Advances in Engineering Software 175 (2023) 103330

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Glasius bio-inspired neural networks based UV-C disinfection path planning improved by preventive deadlock processing algorithm

Daniel Vicente Rodrigo [a], J. Enrique Sierra-García [b,*], Matilde Santos [c]

[a] Complutense University of Madrid, 28040, Madrid, Spain
[b] Electromechanical Engineering Department, University of Burgos, 09006 Burgos, Spain
[c] Institute of Knowledge Technology, Complutense University of Madrid, 28040 Madrid, Spain

ARTICLE INFO

Keywords:
Complete coverage path planning
Mobile robot
UV-C
Deadlocks
Escape routes

ABSTRACT

The COVID-19 pandemic made robot manufacturers explore the idea of combining mobile robotics with UV-C light to automate the disinfection processes. But performing this process in an optimum way introduces some challenges: on the one hand, it is necessary to guarantee that all surfaces receive the radiation level to ensure the disinfection; at the same time, it is necessary to minimize the radiation dose to avoid the damage of the environment. In this work, both challenges are addressed with the design of a complete coverage path planning (CCPP) algorithm. To do it, a novel architecture that combines the glasius bio-inspired neural network (GBNN), a motion strategy, an UV-C estimator, a speed controller, and a pure pursuit controller have been designed. One of the main issues in CCPP is the deadlocks. In this application they may cause a loss of the operation, lack of regularity and high peaks in the radiation dose map, and in the worst case, they can make the robot to get stuck and not complete the disinfection process. To tackle this problem, in this work we propose a preventive deadlock processing algorithm (PDPA) and an escape route generator algorithm (ERGA). Simulation results show how the application of PDPA and the ERGA allow to complete complex maps in an efficient way where the application of GBNN is not enough. Indeed, a 58% more of covered surface is observed. Furthermore, two different motion strategies have been compared: boustrophedon and spiral motion, to check its influence on the performance of the robot navigation.

1. Introduction

COVID-19 pandemic has highlighted the importance of health security strategies. Among these, surface disinfection in health-care settings, as well as other public spaces, transportation, and so on, has shown to be quite successful [1]. However, as disinfection is time consuming and may be even harmful, the automatic execution of this process by robots would be very profitable [2]. Indeed, some mobile robot manufacturers have recently launched disinfection solutions to market [3].

For this purpose, the combination of UV-C light and mobile robotics has been explored as UV-C light provides a fast, clean and efficient inactivation of microorganisms [4]. The complete automation of this process while desirable presents several challenges. On the one hand, it is vital to guarantee a good coverage to ensure all surfaces receive the required dose for disinfection. On the other hand, the radiation must not be too high to prevent premature degradation of the furniture and other items [5]. This is even more crucial in healthcare and hospital environments, where very high-accuracy electronic instruments are present and damage to them must be avoided at all costs. But at the same time, it is that location where disinfection is more necessary to avoid any nosocomial infections [6]. It is important to remark that safety issues are at utmost importance when dealing with UV-C radiation. Thus, the algorithms proposed here have been designed to work in scenarios without humans beings or animals. This way, any possible harm is avoided.

Different path planning methods and algorithms have been proposed to face the complete coverage problem (CCPP). For a further study on this topic [7–9], and [10] can be consulted. Among CCPP algorithms, Glasius Bio-inspired neural network (GBNN) has been widely used to complete coverage path planning of autonomous robots. To mention some works, in [11] GBNN algorithm is applied to the complete coverage path planning of an autonomous underwater vehicle (AUV). The grid map is built by discretizing the two-dimensional underwater environment. In a previous work (with respect to [11]), Zhu et al. propose a method of cooperative complete coverage path planning for multi-AUVs based on the GBNN algorithm, where all AUVs share the common working environment information, and each AUV

* Corresponding author.
E-mail address: jesierra@ubu.es (J.E. Sierra-García).

https://doi.org/10.1016/j.advengsoft.2022.103330
Received 27 July 2022; Received in revised form 13 September 2022; Accepted 19 October 2022
Available online 25 November 2022

deals with other AUVs as moving obstacles [12]. Maintenance operations have been also solved with this approach; for instance, a novel energy-efficient CCPP method based on GBNN for a ship hull inspection robot is proposed in [13]. However, GBNN may lead to unsatisfactory performance in complex environments, so different modifications have been proposed [14,15]. There are some previous works that use CCPP algorithms for different applications in the robotics field. However, to the best of our knowledge, there are not previous works where GBNN is modified and applied for automatic disinfection with UV-C robots. Furthermore, in this work, we improve the GBNN by adding preventive deadlock processing and the automatic generation of escape routes.

In order to test CCPP algorithms for UV-C disinfection, we present a simulation model of the whole scenario. We propose a control architecture that combines GBNN for discretizing the map, a motion strategy to improve the path planning in presence of obstacles, an UV-C radiation dose estimation together with a speed controller in charge of adjusting the robot speed so to get the appropriate radiation dose and, finally, a pure pursuit controller to ensure a correct movement of the mobile robot. Two different motion strategies have been tested to check the influence on the performance of the algorithm: the boustrophedon movement and the spiral one.

One of the main challenges of CCPP algorithms, especially in complex environments with many obstacles, is the deadlocks that may cause the robot to get stuck and fail to complete the navigation of the map. In our case, deadlocks cause operation loss as the robot needs more time and energy to complete the map, and they may also lead to higher peaks in the radiation map as the robot passes several times through the same place, reducing the useful life of the furniture or damaging it. In the worst case, it can cause the robot to stuck and not to complete the disinfection process. To address this problem, in this work we have designed a strategy which includes a preventive deadlock processing algorithm (PDPA) and an escape route generator algorithm (ERGA). On the one hand, by the preventive approach the robot anticipates to the deadlocks by inspecting the surrounding cells, and react to avoid the blockage. On the other hand, when a deadlock is estimated, the algorithm obtains the optimum escape route to reach the closest non-disinfected zone by the ERGA. Simulation results show how the application of PDPA and the ERGA allow to complete complex maps where application of GBNN is not enough. Furthermore, the radiation levels obtained are more regular.

Hence, the main contributions of this work can be summarized as follows:

- Simulation model to test CCPP algorithms for UV-C disinfection.
- Control architecture which combines GBNN, UV-C estimation, speed controller, and pure pursuit control to enable UV-C CCPP.
- Development of preventive deadlock processing novel strategy.
- Development of escape route generator algorithm.
- Improvement of the GBNN by the integration of preventive deadlock processing and escape route generator.
- Comparison of the performance with spiral and boustrophedon movement strategies.

The rest of the paper is structured as follows. Section 2 describes how the UV-C radiation and the robot have been modeled. The description of the control architecture is explained in Section 3. Section 4 details how the algorithms for preventive deadlock processing and the escape route generator work. Results are discussed in Section 5. Finally, the paper ends with the conclusions and the future works.

2. Description of the system model

2.1. UV-C disinfection

Ultraviolet light is a type of electromagnetic radiation, invisible to humans. The electromagnetic spectrum includes many types of radiation, each one defined by a range according to wavelength or frequency. The wavelength of the UV radiation varies from 100 nm to 400 nm, and this range of energy can be divided into 4 types: UV-A (320–400 nm), UV-B (280–320 nm), UV-C (200–280 nm), UV-V (100–200 nm). UV-C is the UV type used for disinfection purposes.

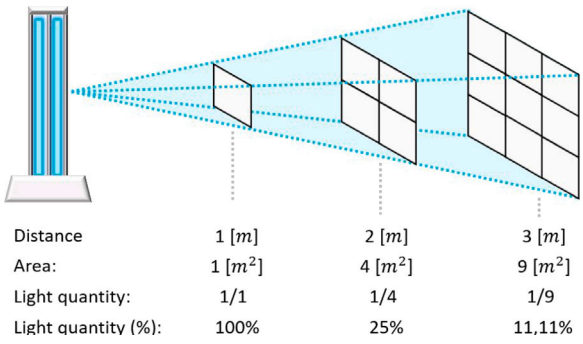


Fig. 1. Propagation of UV-C light according to the inverse square law [19].

UV disinfection technologies play an essential role in the cleaning and disinfection of environmental surfaces. The term UVGI (UltraViolet Germicidal Irradiation) refers to the fact of using UV radiation energy to inactivate fungi, bacteria, or viruses, applied at a given location. The UV-C light provides a fast and efficient inactivation of microorganisms through a physical process. It 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). In [16], results from several studies that have been carried out on Coronaviruses exposed to ultraviolet light are summarized. In these studies, D_{90} value normally denotes the UV dose for 90% inactivation. The range of D_{90} values for coronaviruses is 7–241 J/m^2, whose average, 67 J/m^2, should describe the ultraviolet susceptibility of the SARS-CoV-2 virus. However, UV dose requirements for SARS-CoV-2 removal must be guaranteed in the sterilization process in a human-safe way [17].

UV germicidal lamps are used to achieve this disinfection. An alternative to mercury-based UV lamps is Light Emitting Diodes. UV-LEDs are semiconductor devices that can emit radiation at different wavelengths, including the germicidal range. This disinfection devices are typically characterized 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 [18].

The factors that have the highest impact on disinfection are humidity and distance from the source of radiation. On the one hand, the UV absorbance of the fluid where the light propagates influences the UV radiation dose received. A larger absorbance indicates that the fluid is opaquer and thus, the dose received by the surfaces to be disinfected is lower. In our case, the fluid is the air; the relative humidity in the air determines the concentration of water steam in the air, therefore larger humidity means higher concentration and thus, higher absorbance. On the other hand, the germicidal dose is the product of time and intensity. High intensities over a short period of time and low intensities over a long period of time are fundamentally the same regarding their lethal action on bacteria. The law of the inverse square applies to the germicidal ultraviolet light. Therefore, lethal power decreases as the lamp distance increases (Fig. 1).

In addition, 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 deactivation of a target microorganism can be expressed as (1) [20].

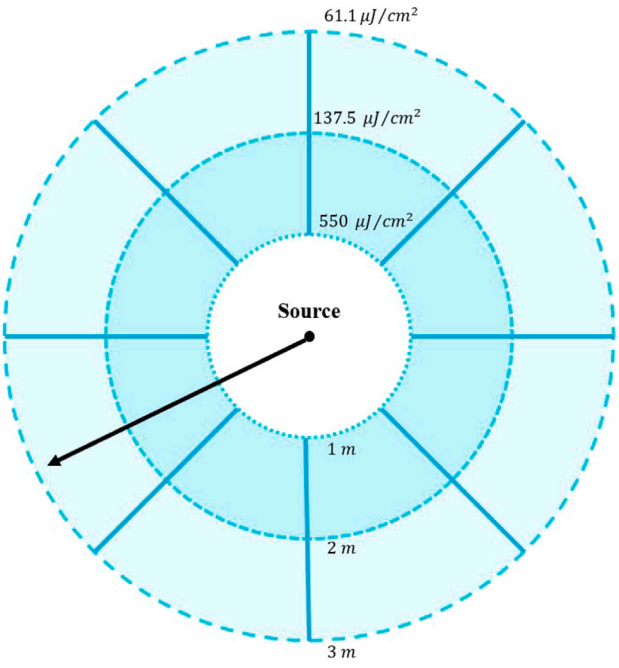


Fig. 2. UV radiation profile approximation (550 μJ/cm²) [19].

As the UV-C is harmful for humans, all safety regulations must be strictly complied when these technologies are applied [21]. We must ensure that all workers who can be exposed to radiation risk receive all the necessary information and training in relation to the result of the risk assessment. It is necessary to wear safety glasses or face shields and clothing that covers all skin; use dosimeters to ensure that the clothing is UV-C opaque; use plastic or glass screens to limit the exposed area, and place caution signs at the entrances to the spaces under disinfection.

2.2. Ultraviolet distribution model

To estimate the disinfection provided by the robot during the mission a model is needed. In this case, the relationship between dose and destruction of a target microorganism is expressed in (1), where N is the number of microorganisms; N_0 is the initial number of microorganisms; k is a constant value [cm²/J]; I is the UV lamp intensity [μW/cm²]; t the time of exposure [s]; and $D_{UV} = I \cdot t$ [μJ/cm²] is the dose received. This equation is further explained in [20].

$$\frac{N(t)}{N_0} = e^{-k \cdot I \cdot t} = e^{-k \cdot D_{UV}} \tag{1}$$

A circular distribution from 8 UV-C lamps as shown in Fig. 2 is considered. It is also assumed that this circular radiation profile only represents a 2D environment, and only follows the inverse square law.

The disinfection process is simulated using the cell matrix with the proposed UV distribution presented in [19]. This circular radiation profile follows the inverse square law, and the radiation emitted by each lamp in this scenario is approximately $I_{UV} = 550$ μW/cm² at a distance of 1 m, that is, $D_{UV} = 550$ μJ/cm² in 1 s. To obtain a 99% SARS-CoV-2 disinfection, we need to administer a dosage of 500 J/m² [22]. In any case, this dosage threshold is a parameter of the algorithm and can be easily updated to disinfect other microorganisms.

It is important to remark that this model considers that the area under the robot does not receive radiation. Furthermore, the radiation model proposed has no distance limitation, i.e., the absorption coefficients of each material are not considered and the radiation goes through the existing obstacles in the map.

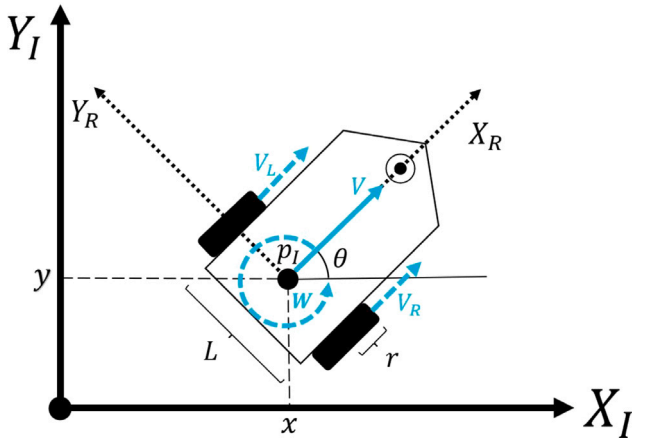


Fig. 3. Kinematic model of a differential robot in the Cartesian coordinate space [19].

The latter assumption is only used to accelerate the simulation time and it only slightly modifies some numerical results. It must be taken into account that the obstacles in the maps we are using are wide, and knowing that the radiation intensity decreases quickly with the inverse square law (see Fig. 2), behind the obstacles the radiation must be low. That is, results are not modified. That is why when this assumption is applied the cells near the obstacles present a very slightly increased radiation regarding the real one. If we do not consider this assumption, the radiation near the obstacles would be slightly lower, without affecting the rest of the algorithm. In any way, this assumption does not affect the way how the PDPA and the ERGA work.

2.3. Robot model

The proposed system combines autonomous mobile robotics for the exploration of environments with the use of UV-C light for disinfection. The differential drive system considered is represented in Fig. 3, where the position $p_I\{x, y, \theta\}$ is expressed in Cartesian coordinates of the inertial frame $\{X_I, Y_I\}$. The relationship between the robot speed in this space, i.e., $\dot{p}_I$, and the velocity in the robot frame $\{X_R, Y_R\}$, $\dot{p}_R$, is defined in [23]. This relationship is used in the differential drive kinematics module of the architecture.

It is a common a practice to assume that there is not lateral slippage, thus the y-component of the velocity in the robot frame is 0. The position and orientation are then the state variables. The linear speed, V, is defined as the average speed of the two wheels. The angular velocity, W (rad/s), is the speed difference of each wheel divided by the distance between them.

The kinematic model is defined at some arbitrary position p in the global inertial frame. The combined action of linear velocity V and angular velocity W describes the movement of the robot (2), where V_L and V_R are the left and right wheel linear velocity in m/s, respectively.

$$\dot{p}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} V \\ 0 \\ W \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ W \end{bmatrix} = \begin{bmatrix} \frac{V_L+V_R}{2}\cos\theta \\ \frac{V_L+V_R}{2}\sin\theta \\ \frac{V_R-V_L}{L} \end{bmatrix} \tag{2}$$

3. System control architecture

The control scheme is shown in Fig. 4. The architecture is based on one of our previous works [19], in this case it is improved to include the PDPA and ERGA. It is mainly composed by two modules, the CCPP algorithm and the path planning. The first one generates a list of waypoints where the robot should go through (*goals*), and the UV percentage that feeds the speed controller in the path planner ($UV_{\%}$).

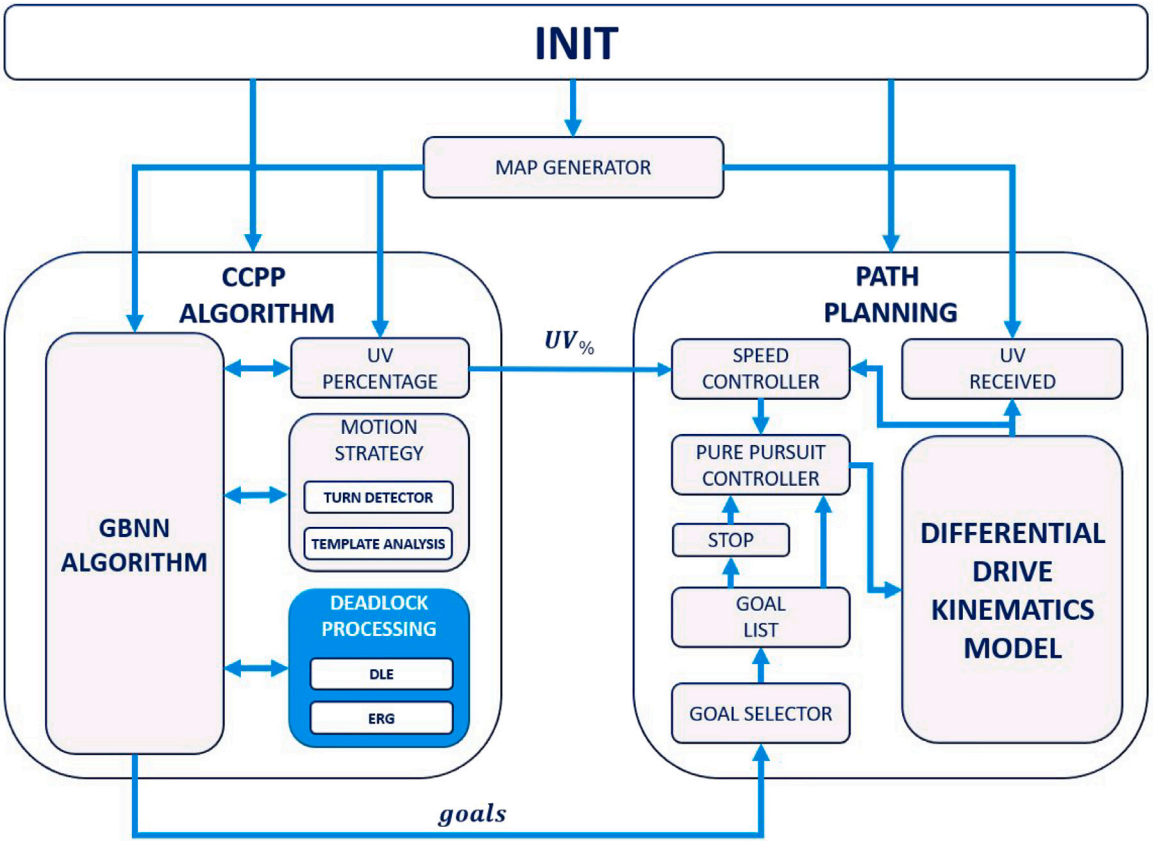


Fig. 4. System architecture diagram, which involves the complete coverage path planning and the simulation environment.

The path planner uses this UV amount to adjust the velocity of the autonomous robot with the speed controller. The list of waypoints are filtered and sequentially given to the pure pursuit controller. The robot in its disinfection task only uses vertical, horizontal and diagonal straight lines to move around the map obtained from the algorithm. For this reason, in order to reduce the list of waypoints used, only those that involve the end of a line (horizontal, vertical or diagonal) are considered. To this end, the list of raw waypoints is analyzed and when a change of orientation of a goal with respect to the previous one is detected, the corresponding cell is added to the definitive list of waypoints. There is not any theoretical limit in the number of waypoints generated. However, in practice, the limitation is related to the RAM memory of the computer where the algorithm runs. The control strategy is validated in simulation, thus the components *INIT* and *MAP GENERATOR* of the Figure. They are used to initialize the simulation and the map. When the controller is deployed in a real robot these components are not present.

The pure pursuit control is a path tracking algorithm. It calculates the target angular velocity to make the robot reach some look-ahead point in front of its current position. This way, this controller constantly pursues a point in front of the robot. The look-ahead distance is how far along the path the robot should look from the current location to obtain the angular speed. 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.

The CCPP algorithm used is based on the enhanced GBNN algorithm briefly explained in the next section and further explained in [19]. This methodology uses 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 a cell in the grid. Each element is connected to its 8 immediate neighbors. Another important component is the preventive deadlock processing, composed by the deadlock estimator (DLE) and the ERGA. They are explained in detail in Section 4.

3.1. Enhanced Glasius Bio-Inspired Neural Network algorithm

In [24], the GBNN algorithm was proposed as a new strategy to perform the complete coverage path planning of autonomous underwater vehicles. This strategy solves the disadvantages of the biological inspired neural network algorithm, reducing the amount of calculation and improving the efficiency of path planning.

In order to improve this algorithm and apply it to space disinfection by means of ultraviolet radiation, the enhanced GBNN algorithm is proposed in [19]. This new approach incorporates a series of modifications, applying templates to ensure that a path is performed in a more efficient way, and focusing its path to reach an adequate dose of ultraviolet radiation in all the accessible points of the environment.

The enhanced GBNN uses the neural activity to determine the different weights between the neuronal connections in combination with several templates and a deadlock detector to increase the algorithm performance. The deadlock detector improves the efficiency as it allows to detect if the robot will get stuck, and through Dijkstra's algorithm, it establishes an escape route to new areas without disinfection.

The behavior of the enhanced GBNN algorithm is detailed in [19]. Using this algorithm, the main task of the robot is to dynamically build the path for complete coverage and to guarantee the disinfection of the workspace. The robot moves to non-visited areas until all regions have been visited. In other words, non-visited regions attract the robot.

Considering the obstacle distribution known, a grid is drawn in the space to be covered, forming cells. A series of flags $flag(k,l)$ are defined for each neural position (k,l). The flag indicates the current state of each cell. The value 0 means non-visited, 1 indicates visited, 2 means that there is an obstacle, and 3 refers to a deadlock. In the same way, the external input I_i is initialized according to the information available

about the environment. The value 0 means visited, +100 indicates not visited, and −100 means there is an obstacle in that cell.

The time evolution of the neural activity is given by (3).

$$na_i(t) = f\left(\sum_{j=1}^{k} w_{ij} \cdot max[na_j(t-1), 0] + I_i\right) \tag{3}$$

Where $na_i(t)$ is the activity of the i-neuron at t; $na_j(t-1)$ is the activity of the neighbor neuron j at $t-1$, which is laterally connected with the i-neuron, and the index k corresponds to the links between the neuron i and its neighboring neurons. The operator $max[a, 0] : \mathbb{R} \to \mathbb{R}$ denotes the biggest value between a and 0.

The function $f(a) : \mathbb{R} \to \mathbb{R}$ is defined as (4),

$$f(a) = \begin{cases} -1, & a < 0 \\ \beta a, & 0 \le a < 1 \\ 1, & a \ge 1 \end{cases} \tag{4}$$

Where $\beta > 0$ is a parameter to adjust the activation function. Regarding the connection weight w_{ij} between neuron i and j, it is defined in Eq. (5), 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.

$$w_{ij} = \begin{cases} e^{-\alpha|q_i - q_j|^2}, & 0 < |q_i - q_j| \le r_0 \\ 0, & |q_i - q_j| > r_0 \end{cases} \tag{5}$$

Therefore, Eq. (3) determines the dynamic of each neuron, attracting the robot to non-visited areas, avoiding already detected obstacles and visited regions.

To ensure a proper operation, 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. 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. The block detector is activated at each iteration. However, while the robot moves along a path generated in response to a deadlock situation, the detector remains inactive. In case the target cell is not reached within the expected number of iterations, the algorithm is re-activated in case a new path needs to be obtained. The operation of the deadlock detector is further explained in the next section.

3.2. Motion strategies

The trajectory is then generated applying the corresponding dynamics to the previous position. Hence, the next position, denoted p_n, is calculated from the current position, (k_c, l_c), (6).

$$p_n = \underset{(m,n)\in \mathcal{N}_r(k_c,l_c)}{\operatorname{argmax}} \{x(m,n) + cy(m,n)\} \tag{6}$$

Where $x(m,n)$ is the activity of the (m,n)-neuron and c is a positive constant. The function $y(m,n)$ is monotonically increasing and is described as the angular difference with the current orientation to reach that cell (7).

$$y(m,n) = 1 - \frac{\Delta\theta(m,n)}{\pi} \tag{7}$$

The term $\Delta\theta(m,n) \in [0, \pi]$ corresponds to the angle of rotation between the current and the possible next moving direction. The general expression is then, (8).

$$\Delta\theta(m,n) = |\theta_n - \theta_c| \tag{8}$$

The robot follows the above strategy to select the next cell, however when several neighbor neurons have the same neural activity or the robot arrives an obstacle, this strategy must be complemented with a motion pattern to ensure all cells are visited. In these cases, there are two basic motion patterns that can be used to achieve the complete coverage, the square spiral motion and the boustrophedon motion. The latter consists of alternatively performing one vertical movement from bottom to top and the next one from top to bottom or vice versa [8] [7]. The spiral motion is to move in a winding or circular motion around a central point. Both motion strategies are shown in Fig. 5, and have been tested and compared in this work.

The coverage of a surface can be achieved using a single robot or a multi-robot tactic according to the environment size. However, both ways have common aspects regarding the coverage 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.

Within the spiral movement, there are two possible variants. First, the robot can start at the center of the environment and move towards the edges in an increasing spiral path. With the second option, the robot starts at one edge of a side and moves towards the center, generating a decreasing spiral path [25,26]. The latter option is the one considered in this article and can be formalized by the algorithm 1.

Algorithm 1: Spiral Algorithm

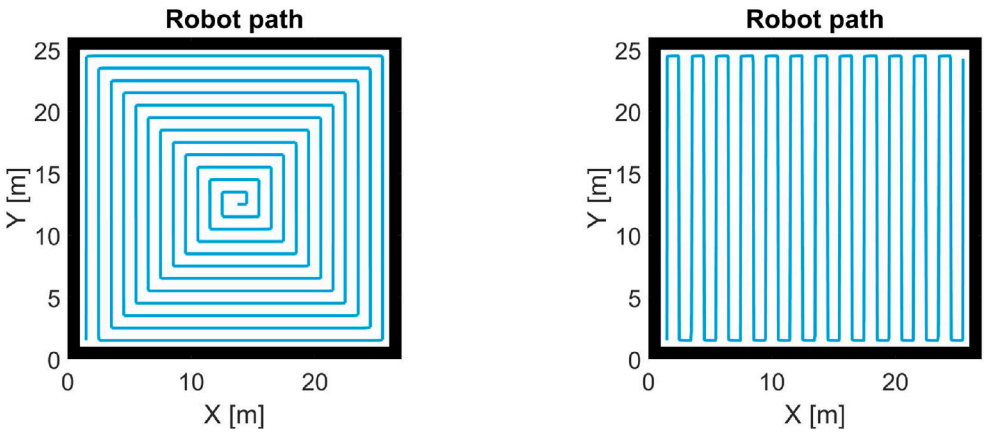
```
1. Start robot motion direction up
2. Check obstacle
   if next cell is obstacle or visited then
      Turn 90° clockwise
   else
      Move forward one cell
   end
```

Boustrophedon refers to the ox-turning in ancient Greek. The algorithm starts by going upwards and then makes the robot turn round at every collision with an obstacle (or wall). The turning direction of the robot continuously changes [25,26]. This motion strategy can be formalized by the algorithm 2.

Algorithm 2: Boustrophedon Algorithm

```
1. Init count = 0
2. Start robot motion direction up
3. Check obstacle
   if next cell is obstacle or visited then
      if count is odd then
         Turn 90° clockwise
         Move forward one cell
         Turn 90° counterclockwise
         count = count + 1
      else
         Turn 90° counterclockwise
         Move forward one cell
         Turn 90° clockwise
         count = count + 1
      end
   else
      Move forward one cell
   end
```

To be efficient in terms of time and energy, the robot must travel along the shortest path, avoiding previously visited areas and unnecessary turns. However, to achieve complete coverage, previously covered cells are sometimes overlapped. To avoid these situations, we have implemented obstacle templates to ensure that the robot faces an obstacle in the most orchestrated way possible. These templates [19] force the robot to adopt a specific path without considering the neural network behavior during the GBNN algorithm process. On the other hand, diagonal movements have been disabled, and their use is only allowed when these movements occur in deadlock situations.



(a) Path with spiral motion (b) Path with boustrophedon motion

Fig. 5. Motion strategy comparison in an empty map.

The generated path will have some specific characteristics according to the motion strategy used for the complete coverage navigation. In most cases, when the environment is full of obstacles, it is impossible to achieve a single pass coverage. In these cases, more advanced strategies are required to complete the disinfection of the workspace, as the deadlock detector and the escape route generator here proposed (Section 4).

3.3. UV estimation and speed control

Fig. 6 shows the dose received in an empty map by using the proposed UV distribution model and the specifications of the disinfection lamp. The UV radiation percentage estimation is obtained during the CCPP by the enhanced GBNN algorithm. This radiation calculation is used to estimate the highest dose of each cell. A grid map is created to store the dose at each cell while the robot moves (Figs. 6(a) and 6(b)). In an empty map, the UV% signal (see Fig. 4) is similar for both types of movements, spiral and boustrophedon. There is only a slight difference in the center of the map in the case of the spiral movement (Fig. 6(a)). With this spiral pattern, the density of radiation tends to grow at the center of the map, as the circles are smaller and smaller. Thus, the control signal, i.e., the UV%, is slightly reduced to correct this effect.

The UV% information is then sent to the robot speed controller that adjusts the velocity to ensure that the final dose is within the desired range (Figs. 6(c) and 6(d)).

The information provided by the coverage algorithm is used to estimate the speed in each section associated to a neuron (cell). Based on the UV percentage matrix ($UV_{\%}$), the intensity of the lamp model (I_{UV}), and the dose required for disinfection ($D_{SARS-CoV-2}$), a speed value is obtained using (9), as shown in the figures Fig. 6(c) and Fig. 6(d).

$$v(x,y) = \rho \cdot UV_{\%}(x,y)\left(\frac{I_{UV}}{D_{SARS-CoV-2}}\right) \tag{9}$$

Where v is the linear velocity of the robot in m/s; ρ is a positive parameter which varies depending on the size and complexity of the map, considering the overall accumulated amount of radiation due to the time spent in the disinfection; $UV_{\%}$ is the estimated maximum percentage of the dose received, taken into account the position of the robot and its eight neighboring cells; I_{UV} is the lamp radiation power in mW/cm^2; and $D_{SARS-CoV-2}$ is the required SARS-CoV-2 dose in mJ/cm^2 to achieve the disinfection of the environment. ρ can be manually adjusted considering the size and the complexity of the map. If the size of the map or the number of obstacles grows it is recommended to increase the value. It is recommended to use values close to 1.

The D_{UV} value is based on the specifications of the UV distribution model. Specifically, a value of 0.55 mJ/cm^2 for the lamp is used. This value comes from the tests performed in [27].

To achieve the inactivation of SARS-CoV-2, it would be convenient 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 [22]. Nevertheless to achieve a 99% reduction, the power should be doubled. For the above reasons, it has been considered appropriate the dose of 500 J/m^2. In any case the recommended radiation dose is given by the parameter $D_{SARS-CoV-2}$ and can be easily updated.

In addition, as shown in the control architecture, a pure pursuit controller calculates the angular velocity that moves the robot from its current position to the next waypoints ($[x\ y\ \theta]$) [28].

4. Generation of escape routes when deadlocks

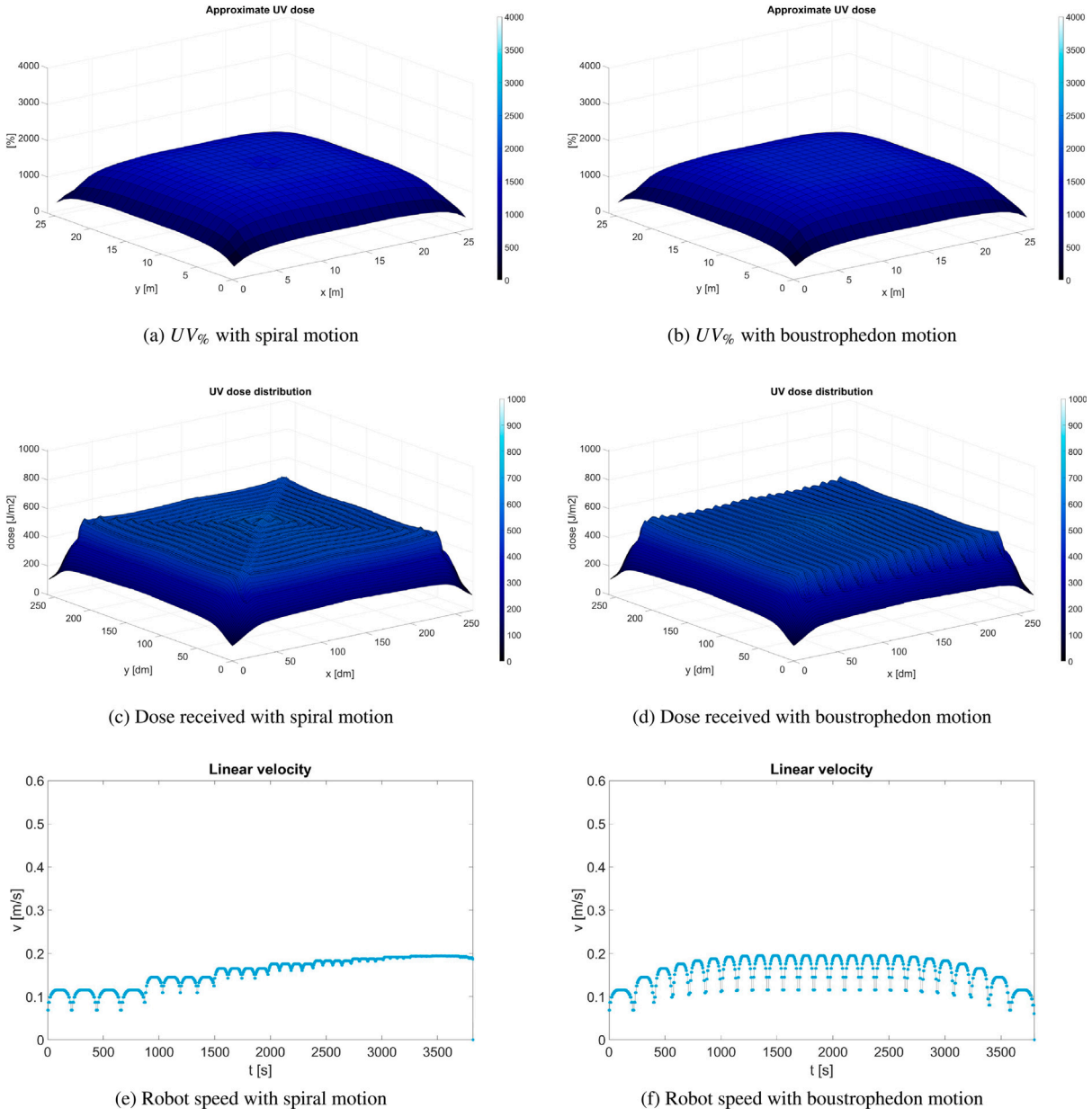
4.1. Deadlock situation

A deadlock is a potential situation that may happen in a complete coverage path planning where no solution can be found, or even it may not exist. Mathematically, it can be defined as follows: Given the 2D Cartesian workspace $\mathcal{W}$ and a neuron $N(m,n)$, a deadlock situation occurs when each neuron in the neighborhood has been previously visited ($flag(k,l) = 1$), or is an obstacle ($flag(k,l) = 2$), or has less neural activity ($na(k,l) < na(m,n)$). In this case, the following characteristic can be asserted [29]:

$$\begin{aligned} &\text{if } \exists(m,n) \mid \forall k \in \{m-1,m,m+1\}, \forall l \in \{n-1,n,n+1\},\\ &\qquad (k,l) \neq (m,n),\ 1 \le m \le N_x, 1 \le n \le N_y,\\ &\qquad \text{s.t. } [(flag(k,l) == 1 \vee 2) \vee x(k,l) < x(m,n)]\\ &\qquad \text{then } f(m,n) := 3 \end{aligned} \tag{10}$$

Where N_x and N_y denote the number of cells in the x-axis and y-axis of the map. Therefore, once the robot is in a deadlock situation, the neural activity of its neighboring neurons is no longer from the activity at the current position, as all the surrounding cells have been previously visited, and hence, disinfected. Consequently, during a deadlock situation the robot has to wait until another neuron is available in the neural activity landscape.

When the robot reaches a deadlock position, the neural network model and the proposed coverage algorithm could sometimes solve automatically this situation. The robot stuck 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. This is automatically performed by GBNN. However, if the environment is very complex, the robot could remain trapped indefinitely without ever finding a way out. To solve this problem, we have implemented the DLE and the ERGA.



(a) $UV_{\%}$ with spiral motion

(b) $UV_{\%}$ with boustrophedon motion

(c) Dose received with spiral motion

(d) Dose received with boustrophedon motion

(e) Robot speed with spiral motion

(f) Robot speed with boustrophedon motion

Fig. 6. Dose received in an empty map, at the end of the disinfection process, with the UV distribution model proposed and lamp power $D_{UV} = 550\ \mu J/cm^2$.

4.2. Escape provided by GBNN

When a deadlock appears the neural network model and the coverage algorithm try to automatically solve the situation. The robot trapped in the deadlock will move to the neighbor cell with the maximum value, even if they have been previously visited, so at each iteration the robot selects a new cell, it never stays in the same position. Performing this movement, it may sometimes find and go back to a previously visited cell and can get out of this situation and then continue with the coverage. Fig. 7 shows a clear example of this. 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.

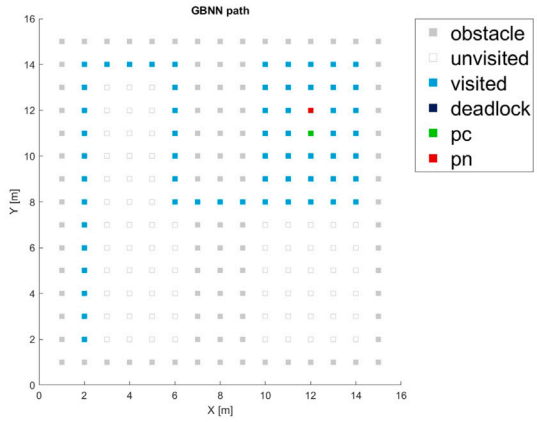
However, in some cases the robot may be changing cells permanently without finding a way to solve the deadlock. This situation is more frequent when the robot goes into closed rooms with only a way to escape.

4.3. Escape provided by preventive deadlock processing

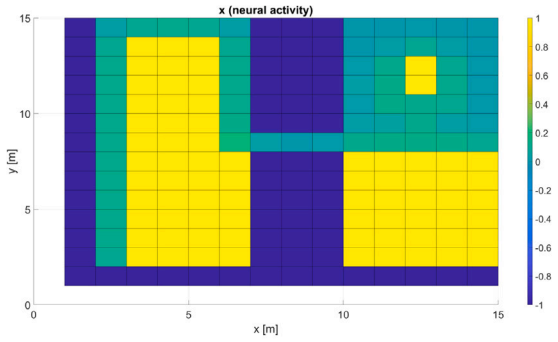
Although standard GBNN may allow the robot to escape in certain deadlock situations, this approach may be too slow or even it may not find the way out. This results in the robot wandering in circles in the workspace until the neural map is enough updated so to find a clear escape route, if it does. Moreover, depending on the complexity and size of the workspace, it might be impossible to access all the non-disinfected areas using the neural approximation.

In an effort to improve the results of the CCPP strategy, the DLE and the ERGA have been implemented and integrated with the main enhanced GBNN algorithm.

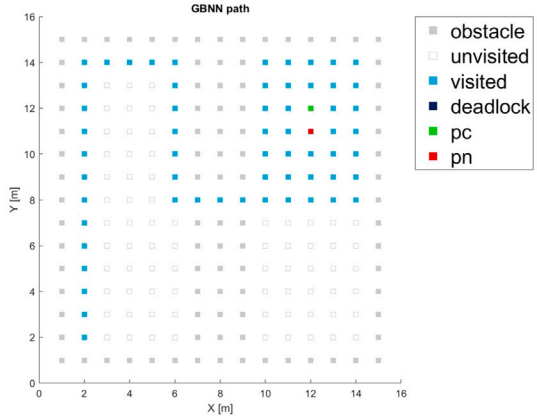
The PDPA explains the steps designed to anticipate deadlock issues. The deadlock estimator is activated at each iteration. However, while the robot moves along a path generated in response to a deadlock



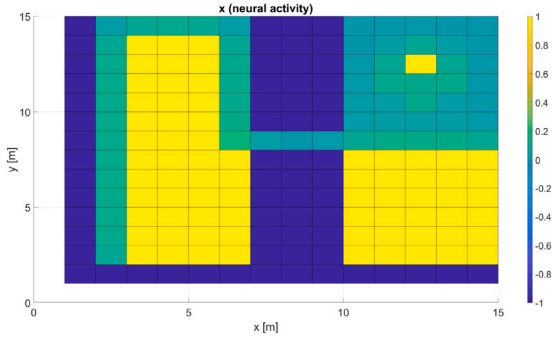
(a) Visited cells during GBNN using spiral motion ($n_{GBNN} = 60$)



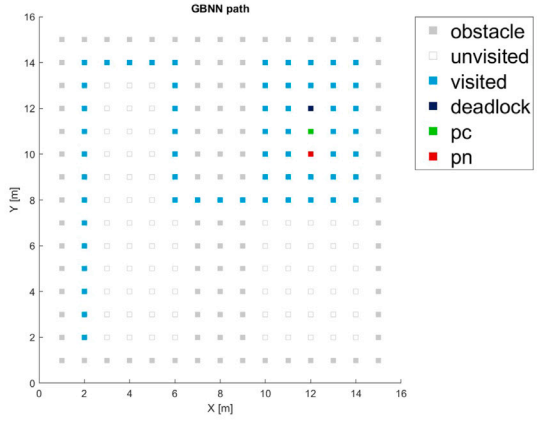
(b) Neural Activity ($n_{GBNN} = 60$)



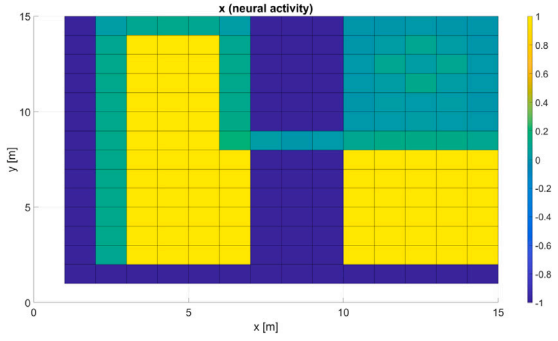
(c) Visited cells during GBNN using spiral motion ($n_{GBNN} = 61$)



(d) Neural Activity ($n_{GBNN} = 61$)



(e) Visited cells during GBNN using spiral motion ($n_{GBNN} = 62$)



(f) Neural Activity ($n_{GBNN} = 62$)

Fig. 7. Deadlock situation solved with a neural approach.

estimation, the DLE remains inactive. If the target cell is not reached within the expected number of iterations, the algorithm is re-activated as it is necessary to obtain a new path.

To ensure the PDPA works in a preventive way, it is important to mention that the deadlock estimation is not obtained with the expression (10), but when (11) is met. That is, as long as the energy of the 24 neurons ($r = 2$) surrounding the current one is equal to 0, the algorithm will automatically anticipate to a future deadlock situation and will react triggering the ERGA.

$$\text{if } I(g,h) == 0, \forall g \in \{k_c - 2, k_c, k_c + 2\}, h \in \{l_c - 2, l_c, l_c + 2\} \quad \text{then } deadlock_{est} = \text{true} \tag{11}$$

Algorithm 3: Preventive Deadlock Processing Algorithm PDPA

1. Analyses the current situation of the robot
   if $deadlock_{est}$ == false *then*
   1.1. Check if the external input I is equal to 0 in the cells close to the current position (k_c, l_c).
   if $I(g,h) == 0 \in \mathcal{N}_n(m,n)$,
   $\forall g \in \{k_c-2, k_c, k_c+2\}$ *and* $h \in \{l_c-2, l_c, l_c+2\}$ *then*
   1.1.1. The status of the robot is updated
   $deadlock_{est}$:= true
   1.1.2. *Call ERGA*. Use route escape generator to modify the neuron map
   1.1.3. The desired target escape position is obtained $(k_e, l_e, iter_e)$
   get k_e, l_e, $iter_e$
   set $iter_c := 0$
   else
   1.2. Update iterations
   $iter_c := iter_c + 1$
   1.3. Check if the robot has reached the target escape position
   if $k_c == k_e$ *and* $l_c == l_e$ *then*
   $deadlock_{est}$:= false
   1.4 Check if the robot is stuck during escape
   if $iter_c > iter_e + 1$ *then*
   Go to step 1.1
   end

Algorithm 4: Escape Route Generator Algorithm ERGA

1. Search for non-visited cells $N(m,n)$
   $N_{f=0}(m,n)$, $\forall\, m \neq k_c, n \neq l_c$, $1 \leq m \leq N_x, 1 \leq n \leq N_y$, s.t $f(m,n) = 0$
2. All possible escape positions are analysed
   for $(k_{esc}, l_{esc}) \in N_{f=0}$
   2.1 Calculate Euclidean distance for each possible escape cell.
   $d_{esc}(index, p_{esc}) = d_E(p_c, p_{esc}) = \sqrt{(k_c - k_{esc})^2 + (l_c - l_{esc})^2}\ \forall\ 1 \leq index \leq (N_{f=0_x} + N_{f=0_y})$
3. Ascending order the distances
   $d_{esc}(index) = d_{esc}(index, p_esc)\ \forall\ 1 \leq index \leq (N_{f=0_x} + N_{f=0_y})$ s.t $d_{esc}(index, p_{esc}) \leq d_{esc}(index+1, p_{esc})$
4. The minimum distance d_{min} is initialized
   set $d_{min} := \infty$
   set $d_{loop} := true$
5. All possible ascending ordered escape positions are analysed
   for $p_{esc} \in d_{esc}$ s.t $d_{loop} == true$
   5.1. Calculate distance with Dijkstra's algorithm
   $[d_{Dijkstra}, p_{Dijkstra}] := Dijkstra(p_c, p_{esc})$
   5.2. Check if $d_{Dijkstra}$ is less than d_{min}
   if $d_{Dijkstra} < d_{min}$ *then*
   5.2.1. Update d_{min}
   set $d_{min} := d_{Dijkstra}$
   set $p_{min} := p_{Dijkstra}$
   5.2.2. Check if $d_{Dijkstra}$ is the same as d_{esc}
   if $d_{Dijkstra} == d_{esc}$ *then*
   set $d_{loop} := false$
6. Neural behaviour is updated
   set $x(m,n) := 1\ \forall\ (m,n) \in p_{min}$
   set $I(m,n) := 100\ \forall\ (m,n) \in p_{min}$

4.4. Escape route generator

Once a deadlock is estimated, as an alternative to the neuronal approach, a solution based on Dijkstra's algorithm is proposed. The algorithm will calculate the shortest path between two nodes within the same graph. As a result, the robot will be able to escape from a deadlock situation as soon as possible.

In order to use the Dijkstra's algorithm during the initialization process, a graph is created from the map. The graph has as many nodes as non-obstacle cells. In addition, an edge between each cell and its neighbor cells is established. All edges have the same constant weight.

When an imminent deadlock situation has been detected, the ERGA starts a search, looking for the shortest path between the current robot position and the remaining non-disinfected cells being $p_{Dijkstra}$ and $d_{Dijkstra}$ the minimum path and the minimum distance obtained by the Disktra's algorithm, respectively.

To summarize, the ERGA initially obtains the Euclidean distance between the current position and the rest of the candidate cells. In this step the obstacles are not considered to compute the distance. These candidates are classified according to their distance from the current position from furthest to closest.

Once this step is made, Dijkstra's algorithm is applied from the current cell to all previously ordered candidates, taking into account the following:

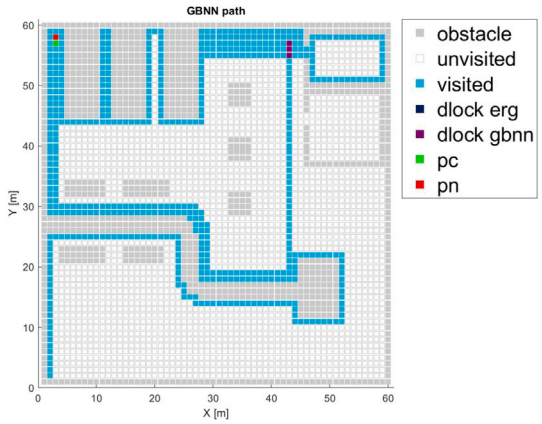
- The Disktra's route computed in this step considers the obstacles, that is the generated route will not cross obstacles. The explanation is that as in the graph there are only connections between non-obstacle cells, the paths returned by the Disktra's algorithm do not include obstacles. Thus, the length of this route is usually different from the euclidean distance to the destination cell.
- The value of the target cell is updated when the distance obtained by this algorithm is the smallest one achieved until now.
- In an attempt to avoid wasting CPU time, when the distance obtained using Dijkstra's is $d_{Dijkstra} \approx |q_i - q_j| \pm 0.1$ m, the search ends and set that candidate as the destination to escape from the deadlock situation. The value 0.1 m is a configurable parameter. The recommended value is 1/10 of the size of the cell in the map. In our case is 1 m, thus this parameter is set to 0.1 m.

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 that path. In this way, the affected cells are restored to their original state of non-disinfected, and the enhanced GBNN algorithm can continue its normal activity to escape the deadlock situation in a natural way.

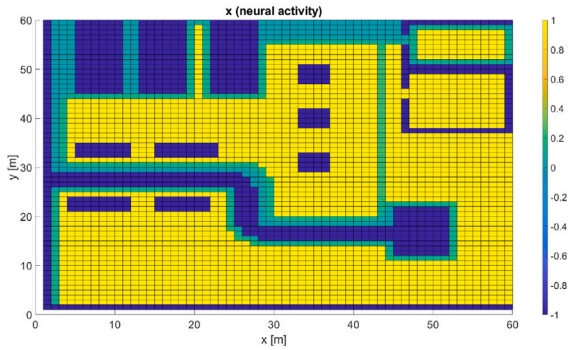
This approach requires a higher computational cost but it optimizes the escape route after a deadlock situation. The Dijkstra's algorithm implementation is based on the code by [30] and the Bioinformatics Toolbox of Matlab was used. The result is that the optimal path between two given nodes is obtained.

To illustrate the way in which the ERGA intervenes in the neural activity to find a escape route from a blocking situation, a capture of the map and the neural activity at two different iterations is shown in the following figure.

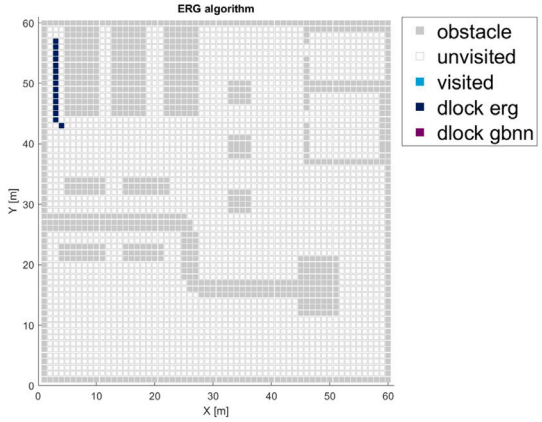
- In Fig. 8(a), when the GBNN algorithm is at $iter = 555$, a deadlock situation occurs in the current position (green cell). The neural activity at this point is shown in Fig. 8(b).
- At this point in time, the ERGA is activated. In Fig. 8(c), the escape route the algorithm has found is examined. This route is represented in the legend as "dlock erg". Once the affected cells are known, the neural activity is initialized as shown in Fig. 8(d). It is possible to see how the color of these cells has changed to yellow, that is, neural activity equal to 1.
- After having modified the neural activity, the enhanced GBNN algorithm is able to get out of the deadlock situation by itself



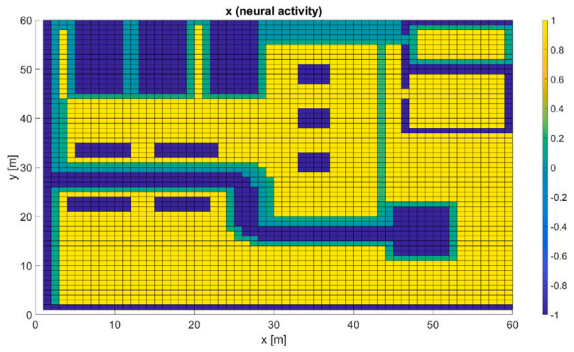
(a) GBNN path (*iter* = 555)



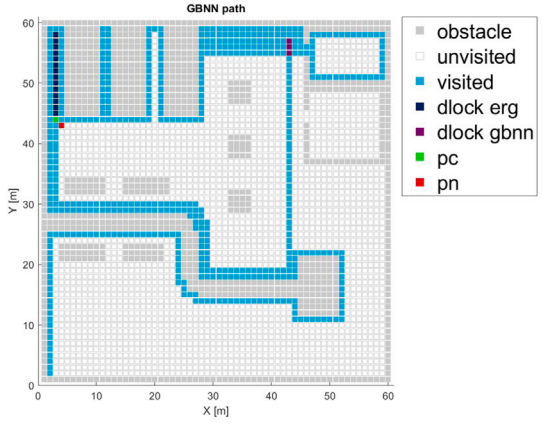
(b) Neural Activity x (*iter* = 555)



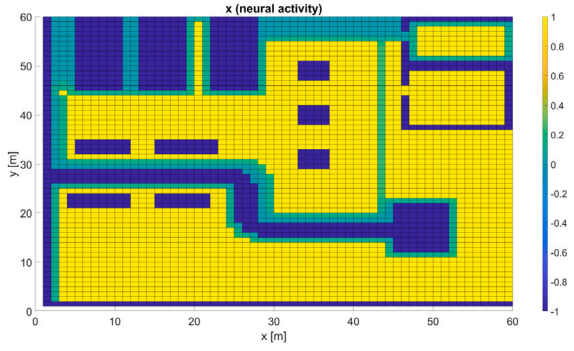
(c) ERGA proposed path (*iter* = 555)



(d) Neural Activity x after ERGA (*iter* = 555)



(e) GBNN path (*iter* = 570)



(f) Neural Activity x (*iter* = 570)

Fig. 8. Deadlock situation solved using ERGA.

(Fig. 8(e)). In this case, the neural activity is updated to 0 by the enhanced GBNN as can be seen in Fig. 8(f).

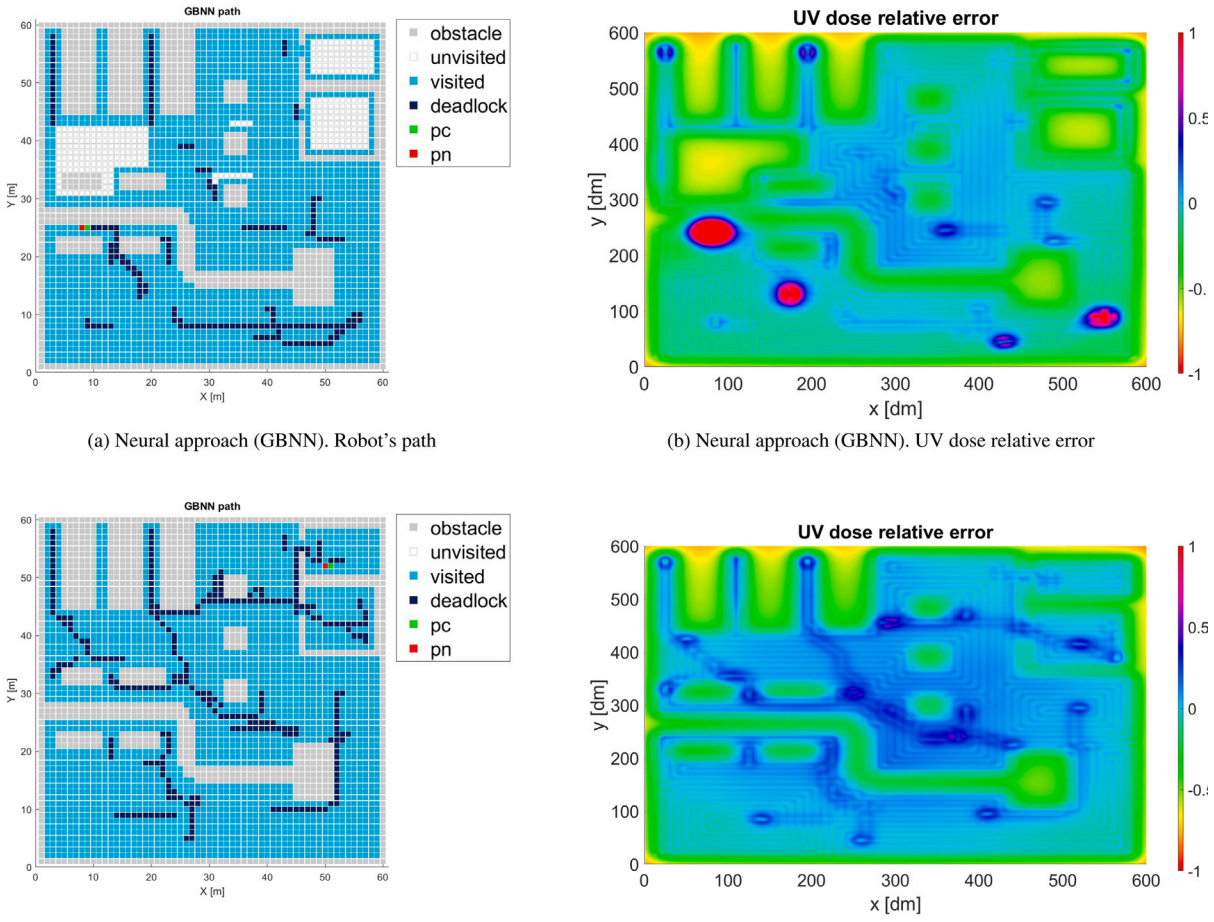
It is worth noting that the other deadlock situation shown in the example in Fig. 8(a) has been solved without the need of the ERGA.

5. Simulation results and discussion

In this section qualitative and quantitative results of the application of control approach are presented and discussed. The computer used for the tests is an Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz with a 2201 MHz clock; 1536 KB cache; and 6 processors. The operating system is Microsoft Windows 10 Education N. The algorithms have been implemented in Matlab/Simulink.

5.1. Preventive deadlock processing vs GBNN

Fig. 9 compares the results obtained when the escape route generator is active and when it is not applied in a scenario with medium



(a) Neural approach (GBNN). Robot's path

(b) Neural approach (GBNN). UV dose relative error

(c) Preventive deadlock processing. Robot's path

(d) Preventive deadlock processing. UV dose relative error

Fig. 9. Deadlock situation solved by different approaches.

obstacle distribution. The images on the left show the obstacles, the visited and unvisited cells, and the cells where a deadlock occurs. The images on the right represent the UV dose relative error regarding the expected dose, err_D. This error is computed by (12), where $D(x, y)$ denotes the dose received in the location (x, y). When err_D is negative, the cell receives less dose than expected and is not correctly disinfected. When it is positive the cell receives more dose than required, however high positive errors must be avoided in order not to degrade the environment. On the other hand, the images in the first row depict the results provided by the neural activity approach (standard GBNN) and the second one by the escape route generator.

$$err_D(x, y) = \frac{D(x, y) - D_{SARS-CoV-2}}{D_{SARS-CoV-2}} \tag{12}$$

As it may be observed, GBNN can solve the first deadlocks. However, when the robot arrives at cell (9,25), -green cell in the figure-, it gets stuck and is not able to go any further. This means that in this case an 11.66% of cells are not visited. Moreover, this issue causes a large growth in the radiation diagram. Nevertheless, using the ERGA, it is always possible to escape from these deadlock situations and the robot completes the map successfully.

Furthermore, when the ERGA is used the radiation surface is more regular and the values are closer to the required dose. By contrast, when the neural activity approach is applied some zones have very low radiation (zones near the unvisited cells with negative err_D) and others have very large radiation values (near the deadlocks with high positive err_D). It is noteworthy to remark that an excessive UV radiation may damage the environment and reduce of the useful life of the elements.

5.2. Spiral vs Boustrophedon performance evaluation

The control proposal has been evaluated with two different movement patterns: spiral and boustrophedon motions. In both cases the ERGA and the deadlock detector have been used. As it will be shown, both movement types give successful results, but the evaluation values are different. To summarize, with spiral motion less deadlocks appear and the robot requires smaller control effort. On the other hand, with the boustrophedon movement the cells are closer to the required dose and the excess of the radiation is lower.

Therefore, in general there is no one motion strategy better than another. One is good at some metrics and the other is better at the rest. The choice of method will depend on the metrics we wish to prioritize.

5.2.1. Metrics

As explained, the robot must complete the map while spreading the required radiation dose, and as much exceeding it as little as possible so as not to damage the environment, meanwhile minimizes the number of turns and the control effort for efficiency reasons. In order to quantitatively evaluate the performance of this control proposal, different metrics related to the movement of the robot and the radiation dose received have been defined:

- t_{robot}: is the total time required to complete the maps, that is, to visit all cells at least once. One of the goals is to reduce this value as much as possible to improve the efficiency of the disinfection procedure.
- $n_{Dijkstra}$: determines the number of times the Dijkstra's algorithm is required to get out of a deadlock situation. It is desirable that

this value is as small as possible. This way the control algorithm requires less computational effort and the radiation level is more regular.

- $\boldsymbol{d_{robot}}$: indicates the total distance traveled by the robot. The robot must complete the map keeping this value as small as possible.
- $\boldsymbol{r_{robot}}$: indicates the total radians rotated by the robot. Like in the total distance, it is intended to reduce as much as possible the number of turns to avoid mechanical robot wear.
- $\boldsymbol{v_{av}}$: denotes the average velocity of the robot, calculated as d_{robot} divided by t_{robot}. Therefore, the smaller this value is, the smoother the movement will be.
- $\boldsymbol{CE}$: indicates the control effort. This metric is usually considered in control studies. CE is based on the angular velocity reference. As expected, many turns may increase the overall consumption and the degradation of the robot. This control effort is expressed in (13), where $w_{ref}(t)$ is the angular speed reference.

$$CE = \sqrt{\frac{\int w_{ref}(t)^2 dt}{t_{robot}}} \tag{13}$$

- $\boldsymbol{tc_{ex}}$: generally, to complete the map the number of visited cells is larger than the available cells. This metric measures this excess of visited cells, and it is defined as the ratio between total traveled cells vs available cells (14). This metric is related to d_{robot} and should be as small as possible.

$$tc_{ex} = \frac{cells_{\ Traveled} - cells_{\ Available}}{cells_{\ Available}} \cdot 100 \tag{14}$$

The ratio between complete cells vs available cells is not considered because, as the map is always complete, all cells are explored, so this value would be 100% in all cases.

- $\boldsymbol{D_{max}}$: is the maximum radiation dose received by a cell of the map. It should be very close to the specified.
- $\boldsymbol{D_{min}}$: is the minimum radiation dose received in a cell of the map. This value should be very close to the required dose.
- $\boldsymbol{D_N}$: is the percentage of cells that receive the desired dose range. It should be noticed that the purpose of the disinfection is to maximize the percentage of cells with a UV radiation dose in the expected range, i.e., (15), where $D_{SARS-CoV-2}$ is the SARS-CoV-2 required dose to achieve the expected disinfection, and D is the final UV dose received.

$$D_N = \left[\frac{1}{cells_{\ Available}} \sum_{(i,j)} 1\right] \cdot 100, \quad (i,j) \in \mathbb{N}^2 \mid \quad 0.9 D_{SARS-CoV-2} < D(i,j) < 1.1 D_{SARS-CoV-2} \tag{15}$$

- $\boldsymbol{D_H}$: is the percentage of cells whose dose is higher than $1.1 D_{SARS-CoV-2}$. It is desirable to reduce this value as much as possible to avoid environment damages.
- $\boldsymbol{D_L}$: is the percentage of cells whose dose is lower than $0.9 D_{SARS-CoV-2}$. It is desirable to reduce this as much as possible to guarantee a correct disinfection.
- $\boldsymbol{NC}$: is the percentage of non-visited cells at the end of the simulation if the PDPA is not used. That is, the standard GBNN is used instead of DLE+ERGA. This metric informs us about the benefit of the use of PDPA.

5.2.2. Simulation scenarios

To evaluate the performance of the proposal, four scenarios of growing complexity have been considered: empty, low, medium and high (Fig. 10). When the complexity increases the size of the map and the number of obstacles grow. These scenarios have been designed to create challenging situations where the robot goes into some zones where there is only a way to escape.

In all these scenarios, the initial position is set to $p = (2, 2, \frac{\pi}{2})$, and the parameters of the model $[\alpha, \beta, \rho_0, c, E]$ are $[2, 0.6, 7, 0.1, 100]$. The intensity of the lamp is constant and provides $I_{UV} = 5.5\ \mathrm{W/m^2}$. In addition, the value $D_{SARS-CoV2} = 500\ \mathrm{J/m^2}$ is set as the reference value to achieve disinfection; $d_{LA} = 0.2$ as the anticipation value for the pure pursuit controller. The sample time for simulation is $dt = 0.1$ ms. As the size of the map changes depending on the scenario, the value ρ of the speed controller has been adjusted for each one: 1.11 is used for the low complexity scenario, 1.25 for medium one, and 1.43 for the most complex one.

5.2.3. Comparison of results

Table 1 shows all metrics obtained when the control approach is applied to the four different scenarios, with both the spiral movement and the boustrophedon one. The ρ value used in each scenario is indicated together the name. For each scenario the best results have been boldfaced in the Table. For D_H the minimum is the best; for D_{min}, the best is the value closest to the expected value, that is, 550. As all the results are below this expected dose, the best value of D_{min} is the maximum.

Looking at the times required by the robot in each of the experiments, the type of motion strategy has no influence on it. This is due to the speed controller described in (9), as it allows the speed to be increased or decreased so that the radiation absorbed in the environment is appropriate. That is, thanks to the algorithm, the disinfection process can be successfully completed by adjusting the speed according to the number of times a robot passes through the same area.

If we look at the number of existing deadlock situations, as expected this number increases as the complexity of the map increases. a higher number of deadlock situations means a greater distance traveled and a higher number of turns. However, whenever boustrophedon motion is used, the number of deadlocks is always higher.

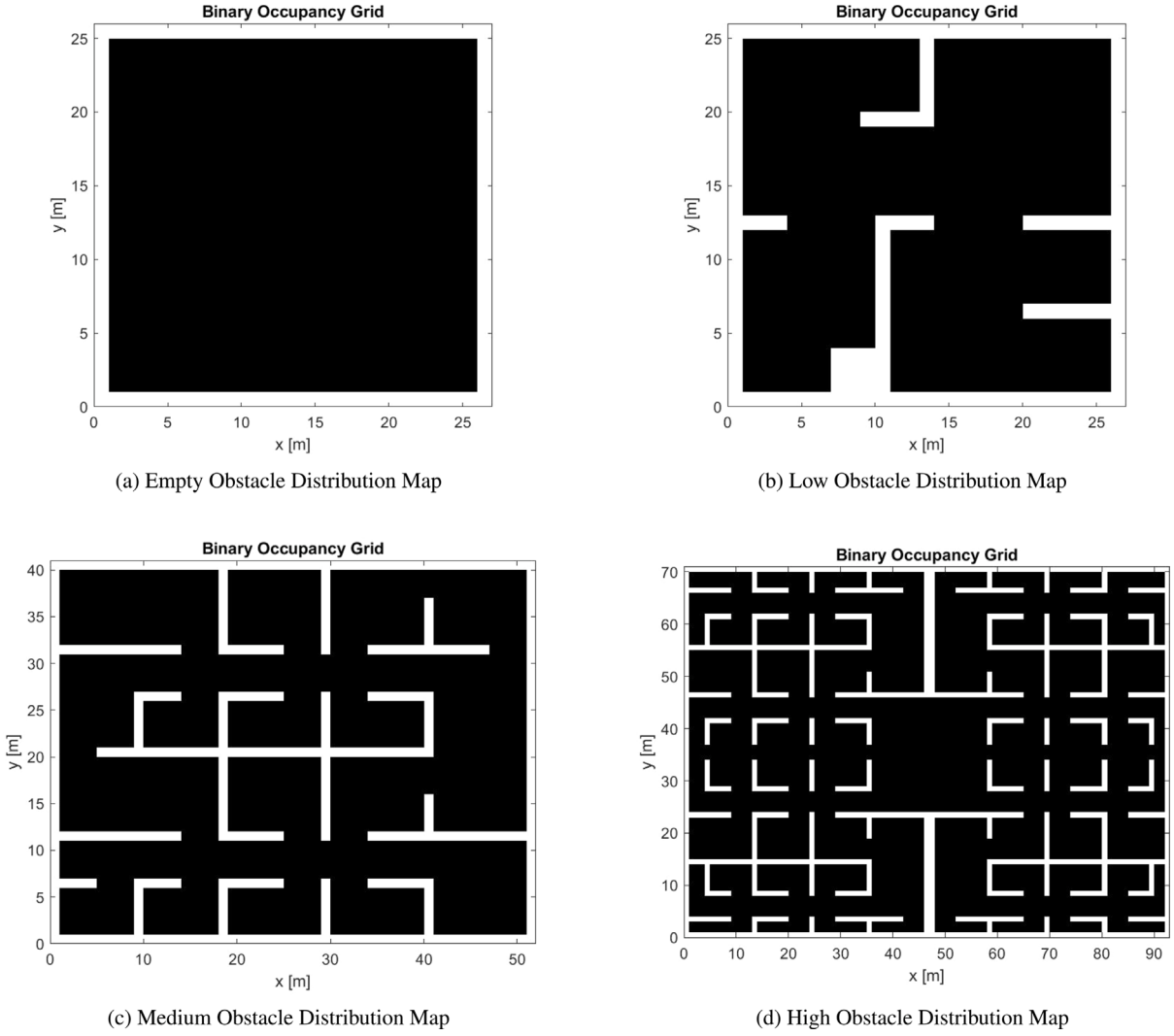
Although the distance traveled is quite similar for both strategies, the difference between the radians turned by the robot for each motion increases as the complexity of the map grows. For the high obstacle distribution scenario, the difference is approximately 980 rad. Since one revolution is $2pi$ radians, the robot using boustrophedon motion turns approximately 160 revolutions more than the robot using spiral motion. Therefore, using boustrophedon motion would result in more fatigue and higher energy consumption.

In almost all cases the maximum dose (D_{max}) is smaller with the boustrophedon movement although in all cases radiation peaks are produced, mainly due to the turns. It is often the case that in cells that are adjacent to an obstacle, the received dose is below the threshold, as they are typically hardly passed through when the robot escapes from a deadlock situation (diagonal movements are often used to get to a non-disinfected area faster). However, in the case of the minimum dose (D_{min}), both movement types produce similar results.

As the maximum dose is smaller with boustrophedon, this strategy also gives larger percentage of cells with the desired dose range D_N. It may be explained as the spiral movement tends to create peaks in the center of the spiral, especially at the beginning of the ellipse.

By inspecting NC in the table, it is possible to realize how this value grows with the complexity of the map. In other words, if the complexity of the map is high, the GBNN is not enough to solve the deadlocks and the application of the DLE and ERGA is necessary. Another interesting result is that when only GBNN is used, the boustrophedon movement works worse than the spiral one.

In addition to the quantitative results, graphical results have been also obtained. Fig. 11 shows the results when the control algorithm is used with the high complexity map and the two types of movement. Figures on the left (a, c, e) represent the results with spiral motion and figures (b, d, f) with the boustrophedon. On the other hand, figures of the first row show the path, the second row represents the cells and found deadlocks, and figures in the last row show the radiation dose administered.



(a) Empty Obstacle Distribution Map

(b) Low Obstacle Distribution Map

(c) Medium Obstacle Distribution Map

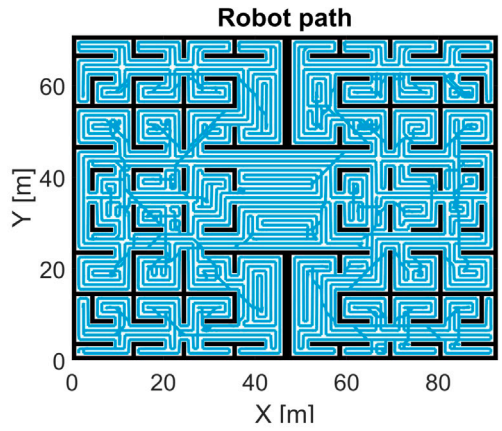
(d) High Obstacle Distribution Map

Fig. 10. Working environments used in the simulation for the analysis of the disinfection process.

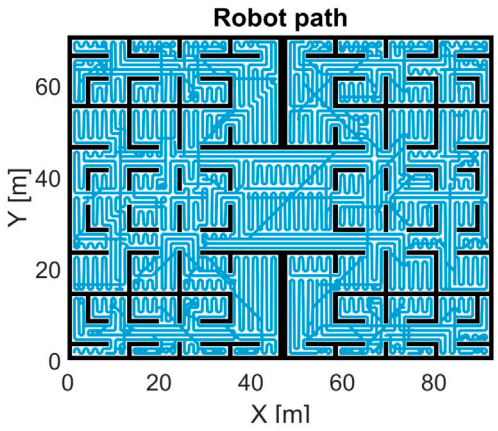
Table 1
Disinfection process analysis.

Empty Obstacle Distribution Map ($\rho \approx 1.11$)													
$Motion$	t_{robot}	$n_{Dijkstra}$	$d_{robot}[m]$	$r_{robot}[rad]$	$v_{av}[m/s]$	CE	$tc_{ex}[\%]$	$D_{max}[J/m^2]$	$D_{min}[J/m^2]$	$D_N[\%]$	$D_H[\%]$	$D_L[\%]$	$NC[\%]$
Spiral	01 : 03 : 38	0	596.70	**73.86**	0.16	0.23	0	596.53	**237.77**	73.45	13.97	12.58	0.00
Boustrophedon	**01 : 03 : 18**	0	**595.85**	75.20	0.16	0.23	0	**577.89**	188.83	**78.15**	**9.49**	**12.36**	0.00
Low Obstacle Distribution Map ($\rho \approx 1.11$)													
$Motion$	t_{robot}	$n_{Dijkstra}$	$d_{robot}[m]$	$r_{robot}[rad]$	$v_{av}[m/s]$	CE	$tc_{ex}[\%]$	$D_{max}[J/m^2]$	$D_{min}[J/m^2]$	$D_N[\%]$	$D_H[\%]$	$D_L[\%]$	$NC[\%]$
Spiral	**01 : 01 : 33**	**9**	639.75	**306.41**	0.17	**0.48**	15.24	798.48	**231.44**	57.28	**26.36**	16.36	**0.00**
Boustrophedon	01 : 01 : 54	10	**637.61**	377.18	0.17	0.54	15.24	**695.46**	204.91	**65.13**	20.64	**14.23**	14.34
Medium Obstacle Distribution Map ($\rho = 1.25$)													
$Motion$	t_{robot}	$n_{Dijkstra}$	$d_{robot}[m]$	$r_{robot}[rad]$	$v_{av}[m/s]$	CE	tc_{ex}	$D_{max}[J/m^2]$	$D_{min}[J/m^2]$	$D_N[\%]$	$D_H[\%]$	$D_L[\%]$	$NC[\%]$
Spiral	**02 : 47 : 36**	**47**	**2235.91**	**1168.96**	**0.22**	**0.57**	**25.74**	**719.98**	228.93	55.66	31.56	12.77	**54.57**
Boustrophedon	02 : 48 : 10	55	2316.56	1406.77	0.23	0.63	30.69	722.39	**233.14**	**61.16**	**28.21**	**10.63**	80.31
High Obstacle Distribution Map ($\rho \approx 1.43$)													
$Motion$	t_{robot}	$n_{Dijkstra}$	$d_{robot}[m]$	$r_{robot}[rad]$	$v_{av}[m/s]$	CE	tc_{ex}	$D_{max}[J/m^2]$	$D_{min}[J/m^2]$	$D_N[\%]$	$D_H[\%]$	$D_L[\%]$	$NC[\%]$
Spiral	**07 : 27 : 10**	**119**	**6794.35**	**3380.95**	**0.25**	**0.59**	**22.26**	739.34	224.57	57.67	29.96	12.37	**58.37**
Boustrophedon	07 : 28 : 39	161	7070.34	4358.31	0.26	0.68	27.47	**702.93**	**224.54**	**61.28**	**28.18**	**10.54**	99.02

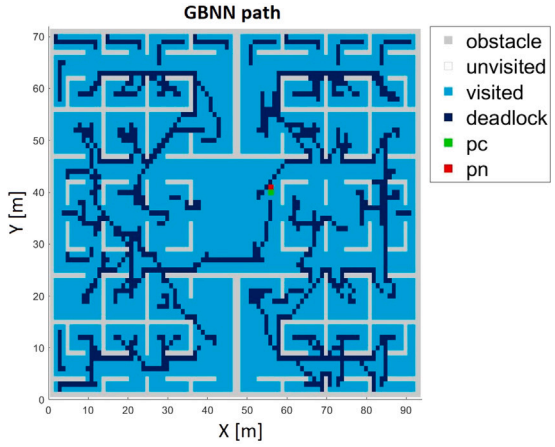
Some numerical results can be confirmed in the figures. For instance, the boustrophedon tends to generate more turns, and this is even more evident in the narrow aisles if they are not aligned with the main direction of the movement. It is possible to see this at the top left. In addition, when the boustrophedon is used, there are differences between following the longest or the shortest axis in the workspace. If the robot moves across the longer side of the room it will make more turns than in the other case. This dependency with the direction does



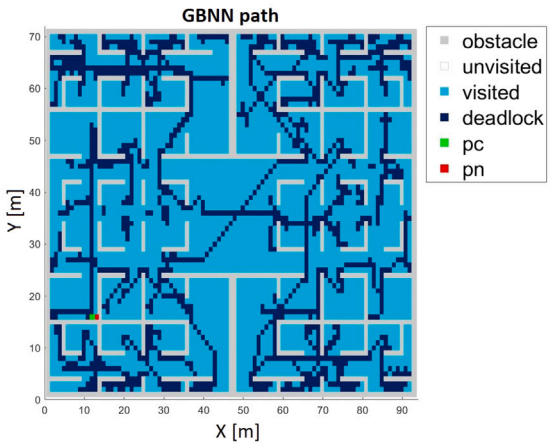
(a) Differential Robot Path (Spiral motion)



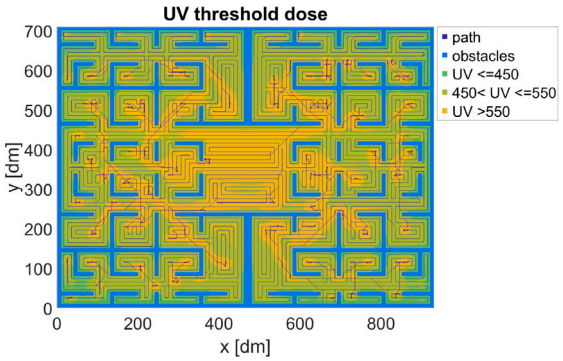
(b) Differential Robot Path (Boustrophedon motion)



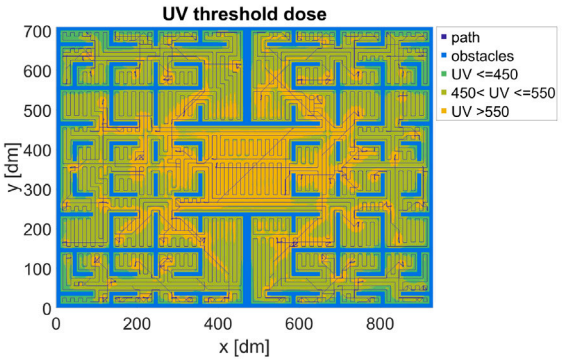
(c) Enhanced GBNN path (Spiral motion)



(d) Enhanced GBNN path (Boustrophedon motion)



(e) D_{UV} received categorized into thresholds (Spiral motion)



(f) D_{UV} received categorized into thresholds (Boustrophedon motion)

Fig. 11. High obstacle distribution map simulation.

not happen with the spiral movement, being this one of its advantages. Moreover, reducing the turns saves the robot battery.

It is also possible to observe that there are fewer deadlocks when the spiral movement is applied. On the whole, deadlocks are more frequent in narrow zones where the robot has only one way to escape. For instance, in the zones located at the top and the bottom of the map.

On the other hand, as far as the radiation map is concerned, the zones with higher values are at the crossing paths where the robot has been before. With both motions there is a zone in the middle of the map with a high dose. Although the radiation dose is similar in both cases, there are more zones where the radiation overpasses the required dose when the spiral is applied.

6. Conclusions and future works

In this paper we have developed a simulation model to test CCPP algorithms for UV-C disinfection. It includes a novel control architecture that combines GBNN for discretizing the map, a motion strategy to improve path planning in the presence of obstacles, a UV-C radiation dose estimation in conjunction with a speed controller that adjusts the robot speed to get the proper radiation dose, and finally, a pure pursuit controller to ensure the mobile robot moves correctly.

To address deadlock issues, an original preventive deadlock processing approach is proposed with the ERGA. The robot is able to anticipate deadlocks by evaluating the surrounding cells and to react so to avoid

getting stuck. When a deadlock is estimated, the method generates the best escape route to the nearest non-disinfected zone using the ERGA.

Two motion techniques were tested to see how they influence the algorithm performance: boustrophedon and spiral movements. Spiral motion reduces deadlocks and allows the robot to move with less control effort. The boustrophedon movement, on the other hand, brings the cells closer to the appropriate dosage and reduces the extra radiation. As a result, we conclude that no motion is superior to another. One excels at some metrics while the other stands out at others. The technique to be used will be determined by the metrics we want to prioritize.

When the PDPA is used, the radiation diagram becomes more regular, and the values better approach the necessary dose. When only GBNN is applied, certain zones have extremely low radiation values (zones near unvisited cells), while others have very high radiation levels (zones near the deadlocks). It is worth noting that excessive UV-C radiation can wear out the environment and shorten the useful life of goods.

As future work lines we may highlight to test the proposal with a real robot, the use of machine learning techniques to predict earlier the deadlock situations, and to incorporate reinforcement learning to improve the escape route generator.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Organization WH, et al. Cleaning and disinfection of environmental surfaces in the context of COVID-19. 2020.

[2] Fuchs FM, Bibinov N, Blanco EV, Pfaender S, Theiß S, Wolter H, et al. Characterization of a robot-assisted UV-C disinfection for the inactivation of surface-associated microorganisms and viruses. J Photochem Photobiol 2022;11:100123.

[3] Disinfection robot market. 2022, https://www.persistencemarketresearch.com/market-research/disinfection-robot-market.asp. [Accessed 13 July 2022].

[4] Sierra Garcia J, Guillen-Grima F, Garcia-Garcia M, Cascajar L, Rodriguez-Merino F. Evaluation of an UVC robot for terminal disinfection of hospital rooms. Antimicrob Resist Infect Control 2021.

[5] Mitxelena-Iribarren O, Mondragon B, Pérez-Lorenzo E, Smerdou C, Guillen-Grima F, Sierra-Garcia JE, et al. Evaluation of the degradation of materials by exposure to germicide UV-C light through colorimetry, tensile strength and surface microstructure analyses. Mater Today Commun 2022;103690.

[6] Jelden KC, Gibbs SG, Smith PW, Hewlett AL, Iwen PC, Schmid KK, et al. Ultraviolet (UV)-reflective paint with ultraviolet germicidal irradiation (UVGI) improves decontamination of nosocomial bacteria on hospital room surfaces. J Occup Environ Hyg 2017;14(6):456–60.

[7] Khan A, Noreen I, Habib Z. On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges. J Inform Sci Eng 2017;33(1). http://dx.doi.org/10.6688/JISE.2017.33.1.7.

[8] Galceran E, Carreras M. A survey on coverage path planning for robotics. Robot Auton Syst 2013;61(12):1258–76. http://dx.doi.org/10.1016/j.robot.2013.09.004.

[9] Li J, Yang SX, Xu Z. A survey on robot path planning using bio-inspired algorithms. In: 2019 IEEE international conference on robotics and biomimetics. IEEE; 2019, p. 2111–6.

[10] Garcia-Aunon P, Roldan J, De Leon J, Del Cerro J, Barrientos A. Practical applications using multi-UAV systems and aerial robotic swarms. Revista Iberoamericana Automática E Informática Ind 2021;18(3):230–41. http://dx.doi.org/10.4995/riai.2020.13560.

[11] Zhu D, Tian C, Sun B, Luo C. Complete coverage path planning of autonomous underwater vehicle based on GBNN algorithm. J Intell Robot Syst 2019;94(1):237–49.

[12] Zhu D, Tian C, Jiang X, Luo C. Multi-AUVs cooperative complete coverage path planning based on GBNN algorithm. In: 2017 29th Chinese control and decision conference. IEEE; 2017, p. 6761–6.

[13] Muthugala MVJ, Samarakoon SBP, Elara MR. Toward energy-efficient online complete coverage path planning of a ship hull maintenance robot based on glasius bio-inspired neural network. Expert Syst Appl 2022;187:115940.

[14] Yao P, Zhao Z. Improved glasius bio-inspired neural network for target search by multi-agents. Inform Sci 2021;568:40–53.

[15] Chen M, Zhu D. Real-time path planning for a robot to track a fast moving target based on improved glasius bio-inspired neural networks. Int J Intell Robot Appl 2019;3(2):186–95.

[16] Kowalski W, Walsh T, Petraitis V. COVID-19 coronavirus ultraviolet susceptibility. Purplesun Inc; 2020.

[17] Raeiszadeh M, Adeli B. A critical review on ultraviolet disinfection systems against COVID-19 outbreak: Applicability, validation, and safety considerations. ACS Photonics 2020;7(11):2941–51. http://dx.doi.org/10.1021/acsphotonics.0c01245.

[18] Kheyrandish A, Taghipour F, Mohseni M. UV-LED radiation modeling and its applications in UV dose determination for water treatment. J Photochem Photobiol A Chem 2018;352:113–21.

[19] Rodrigo DV. Bio-inspired complete coverage path planning for SARS-COV-2 ultraviolet disinfection robots. Complutense University of Madrid; 2021.

[20] Arguelles P. Estimating UV-C sterilization dosage for COVID-19 pandemic mitigation efforts. 2020, Preprint April..

[21] Une 0068:2020 safety requirements for uv-c equipment used for room air disinfection and surfaces. 2020, https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0064094. [Accessed 13 July 2022].

[22] Derraik JG, Anderson WA, Connelly EA, Anderson YC. 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. Int J Environ Res Public Health 2020;17(17):6117. http://dx.doi.org/10.3390/ijerph17176117.

[23] Malu SK, Majumdar J. Kinematics, localization and control of differential drive mobile robot. Int J Environ Res Public Health 2020;17(17):6117. http://dx.doi.org/10.3390/ijerph17176117.

[24] Sun B, Zhu D, Tian C, Luo C. Complete coverage autonomous underwater vehicles path planning based on glasius bio-inspired neural network algorithm for discrete and centralized programming. IEEE Trans Cogn Dev Syst 2019;11(1):73–84. http://dx.doi.org/10.1109/TCDS.2018.2810235.

[25] Gunning R. A performance comparison of coverage algorithms for simple robotic vacuum cleaners. 2018.

[26] Hasan KM, Reza KJ. Path planning algorithm development for autonomous vacuum cleaner robots. In: 2014 International conference on informatics, electronics vision. 2014, p. 1–6. http://dx.doi.org/10.1109/ICIEV.2014.6850799.

[27] Fischer RJ, Morris DH, van Doremalen N, Sarchette S, Matson MJ, Bushmaker T, et al. Effectiveness of N95 respirator decontamination and reuse against SARS-CoV-2 virus. Emerg Infect Diseases 2020;26(9):2253. http://dx.doi.org/10.3201/eid2609.201524.

[28] Coulter RC. Implementation of the pure pursuit path tracking algorithm. Technical Report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST; 1992.

[29] Luo C, Yang SX. A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. IEEE Trans Neural Netw 2008;19(7):1279–98. http://dx.doi.org/10.1109/TNN.2008.2000394.

[30] Javanmard R. Reza Javanmard (2021). Discretizing (tiling) the input map into a grid graph, and finding shortest path in non-convex map. 2016.