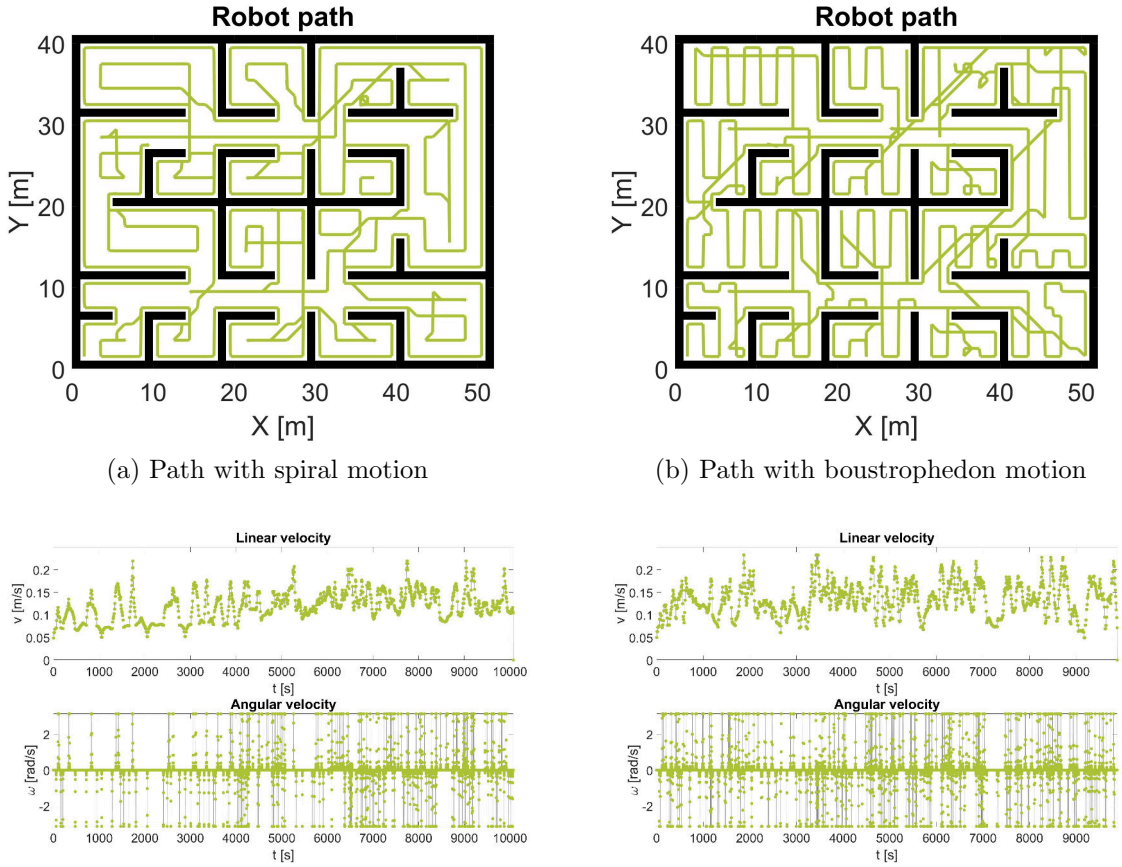


(d) 2D view boustrophedon motion

Figure 5.20: UV % estimated during the complete coverage path planning of Test 4

5.5.3 Simulation analysis

In Figure 4, the path covered by both robots can be seen on the map. Although both routes are similarly complex, it is a fact that the robot with spiral motion uses fewer turns to perform the disinfection. The speed in both cases is also similar. Although when using boustrophedon, the speed is higher, as is usual in all the tests done.



(a) Path with spiral motion

(b) Path with boustrophedon motion

(c) Speed with spiral motion

(d) Speed with boustrophedon motion

Figure 5.21: Robot path and speed for Test 4

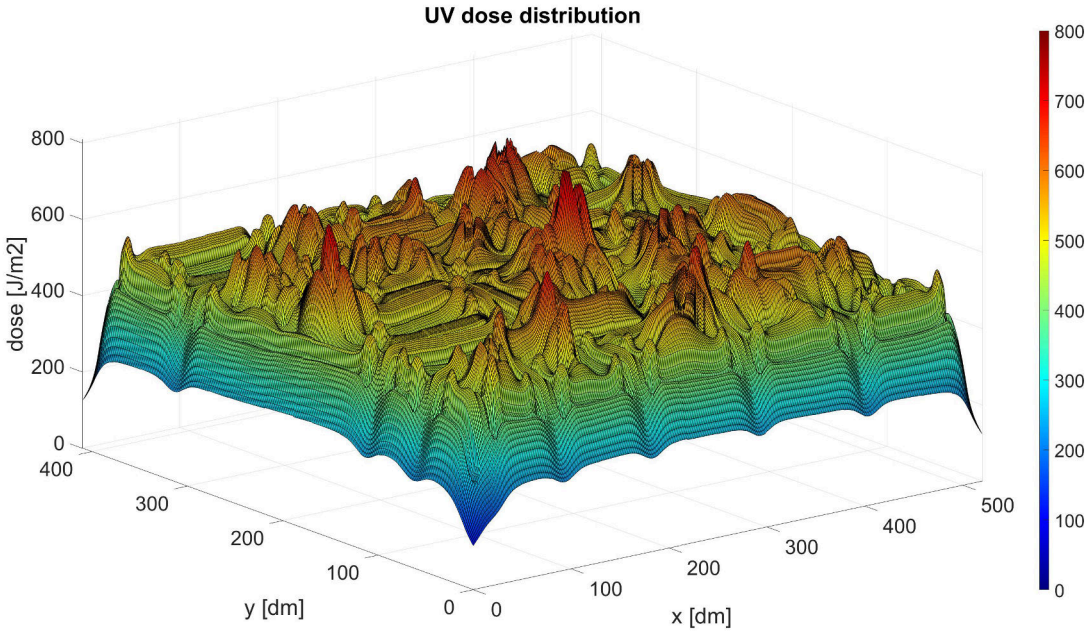
Considering the Table 5.19 and the equivalents in the previous tests, the same evolution could be seen. As the complexity increases, the difference between the distance travelled and the speed used, increases for each type of motion used.

Table 5.19: Robot metrics for Test 4.

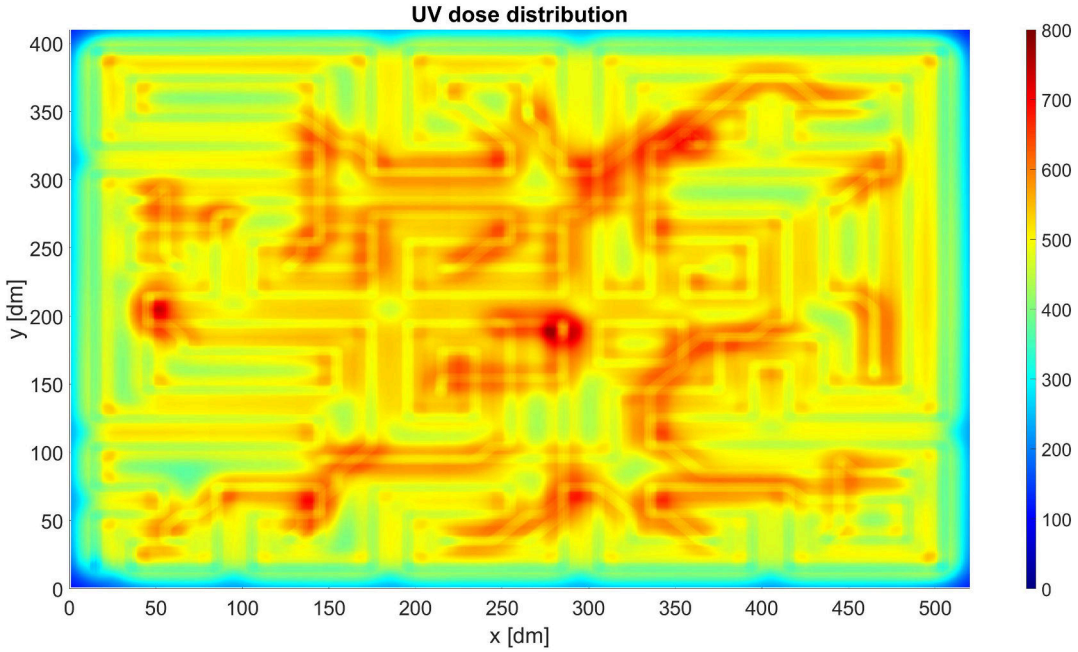
	$\mathbf{n_{goals}}$	$\mathbf{d_{robot}}$	$\mathbf{r_{robot}}$	$\mathbf{v_{max}}$	$\mathbf{v_{min}}$	$\mathbf{v_{av}}$
Spiral	263	1190.47	420.82	0.23	0.04	0.11
Boustrophedon	369	1317.09	584.38	0.25	0.04	0.13

5.5.4 UV disinfection analysis

Due to the complexity of the map, in Figure 5.22, it is evident that the speed controller is not able to smooth the received dose as it might be expected. This result is manifested in a large number of peaks, which appear mainly due to the need to make a large number of turns to disinfect the environment and deadlocks situations.



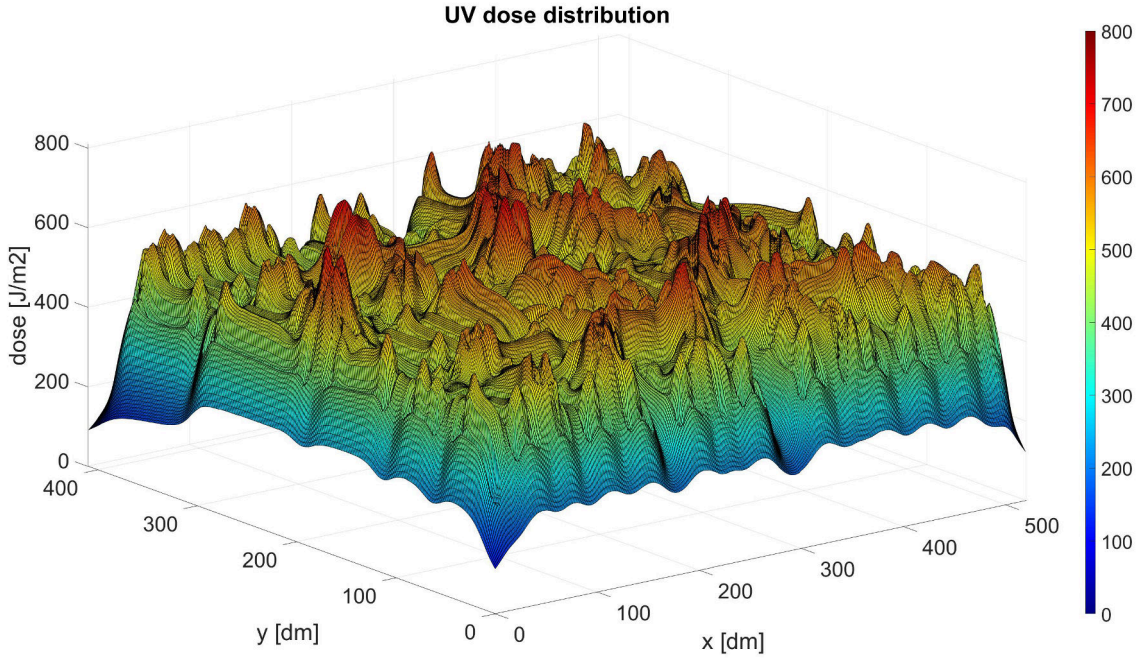
(a) 3D view



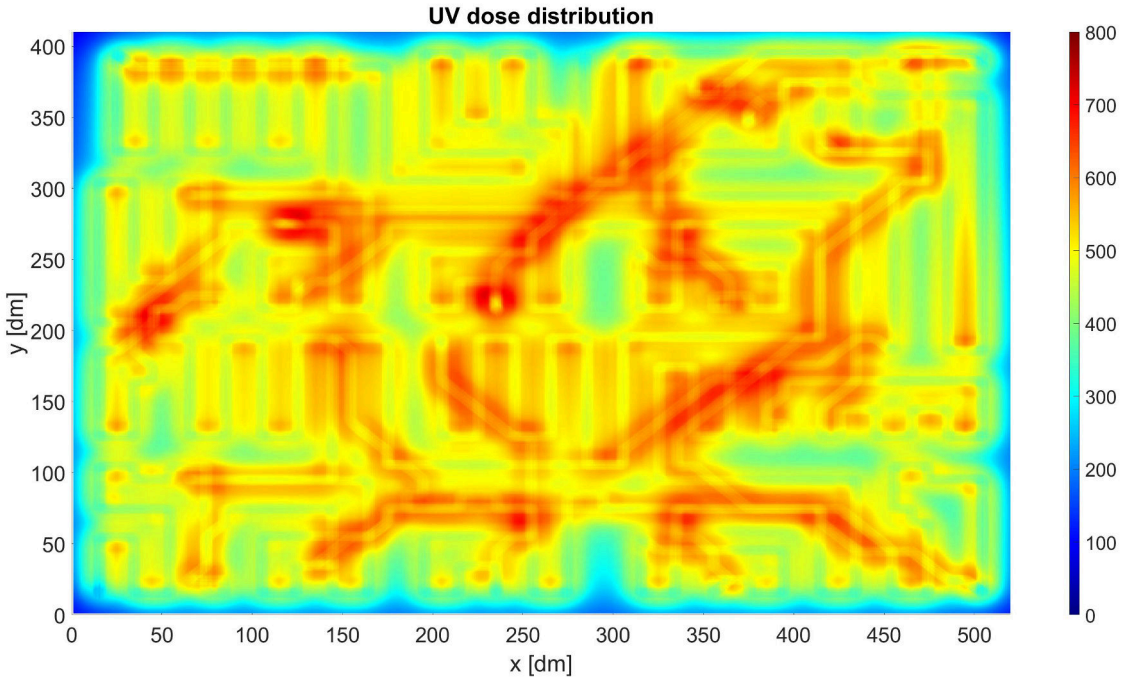
(b) 2D view

Figure 5.22: D_{UV} received with spiral motion for Test 4

Similarly, the undesired effect of the turns is visible as depicted in Figure 5.23. On this route, the number of turns is greater, and they are also closer turns, which causes higher peaks. Also, the unwanted effect of deadlock situations can be seen quite well in the 2D graph.



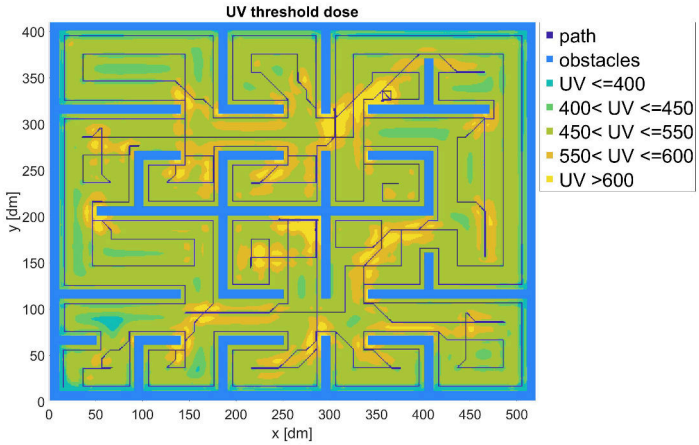
(a) 3D view



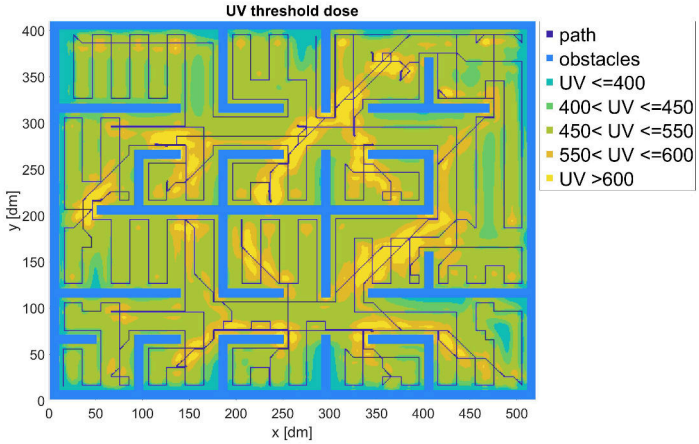
(b) 2D view

Figure 5.23: D_{UV} received with boustrophedon motion for Test 4

The radiation excess is higher than in previous cases, as shown in Figure 5.24. Once the complexity of the map is increased, reaching the desired radiation is not a problem. The matter comes from the regions that exceed this threshold.



(a) Spiral motion



(b) Boustrophedon motion

Figure 5.24: D_{UV} received categorised into thresholds for Test 4

In Table 5.20, it can be seen how almost 20% of the map exceeds the desired threshold. This excess is a problem in terms of energy consumption, as well as in terms of material degradation. On this occasion, the highest peak of radiation is given using spiral motion. However, the $D_{max} - D_{min}$ difference is still lower. Furthermore, as in previous cases, D_N is higher than the value obtained with boustrophedon motion.

Table 5.20: Disinfection metrics for Test 4.

	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{D_{VL}}$	$\mathbf{D_L}$	$\mathbf{D_N}$	$\mathbf{D_H}$	$\mathbf{D_{VH}}$
Spiral	776.04	250.75	3.43%	11.62%	66.91%	14.67%	3.37%
Boustrophedon	722.35	164.19	5.87%	13.60%	58.83%	15.57%	6.13%

5.6 Test 5: Sorting facility warehouse map

5.6.1 General

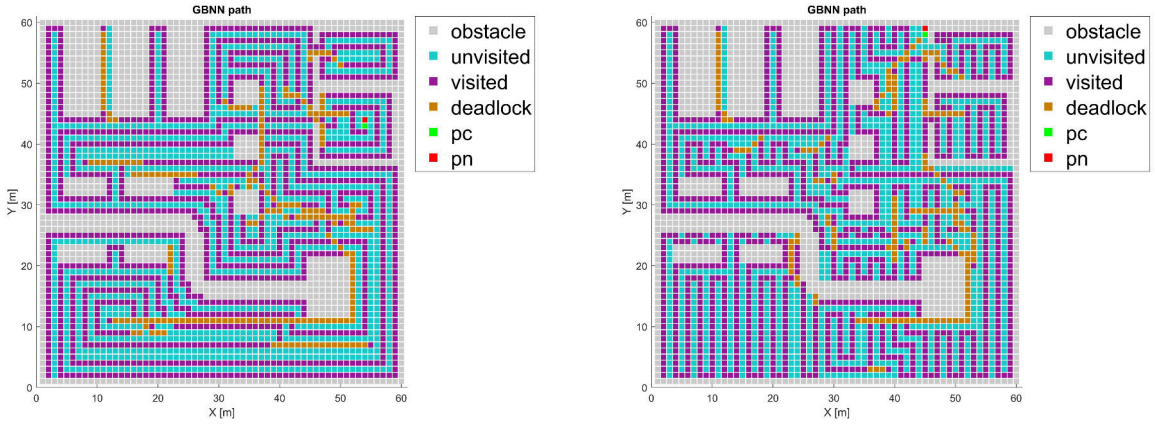
The last map used corresponds to the environment that could be found in some factories. A conveyor, work tables, storage shelves, and small offices can be seen in the workspace. It is the biggest map, although of similar complexity to the previous one. On this occasion, the robot takes more than four and a half hours to complete the disinfection process.

Table 5.21: Time measurement for Test 5.

	$\mathbf{t_{CPU}}$	$\mathbf{t_{sim}}$	$\mathbf{t_{GBNN}}$	$\mathbf{t_{robot}}$
Spiral	00 : 23 : 29	00 : 19 : 22	00 : 02 : 19	04 : 38 : 44
Boustrophedon	00 : 22 : 38	00 : 18 : 34	00 : 02 : 14	04 : 35 : 48

5.6.2 GBNN algorithm analysis

The paths involved in each case are shown in Figure 5.25. Both routes involve deadlock situations in practically the same areas. Anyway, it is a map that shows quite well the performance of each motion since most of the paths have been generated without having to use Dijkstra's algorithm.



(a) Spiral

(b) Boustrophedon

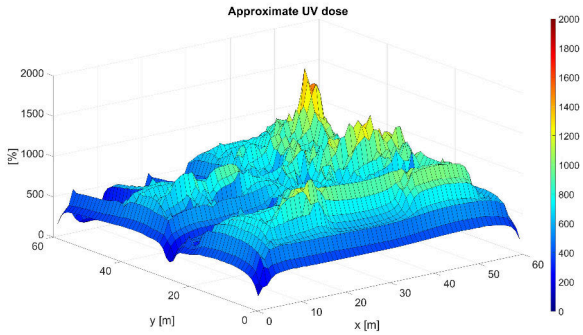
Figure 5.25: Path for Test 5, generated by complete coverage path planning

For the first time, the robot using boustrophedon employs fewer iterations in finishing the path. Once again, it is the robot with a lower t_{robot}, as in all other cases. In Table 5.22, it can be seen that the number of deadlock situations is the same in both cases. This fact can be verified in the previous figure.

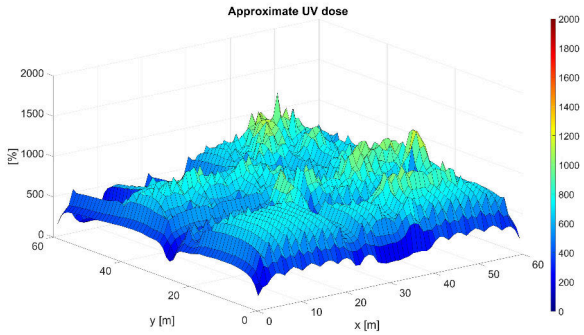
Table 5.22: GBNN algorithm metrics for Test 5 with spiral motion.

	$\mathbf{n_{GBNN}}$	$\mathbf{n_{Dijkstra}}$	$\mathbf{d_e}$	$\mathbf{r_e}$
Spiral	1579	18	1591.01	339.29
Boustrophedon	1539	18	1552.67	469.67

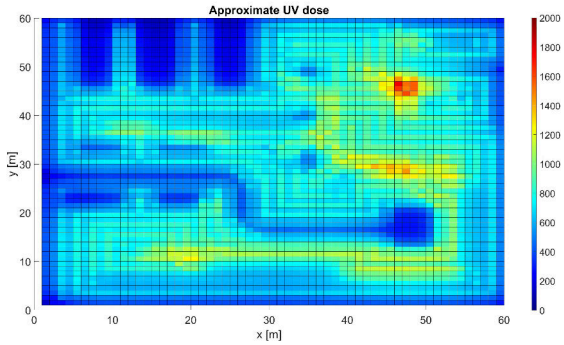
In Figure 5.26, it can be seen that with both techniques, a uniform distribution is achieved. One reason for this is the density of obstacles on the map. The size of the environment has been increased, but not the number of obstacles that appear in it. For this reason, deadlock situations have been reduced considerably. However, some peaks appear which exceed the desired threshold. The highest peaks occur using spiral motion.



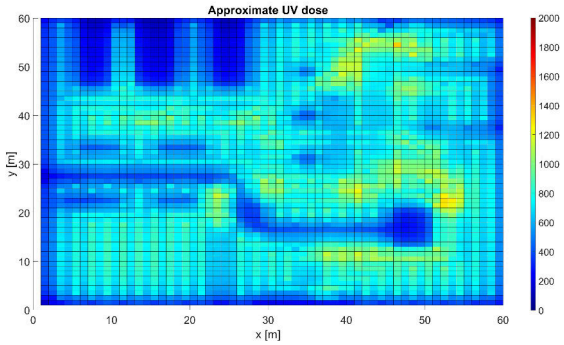
(a) 3D view spiral motion



(b) 3D view boustrophedon motion



(c) 2D view spiral motion

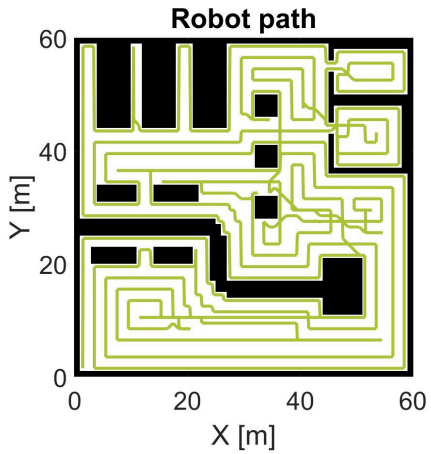


(d) 2D view boustrophedon motion

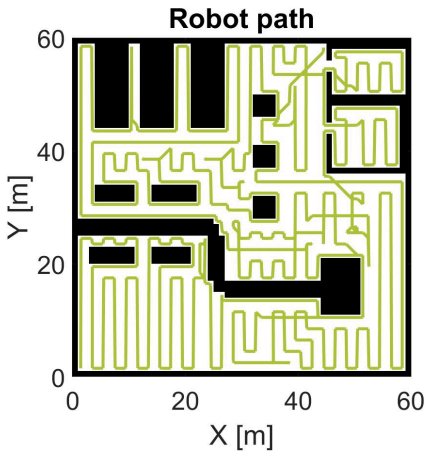
Figure 5.26: UV % estimated during the complete coverage path planning of Test 5

5.6.3 Simulation analysis

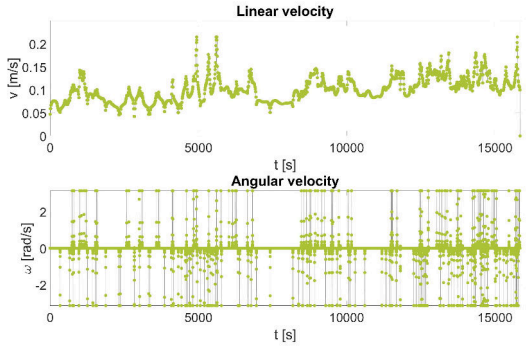
The path followed by each robot is shown in Figure 5.27. In both cases the essence of each motion involved can be appreciated. On this occasion, it is using spiral motion when the robot has higher speeds. Although, in general, the speed used is quite similar in both scenarios.



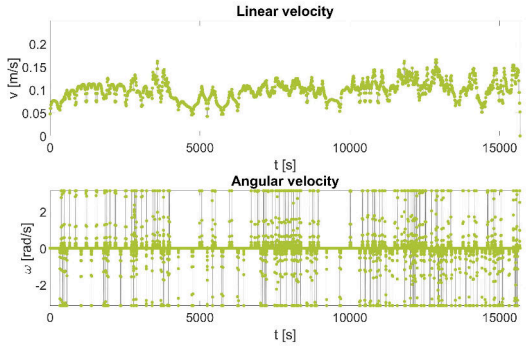
(a) Path with spiral motion



(b) Path with boustrophedon motion



(c) Speed with spiral motion



(d) Speed with boustrophedon motion

Figure 5.27: Robot path and speed for Test 5

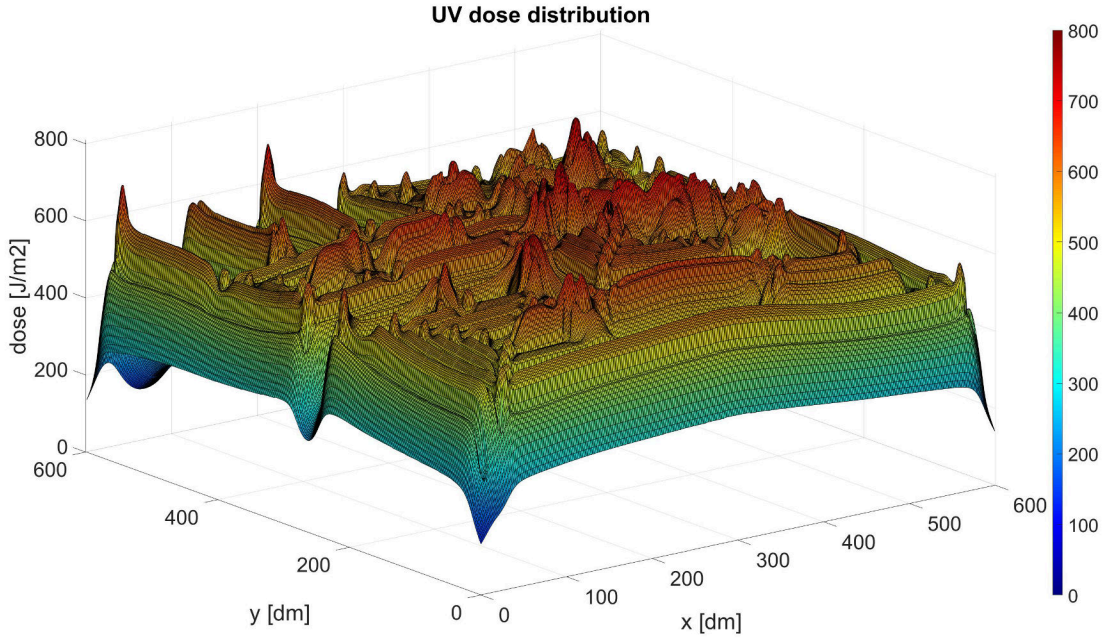
As can be seen in Table 5.23, on this occasion, the path that presents a smaller route is the robot that uses boustrophedon motion. In previous maps, the path with the most distance was also the one that required the least time. On this occasion, this is not the case, and the t_{robot} for the case of spiral motion is still higher.

Table 5.23: Robot metrics for Test 5.

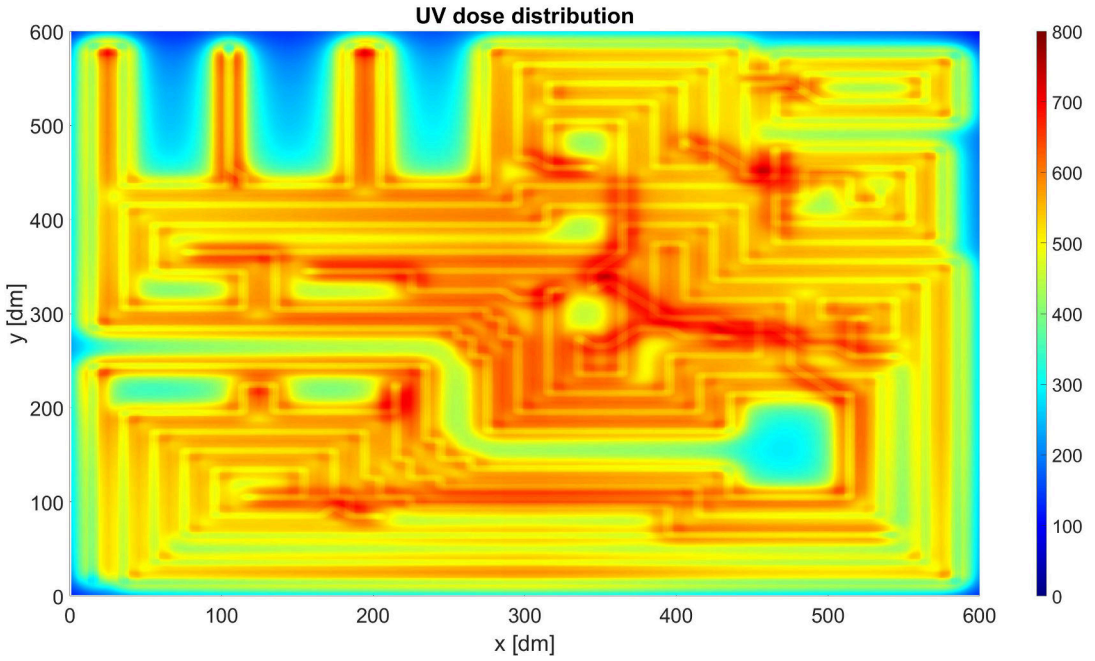
	$\mathbf{n_{goals}}$	$\mathbf{d_{robot}}$	$\mathbf{r_{robot}}$	$\mathbf{v_{max}}$	$\mathbf{v_{min}}$	$\mathbf{v_{av}}$
Spiral	218	1581.39	347.84	0.23	0.04	0.09
Boustrophedon	298	1536.21	477.06	0.18	0.04	0.09

5.6.4 UV disinfection analysis

In Figure 5.28, it can be appreciated how the speed controller is unable to reduce all the existing peaks. For this reason, a vast part of the workspace exceeds the desired threshold. In the central area of the map, a vast affected region appears, in which the waste of energy is maximum, as well as the possible degradation of the surrounding materials.



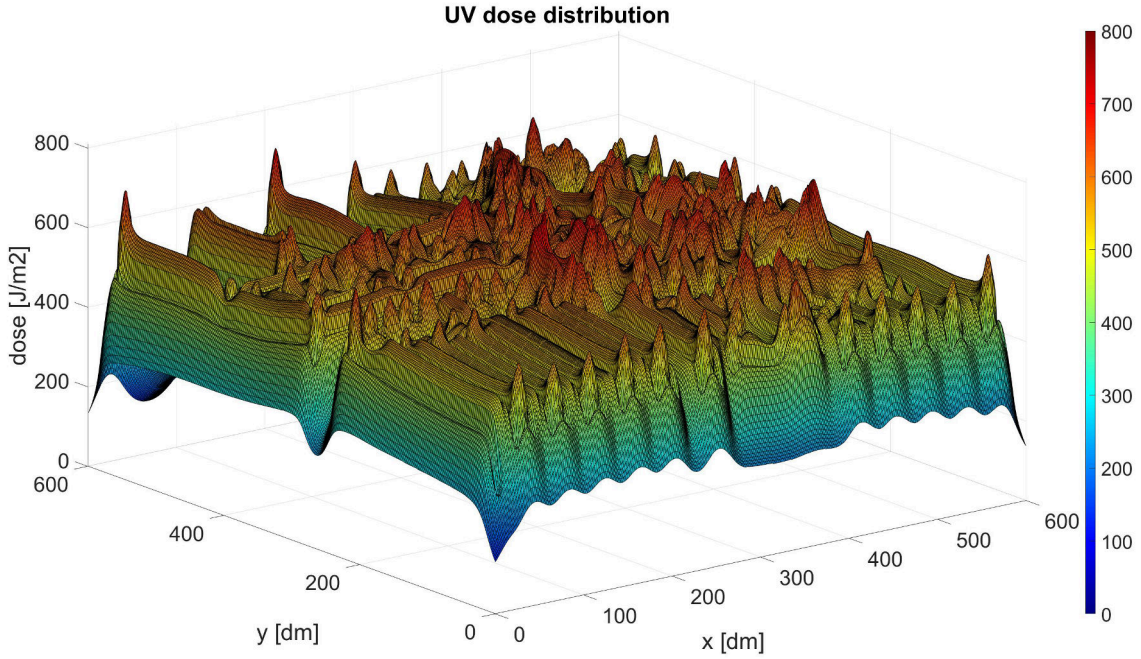
(a) 3D view



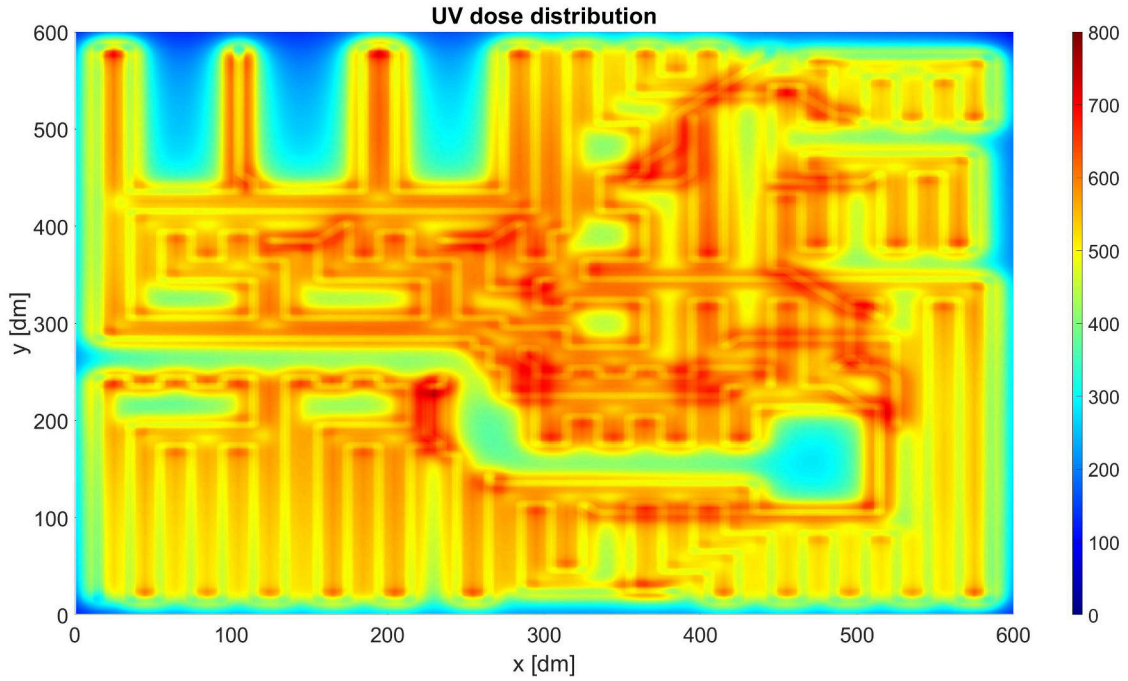
(b) 2D view

Figure 5.28: D_{UV} received with spiral motion for Test 5

In the same way, the previous effect can be seen in Figure 5.29. However, in this case, the many peaks that appear are of smaller size. Only the central area of the map is where this negative effect can be seen. This excessive dose is produced by the lack of efficiency of the speed controller.



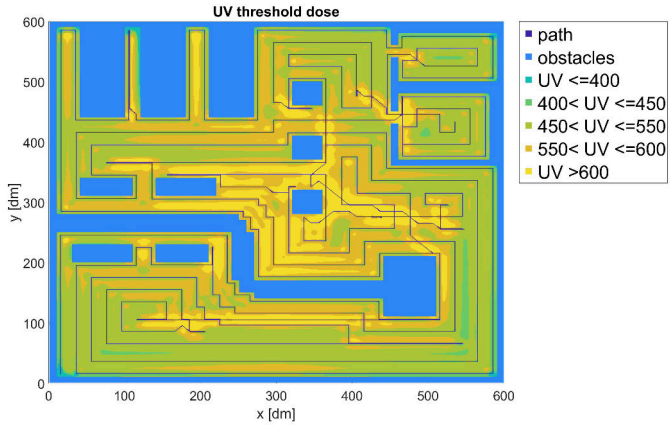
(a) 3D view



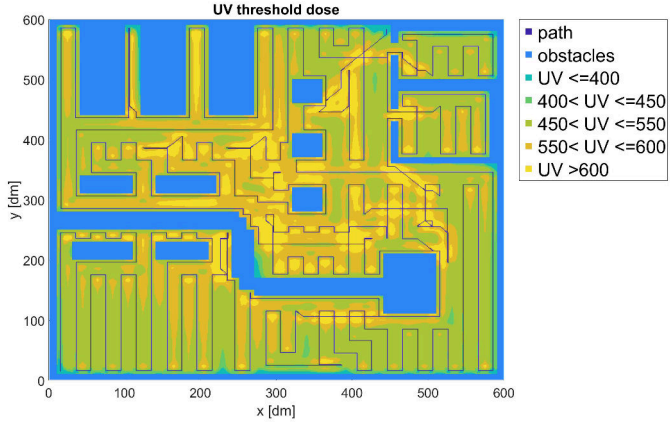
(b) 2D view

Figure 5.29: D_{UV} received with boustrophedon motion for Test 5

This unwanted excess radiation can be analysed in Figure 5.30. As mentioned above, the central region on the map is the most affected. There is an excess dose in the whole environment. Throughout all the tests, it has been possible to see how this effect becomes more aggravated as the map size increases.



(a) Spiral motion



(b) Boustrophedon motion

Figure 5.30: D_{UV} received categorised into thresholds for Test 5

In Table 5.24 it can be seen that practically more than 40% of the dose exceeds the desired threshold, and only 50% remains at the expected threshold. In this map, both solutions achieve similar disinfection percentages, despite the motion used. It is important to remember that this excess of radiation results in a waste of energy. Besides, it produces an increase in the degradation of the materials.

Table 5.24: Disinfection metrics for Test 5.

	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{D_{VL}}$	$\mathbf{D_{L}}$	$\mathbf{D_{N}}$	$\mathbf{D_{H}}$	$\mathbf{D_{VH}}$
Spiral	748.72	215.51	0.38%	5.13%	53.19%	28.58%	12.72%
Boustrophedon	757.92	215.87	1.03%	4.82%	53.49%	29.83%	10.82%

5.7 Summary

A series of conclusions can be learned from all the tests done in this chapter:

- Both of the motion techniques can achieve the complete disinfection of the environment. However, during the tests, it was found that D_N is usually higher using spiral motion.
- The GBNN algorithm is a minimal fraction of all the calculations needed to perform the disinfection simulation.
- The routes generated by boustrophedon motion are usually larger than those generated by spiral motion. However, in all the tests, the robot required less time to complete the coverage when using boustrophedon type movements.
- The speed in the robot has been higher when using boustrophedon motion. However, the average speed has been approximately the same, independently of the movement technique used.
- As the number of obstacles on the map increased, the need to use Dijkstra's algorithm to escape from deadlock situations increased.
- As the map size increased, the effectiveness of the proposed speed controller decreased. This effect causes the radiation doses to exceed the established threshold.

Chapter 6

Final considerations

6.1 Conclusions

In this project, a solution to the challenge of generating disinfection paths has been implemented. A path is obtained using neuronal activity, starting from a known environment. The robot can disinfect all areas of the map using this route.

For this purpose, a Glasius Bio-inspired Neural Network (GBNN) algorithm adapted to the needs of disinfection employing UV radiation has been used. Two different motions have been applied when moving around the workspace, spiral, and boustrophedon. These motions are used to establish a comparison based on the results obtained.

The behaviour that the robot would have when performing a complete disinfection process is simulated. An approximate model has been used to infer the dose received. This approach provides a representative demonstration of the results obtained with each of the available motion techniques.

It is worth mentioning that similar results are achieved with both strategies. Better disinfection times have been reached using boustrophedon motion, which translates into significant energy consumption reduction. However, using spiral motion, the dose percentage that the environment receives within the desired limit is higher. Besides, the time difference between the two techniques is not very significant, so it is worth achieving a better environment.

To escape from deadlock situations Dijkstra's algorithm has been necessary with both techniques. It has been verified that the GBNN algorithm implemented requires less calculation time than the simulation process itself. Therefore, combining Dijkstra with the main GBNN algorithm has resulted in considerably more advantages than disadvantages since it barely affects the computing time. It is worth emphasizing the aim of avoiding the robot making unnecessary movements. For these purposes, a strategy to detect deadlock situations before they occur has been followed.

When using environments, and therefore maps, more and more complex, it was possible to observe how the speed controller lost efficiency. It must be considered that the proposed controller is not very complex, so it was expected that it would not work properly on all the proposed maps. A straightforward solution could be making variable the fitting parameter (ρ) of the speed controller, according to the map complexity. Even so, as it will be seen in the following section, the controller is one of the components to be improved in future works.

Another essential aspect to consider is the analyse of the dose received in the environment. The proposed model does not consider the existence of obstacles. It is being assumed that radiation is capable of passing through walls.

To conclude, an algorithm to generate paths for the disinfection of 2D environments has been developed, using different motion techniques. To optimize the disinfection, the system has a speed controller to adapt the dose received in the workspace. All of the above has been verified and validated in different-complexity environments.

6.2 Further research

There are several possible lines of action, which appear after the development of the proposed work. Some are presented as improvements on what has already been implemented, and others involve new research alternatives.

Optimizing the use of templates:

When generating the disinfection path on each map, it has been necessary to use a series of templates. These templates have made it possible for the robot to perform a specific movement in a particular situation. However, it would be desirable to develop more generic templates to cover as many situations as possible.

Combination of motion techniques:

Depending on the distribution of obstacles in a particular area of the map, a specific type of motion may be desirable. It would be convenient to use the advantages of both strategies when generating the route that allows the complete disinfection of the environment.

Improving the UV radiation model:

The proposed radiation model is only an approximation. It has made it possible to analyse the different paths generated and to compare the motions used. A model more in accordance with the existing UV lamps on the market would improve the perception of the disinfection performed. Likewise, absorption coefficients could be used to consider the properties of the material to be disinfected.

Improve the speed controller:

It is proposed to design a new controller, given the ineffectiveness of the current controller on certain occasions. This controller must guarantee a specific interval of received dose, independently of the map, to be disinfected. A solution based on fuzzy logic could fit very well with the proposed application. Although any other intelligent controller that contemplates the fundamental characteristics to perform this disinfection could be equally appropriate.

Optimal Starting Point:

Implement an algorithm to find the optimal starting point for the GBNN algorithms. A point that reduces the distance covered, the deadlock situations and guarantees the complete disinfection of the environment.

3D environments:

In this project, work has been done in a 2D environment, as the robot only moves on the floor. Nevertheless, the pattern of ultraviolet radiation presented by the disinfection lamps works in a 3D environment. Therefore, it would be appropriate to generate a disinfection model that includes these three dimensions when analysing the workspace coverage.

Multi-robot strategies:

As in many other areas of research, it is possible to work with several robots in the same environment. Full coverage through a cooperative strategy refers mainly to the fact that the robots work as a team, so it can effectively improve work efficiency and reduce the time needed for a full-coverage mission.

Bibliography

[1] Khan, A., Noreen, I., & Habib, Z. (2017). *On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges*. J. Inf. Sci. Eng., 33(1), 101-121.

[2] Cao, Z. L., Huang, Y., & Hall, E. L. (1988). *Region filling operations with random obstacle avoidance for mobile robots*. Journal of Robotic systems, 5(2), 87-102.

[3] Sariff, N., & Buniyamin, N. (2006). *An overview of autonomous mobile robot path planning algorithms*. In 2006 4th student conference on research and development (pp. 183-188). IEEE.

[4] Felner, A., Stern, R., & Kraus, S. (2002). *PHA* performing A* in unknown physical environments*. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1 (pp. 240-247).

[5] Van Den Berg, J. P., & Overmars, M. H. (2005). *Roadmap-based motion planning in dynamic environments*. IEEE Transactions on Robotics, 21(5), 885-897.

[6] Jin, J., & Tang, L. (2010). *Optimal coverage path planning for arable farming on 2D surfaces*. Transactions of the ASABE, 53(1), 283-295.

[7] Hameed, I. A., Bochtis, D. D., & Sorensen, C. G. (2011). *Driving angle and track sequence optimization for operational path planning using genetic algorithms*. Applied Engineering in Agriculture, 27(6), 1077-1086.

[8] Pratama, P. S., Kim, J. W., Kim, H. K., Yoon, S. M., Yeu, T. K., Hong, S., ... & Kim, S. B. (2015). *Path planning algorithm to minimize an overlapped path and turning number for an underwater mining robot*. In 2015 15th International Conference on Control, Automation and Systems (ICCAS) (pp. 499-504). IEEE.

[9] Viet, H. H., Dang, V. H., Laskar, M. N. U., & Chung, T. (2013). *BA*: an online complete coverage algorithm for cleaning robots*. Applied intelligence, 39(2), 217-235.

[10] Galceran, E., & Carreras, M. (2013). *A survey on coverage path planning for robotics*. Robotics and Autonomous systems, 61(12), 1258-1276.

[11] Liu, Y., Lin, X., & Zhu, S. (2008, June). *Combined coverage path planning for autonomous cleaning robots in unstructured environments*. In 2008 7th World Congress on Intelligent Control and Automation (pp. 8271-8276). IEEE.

[12] Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N., & Hull, D. (2002). *Morse decompositions for coverage tasks*. The international journal of robotics research, 21(4), 331-344.

[13] Zelinsky, A., Jarvis, R. A., Byrne, J. C., & Yuta, S. (1993). *Planning paths of complete coverage of an unstructured environment by a mobile robot*. In Proceedings of international conference on advanced robotics (Vol. 13, pp. 533-538).

[14] Gabriely, Y., & Rimon, E. (2002, May). *Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot*. In Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292) (Vol. 1, pp. 954-960). IEEE.

[15] Luo, C., & Yang, S. X. (2008). *A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments*. IEEE Transactions on Neural Networks, 19(7), 1279-1298.

[16] Yang, K., Moon, S., Yoo, S., Kang, J., Doh, N. L., Kim, H. B., & Joo, S. (2014). *Spline-based RRT path planner for non-holonomic robots*. Journal of Intelligent & Robotic Systems, 73(1-4), 763-782.

[17] Xu, L. (2011). *Graph planning for environmental coverage*. PhD thesis, Carnegie Mellon University.

[18] Lee, T. S., Choi, J. S., Lee, J. H., & Lee, B. H. (2009, October). *3-d terrain covering and map building algorithm for an auv*. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4420-4425). IEEE.

[19] Cheng, P., Keller, J., & Kumar, V. (2008, September). *Time-optimal UAV trajectory planning for 3D urban structure coverage*. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 2750-2757). IEEE.

[20] *Información iluminación UV para desinfección*. Retrieved December 5, 2020, from Aliada Dental website: `https://www.aliadadental.es/archivos/categorias/1602246103.pdf`

[21] Soehnge, H., Ouhtit, A., & Ananthaswamy, O. N. (1997). *Mechanisms of induction of skin cancer by UV radiation*. Front Biosci, 2, D538-D551.

[22] *How UV photo-oxidizes organic contaminants and is effective against bacteria*. Retrieved December 5, 2020, from Veolia website: http://www.veoliawatertechnologies.ie/scientific/uv/

[23] Kowalski, W., Walsh, T., & Petraitis, V. (2020). *COVID-19 coronavirus ultraviolet susceptibility*. Purplesun Inc.

[24] Reed, N. G. (2010). *The history of ultraviolet germicidal irradiation for air disinfection*. Public health reports, 125(1), 15-27.

[25] *Robotics System Toolbox*. Retrieved December 5, 2020, from MathWorks website: https://uk.mathworks.com/help/robotics/index.html?s_tid=CRUX_lftnav

[26] *Get Started with Parallel Computing Toolbox*. Retrieved December 5, 2020, from MathWorks website: https://uk.mathworks.com/help/parallel-computing/getting-started-with-parallel-computing-toolbox.html

[27] *Bioinformatics Toolbox*. Retrieved December 5, 2020, from MathWorks website: https://uk.mathworks.com/products/bioinfo.html

[28] Javanmard, R. (2016). *Discretizing (tiling) the input map into a grid graph, and finding shortest path in non-convex map*. Retrieved December 5, 2020, from MATLAB File Exchange website: https://www.mathworks.com/matlabcentral/fileexchange/55519-discretizing-tiling-the-input-map-into-a-grid-graph-and-finding-shortest-path-in-non-convex-map.

[29] *Commercial & Industrial disinfection solution*. Retrieved December 5, 2020, from ZENZOE website: http://zenzoe.tech/download/ZENZOE_Commercial_Industrial.pdf

[30] Malu, S. K., & Majumdar, J. (2014). *Kinematics, localization and control of differential drive mobile robot. Global Journal of Research In Engineering*.

[31] *Luminaria fluorescente*. Retrieved December 5, 2020, from Wikipedia website: https://es.wikipedia.org/wiki/Luminaria_fluorescente

[32] *Información de la luz ultravioleta*. Retrieved December 5, 2020, from Lenntech website: https://www.lenntech.es/uv-informacion.htm

[33] Kheyrandish, A., Taghipour, F., & Mohseni, M. (2018). *UV-LED radiation modeling and its applications in UV dose determination for water treatment*. Journal of Photochemistry and Photobiology A: Chemistry, 352, 113-121.

[34] *Desinfección del Coronavirus en ambientes interiores*. Retrieved December 5, 2020, from Monsolar Ingenieros website: https://www.monsolar.net/

desinfeccion-del-coronavirus-en-ambientes-interiores-luz-ultraviole
ta-con-efecto-germicida-uvg/.

[35] Derraik, J. G., Anderson, W. A., Connelly, E. A., & Anderson, Y. C. (2020). *Rapid review of SARS-CoV-1 and SARS-CoV-2 viability, susceptibility to treatment, and the disinfection and reuse of PPE, particularly filtering facepiece respirators.* International journal of environmental research and public health, 17(17), 6117.

[36] Fischer, R. J., Morris, D. H., van Doremalen, N., Sarchette, S., Matson, M. J., Bushmaker, T., ... & Judson, S. D. (2020). *Effectiveness of N95 respirator decontamination and reuse against SARS-CoV-2 virus.* Emerging infectious diseases, 26(9), 2253.

[37] Luo, C., Yang, S. X., Li, X., & Meng, M. Q. H. (2016). *Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots.* IEEE Transactions on Industrial Electronics, 64(1), 750-760.

[38] Hodgkin, A. L., & Huxley, A. F. (1952). *A quantitative description of membrane current and its application to conduction and excitation in nerve.* The Journal of physiology, 117(4), 500.

[39] Grossberg, S. (1988). *Nonlinear neural networks: Principles, mechanisms, and architectures.* Neural networks, 1(1), 17-61.

[40] Bacciotti, A., & Rosier, L. (2006). *Liapunov functions and stability in control theory.* Springer Science & Business Media.

[41] Cohen, M. A., & Grossberg, S. (1983). *Absolute stability of global pattern formation and parallel memory storage by competitive neural networks.* IEEE transactions on systems, man, and cybernetics, (5), 815-826.

[42] Sun, B., Zhu, D., Tian, C., & Luo, C. (2018). *Complete coverage autonomous underwater vehicles path planning based on Glasius bio-inspired neural network algorithm for discrete and centralized programming.* IEEE Transactions on Cognitive and Developmental Systems, 11(1), 73-84.

[43] Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., & Nosovic, N. (2012). *Dijkstra's shortest path algorithm serial and parallel execution performance analysis.* In 2012 proceedings of the 35th international convention MIPRO (pp. 1811-1815). IEEE.

[44] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms (3-rd edition).* MIT Press and McGraw-Hill.

[45] Fernández, S. (2020). *Diseño de soluciones UV-C para la prevención del Covid-19*. Retrieved December 5, 2020, from BWT website: http://www.bwt-iberica.com/DOCS/WEBINAR_BWT_Dise%C3%B1o_soluciones-12.05.2020.pdf.

[46] Mallapragada, P. K. (2011). *Logger*. Retrieved December 5, 2020, from MATLAB File Exchange website: https://www.mathworks.com/matlabcentral/fileexchange/33099-logger.

Acronyms

BINN: Bio-Inspired Neural Network

CCPP: Complete Coverage Path Planning

GBNN: Glasius Bio-Inspired Neural Network

LED: Light emitting diodes

NN: Neural Network

UV: Ultraviolet

UVGI: Ultraviolet germicidal irradiation

Appendix A

Environments used in simulation

This appendix includes the different maps used in each of the tests. The majority of the maps used are provided by MATLAB. Only the workspace used in test 2 has been recreated from the experiments performed in [15]. The behaviour of the algorithm proposed to obtain a path is evaluated from these maps, as well as the disinfection coverage of the environment

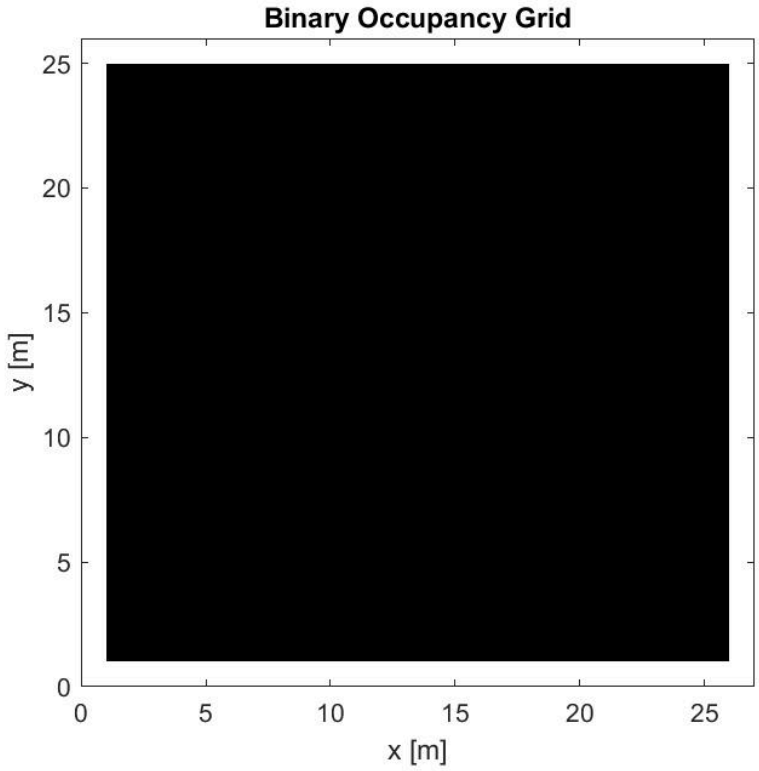


Figure A.1: An empty map used in Test 1 [27 x 26] m

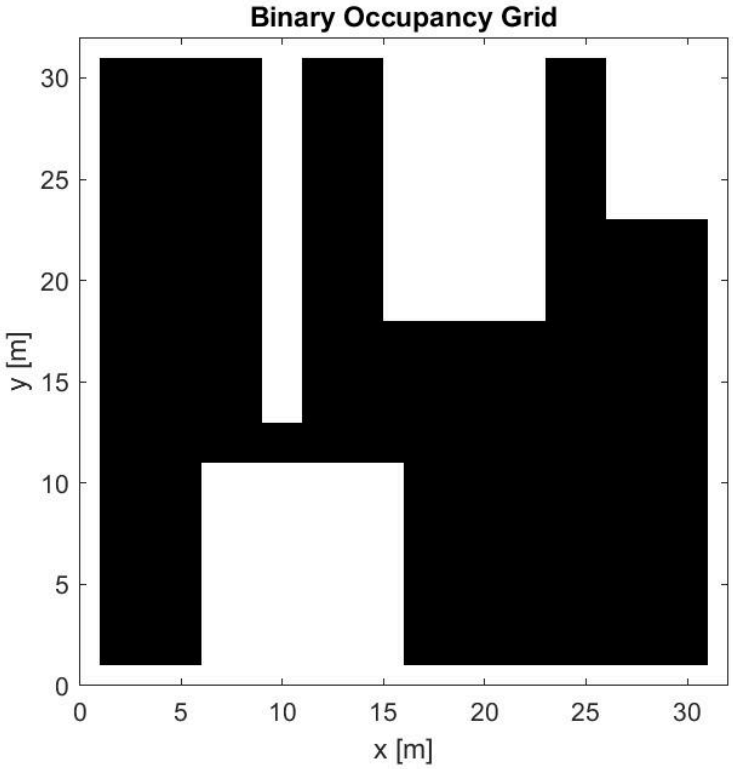


Figure A.2: Map from [15] used in Test 2 [32 x 32] m

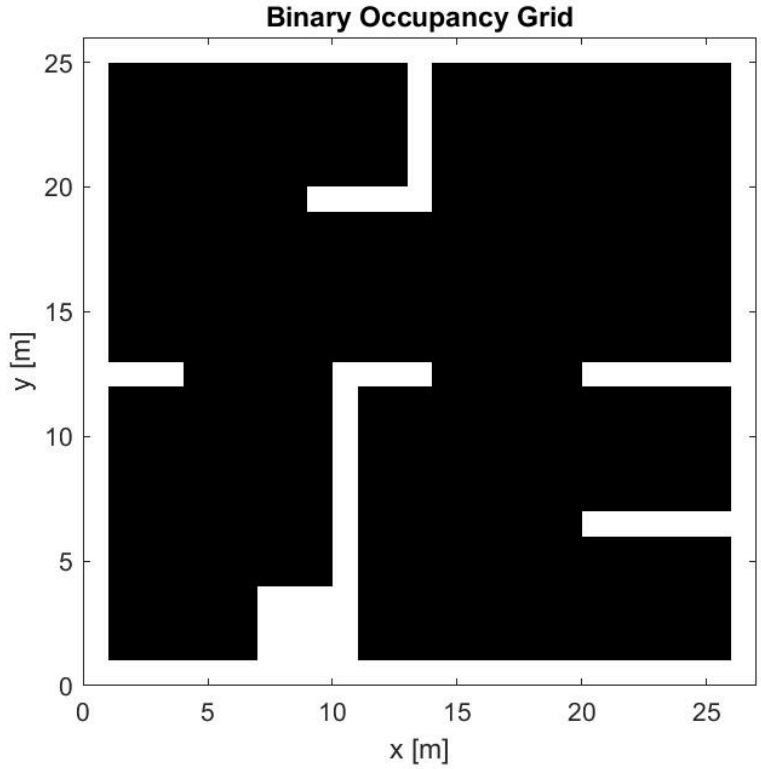


Figure A.3: Map with a simple wall distribution used in Test 3 [27 x 26] m

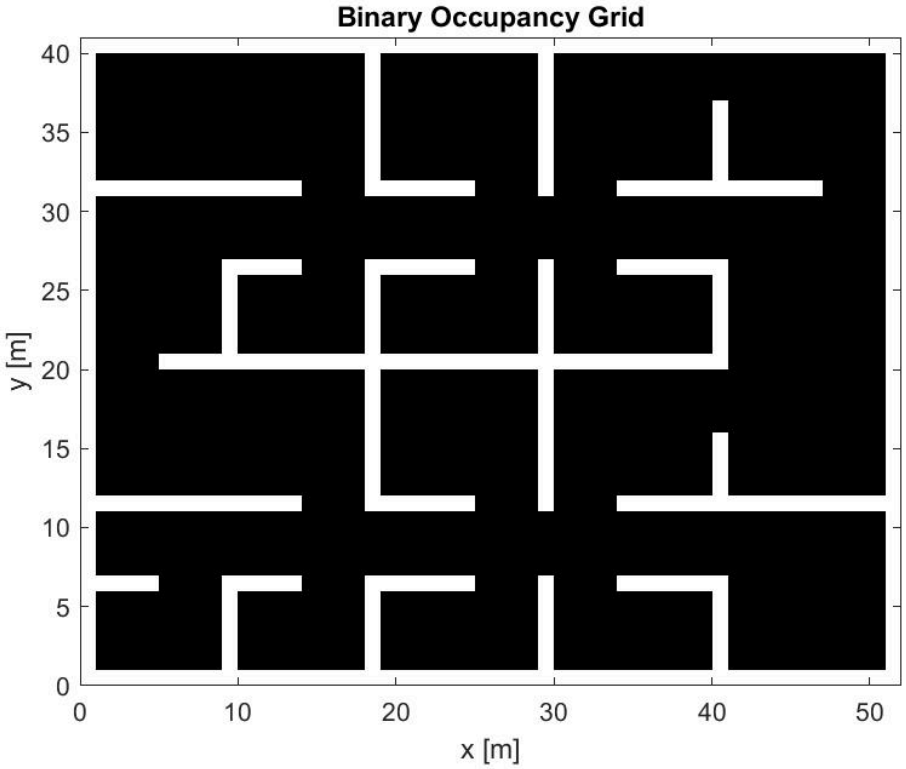


Figure A.4: Map with complex wall distribution used in Test 4 [52 x 41] m

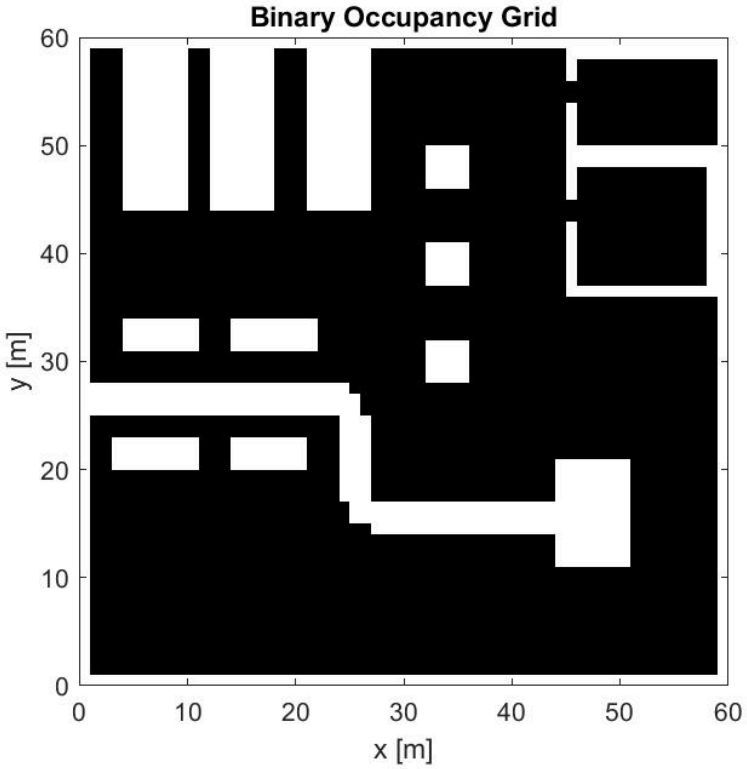
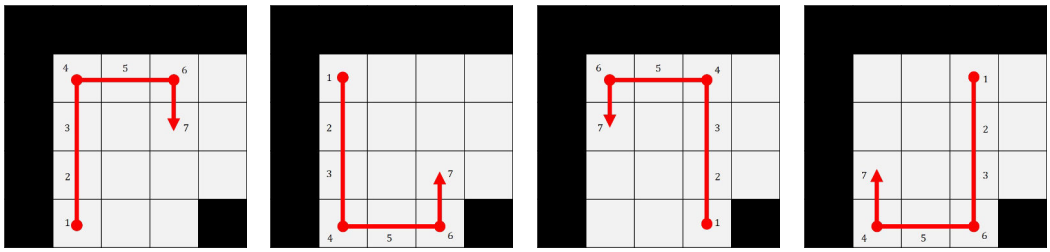


Figure A.5: A typical warehouse of a distribution facility used in Test 5 [60 x 60] m

Appendix B

Templates

In the case of using boustrophedon motion, the robot has to use a specific pattern. The characteristic curve shown in Figure B.1 is the movement that defines this type of motion technique.



(a) Top-Right (b) Bottom-Right (c) Top-Left (d) Bottom-Left

Figure B.1: Boustrophedon motion templates

Furthermore, on some occasions, it will be necessary to make some diagonal movements. In the figure below, it can be seen that when there are no obstacles, the movement can be completed without any problem.

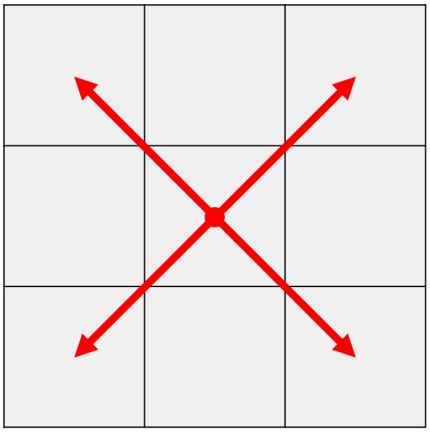
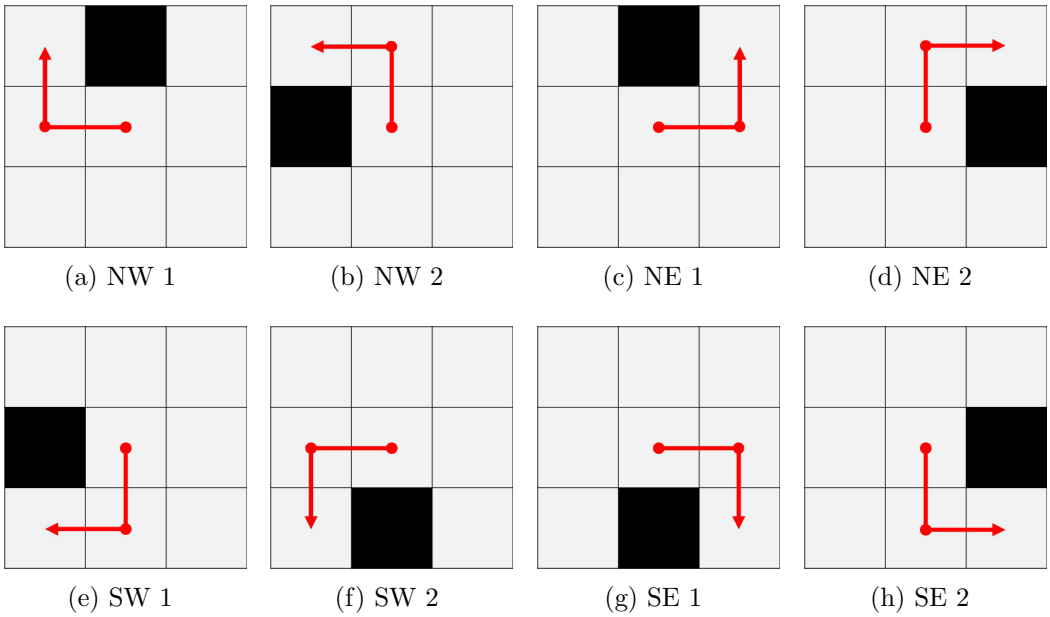


Figure B.2: Diagonal movement without obstacles

However, considering the size of the robot, when there are obstacles, it is recommended to execute an L-shaped movement to go surround it. In Figure B.3, the most common cases appear.



(a) NW 1 (b) NW 2 (c) NE 1 (d) NE 2

(e) SW 1 (f) SW 2 (g) SE 1 (h) SE 2

Figure B.3: Diagonal movement templates

It might occur that the obstacles situation does not correspond to the previous cases. Sometimes the diagonal movement required is similar to that shown in Figure B.4. If this happens, it is because it is the only way to advance. The algorithm notifies this fact through the logger for further analysis once the test is complete. In the proposed maps, this situation never occurs.

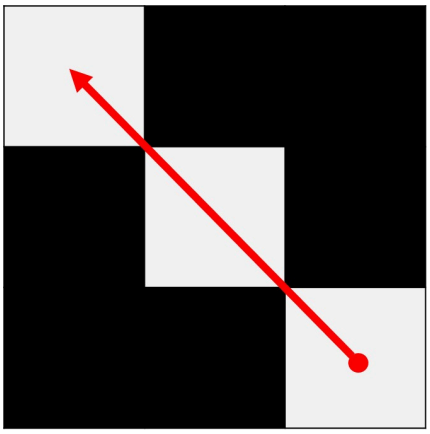
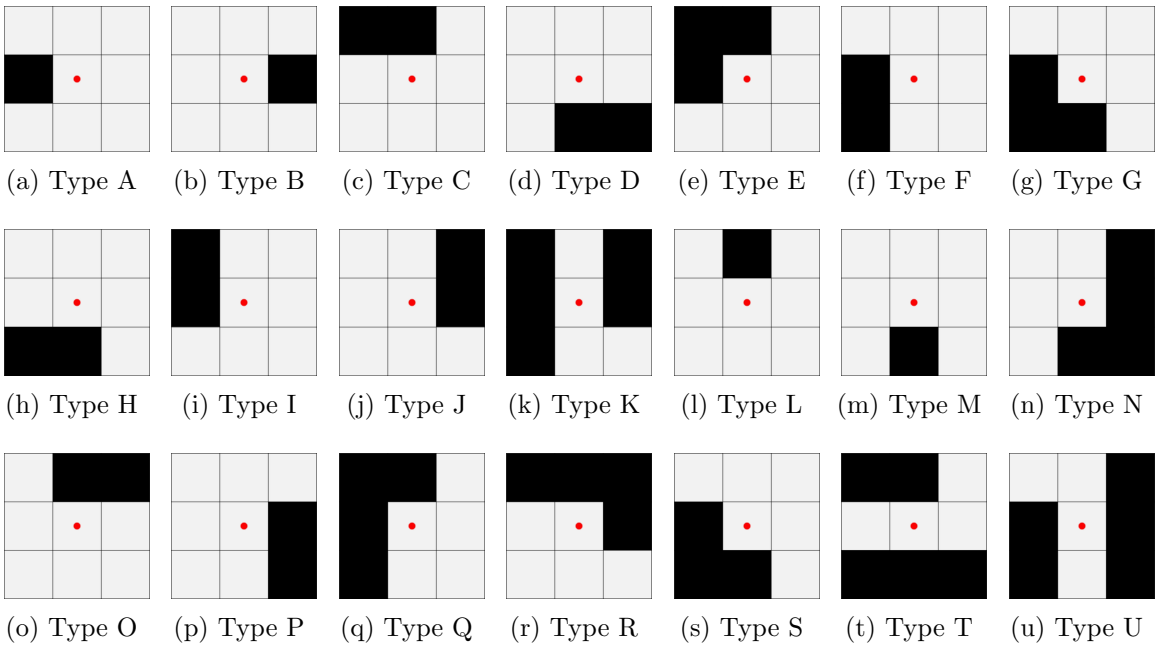


Figure B.4: Narrowing situation that requires diagonal movement

Lastly, to facilitate following a path efficiently, obstacle templates are used. This template forces the robot to adopt a specific path without considering the neural network behaviour. In Figure B.5, the most common situations that have occurred in the maps used for testing appear.



(a) Type A (b) Type B (c) Type C (d) Type D (e) Type E (f) Type F (g) Type G

(h) Type H (i) Type I (j) Type J (k) Type K (l) Type L (m) Type M (n) Type N

(o) Type O (p) Type P (q) Type Q (r) Type R (s) Type S (t) Type T (u) Type U

Figure B.5: Obstacle templates

Therefore, the obstacle templates do not cover all possible situations. Moreover, it is also considered where the robot comes from, when deciding where it should go. There are several possible movement patterns for the same obstacle situation. The pattern is calculated according to the previous state of the robot.

The pattern of movement applied in each situation always tries to go around the obstacle. When working with new maps, new templates will probably need to be added.



UNIVERSIDAD
COMPLUTENSE
MADRID

UNED