applied sciences

MDPI

Article

Path Optimization Using Metaheuristic Techniques for a Surveillance Robot

Mario Peñacoba [1,*], Jesús Enrique Sierra-García [1,*], Matilde Santos [2] and Ioannis Mariolis [3]

[1] Department of Digitalization, University of Burgos, 09001 Burgos, Spain
[2] Institute of Knowledge Technology, Complutense University of Madrid, 28040 Madrid, Spain
[3] Centre for Research and Technology Hellas, Information Technologies Institute, 570 01 Thessaloniki, Greece
* Correspondence: mpenacoba@ubu.es (M.P.); jesierra@ubu.es (J.E.S.-G.)

Abstract: This paper presents an innovative approach to optimize the trajectories of a robotic surveillance system, employing three different optimization methods: genetic algorithm (GA), particle swarm optimization (PSO), and pattern search (PS). The research addresses the challenge of efficiently planning routes for a LiDAR-equipped mobile robot to effectively cover target areas taking into account the capabilities and limitations of sensors and robots. The findings demonstrate the effectiveness of these trajectory optimization approaches, significantly improving detection efficiency and coverage of critical areas. Furthermore, it is observed that, among the three techniques, pattern search quickly obtains feasible solutions in environments with good initial trajectories. On the contrary, in cases where the initial trajectory is suboptimal or the environment is complex, PSO works better. For example, in the high complexity map evaluated, PSO achieves 86.7% spatial coverage, compared to 85% and 84% for PS and GA, respectively. On low- and medium-complexity maps, PS is 15.7 and 18 s faster in trajectory optimization than the second fastest algorithm, which is PSO in both cases. Furthermore, the fitness function of this proposal has been compared with that of previous works, obtaining better results.

Keywords: robotics; surveillance; inspection; optimization; genetic algorithm; particle swarm; pattern search

check for updates

Citation: Peñacoba, M.; Sierra-García, J.E.; Santos, M.; Mariolis, I. Path Optimization Using Metaheuristic Techniques for a Surveillance Robot. *Appl. Sci.* **2023**, *13*, 11182. https://doi.org/10.3390/app132011182

Academic Editors: Dwayne McDaniel and Leonel E. Lagos

Received: 31 August 2023
Revised: 28 September 2023
Accepted: 10 October 2023
Published: 11 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

It is undeniable that along with any type of industrial application there is an associated monitoring task. This is essential to avoid accidents, sabotage, or theft that could disrupt normal operation and, in the worst case, lead the industry to bankruptcy. For this reason, for decades a part of the personnel of companies has been dedicated to surveillance and control tasks. However, with recent advances in mobile robotics and optimization algorithms, it is possible to increase the efficiency of such tasks, reducing human errors and improving performance, quality, and monitoring.

In order to design a procedure capable of improving the quality of these tasks beyond human capabilities, it is imperative to address all the challenges that this implies. Specifically, within the scope of trajectory optimization, efficient route planning for autonomous vehicles is essential to minimize labor time while providing comprehensive spatial surveillance in the most efficient way possible.

In the current market there are robots that receive orders to navigate freely to a specific point (usually called free navigation) or to navigate following predefined trajectories. Both approaches allow the creation of paths that run through a given space, facilitating the implementation of surveillance applications. However, waypoints must be chosen judiciously to maximize efficiency, that is, to cover the maximum area in the shortest possible time [1].

To address this important issue, this paper proposes the utilization of different heuristic techniques, such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and

Pattern Search (PS), as tools to optimize these route waypoints and develop a complete coverage path planning (CCPP) approach. These approaches were chosen due to their capability to solve complex and dynamic challenges in trajectory planning [2].

In particular, this research explores how each evolutive approach influences the quality and efficiency of surveillance coverage, considering the characteristics of a Light Detection and Ranging (LiDAR) sensor, and a commercial mobile robot. A LiDAR is a remote sensing technology that uses laser light to measure distances, typically employed in applications such as mapping, surveying, and object detection. The proposed methodology aims to minimize robot displacements while achieving complete spatial coverage in the shortest time possible for different maps. The effectiveness of the three aforementioned optimization methods (GAs, PSO, and PS) has been studied and compared in rooms of different sizes and with obstacles, for the devices available in these use cases.

As the methodology proposed here is general, with minor and simple adjustments it can be applied to any environment, with different types of sensors and for any differential robot. It could even be extended to other surveillance sensors, such as a smoke or gas detector [3]. To replace the sensor, it would be sufficient to adjust the range, pitch angle, and sweep angle parameters in the surveillance parameter calculation code. On the other hand, to use another robot it would be necessary to adjust the parameters of the length and angular acceleration ramps in the trajectory simulation code. All of these changes can be made quickly and easily.

The main contributions of this work can be summarized as follows:

- Development of a new methodology to solve the CCPP problem based on polyline optimization. An innovative methodology is introduced that optimizes the route planning of a mobile robot. The approach considers obstacles in the environment, maximizes coverage, and minimizes time. GA, PS, and PSO were applied as optimization techniques to solve this problem efficiently.
- Application of the CCPP methodology to an autonomous surveillance robot with LiDAR technology.
- A complete set of simulation experiments are carried out to evaluate the performance of the proposed methodology. The obtained optimized trajectories outperform the initial suboptimal trajectories, showing a significant reduction in travel time and an increase in coverage percentage when the entire target space cannot be covered with the specified parameters. The fitness function used here is compared with that of other previous works and provides better results.
- Three evolutionary comparison techniques are compared. The results show that, although all are efficient in this CCPP problem, PS gives superior performance in simple and medium-complexity environments. However, the PSO method excels in very complex environments.

The organization of this paper is as follows. A brief state of the art is presented in Section 2. The case of use, including the experimental setup, is detailed in Section 3. The results obtained with the three proposed heuristic optimization approaches are presented and compared in Section 4. The strengths and weaknesses of each method are identified in terms of coverage, mission duration, and event detection efficiency. The paper ends with the conclusions and directions for future work.

2. Related Works

The need for safety in both outdoor and indoor environments has led to growth in the development of mobile robots with the ability to explore surfaces. Surveillance robots are used to monitor the behavior, activities, and other changing information for the general purpose of managing, directing, or protecting one's assets or position [4]. Mobile robots assigned the task of surveillance play a crucial role in different sectors: security, health, rescue, agriculture, institutions, etc. [5]. As an example of this variety of uses, an application of surveillance of engine rooms in a ship by autonomous mobile robots to detect fire is presented in [6]. A map of the engine room was created using an autonomous

robot, and when a destination was set on the map, a path was found and the engine room was surveilled autonomously. The paper by Zhang et al., reviews the latest research on AGVs and AMRs, and discusses visual tracking control technologies in civil engineering applications [7].

Security robots are commonly used to protect and safeguard a location, some valuable assets, or personal property against danger, damage, loss, and crime. Most of the works found in the literature on this specific application of mobile robots work with aerial vehicles (UAVs). The papers by Stolfi et al., present a surveillance system for early detection of individuals using a swarm mobility model, in the first paper, and a swarm of unmanned aerial vehicles with a Competitive Coevolutionary GA that aims to maximize intruder detection in the second reference [8,9]. Sun et al. designed a two-mode monitoring systems for industrial facilities and their adjacent territories based on the application of unmanned aerial vehicle (UAV), Internet of Things (IoT), and digital twin (DT) technologies [10]. Prykhodchenko et al. present a single-robot solution for the surveillance of buildings [11]. The robot operates with a combination of different people detection techniques for detecting and tracking humans. The paper by Lee and Shih proposes an autonomous robotic system based on a convolutional neural network to perform visual perception and control tasks [12]. The visual perception aims to identify all objects moving in the scene and to verify whether the target is an authorized person.

There are different systems for security and surveillance which are currently available in the market. According to [13], conventional patrolling lacks an integrated multi-sensing system that coordinates various technologies for surveillance and detection of human intruder movement in the different scenarios. They usually are equipped with cameras. Some of them are controlled remotely [14]. Nevertheless, Light Detection and Ranging (LiDAR) sensors are affordable in terms of the acquisition price and processing requirements. Examples of applications of this sensor can be found in [15], where the authors describe a mobile robot that operates in an indoor environment and it is capable of tracking pairs of legs in a cluttered environment using a 2D LiDAR scanner. The already mentioned paper by Kim and Bae creates a map of the engine room using LiDAR technology [6]. The authors in [16] systematically review and analyze in mobile robotics the use 3D ToF LiDARs in research and industrial applications. In [17], a multimodal sensor module for multiple fixed and mobile agents in outdoor environments is proposed. It has a vision sensor and four-channel sound that are synchronized and integrates them using a 3D LiDAR and a calibration method. This sensor equipment enables integrated data collection for a 24 h monitoring of the outdoor surveillance area. A graph theoretic approach including heuristic algorithms for optimal point-to-point navigation using a LiDAR sensor is presented in [18], ensuring total workspace coverage and minimization of action performed by an indoor robot.

Path planning is the core task for the AGV system, and it generates the path from origin to destination. However, the implementation of path planning and trajectory tracking by autonomous robots generates a series of optimization problems that can be dealt with using various techniques [19]. In Ayvali et al., stochastic trajectory optimization methods are used. The authors introduce the ergodicity metric as an objective in a sampling-based stochastic trajectory optimization framework for a mobile robot. In their approach, they construct a probability distribution over feasible trajectories and search for the optimal trajectory using the cross-entropy method [20]. A different approach consists of addressing these challenges with heuristic optimization techniques.

In this work, we apply and compare three different heuristic optimization techniques to the CCPP problem, namely, GA, PS, and PSO. Briefly, their main characteristics regarding the path planning problem are as follows.

GAs are computer programs that mimic the processes of biological evolution (selection, crossover, mutation) to solve problems and model evolutionary systems [21]. This feature makes them suitable for solving optimization problems that require an adaptive computer program. In the case of path planning, they allow researchers to easily introduce a fitness

function with more than one objective that also maintains several possible solutions at the same time.

In PSO, a group of simple entities called "particles" are positioned within the search space of a given problem or function. Each particle evaluates the objective function at its current location and determines its path through the search space by fusing certain aspects of its individual history, the location that has the best fitness, and those of one or more peers within the swarm [22]. The search approach used by this method is particularly suitable for the problem of optimizing a trajectory in a complex environment since it leads to the rapid identification of a solution, as shown in the results obtained in this problem.

PS is an effective technique for exploring the minima of a function, even when that function is not differentiable, exhibits stochastic behavior, or is not necessarily continuous [23]. The PS algorithm explores a set of points surrounding the current location and actively searches for a point within this set where the value of the fitness function is less than at the current point. This feature makes this method suitable for finding routes as long as the initial route is close to a good solution. This suggests that this method will work effectively in small environments with few obstacles.

Some other works have explored these techniques. For instance, the work in [24] proposes an hybrid PSO-SA algorithm for the optimization of AGV path planning. In [25], a comprehensive review of methodologies for path planning and optimization of mobile robots is provided. It includes not only the classic but state-of-the-art techniques such as artificial potential fields, GA, swarm intelligence, and machine learning-based methods. The paper by Xiao et al. focuses on the indoor AGV path-planning problem in large-scale, complex environments and proposes an efficient path-planning algorithm (IACO-DWA) that incorporates the ant colony algorithm (ACO) and dynamic window approach (DWA) to achieve multi-objective path optimization [26]. First, an improved ant colony algorithm (IACO) is proposed to plan a global path for AGVs that satisfies a shorter path and fewer turns. Then, local optimization is performed between adjacent key nodes by improving and extending the evaluation function of the traditional dynamic window method (IDWA), which further improves path security and smoothness. An overview of navigation strategies for mobile robots that utilize three classical approaches, roadmap (RM), cell decomposition (CD), and artificial potential fields (APF), in addition to eleven heuristic approaches, including GA, ACO, artificial bee colony (ABC), gray wolf optimization (GWO), etc., which may be used in various environmental situations is presented in [27].

The main aspects that this work deals with in relation to those previously mentioned can be presented in three groups: the monitored environment, the surveillance method, and the optimization of path planning. Regarding the first, the different places that are monitored are for indoor or outdoor spaces. In our case, given the characteristics of the robot and the sensor, the proposed methodology can be applied in both environments. Regarding the method used to carry out the surveillance task, in this work a mobile ground robot with a built-in 3D LiDAR (https://www.sick.com/es/es/sensores-lidar/sensores-3d-lidar/multiscan100/c/g574914 (accessed on 27 September 2023)) is proposed, but as it has been shown in other papers, this task could be carried out with aerial vehicles, or with a swarm of robots, and also using different sensors. Finally, there are many optimization methods that have been proven effective and each of them has advantages and disadvantages, as has been shown in this work in which three of them, GA, PSO and PS, that are well-known and easily implemented are compared.

3. Description of the Experimental Setup and Simulation Scenarios

Effective surveillance is crucial in security and monitoring applications. In robotics, a common challenge is to determine the optimal trajectory that ensures complete coverage of the target area while minimizing redundancy and maximizing detection time. It is essential to design routes that are efficient in terms of distance traveled and time invested, as well as to identify the path that consumes less energy, with less wear and, at the same time, that

performs more work in the minimum time, in order to provide the greatest value to the system.

This study explores how to apply advanced surveillance trajectory optimization techniques, such as GA, PSO, and PS, to improve the surveillance trajectories of a Hussar mobile robot [28] equipped with a 6 m range 3D 360 LiDAR [29].

The Hussar robot is a square-bodied delivery robot with extensive mobility, positioning, and remote trajectory tracking capabilities. It is also compact and versatile, allowing it to effortlessly navigate through tight spaces and obstacle-filled environments, making it ideal for areas with narrow passages. It has dimensions ($L{\cdot}W{\cdot}H$) of 450 mm $\times$ 450 mm·317 mm. It has an important autonomy of 6 h and an automated recharging system.

As for the robot's mobility, it is capable of moving with speeds in a range from 0.2 to 0.8 m/s. However, the version available in the laboratory is limited to 0.5 m/s. Therefore, in the experiments its speed is considered constant and has been set at the maximum of 0.5 m/s.

The robot is equipped with an Android 5.1 operating system and can establish communication with the on-board computer via a PC, Raspberry Pi, or Arduino using ROS (Robot Operating System, http://www.reemanrobot.com/robot-chassis/square-robot-chassis/automatic-square-robot-chassis.html (accessed on 27 September 2023)).

Given the need to detect and distinguish obstacles and people from other types of potentially malicious objects, a 3D LiDAR type sensor is necessary. The detection data from the LiDAR are intersected with the static map of the environment stored inside the robot. This process helps identify if an object is an obstacle or a potential alert.

The 3D multiScan100 from SICK (SICK, Waldkirch, Germany) [30] was chosen, which offers a 360-degree range, operates in three dimensions, and has a distance range of up to 10 m with 10% reflectance.

For security reasons, the distance that has been programmed in the simulation environment is 4 m. This ensures that the distance between two consecutives sensing samples in the same sensing sweep is below 1 cm.

For data transmission, this device has an Ethernet communication interface.

The system architecture (Figure 1) is composed of three elements: the LiDAR multiScan100 (from SICK, Waldrick, Germany), a Raspberry Pi 4 (from Raspberry Pi Foundation, Cambridge, UK), and Reeman's Hussar robot (from Reeman Robotics, Shenzen, China). The process by which objects are detected is as follows: the LiDAR, powered by the same source as the Raspberry Pi, collects positional data of objects in the environment and transmits them to the Raspberry Pi as a point cloud. The Raspberry Pi compares the positions collected by the multiScan100 with the map. If an object not present in the map is detected, the system issues an alarm.

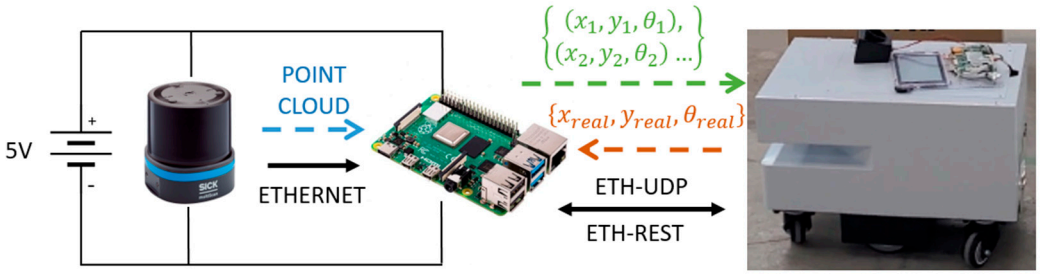
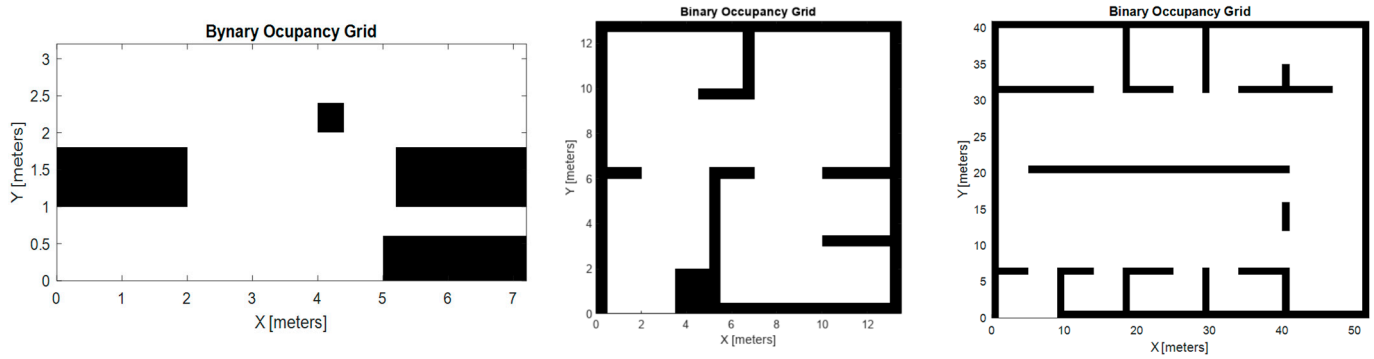


Figure 1. System architecture.

On the other hand, the navigation algorithm is also integrated in the Raspberry, and it works as follows: the autonomous robot communicates its current position, and the control device recognizes it and commands the next position in the navigation path. This approach requires the robot to function properly; for instance, if a wheel breaks down and the movement of the robot becomes impossible, it will not be possible to perform the surveillance correctly.

The simulation environments are three rectangular spaces of increasing complexity: a low—(Figure 2a), medium—(Figure 2b), and a high—(Figure 2c) complexity room.



(**a**) (**b**) (**c**)

Figure 2. Low-complexity room (**a**), medium-complexity room (**b**), and high-complexity room (**c**).

The rooms shown in Figure 2a–c correspond to environments of four, twelve, and thirty-three obstacles, respectively. These compartments represent real spaces. The first one simulates a room with two beds, a column, and a piece of furniture attached to the wall. The next two were designed as an office, that is, a room with different compartments separated by weak walls. The spaces in Figure 2b,c were implemented by adapting two example maps taken from Matlab R2020b Update 8, the computational software package used for the simulation experiments.

3.1. Differential Robot Model

Furthermore, to obtain useful results close to reality, a simulation environment was created to replicate the real movement of the Hussar robot in the defined spaces. The construction of this environment required, on the one hand, the definition of the type of robot and its movement, as well as its modeling and parameters to control. On the other hand, the surveillance sensor and its range were modeled.

The kinematic of this robot is shown in Figure 3 (adapted from [31]).

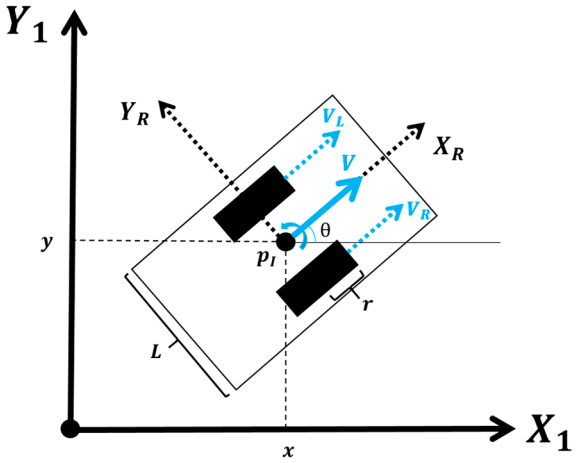


Figure 3. Kinematic model of the Hussar robot.

In this figure the position of the robot is represented by its Cartesian coordinates $p_I = \{x, y, \theta\}$ in the inertial frame $\{X_I, Y_I\}$. The transverse velocity Y_R is assumed to be 0 because there is no lateral slip.

In this model, the longitudinal velocity V and angular velocity W are the result of the combination of the linear velocities of each wheel, V_R and V_L (the subindexes correspond to right and left, respectively).

$$V = \frac{V_R + V_L}{2}, W = \frac{V_R - V_L}{L} \tag{1}$$

In addition, it must be considered that the left and right wheel speed are adjusted by a controller, to follow a longitudinal and angular speed profile. This can be easily modelled by Equations (2) and (3).

$$\begin{cases} V(t) = V_0 + \alpha_V(t - t_0) & V < V_{ref} \\ V(t) = V_{ref} & V = V_{ref} \end{cases} \tag{2}$$

$$\begin{cases} W(t) = W_0 + \alpha_W(t - t_0) & W < W_{ref} \\ W(t) = W_{ref} & W = W_{ref} \end{cases} \tag{3}$$

It is assumed that the speed (longitudinal and angular) follows a trapezoidal profile. The longitudinal acceleration is denoted by α_V and the angular acceleration is α_W. These values can be different for acceleration and deceleration. This way the speed increases or decreases linearly until it reaches the reference speed, V_{ref} for longitudinal and W_{ref} for angular speed.

The speed profiles defined by Equations (2) and (3) are shown in Figure 4 (red line, longitudinal speed, and blue line, angular speed). The black dashed line indicates the acceleration starting time.

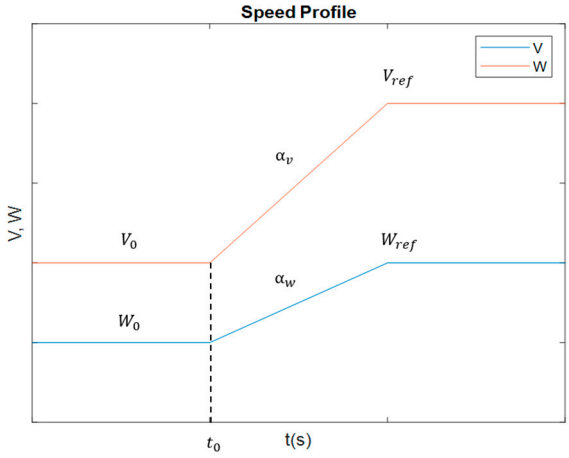


Figure 4. Speed profiles of the robot.

It should be noted that this way of inserting and modeling speed profiles is common among mobile robots, but any other profile could be used. To do this, it would be enough to modify the acceleration and maximum speed parameters (longitudinal and angular).

If p_I is considered an arbitrary position in the global inertia frame, the kinematic model can be given by Equation (4).

$$\dot{p}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{bmatrix} V \\ 0 \\ W \end{bmatrix} \tag{4}$$

The state vector is determined by $[x, y, \theta]$. All Cartesian coordinates are in meters and angles are in radians. Likewise, linear velocities are expressed in meters per second and angular velocities in radians per second.

3.2. Modeling and Simulation of the Detection System

The detection system, multiScan100 LiDAR sensing system [30], has been simulated by successively creating straight lines with a defined pitch angle starting from a certain point (location of the sensor robot). The possible intersection of the LiDAR with any obstacle is calculated by segmenting each colliding line when it intersects with it. In other works that used and modeled this sensor, the coverage did not consider the obstacles, so the radiation passed through them, losing precision in the simulation [31,32]. In this work, this issue has been solved.

This methodology is very suitable for calculating the visibility of the environment. Lines of sight whose coordinates match those of the spatial matrix cells are classified as "seen" and are, therefore, considered covered. Figure 5 illustrates the modeling of the sensor's operation with a scanning angle of 1 degree and a range of 4 m, encountering an obstacle that simulates a bed. The blue lines represent the space detected and, therefore, covered by the robot.

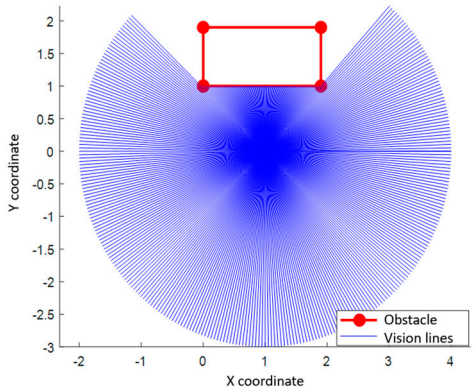


Figure 5. Space sensed by the LiDAR in the presence of obstacles.

Equations (5) and (6) illustrate the mathematical process followed in creating the lines of sight. It shows two different calculations that depend on whether there has been a line-obstacle intersection. If this occurs, the visible object truncates the range of the line. Otherwise, the laser visibility is extended to its maximum range. In that equation, $\{X_{FINAL}, Y_{FINAL}\}$ are the final coordinates of each line; $\{x_n,\ y_n\}$ refers to the position of the robot and, therefore, the starting point of the lines of sight; α refers to the set of sensor angles; r_{eff} denotes the sensor range in m (Equation (5)); and $\left\{x_n^{int}(\alpha), y_n^{int}(\alpha)\right\}$ refers to the closest intersection point with an obstacle. The minimum value of angle α is α_{min}, the maximum value is α_{max}, and the distance between two consecutive angles is the angular resolution, indicated as α_r.

$$r_{eff}(\alpha) = \begin{cases} r_{max} & if\ \nexists \text{ intersection} \\ \sqrt{\left(x_n - x_n^{int}(\alpha)\right)^2 + \left(x_n - y_n^{int}(\alpha)\right)^2} & if\ \exists \text{ intersection} \end{cases} \quad (5)$$

If there is an intersection with an obstacle, there is a reduction in effective sensor range r_{eff} (3). If there is no intersection, then the effective range is the maximum LiDAR range, i.e., r_{max}. The end point of each vision line is calculated as shown in Equation (6).

$$\{X_{FINAL}, Y_{FINAL}\} = \begin{cases} X_{FINAL} = x_n + r_{eff}(\alpha) \cdot \cos(\alpha) \\ Y_{FINAL} = y_n + r_{eff}(\alpha) \cdot \sin(\alpha) \end{cases} \quad (6)$$

Finally, given the initial point $P\{x_n,\ y_n\}$ and the set of endpoints of the straight lines $Q\{X_{FINAL}, Y_{FINAL}\}$, the set of points that belong to the segments PQ, that is, $\{X, Y\} \in PQ$, are used to mark the sensed area in the matrix V_M which represents the simulation environment by $V_M(X, Y) = 1$.

To classify the object viewed as an obstacle or a malicious object, a 360-degree LiDAR was used to identify its position. This is compared with the environment map created by the LiDAR built into the robot and if there is a mismatch between the two, it is detected as an unwanted object.

Table 1 presents the robot and sensor parameters used in this research. These parameters are those of the Hussar robot and LiDAR multiscan100. The last column specifies whether the parameter is related to the robot or the sensor. If this proposal is implemented with a different robot or sensor, the parameters in this table must be updated to the values of the new devices.

Table 1. Summary of the main parameters of the robot and sensor.

Parameter	Symbol	Value	Hardware
Linear speed	V_{ref}	0.5 $\frac{m}{s}$	Robot
Angular speed	W_{ref}	0.5 $\frac{rad}{s}$	Robot
Linear acceleration	α_v	0.5 $\frac{m}{s^2}$	Robot
Angular acceleration	α_w	0.5 $\frac{rad}{s^2}$	Robot
Sensor distance range	r_{max}	4 m	Sensor
Sensor angular resolution	α_r	1°	Sensor
Minimum sensing angle	α_{min}	360°	Sensor
Maximum sensing angle	α_{max}	360°	Sensor

4. Optimization Methodology

The objective of this study is to find the optimal trajectory in a defined space so that an automatic surveillance robot can work as efficiently as possible. For this purpose, a series of waypoints must be defined to establish a surveillance route. The robot works in the following way: it first recognizes a fixed set of points as coordinates and passes through them sequentially [32]. The robot follows the movement given by the state machine, shown in Figure 6, that has been coded. The right side of the figure shows the states reached by the machine along a trajectory.

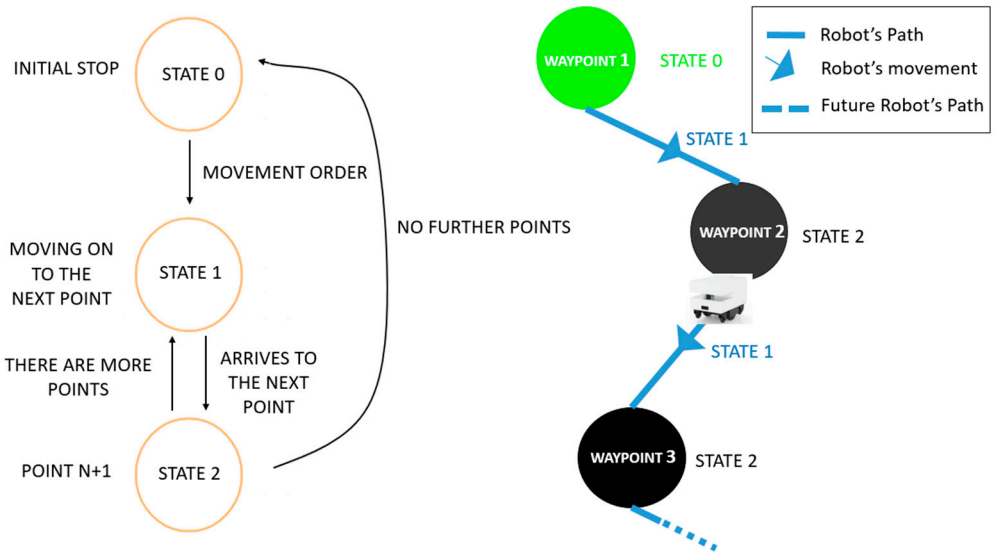


Figure 6. State diagram.

Each point consists of three reference parameters: coordinates $(x,\ y)$ and angle θ, that is, (x_r, y_r, θ_r), where the sub index r indicates that the values are reference points for the robot. In order not to make the program execution time too long, the angles have been fixed. These do not affect the trajectory since the sensing area is circular, so they can only influence the computational time. With the angles fixed, the variables that control the algorithms are reduced to the position coordinates (x_r, y_r).

4.1. Software Architecture

The implementation of the software architecture for the integration of the different optimization algorithms with the robot simulator and the LiDAR is shown in Figure 7.

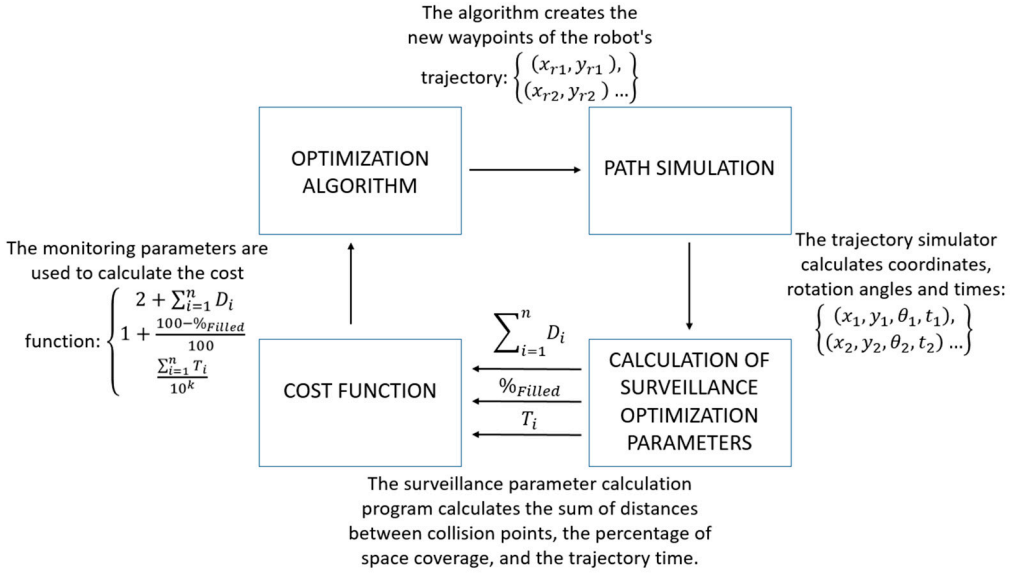


Figure 7. Software architecture.

The real robot has a program that supports XML files that contain the parameters of the points mentioned in the previous section, i.e., $\{x_r, y_r, \theta_r\}$. Thus, a computational code that converts these points from matrix format to this type of file was programmed.

Each time the optimization algorithms provide a new combination of parameters, a program is called that converts it into XML. This file is read by the Python version 3.11.3. trajectory simulation program, exactly as the real robot would do. The simulator then stores the robot's position at each instant of time, which is used by another Matlab version is R2020b Update 8 program that detects whether there has been a collision with an obstacle and calculates the distances between intersections, the percentage of vision (percentage that has been surveilled), and the trajectory time. Then, these parameters are used in the cost function. In fact, the cost function of the optimization algorithm differentiates three cases: (i) there is a collision with one or more obstacles; (ii) there is no collision, but coverage of the space is not completed; and (iii) there is no collision, and coverage is completed.

4.2. Metaheuristic Optimization Techniques Applied

The three optimization techniques that are applied and compared are the following.

– Genetic Algorithms: GAs emulate biological evolution to find optimal solutions in complex problems. They use selection, crossover, and mutation to refine a population of potential solutions iteratively. In surveillance trajectory design, they help generate and evolve efficient paths considering coverage, time, and distance limitations. By evaluating fitness and evolving generations, they converge towards improved strategies. GAs are versatile tools for optimization challenges [33].

GA utilized the following characteristic parameters: a crossover fraction of {0.8}, a population size of 200, a constraint tolerance of $\{1 \times 10^{-3}\}$, an elite count of 10, a function tolerance of $\{1 \times 10^{-4}\}$, a migration fraction of {0.2}, and a migration interval of {20}.

– Particle Swarm Optimization: PSO mimics swarm behavior, where particles adjust their positions and velocities to find optimal solutions in multi-dimensional spaces. It combines individual and collective knowledge to converge towards better solutions iteratively. PSO's dynamic balance between exploration and exploitation makes it effective for complex optimization problems. In surveillance trajectory refinement,

PSO iteratively adjusts trajectory parameters to enhance convergence towards optimal solutions, optimizing coverage, time, and constraints [34].

In this case, the following parameters were used: the minimum neighbor fraction {0.25}, the self-adjustment weight {1.49}, and the social adjustment weight {1.49}.

– Pattern Search: Leveraging PS for optimizing Hussar robot's trajectory involves systematic environment exploration to identify features like obstacles and open areas. As the two algorithms explained previously, it initiates with an initial path, iteratively adjusting it using diverse patterns aligned with identified features. Patterns leading to improvements are retained, while ineffective ones are discarded. Continuously refining the trajectory via PS helps the Hussar robot adapt to environmental nuances, ensuring effective area coverage. This dynamic approach tailors the trajectory to specific conditions, enhancing surveillance performance [35].

Finally, in the case of PS, the following parameters were used: a constraint tolerance of $\{1 \times 10^{-6}\}$, an initial mesh size of {1}, an initial penalty of {10}, a mesh contractor factor of {0.5}, a mesh expansion factor of {2}, a mesh tolerance of $\{1 \times 10^{-6}\}$, a penalty factor of {100}, and a step tolerance of $\{1 \times 10^{-6}\}$.

4.3. Optimization Problem

The optimization of the trajectory requires the definition of a cost function that the three algorithms must minimize. It differentiates the three different cases already mentioned: searching no shock; maximizing coverage without shock; and reducing time without shock and 100% coverage.

What the optimization algorithm should aim for is to find the trajectory without collisions and with total coverage of the surface, that is, case (iii), and in the minimum time.

Equation (7) shows the definition of the fitness function for the three cases, where the first row corresponds to the one in which intersections with obstacles occur (it will always be greater than 2). The second row represents the case of no collision, but the coverage is not 100% (values between 1 and 2). Finally, if there are no collisions and the entire space is covered, time is minimized (third row). The value of this last case is below 1 since the constant of the denominator (k) is calculated according to the time it takes for a standard route to be completed in the scenario that is evaluated.

$$f_c = \begin{cases} 2 + \sum_{i=1}^{n} D_i & if \sum_{i=1}^{n} D_i \neq 0 \\ 1 + \frac{100 - \%_{\text{Filled}}}{100} & if \sum_{i=1}^{n} D_i = 0 \;\&\; \%_{\text{Filled}} < 100 \\ \frac{\sum_{i=1}^{n_w} T_i}{10^k} & if \sum_{i=1}^{n} D_i = 0 \;\&\; \%_{\text{Filled}} = 100 \end{cases} \quad (7)$$

In Equation (7), n is the number of obstacles and D_i is the distance that the trajectory of the vehicle crosses the obstacle i.

The percentage of visualization of the environment $\%_{\text{Filled}}$ is defined by Equation (8), considering that Nx is the number of cells in the x-axis and Ny is the number of cells in the y-axis of the map. It is noteworthy to remember that V_M denotes the matrix with the cells of the map which represents the simulation environment, and a value one in a cell means that the cell has been sensed by the sensor.

$$\%_{\text{Filled}} = \frac{V_M(i,j) = 1}{Nx \times Ny} \qquad (i,j)\epsilon(\mathbb{N} < Nx, \mathbb{N} < Ny) \quad (8)$$

The time it takes the robot to get from one point to its next waypoint is T_i, and n_w is the number of waypoints; thus, $\sum_{i=1}^{n_w} T_i$ is the total time it takes to complete the trajectory. Finally, k is a constant that defines the denominator of the third case of the cost function as a multiple of 10. In this way, k ensures that the cost function in this case is always less than one.

In addition, a number of constraints are imposed on the values of the parameters to be optimized. These parameters correspond to the Cartesian coordinates $\{x,\ y\}$ in the $\{X,\ Y\}$ plane, with values belonging to the intervals $\{x_{\min},\ x_{\max}\}$ and $\{y_{\min},\ y_{\max}\}$, respectively.

It should be noted that the speed of the robot has been set to 0.5 m/s in the trajectory simulation algorithm, as this is the maximum speed of the real mobile robot that will perform the surveillance task. In this robot the command to reach a new point receives $\{x_r, y_r, \theta_r\}$. However, in order to accelerate the optimization process, the angle θ_r is set to 0. This does not affect the surveillance as the sensing range of the sensor is 360°. If the sensing pattern was different than 360°, it would be recommended to include the angle as a variable to be optimized.

5. Simulation Results

For each of the three scenarios (rooms), the three optimization algorithms have been implemented with Matlab software and their efficiency has been compared using the following metrics:

1. Non coll. Time: Time it takes the algorithm to make the robot avoid obstacles.
2. Non coll. Iterations: Number of iterations it takes the algorithm to make the robot avoid obstacles.
3. Time 100%: Minimum time it takes to reach 100% coverage.
4. Iterations 100%: Number of iterations it takes to reach 100% coverage.
5. Best Dist.: Best trajectory distance (lowest).
6. Best %: Best percentage of coverage (maximum).

The first two metrics are related to the obstacle avoidance objective (first part of the cost function). The next two metrics evaluate the time it takes the algorithm to reach full coverage of the space. In other words, they are used to compare the speed of the different methods. Finally, the best result in terms of distance, and the best percentage of coverage are obtained. The latter refers to the possible case in which a complete coverage could not be achieved, either because the space is too large, or because of the lack of waypoints in the trajectory. Furthermore, the evolution of the cost function for each scenario and optimization algorithm is also presented.

In addition, the GA, PSO, and PS algorithms need an initial path on which to start the calculation. The criterion followed to implement it is that two starting restrictions are determined: a waypoint cannot be separated from the immediately preceding one by a distance greater than the sensor range, and the direction when traveling the path must be clockwise.

Finally, another restriction has been imposed on the number of waypoints for each trajectory. The number of crossing points for each scenario is as follows: eight for the small room, twelve for the medium room, and thirty-two for the large and more complex room.

Tables 2–4 show the metrics obtained for the different scenarios. The computer model used to obtain all these results has been a Dell Vostro 5471. It has 8 GB of RAM and an i7 8th generation processor. The version of the simulation tool was Matlab 2020b.

Table 2. Low-complexity room results.

Technique	Non Coll. Time	Non Coll. Iterations	Time 100%	Iterations 100%	Best Dist.	Best %
GA	0:05:48	115	0:05:48	115	8460	100%
SP	0:03:25	21	0:03:25	21	8350	100%
PS	0:03:03	6	0:03:03	6	6220	100%

Table 3. Medium-complexity room results.

Technique	Non Coll. Time	Non Coll. Iterations	Time 100%	Iterations 100%	Best Dist.	Best %
GA	0:29:49	737	3:13:27	1423	36,380	100.00%
SP	0:13:55	351	0:13:55	351	2943	100.00%
PS	0:06:50	41	0:06:50	41	2254	100.00%

Table 4. High-complexity room results.

Technique	Non Coll. Time	Non Coll. Iterations	Time 100%	Iterations 100%	Best Dist.	Best %
GA	2:51:35	2540	NC	NC	22,102	84.00%
SP	1:36:01	1163	NC	NC	22,158	86.70%
PS	1:26:15	950	NC	NC	21,982	85.00%

5.1. Results in Scenario (a): Low-Complexity Room

Table 2 shows that for the low-complexity room, the PS algorithm is the best in all the metrics. It presents a considerably lower number of iterations to reach 100% coverage, less computation time, and has achieved the path with the shortest distance. Figure 8 shows that it also obtains the lowest value of the cost function, i.e., the shortest trajectory travel time.

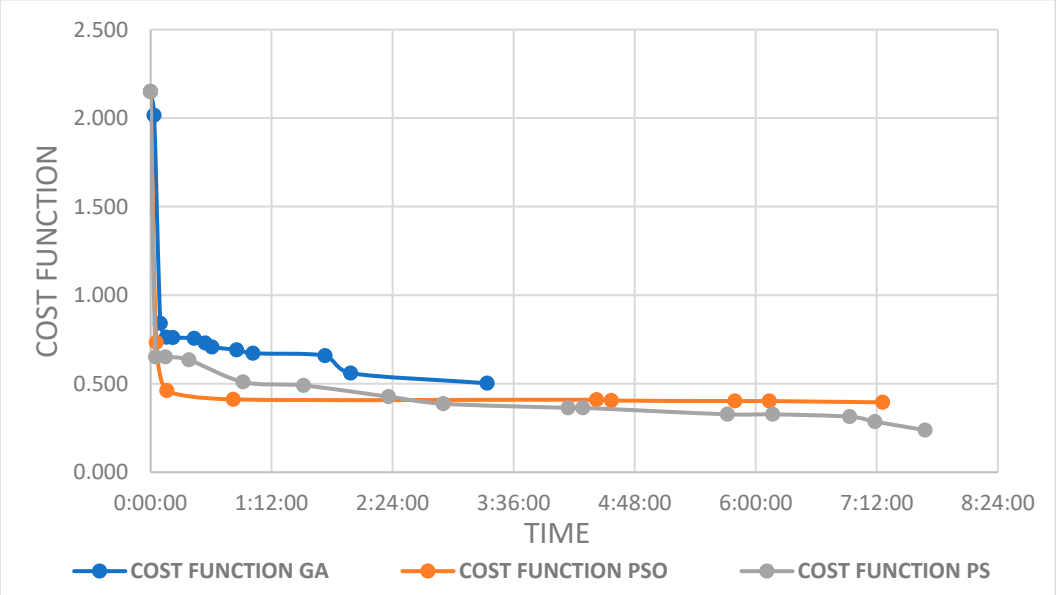


Figure 8. Low-complexity room. Cost function evolution.

Experimenting in a low-complexity environment provides valuable information about how optimization algorithms behave when the solution to be found is not far from the initial one. To show this, it has been considered interesting to see how the cost function varies with respect to time (Figure 8). In Figure 8 it is possible to see how the Pattern Search algorithm is the best option for environments with these characteristics. This result is in line with what was expected, since the methodology of this algorithm is based on finding solutions close to the initial one and, in a small space, this characteristic is very useful. In contrast, the GA and PSO search methods are adequate but not optimal.

In this case, it is clear that addressing a small environment, the algorithm has quickly converged to a plausible solution. It can be also seen that the initial trajectory chosen for this case gives a value above the non-collision threshold and hence, in a relatively small time it is able to start optimizing the travel time.

The best trajectories after eight hours of computation are shown in Figure 9 for each technique, GA (a), PSO (b), and PS (c). The initial trajectory is shown in blue and the optimized one in red.

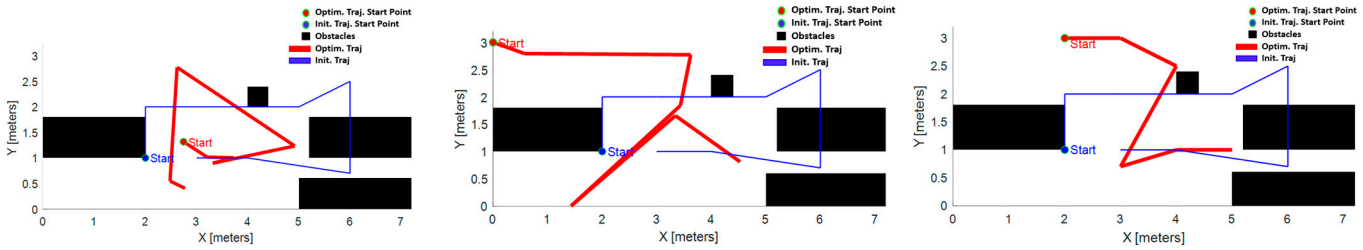


Figure 9. Trajectories in the low-complexity room: Genetic Algorithm trajectory (**a**), Particle Swarm trajectory (**b**) and Pattern Search trajectory (**c**).

As expected in view of the quantitative results, the PS algorithm presents the best trajectory, without abrupt changes or any repeated displacements that worsen the efficiency. On the contrary, the solutions offered by the other two optimization algorithms present some of these inefficiencies. This result is reflected in the final trajectory time, which is considerably lower with PS than with GA or PSO.

In summary, the qualities of the solutions generated are strongly linked to the solution search method applied. As the results reveal, particularly in the context of a low complexity environment, a local search algorithm like PS emerges as the most favorable option. This algorithm excels in situations where fine-tuning within a limited search space is critical to achieving optimal results.

To show the validity of the proposal in a real-world environment, where LiDAR may be affected by noise in point cloud data, and the presence of moving objects could influence the robot's LiDAR mapping, the low-complexity environment was recreated in the laboratory. Figure 10a shows the real environment without the presence of external obstacles, and Figure 10b shows the mapping of the environment. Figure 10b shows a clear distinction between the points that the LiDAR detects in real time (orange) and those that it has stored in memory (black). The robot is represented by the green arrow.

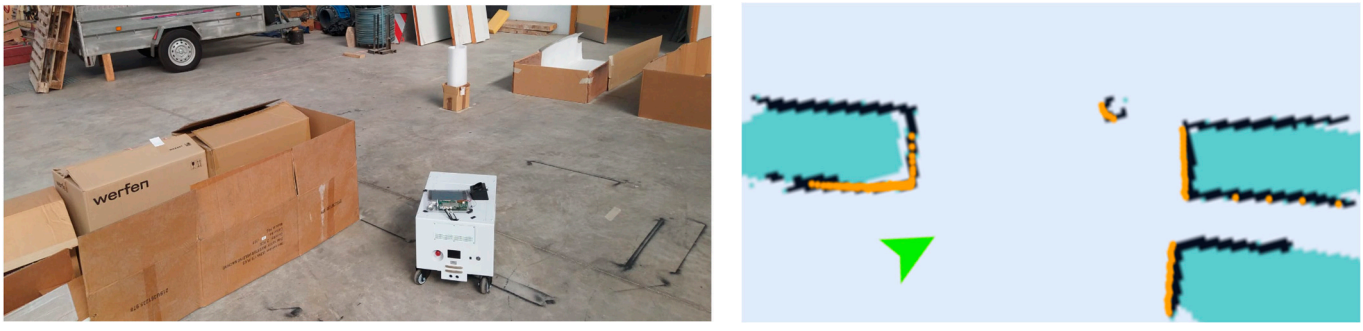
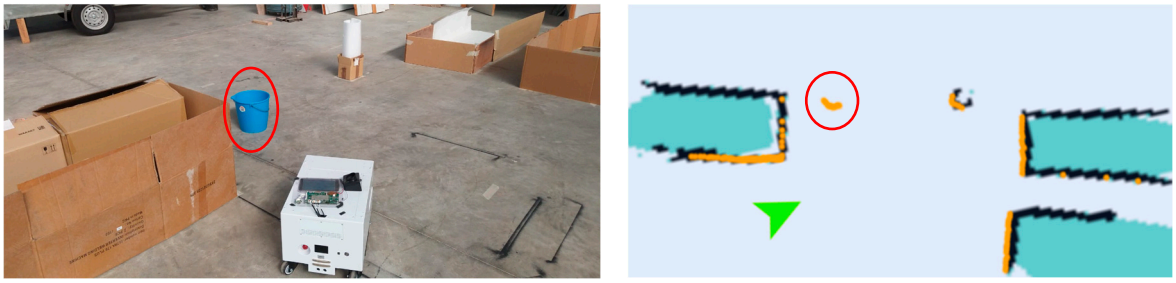


Figure 10. Experiments in the low-complexity room. Real environment (**a**), and mapped environment (**b**).

After calibrating the environment, an obstacle (the blue bucket circled in red) was introduced and the sensor's reaction was observed. Figure 11a shows the object introduced into the real environment and Figure 11b shows how the laser sees it. As expected, the bucket points are shown in orange. This object is not identified as part of the map and, therefore, appears as a series of orange dots (current view) with no correspondence to the black dots (mapped view). The points have been marked with a red circle to facilitate identification.



(**a**) (**b**)

Figure 11. Experiments in the low-complexity room with an object. Real environment (**a**), and mapped environment (**b**).

5.2. Results in Scenario (b): Medium-Complexity Room

In the case of a space of medium complexity, the Pattern Search algorithm has been the most efficient in terms of optimization time, iterations, and final trajectory, as in the previous case. The GA has turned out to be the slowest. This observation is a direct extension of the explanation described above. In the context of a moderately complex environment, the preference for a local search algorithm remains the best option, reflecting the need for detailed adjustments within this specific environment. However, it is worth noting that the performance disparities between GA, PSO, and PS are not as pronounced as in the low-complexity scenario.

The results in this medium-complexity environment suggest that as the complexity and dimensions of the space increase, global search algorithms begin to show more favorable final results. Furthermore, it can be noted that in this case, as well as in the other environments studied in this article, the GA exhibits a relatively longer convergence time and a greater propensity to require more iterations to find solutions.

The graph shown in Figure 12 matches the data in Table 3. PS is the fastest to find a good solution, while the GA is the slowest. In general terms, after eight hours of simulation, the best results are also offered by PS.

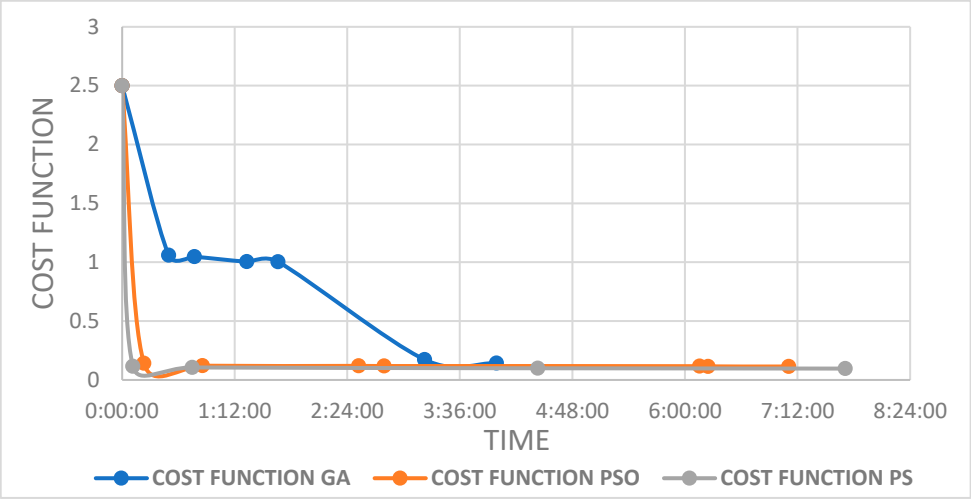
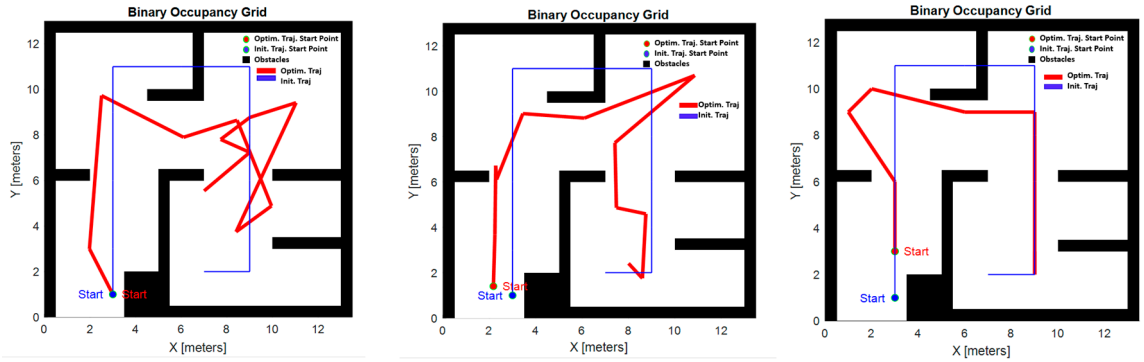


Figure 12. Medium-complexity room. Cost function evolution.

Figure 13a–c show that the best trajectory is the one obtained with PS, that is simpler and more straightforward, in contrast to the GA and PSO, where the robot makes many inefficient displacements. In particular, the final trajectory of the GA is the worst.



(**a**) (**b**) (**c**)

Figure 13. Trajectories in the medium-complexity room: Genetic Algorithm trajectory (**a**), Particle Swarm trajectory (**b**), and Pattern Search trajectory (**c**).

5.3. Results in Scenario (c): High-Complexity Room

According to Table 4, the PS algorithm is able to find a good solution faster and with fewer iterations than the others. However, in this more complex case it is the PSO algorithm that obtains the best solution in eight hours. It gives 86.7% of covered space, while Pattern Search and the GA reach 85% and 84%, respectively.

With a significantly greater complexity and size of the environment than in the previous two cases, the results change. It is no longer the local search (PS) algorithm that offers the best results but rather PSO gives the route with the greatest coverage. It should be noted that, given the size of the space, the lack of reference points, and the limited range of the radar, in this experiment the quality of the optimization was evaluated based on the percentage of the trajectory traveled.

Looking at Table 4, PS was the fastest algorithm to find a solution without collisions with obstacles. This can be attributed to the good choice of the initial trajectory. Figure 13c shows the initial trajectory (blue) and a small number of collisions can be observed. On the other hand, if the initial solution had been very far from a possible optimized solution, PS would have been the worst option. It may not even converge to a feasible trajectory.

Finally, the GA is the one that has given the worst results in all three cases. In summary, the best option for a simple case is PS and for a complex case PSO. This final conclusion makes it clear that a hybrid PSO-PS algorithm can be a very good option for this problem. It could converge to an optimal solution in a complex environment, and once the possible outcome is found, the local feature of PS will refine it.

This room is significantly larger than the others and thus, the algorithms take much longer to converge to a good solution. The PS algorithm is the fastest.

On the other hand, the GA is the one that takes longer to find an optimized solution. It should be recalled that this has also been the case in the two previous cases. This is shown in Figure 14.

Figure 15a–c show the different trajectories offered by each algorithm after eight hours of computation. In the case of the complex room, the PSO gives the most optimized trajectory, but the difference between the values of the cost functions is very small. As in previous cases, those that present the greatest double rerun distance are the ones that give the most inefficient results. In cases like this, with a large space, the optimization is slower and in eight hours of computation it is not possible to reach the most optimal solution. Even so, it is interesting to get a glimpse of how the algorithms behave in this type of environment. It should be noted that, although in this case the best behavior is that of PSO, all three methods provided good trajectories.

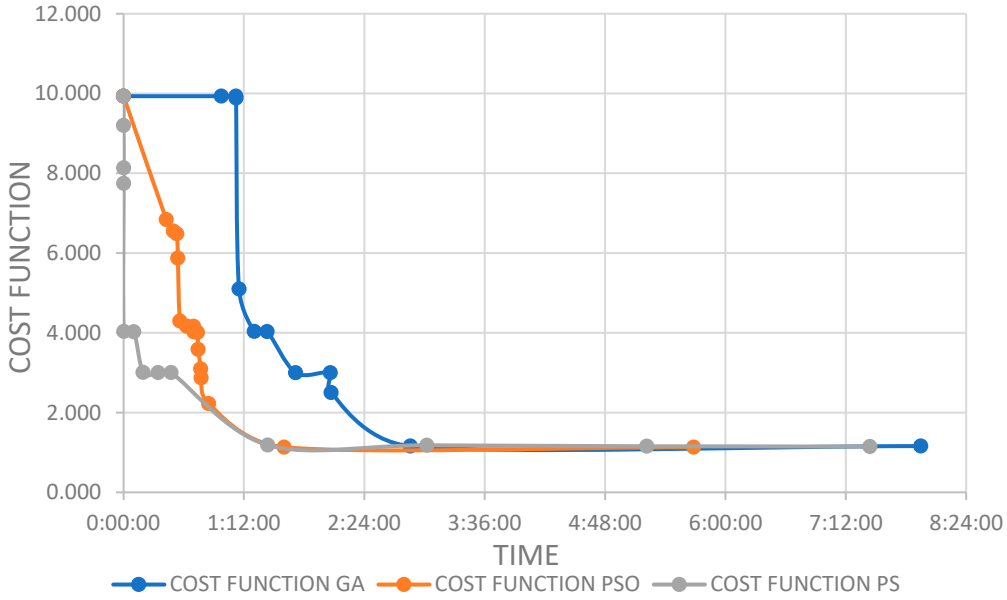


Figure 14. High-complexity room. Cost function evolution.

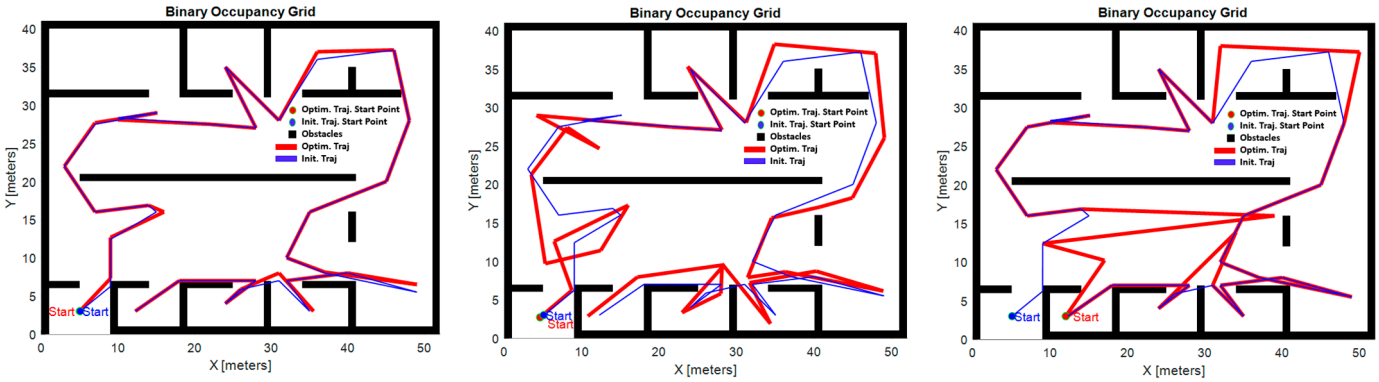


Figure 15. Trajectories in the high-complexity room: Genetic Algorithm trajectory (**a**), Particle Swarm trajectory (**b**), and Pattern Search trajectory (**c**).

After eight hours of calculation, a possible trajectory was obtained by each algorithm. None of them achieved total coverage because in such a large space, they would need more points (>32), more time, or a greater sensor range. However, given the impossibility of total coverage, the algorithms obtained high coverage. The GA achieved 84%, the PSO method covered 86.7% of the space and with PS we reached 85%. Figure 16 shows the best solutions found for each algorithm.

As previously mentioned, the PSO algorithm has obtained the best trajectory. This can be seen in Figure 16a–c. If we compare them, we can see that in Figure 16b there are fewer dark blue areas than in Figure 16a,c, which represent uncovered space.

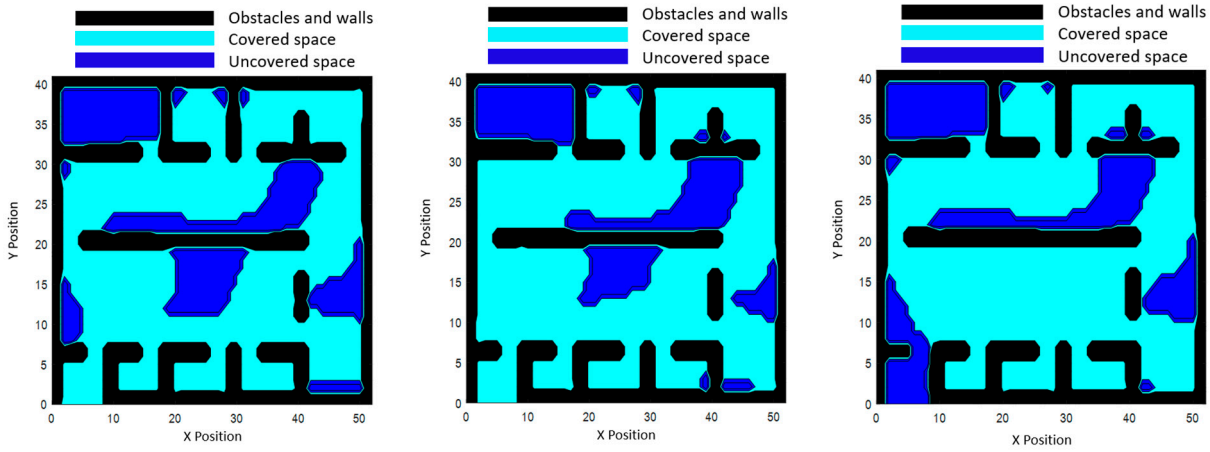


Figure 16. High-complexity room coverage: Genetic Algorithm coverage (**a**), Particle Swarm coverage (**b**), and Pattern Search coverage (**c**).

5.4. Optimization Methods Comparison

To conclude this section of results, it was considered interesting to compare them with those collected in previous works. Specifically, the research carried out by Fetanat et al. was selected as a benchmark [36] because these authors also used GA, PSO, and PS.

Unlike our study that seeks to cover a specific space, these authors addressed the problem of finding the shortest path to reach a destination point, but they evaluated the same algorithms and achieved results consistent with ours. Their case study covered a 100x100 grid, similar in number of cells to our high-complexity case. As in our study, they observed that PSO effectively minimized the cost function, followed by PS, and finally GA. Furthermore, it was observed that PS was the fastest method to find a satisfactory solution, followed by GA and PSO. The difference with our study lies in the fact that GA managed to find a viable solution faster than PSO. However, this discrepancy can be attributed to differences between the two problems.

To make a fairer comparison, the cost function used in that article was applied to our problem. Since the objective of the two studies is not the same, it was necessary to modify it slightly. The sum responsible for minimizing sudden changes in direction was replaced by the percentage of vigilance (Equation (9)). This cost function was applied to our problem to analyze which of the two is more effective.

$$f_c = \begin{cases} \alpha \sum_{i=1}^{n-1} L_i + \beta(100 - \%_{\text{Filled}}) + \gamma f_{\text{semi}} & if \sum_{i=1}^{n} D_i \neq 0 \\ \alpha \sum_{i=1}^{n-1} D_i + \beta(100 - \%_{\text{Filled}}) & if \sum_{i=1}^{n} D_i = 0 \end{cases} \quad (9)$$

where L_i is the length of the i-segment, f_{semi} is a penalty factor (1000) in case of collision with an obstacle, and α, β, and γ are weighting coefficients that establish the minimization importance of each term. In this case, all of them take the value of 1. As the function considers the summatory of L_i, the total travelling distance is minimized.

Table 5 compares the results provided by the fitness function of Equation (9) with the one proposed in this work (7). The results have been obtained in the low-complexity room for two hours of simulation.

Table 5. Comparison of results obtained with the fitness function proposed in this work and adapted from [36]. Best results have been boldfaced.

	Fitness Function Adapted from [36]			Fitness Function of This Work		
Technique	**Trajectory Time (s)**	**Best Dist.**	**Best %**	**Trajectory Time (s)**	**Best Dist.**	**Best %**
GA	85.04	21.82	100%	**56**	**10.59**	100%
SP	80.19	16.21	100%	**41.1**	**10.28**	100%
PS	51.97	**7.94**	100%	**49**	14.41	100%

The comparison shows that our method is more efficient in terms of travel time reduction. In the case of PS, it can be seen that the total distance is greater than in the other two cases. This is because the path length has not been minimized in this study. The other two algorithms, GA and PSO, found solutions with more changes in direction, while PS proposed straighter trajectories.

The fitness function adapted from [36], on the other hand, minimized the distance, but still only in the case of PS a longer trajectory was found. An interesting point here is that GA and PSO obtain better lengths even with a fitness function that does not minimize this parameter directly (7). The total trajectory time, a parameter that minimizes the proposed function, is lower in all cases.

6. Conclusions and Future Works

In this paper we studied the optimization of surveillance trajectories in different scenarios for a differential mobile robot using three heuristic algorithms: GA, PSO, and PS. To do so, the calculation and simulation of the robot's model and movements were simulated using Matlab and Python. Each optimization technique was tested for a given time (eight hours). During this time, it was verified that the algorithms were able to avoid obstacles in the different scenarios, cover the largest possible percentage of the space, and, finally, optimize the trajectory in terms of time. These processes were carried out with Reeman's Hussar robot using the LiDAR multiScan100. However, the study presented in this work can be used for any type of robot with any type of surveillance system (vision, gas detector, etc.).

Having analyzed the results, it can be concluded that all three algorithms give excellent results, but PS stands out from the other two as it was able to find solutions faster and, except in the case of the more complex room, offer better final solutions.

For future lines of research, we intend to investigate the hybridization between PSO and PS. This method works as follows: PSO approaches a good solution and when it has done so, PS refines the results. In addition, in the future we will test other metaheuristic techniques and intelligent techniques such as reinforcement learning.

Author Contributions: M.P.: conceptualization, methodology, software, validation, writing—original draft preparation, writing—review and editing. J.E.S.-G.: conceptualization, methodology, formal analysis, supervision, software, writing—original draft preparation, writing—review and editing. M.S.: conceptualization, validation, writing—review and editing, supervision. I.M.: conceptualization, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the European Commission, under European Project CoLLaboratE, grant number 820767.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Juan, V.S.; Santos, M.; Andújar, J.M. Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity* **2018**, *2018*, 6879419. [CrossRef]
2. Tiseni, L.; Chiaradia, D.; Gabardi, M.; Solazzi, M.; Leonardis, D.; Frisoli, A. UV-C mobile robots with optimized path planning: Algorithm design and on-field measurements to improve surface disinfection against SARS-CoV-2. *IEEE Robot. Autom. Mag.* **2021**, *28*, 59–70. [CrossRef]
3. Perminov, S.; Mikhailovskiy, N.; Sedunin, A.; Okunevich, I.; Kalinov, I.; Kurenkov, M.; Tsetserukou, D. Ultrabot: Autonomous mobile robot for indoor uv-c disinfection. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; IEEE: New York, NY, USA, 2021; pp. 2147–2152.
4. Chun, W.H.; Papanikolopoulos, N. Robot surveillance and security. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1605–1626.
5. Guevara, C.; Santos, M. Surveillance routing of COVID-19 infection spread using an intelligent infectious diseases algorithm. *IEEE Access* **2020**, *8*, 201925–201936. [CrossRef] [PubMed]
6. Kim, S.D.; Bae, C.O. Unmanned Engine Room Surveillance Using an Autonomous Mobile Robot. *J. Mar. Sci. Eng.* **2023**, *11*, 634. [CrossRef]
7. Zhang, J.; Yang, X.; Wang, W.; Guan, J.; Ding, L.; Lee, V.C. Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering. *Autom. Constr.* **2023**, *146*, 104699. [CrossRef]
8. Stolfi, D.H.; Brust, M.R.; Danoy, G.; Bouvry, P. UAV-UGV-UMV multi-swarms for cooperative surveillance. *Front. Robot. AI* **2021**, *8*, 616950. [CrossRef] [PubMed]
9. Stolfi, D.H.; Brust, M.R.; Danoy, G.; Bouvry, P. A competitive Predator–Prey approach to enhance surveillance by UAV swarms. *Appl. Soft Comput.* **2021**, *111*, 107701. [CrossRef]
10. Sun, Y.; Fesenko, H.; Kharchenko, V.; Zhong, L.; Kliushnikov, I.; Illiashenko, O.; Morozova, O.; Sachenko, A. UAV and IoT-based systems for the monitoring of industrial facilities using digital twins: Methodology, reliability models, and application. *Sensors* **2022**, *22*, 6444. [CrossRef] [PubMed]
11. Prykhodchenko, R.; Rocha, R.P.; Couceiro, M.S. People detection by mobile robots doing automatic guard patrols. In Proceedings of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Ponta Delgada, Portugal, 15–17 April 2020; IEEE: New York, NY, USA, 2020; pp. 300–305.
12. Lee, M.F.R.; Shih, Z.S. Autonomous Surveillance for an Indoor Security Robot. *Processes* **2022**, *10*, 2175. [CrossRef]
13. Arjun, D.; Indukala, P.K.; Menon, K.U. Border surveillance and intruder detection using wireless sensor networks: A brief survey. In Proceedings of the 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 6–8 April 2017; IEEE: New York, NY, USA, 2017; pp. 1125–1130.
14. Tharmalingam, K.; Secco, E.L. A Surveillance Mobile Robot Based on Low-Cost Embedded Computers. In Proceedings of the 3rd International Conference on Artificial Intelligence: Advances and Applications: ICAIAA 2022, Jaipur, India, 23–24 April 2022; Springer Nature: Singapore, 2023; pp. 323–334.
15. Guerrero-Higueras, Á.M.; Álvarez-Aparicio, C.; Calvo Olivera, M.C.; Rodríguez-Lera, F.J.; Fernández-Llamas, C.; Rico, F.M.; Matellán, V. Tracking people in a mobile robot from 2d lidar scans using full convolutional neural networks for security in cluttered environments. *Front. Neurorobotics* **2019**, *12*, 85. [CrossRef] [PubMed]
16. Yang, T.; Li, Y.; Zhao, C.; Yao, D.; Chen, G.; Sun, L.; Krajnik, T.; Yan, Z. 3D ToF LiDAR in mobile robotics: A review. *arXiv Preprint* **2022**, arXiv:2202.11025.
17. Uhm, T.; Park, J.; Lee, J.; Bae, G.; Ki, G.; Choi, Y. Design of multimodal sensor module for outdoor robot surveillance system. *Electronics* **2022**, *11*, 2214. [CrossRef]
18. Chaudhuri, R.; Acharjee, J.; Deb, S. Avenues of Graph Theoretic Approach of Analysing the LIDAR Data for Point-To-Point Floor Exploration by Indoor AGV. In *Machine Learning, Image Processing, Network Security and Data Sciences, Proceedings of the 3rd International Conference on MIND 2021, Online, 11–12 December 2021*; Springer Nature: Singapore, 2023; pp. 827–838.
19. Poduval, D.R.; Rajalakshmy, P. A review paper on autonomous mobile robots. In Proceedings of the AIP Conference Proceedings, 5 December 2022; AIP Publishing: Long Island, NY, USA, 2022; Volume 2670.
20. Ayvali, E.; Salman, H.; Choset, H. Ergodic coverage in constrained environments using stochastic trajectory optimization. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: New York, NY, USA, 2017; pp. 5204–5210.
21. Mitchell, M. *Genetic Algorithms: An Overview*; Complex: New York, NY, USA, 1995; Volume 1, pp. 31–39.
22. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization: An overview. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
23. Güneş, F.; Tokan, F. Pattern search optimization with applications on synthesis of linear antenna arrays. *Expert Syst. Appl.* **2010**, *37*, 4698–4705. [CrossRef]
24. Lin, S.; Liu, A.; Wang, J.; Kong, X. An intelligence-based hybrid PSO-SA for mobile robot path planning in warehouse. *J. Comput. Sci.* **2023**, *67*, 101938. [CrossRef]
25. Sahoo, S.K.; Choudhury, B.B. A review of methodologies for path planning and optimization of mobile robots. *J. Process Manag. New Technol.* **2023**, *11*, 122–140. [CrossRef]
26. Xiao, J.; Yu, X.; Sun, K.; Zhou, Z.; Zhou, G. Multiobjective path optimization of an indoor AGV based on an improved ACO-DWA. *Math. Biosci. Eng.* **2022**, *19*, 12532–12557. [CrossRef] [PubMed]

27. Abdulsaheb, J.A.; Kadhim, D.J. Classical and heuristic approaches for mobile robot path planning: A survey. *Robotics* **2023**, *12*, 93. [CrossRef]
28. Reeman Robotics. Hussar Autonomous Robot. 2023. Available online: http://www.reemanrobot.com/robot-chassis/square-robot-chassis/automatic-square-robot-chassis.html (accessed on 27 September 2023).
29. Beltrán, J.; Guindel, C.; Moreno, F.M.; Cruzado, D.; Garcia, F.; De La Escalera, A. Birdnet: A 3d object detection framework from lidar information. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; IEEE: New York, NY, USA, 2018; pp. 3517–3523.
30. SICK. Multiscan100—Sensores 3D LiDAR. SICK, 2023. Available online: https://www.sick.com/es/es/sensores-lidar/sensores-3d-lidar/multiscan100/c/g574914 (accessed on 27 September 2023).
31. Rodrigo, D.V.; Sierra García, J.E.; Santos, M. *Glasius Bioinspired Neural Networks Based UV-C Disinfection Path Planning Improved by Preventive Deadlock Processing Algorithm*; Elsevier: Amsterdam, The Netherlands, 2023.
32. Peñacoba, M.; Sierra, J.E. Optimization of UV-C disinfection robot trajectories by genetic algorithm. In Proceedings of the Actas de XVIII Simposio de Control Inteligente (SCI2023), Valencia, Spain, 28–30 June 2023.
33. Kumar, M.; Husain, D.M.; Upreti, N.; Gupta, D. Genetic algorithm: Review and application. *Int. J. Inf. Technol. Knowl. Manag.* **2010**, *2*, 451–454. [CrossRef]
34. Bai, Q. Analysis of particle swarm optimization algorithm. *Comput. Inf. Sci.* **2010**, *3*, 180. [CrossRef]
35. Wetter, M.; Wright, J. Comparison of a generalized pattern search and a genetic algorithm optimization method. In Proceedings of the 8th IBPSA Conference, Eindhoven, The Netherlands, 11–14 August 2003; Volume 3, pp. 1401–1408.
36. Fetanat, M.; Haghzad, S.; Shouraki, S.B. Optimization of dynamic mobile robot path planning based on evolutionary methods. In Proceedings of the 2015 AI & Robotics (IRANOPEN), Qazvin, Iran, 12 April 2015; IEEE: New York, NY, USA, 2015; pp. 1–7.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.