QUBAHAN ACADEMIC JOURNAL
VOL. 6, NO. 1, March 2026
https://doi.org/10.48161/qaj.v6n1a2235

An Extensible Intelligent Simulation System for Robot Path Planning in Wireless Sensor Networks

Abdulaziz Shehab [1, 2*], **Abdelhady Naguib** [3, 4], **A. S. Abohamama** [5, 6] **and Ahmed Elashry** [7, 8]

[1] Department of Information Systems, College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia;
[2] Department of Information Systems, Mansoura University, Mansoura 35516, Egypt;
[3] Department of Computer Science, College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia;
[4] Department of Systems and Computers Engineering, Faculty of Engineering, Al-Azhar University, Cairo 11884, Egypt;
[5] Department of Computer Science, Mansoura University, Mansoura 35516, Egypt;
[6] Department of Computer Science, Arab East Colleges, Riyadh 53354, Saudi Arabia;
[7] Faculty of Computer Studies, Arab Open University, Riyadh 13311, Saudi Arabia;
[8] Information Systems Department, computers and information Faculty, Kafr El-Sheikh University, Kafr El-Sheikh 33516, Egypt.

* **Corresponding author:** abdulaziz_shehab@mans.edu.eg.

ABSTRACT: Optimizing robot trajectories in dynamic and complex environments remains a significant challenge, particularly within wireless sensor networks (WSNs), where many existing simulation tools rely on predefined and inflexible scenarios. This paper presents an extensible graphical user interface (GUI)–based simulation system that integrates robot animation and scenario generation within a unified framework. The proposed system adopts a modular, plugin-based architecture that enables the seamless incorporation of diverse path-planning and trajectory optimization models without modifying the core infrastructure. In addition, an intelligent path-planning approach based on a Genetic Algorithm enhanced with Bézier curve modeling is integrated to support smooth and collision-free robot navigation in dynamic environments. Experimental evaluation demonstrates that the proposed framework provides an effective, scalable, and adaptable platform for conducting reproducible robot trajectory simulations in both research-oriented and practical applications. Overall, the proposed GUI-based and modular simulation framework provides a flexible, extensible, and reproducible platform for evaluating intelligent robot trajectory planning in dynamic WSN environments.

Keywords: Bézier curves, genetic algorithm (GA), graphical user interface (GUI), robot path planning, trajectory simulation, scenario generation, WSN localization.

I. INTRODUCTION

1. BACKGROUND

Wireless Sensor Networks (WSNs) have become a cornerstone technology for smart and ambient intelligence environments, enabling a wide range of applications such as environmental monitoring, healthcare, intelligent transportation, and industrial automation. Due to their typically random deployment, many sensor nodes in WSNs lack prior position information, making node localization a critical challenge.

Accurate localization is essential for efficient routing, resource management, and context-aware services [1, 2]. Existing localization techniques are generally classified into range-free and range-based approaches. While range-free methods rely on network connectivity and topology, range-based methods exploit physical signal measurements such as RSSI, ToA, or TDoA. Although range-based techniques usually achieve higher accuracy, they often require additional hardware or incur increased energy consumption. To address this trade-off, recent studies have proposed hybrid localization schemes that balance accuracy and efficiency, particularly for IoT-enabled environments [3, 4]. In parallel, robot path planning has gained increasing attention in WSN-based applications. Path planning aims to generate a feasible and collision-free trajectory that guides a mobile robot from a starting point to a target location while accounting for obstacles and environmental dynamics. Traditional path-planning approaches include heuristic search and pre-computation algorithms; however, modern applications operating in dynamic and complex environments require more adaptive and intelligent strategies. Recent surveys indicate that deep reinforcement learning (DRL) and metaheuristic optimization methods, such as genetic algorithms, have become dominant techniques for addressing these challenges [5, 6]. DRL-based approaches have demonstrated strong adaptability for real-time navigation in uncertain environments [7, 8], whereas genetic and evolutionary algorithms remain widely used due to their ability to generate smooth, short, and energy-efficient trajectories in both static and dynamic scenarios [9, 10]. Consequently, the integration of intelligent path planning with localization has become a key requirement for robust and scalable WSN-based systems [11].

In recent years, the role of digital innovation and interactive software platforms has become increasingly important in supporting experimentation, reproducibility, and system prototyping in intelligent and autonomous systems research. Modern simulation environments are no longer limited to algorithmic validation, but are increasingly designed as flexible digital tools that integrate visualization, configurability, and extensibility to support research-driven experimentation. Recent studies on digital innovation and cloud-enabled technologies highlight the importance of adaptable and scalable platforms that facilitate systematic experimentation and knowledge transfer across complex systems. In this context, the proposed GUI-based simulation system is aligned with contemporary digital innovation trends by providing an extensible and interactive environment for robotic trajectory planning and WSN-based experimentation [42].

2. MOTIVATION

Despite the significant progress in localization and path-planning algorithms, many existing simulation tools rely on rigid and predefined scenario descriptions, which limit adaptability and extensibility. Modifying trajectory models or integrating new planning strategies often requires substantial changes to the simulation core, hindering reproducibility and systematic comparison. Furthermore, many available platforms lack intuitive graphical interfaces that integrate scenario generation, trajectory visualization, and performance analysis within a unified environment. These limitations restrict the effective evaluation of emerging intelligent and optimization-based path-planning techniques, particularly in dynamic and complex WSN environments.

3. PROBLEM STATEMENT

From a formal perspective, the path-planning problem involves generating a feasible trajectory that enables a mobile robot to move from an initial position to a target destination while avoiding obstacles within a predefined environment. Path-planning strategies are commonly classified as static, where obstacle positions remain fixed, or dynamic, where obstacles may move, appear, or disappear, requiring continuous map updates. In many models, the robot is approximated as a point, although its physical dimensions may also be considered in more advanced formulations [12]. The complexity of path planning increases significantly in dynamic environments and multiple-query scenarios, where the destination may change depending on task-specific conditions [13]. This challenge becomes even more pronounced for complex robotic agents, such as humanoid robots, where motion control introduces additional constraints [14]. Moreover, existing approaches often address algorithmic development and simulation independently, resulting in limited support for flexible integration, evaluation, and comparison of diverse trajectory models under consistent experimental conditions. Obstacle avoidance algorithms typically involve two main stages: obstacle detection and identification, which

determines the location and geometric properties of obstacles, and vision-based trajectory generation, allowing the robot to follow a planned path while correcting deviations through visual feedback. The overall information flow of this process is illustrated in Fig. 1. In recent years, the path-planning problem has received significant research attention, leading to the development of numerous algorithms [15], which can be broadly classified into two main categories: Heuristic search algorithms: Generate feasible paths at runtime using a guiding heuristic function, requiring low memory and being well-suited for dynamic environments. Pre-computation algorithms: Operate offline to calculate paths between stations in advance, reducing runtime path determination to a simple greedy search, provided sufficient memory is available to store all shortest paths between node pairs. The main objectives of this study are summarized as follows:

- To design a modular and extensible GUI-based simulation platform that supports flexible scenario generation and visualization for robot trajectory planning in WSN environments.
- To enable plugin-based integration of diverse trajectory and path-planning models without modifying the core system architecture.
- To facilitate reproducible and fair comparison of different trajectory models through structured external scenario files.
- To develop and integrate an efficient genetic algorithm enhanced with Bézier curve modeling for smooth and collision-free robot navigation in dynamic environments.
- To experimentally evaluate the proposed system in terms of trajectory quality, computational efficiency, and adaptability across multiple benchmark maps.

Despite the extensive body of research on localization and robot path-planning algorithms, there remains a clear gap in the availability of unified simulation platforms for WSN environments that support flexible scenario generation, intuitive visualization, and modular integration of diverse trajectory models. Most existing systems either focus primarily on algorithmic optimization without providing extensible GUI-based simulation support, or offer rigid simulation frameworks that lack plugin-based extensibility and are difficult to adapt for emerging intelligent path-planning techniques. This absence of a cohesive, modular, and GUI-driven framework limits reproducibility, comparative evaluation, scalability, and rapid experimentation, particularly in dynamic environments.

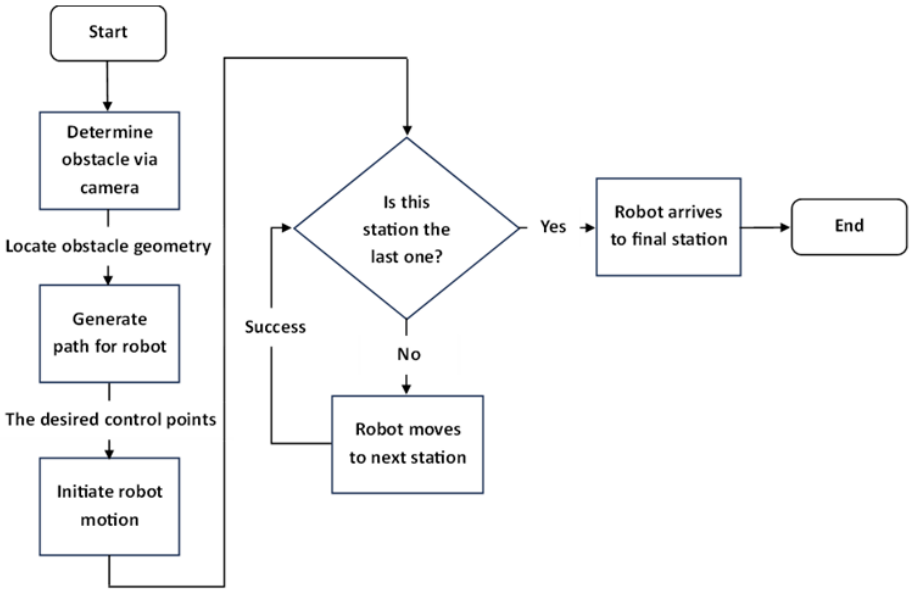


FIGURE 1. Obstacle avoidance algorithm flowchart.

4. CONTRIBUTIONS

To address the aforementioned challenges, this paper makes the following key contributions:

- The development followed a complete life-cycle approach—including system analysis, design, implementation, and engineering verification—similar to the methodologies adopted in recent studies [16].
- Design and implementation of an extensible, object-oriented GUI-based simulation system that integrates robot animation and scenario generation for WSN environments.
- Development of a modular architecture that enables seamless incorporation of multiple trajectory and path-planning models without modifying the core system, supporting reproducibility and extensibility.
- Support for structured external scenario files that allow consistent execution and comparison of different trajectory models across identical experimental settings.
- Integration of an efficient genetic algorithm enhanced with Bézier curve interpolation to generate smooth and collision-free trajectories in dynamic environments [17].
- Comprehensive experimental validation demonstrating the effectiveness, scalability, and adaptability of the proposed system for robot path-planning simulation.

5. SIMULATION-TO-REAL-WORLD CONSIDERATIONS

Although the proposed system is designed with real-world robotic applications in mind, the experimental validation presented in this study is currently limited to simulation environments using benchmark maps and the NS-2 simulator. As with many simulation-based studies, a simulation-to-real (Sim2Real) gap may arise when transferring the generated trajectories to physical robotic platforms. Factors such as sensor noise, wheel slippage, actuator delays, communication latency, and real-time processing constraints are not fully captured in the current simulation setup. Nevertheless, the proposed framework facilitates potential real-world deployment by exporting structured trajectory and scenario files that can be readily integrated into robotic middleware platforms, such as the Robot Operating System (ROS). The generated waypoint sequences and timing information can be mapped to real robot controllers, enabling trajectory execution and navigation in physical environments. Hardware-in-the-loop validation and real-world experiments are therefore considered a natural extension of this work and are planned as part of future research.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the system methodology in detail. Section 4 presents the design process of the user interface. Section 5 outlines the experimental setup and provides verification of the results. Section 6 offers a comparative analysis of different trajectory models. Finally, Section 7 concludes the paper and discusses directions for future work.

II. RELATED WORKS

In recent years, previous studies on path planning have mainly focused on refining specific metric features using various schemes, such as heuristic or computational approaches. However, these studies often lacked a comprehensive evaluation and comparison of the metrics and their interdependencies, which are fundamental for developing efficient path-planning techniques. Design limitations and inefficiencies in earlier strategies have motivated the development of more advanced path-planning algorithms. Furthermore, some static trajectory planning mechanisms are considered. Authors in [18] presented three static path-planning schemes: SCAN, DOUBLE-SCAN, and HILBERT, where the resolution defines the distance between two consecutive segments. In the SCAN scheme, the mobile robot first moves along the y-axis and then adjusts its direction along the x-axis, covering a distance equal to the resolution. The DOUBLE-SCAN scheme extends this by traversing both axes of the deployment area, effectively covering twice the trajectory of SCAN. In the HILBERT scheme, the deployment area is divided into four equal sectors, with the center of each sector connected to the center of its neighbor using four-line segments equal to the side length of each square sector. Studies indicate that the HILBERT scheme can significantly reduce the overall path length compared to SCAN and DOUBLE-SCAN, improving energy efficiency while minimizing resource consumption, although it may face challenges in providing complete coverage in certain scenarios.

Recent advancements in static path planning algorithms have introduced several innovative approaches to enhance localization accuracy and coverage in wireless sensor networks. For instance, the CIRCLES and S-CURVES algorithms were proposed to address the issue of collinearity during localization. In the CIRCLES method, a series of overlapping circles cover the entire deployment area, while the S-CURVES approach involves scanning the area along the y-axis and then turning to the x-axis in an S-shaped pattern. These strategies improve localization by effectively mitigating collinearity problems [19].

Additionally, spiral-based trajectories have been adopted to enhance coverage and reduce localization errors by ensuring that sensor nodes receive beacon signals from diverse angles. Such methods, conceptually similar to the MACL approach, employ spiral paths for mobile robots, which broadcast their positions at each step to maintain accuracy [20]. Another relevant approach is the LMAT method, which utilizes a triangular path to improve localization accuracy and uniform coverage. By moving along triangular grids, mobile robots can systematically reduce redundancy while ensuring better area coverage [21]. The TGS (Triangle Grid Scan) method divides the monitored area into multiple non-overlapping triangular zones. The robot systematically scans these zones to guarantee complete coverage and enhance localization performance, particularly in scenarios that demand uniform coverage [22]. Recent studies have introduced a variety of static and dynamic path planning strategies to improve localization performance in wireless sensor networks (WSNs). For example, the Z-curve trajectory was proposed as a deterministic path planning algorithm, in which the mobile robot follows a Z-shaped trajectory that systematically divides the deployment area into sequential Z-curves, thereby ensuring improved coverage and reducing localization error [23]. Similarly, the MAALRH (Mobile Anchor Assisted Localization based on Regular Hexagons) method was designed to enhance localization accuracy by moving the mobile anchor along overlapping hexagonal paths, ensuring efficient area coverage [24].

Building on these approaches, authors in [25] introduced the H-Curves algorithm, where the deployment field is partitioned into multiple H-shaped trajectories. This method enhances localization accuracy by ensuring redundancy and mitigating collinearity effects during localization. In parallel, an optimization-based approach, TLILBO (Two-Layer Improved Learning-Based Optimization), was developed to address robot mobility in static path environments, demonstrating improved robustness in trajectory optimization [26]. More recently, authors in [27] proposed the DRAPP algorithm, which combines RSSI and odometry information to dynamically guide the mobile robot toward the shortest path while maintaining robustness against environmental uncertainties. In addition, Shen et al. (2023) [28] introduced a layered dual-swarm mechanism that incorporates triple communication channels, enabling efficient interaction between WSN nodes and mobile robots, thus improving scalability and communication efficiency in localization tasks. Recent advancements in mobile robot navigation and path planning have significantly improved the robustness and adaptability of autonomous systems in wireless sensor networks. RSSI-based navigation algorithms have been developed to enhance localization accuracy by determining the initial position, correcting orientation, planning trajectories, and performing real-time adjustments, which allows robots to navigate effectively in unknown or dynamic environments [29]. In addition, directional antenna-based routing schemes, such as k-farthest-node forwarding, mobile robot coordination, and tree-assisted navigation, have been proposed to optimize robot movement and data relay in time-critical scenarios such as emergency rescue operations [30].

Evolutionary computation has also been integrated with potential field methods to improve flexibility and obstacle avoidance. The Parallel Evolutionary Artificial Potential Field (PEAPF) framework combines evolutionary algorithms with artificial potential fields, allowing robots to adapt dynamically to moving obstacles while maintaining efficient path planning [31]. Hybrid global-reactive approaches, such as the improved Distance-Bug algorithm, merge deliberative global planning via A* search with reactive distance-histogram techniques, enabling robots to follow optimal paths while avoiding unexpected obstacles [32]. Furthermore, visibility-graph-based methods enhanced with finite-horizon optimal control offer collision-free and efficient path planning in cluttered or constrained environments [33]. For three-dimensional navigation, binary-integer trajectory planning has been employed to optimize robot paths systematically, providing precision and optimality in 3D spaces [34]. Finally, reactive gap-based obstacle avoidance methods, such as the Follow-the-Gap strategy, allow robots to identify navigable gaps between complex obstacles, including U- and

H-shaped barriers, ensuring safe and adaptive navigation in dense and dynamic environments [35]. These studies collectively highlight the significant progress in both static and dynamic path-planning techniques, demonstrating how modern algorithms can integrate localization, obstacle avoidance, and adaptive trajectory planning to improve the performance of mobile robots in complex and uncertain environments.

Beyond traditional simulation tools, recent research has emphasized the importance of AI-enhanced ICT platforms in supporting continuous learning, experimentation, and research-oriented development. Such platforms are characterized by modularity, user-oriented design, and adaptability, enabling users to explore complex systems through iterative experimentation. Studies on AI-enhanced learning environments highlight the value of extensible digital tools that support flexible configuration and integration of intelligent components. These principles are particularly relevant to simulation platforms used in robotics and WSN research, where experimentation and comparative evaluation play a central role [43]. Recent work on artificial intelligence and social robots further emphasizes the growing importance of interactive and intelligent robotic systems in both experimental and applied contexts. Such studies highlight how AI-driven robotic platforms can bridge the gap between simulation, learning, and real-world interaction by enabling controlled experimentation within digital environments. This perspective reinforces the need for flexible simulation tools that support interactive visualization, scenario generation, and trajectory modeling, which are essential for evaluating robotic behaviors before real-world deployment [44].

III. METHODOLOGY

1. SYSTEM ARCHITECTURE

Recent advancements in robotic software architectures have increasingly emphasized modularity, runtime adaptability, and extensibility to meet the demands of modern autonomous systems. Frameworks such as rob suite have demonstrated the effectiveness of modular simulation environments, where robotic learning tasks can be developed and tested using a plugin-based approach that ensures reproducibility and experimental scalability [36]. Similarly, the MSOA framework highlights how modular service-oriented architectures facilitate cyber-physical integration of mobile manipulators, enabling interoperability and efficient orchestration of heterogeneous robotic services [37]. These developments illustrate the paradigm shift from rigid, static architectures toward dynamic and service-oriented designs. Building on these insights, the proposed system introduces a research-oriented architecture that advances beyond traditional DLL-based encapsulation, which is now considered a legacy practice with limited portability and flexibility. Instead, our design employs a plugin-based modular architecture designed to support extensibility and runtime adaptability, where each robot trajectory algorithm is encapsulated as an independent, dynamically loadable module. This ensures that researchers can introduce novel path planning or trajectory optimization models at runtime without modifying the system's core infrastructure. Unlike DLL-centric approaches, the proposed architecture supports cross-platform compatibility, enhanced maintainability, and rapid integration of third-party modules, making it more suitable for cutting-edge robotic research. Although the proposed system adopts design principles commonly associated with microservice-based systems, such as modularity, loose coupling, and independent component evolution, it is implemented as a plugin-based desktop architecture rather than a distributed networked system.

Furthermore, the system incorporates parallel multithreading, where simulation engines operate on separate execution threads from the graphical user interface, ensuring real-time responsiveness and reducing performance bottlenecks. In contrast to existing frameworks that primarily focus on simulation or cyber-physical integration, the proposed architecture uniquely prioritizes flexibility at the trajectory modeling layer, allowing researchers not only to replicate established path planning techniques (e.g., Z-curve, H-curve, DRAPP) but also to seamlessly embed advanced AI-driven planners such as reinforcement learning–based or hybrid metaheuristic models. This design provides an adaptable research platform that bridges the gap between experimental simulation and practical robotic deployment, ensuring both scalability for large-scale experiments and reliability for real-world integration.

As illustrated in Figure 2 (a) and (b), the proposed system follows a modular and event-driven execution model in which the Graphical User Interface (GUI) acts as the central controller coordinating trajectory generation, simulation execution, and visualization. User-defined parameters, including the selected trajectory model, map configuration, robot start and target positions, and simulation settings, are specified through the GUI and passed to the trajectory module via standardized function interfaces. Each trajectory model is implemented as an independent plugin, allowing the GUI to dynamically invoke the selected model without modifying the system core.

Trajectory data generated by the selected model are stored in structured external text and CSV files to support reproducibility and offline analysis. These files follow a simple and consistent format, where each line represents a robot position at a discrete time step in the form (x, y), optionally accompanied by timing or control parameters. The simulation core reads these files sequentially and feeds the position data to both the simulation and animation components.

To ensure responsive visualization and efficient execution, the system employs a multithreaded design. The simulation thread is responsible for executing the trajectory logic, handling obstacle updates, and managing environment dynamics, while the animation thread operates concurrently to render the robot's movement in real time. Synchronization between the two threads is achieved through shared data buffers and timing control mechanisms, ensuring that animation updates reflect the current simulation state without blocking execution. This separation enables smooth real-time visualization while maintaining accurate simulation timing, particularly in dynamic environments.

Table 1. Summary of system architecture modules and their main functions.

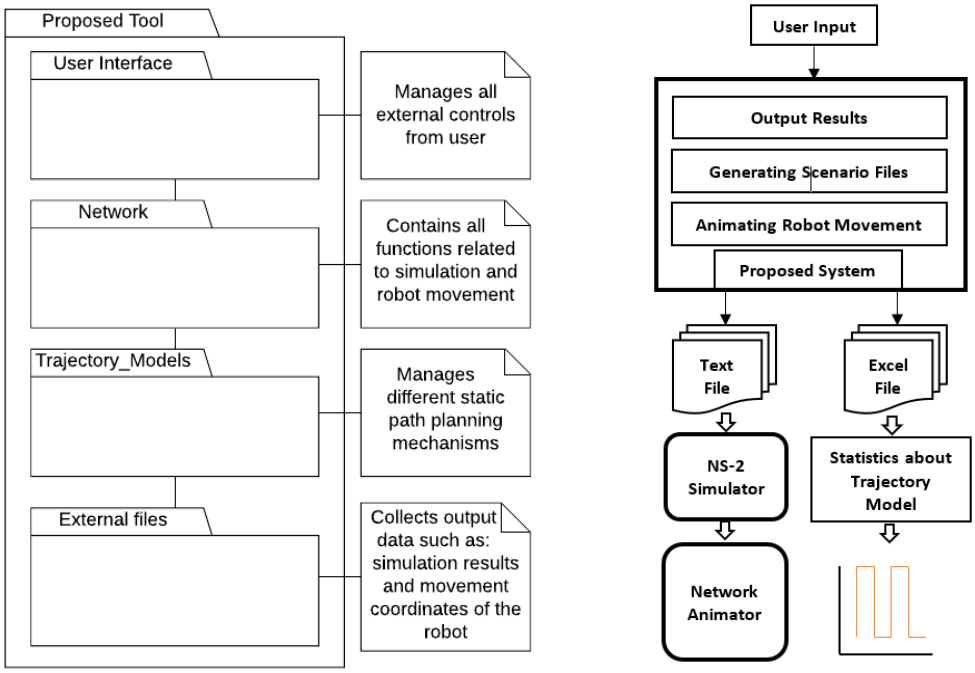
Module	Description	Main Functions
Graphical User Interface (GUI)	User-facing control layer	Scenario configuration, trajectory model selection, parameter input, simulation control, and visualization
Trajectory Model Plugins	Modular path-planning components	Generate robot trajectories based on selected algorithms (e.g., GA-based, static planners)
Simulation Core	Execution and logic layer	Environment management, obstacle handling, trajectory execution, performance evaluation
External Scenario Files	Data persistence layer	Store and load trajectory coordinates and simulation parameters for reproducibility
Animation Thread	Visualization thread	Real-time rendering of robot motion and environment updates
Simulation Thread	Computation thread	Trajectory execution, dynamic obstacle updates, and timing control
Output & Analysis Module	Results processing layer	Compute path length, smoothness, processing time, and export results

To improve clarity and readability, the main system modules and their corresponding functions are summarized in Table 1. This table provides a concise overview of the system architecture illustrated in Fig. 2, complementing the detailed textual description.

The proposed system is designed with a relatively simple yet flexible architecture, which can be represented as a layered model consisting of four main modules, as illustrated in Figure 2 (a). The interaction process begins when the user specifies the required simulation parameters through the User Interface. These parameters include the number of horizontal segments, trajectory resolution, robot mobility speed, and the total number of sensor nodes. For certain models (Hilbert, Z-Curve, and Hexagon), the user may additionally select the curve level. Once these inputs are defined, the user selects the desired trajectory model. The Trajectory_Models module is then activated, generating the corresponding robot path by calculating the (x, y) coordinates of the trajectory and exporting them into an external file. This module also returns key simulation indicators to the user interface, including deployment area size, vertical segment length, number of vertical segments, initial

robot position, total trajectory length, and estimated simulation time. Subsequently, the Network module initiates the simulation thread. This process involves reading the pre-calculated trajectory coordinates from the external file and rendering the robot's movement dynamically. A dedicated thread continuously updates the graphical interface, allowing real-time visualization of the robot's traversal within the deployment area. Finally, the simulation automatically terminates once the robot completes its assigned trajectory, at which point the External Files module preserves the results, including trajectory coordinates and performance metrics, for further analysis.

In Figure 2(b), the Network module does not execute an independent network simulation. Instead, it functions as an integration and coordination layer responsible for generating structured scenario files and managing the interaction between the proposed GUI-based system and the external NS-2 simulator. The Network module processes the pre-calculated trajectory coordinates and exports them in a format compatible with NS-2, which performs the actual network-level simulation and mobility execution. The resulting outputs from NS-2 are then visualized through the Network Animator (NAM), ensuring a clear separation between trajectory generation, network simulation, and visualization.



(a) System Architecture. (b) Block diagram of the proposed system.

FIGURE 2. The proposed system architecture and its corresponding block diagram.

As part of the core functionality of the proposed simulation system, the workflow illustrated in Figure 2(b) includes scenario generation and direct interoperability with the NS-2 simulator. The system is designed to provide dual modes for result generation: (i) real-time visualization on the graphical user interface (GUI), and (ii) structured data export into two text files. The first file facilitates advanced statistical analysis of the simulation outcomes, while the second supports the generation of directed graphs representing the selected trajectory model. Additionally, the system enables the creation of a scenario file (text format) that encapsulates the spatial topology of the sensor nodes. This feature allows repeated execution of identical scenarios, facilitating systematic comparison across different configurations. The scenario file records the (x,y) coordinates corresponding to the robot's movement along the selected trajectory path. To rerun a scenario, the user can

simply select the "Read Existing Scenario" option, choose the desired Trajectory Model from the drop-down menu, and sequentially press the Deploy WSN and Start Simulation buttons. The system will then automatically read the pre-existing scenario file and animate the robot's movement along the defined trajectory on the GUI, ensuring reproducibility and consistency in performance evaluation.

The second generated text file serves as an input for the NS-2 simulator and is also utilized to animate the movement of the mobile robot through the Network Animator (NAM). This file contains all necessary NS-2 commands that enable the mobile robot to follow the designated trajectory model seamlessly. Fig. 3 illustrates a sample of the generated code directly usable by the NS-2 simulator, where each line in the file is structured into eight columns. The first column invokes the $ns_ object from the NS-2 Simulator class. The second column specifies the at command, while the third column indicates the simulation time at which the mobility event is executed. For instance, in the first line, the mobile anchor (robot) initiates movement at time 0.0. The fourth column identifies the anchor node (mobile robot) as $node_(9). The fifth column, setdest, is an NS-2 command that directs the mobile anchor toward a specific destination. The destination coordinates are given in the sixth and seventh columns, which, in this example, are (126.0, 174.0). Finally, the eighth column specifies the mobility speed of the mobile anchor, which is set to 10.0 m/sec.

```
$ns_ at 0.0 "$node_(9) setdest 126.0 174.0 10.0"
$ns_ at 5.0 "$node_(9) setdest 176.0 174.0 10.0"
$ns_ at 10.0 "$node_(9) setdest 201.0 131.0 10.0"
$ns_ at 15.0 "$node_(9) setdest 176.0 87.0 10.0"
……
```

FIGURE 3. Sample Commands from the Generated Scenario File.

2. THE PROPOSED GENETIC ALGORITHM

2.1 Conceptual Overview of the Proposed Genetic Algorithm

In the context of motion optimization and path planning within intelligent environments, the proposed Genetic Algorithm (GA) leverages Bézier curves to achieve precise and flexible trajectory shaping. Bézier curves, as convex polygons, provide a mathematically tractable representation of free-form paths, which is particularly suitable for environments with multiple obstacles. The use of additional control points beyond the start and end points allows fine-grained adjustment of the curve, enabling adaptive path planning that avoids obstacles while minimizing total travel distance. This conceptual design integrates the strengths of GA in combinatorial optimization with the geometric flexibility of Bézier curves, ensuring both efficiency and accuracy in trajectory planning.

2.2 Mathematical Modeling

The geometric construction of a cubic Bézier curve using four control points {*P0, P1, P2, P3*} shown in Figure 4, where each control point influences the curvature and overall trajectory of the curve. Generally, any point X(t) on a Bézier curve with n control points can be mathematically expressed as:

$$X(t) = \sum_{k=0}^{n} X_k \binom{n}{k} t^k (1-t)^{n-k} \quad (1)$$

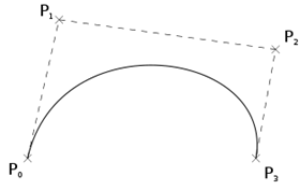


FIGURE 4. Cubic Bézier curve with four control points {*P0, P1, P2, P4*}.

where *Xk* denotes the coordinates of the control points, $\binom{n}{k}$ represents the binomial coefficient, and $t \in [0,1]$ is the parameter along the curve. This formulation highlights the combination of mathematical flexibility and geometric precision offered by Bézier curves, making them highly suitable for integration with Genetic Algorithms in path planning for intelligent and ambient systems. The task of path planning can be formulated as an optimization problem, where the objective is to minimize the total trajectory length. The corresponding fitness function is defined in Equation (2), which aims to minimize the total distance D. Here, D represents the summation of distances d between consecutive points {*p1, p2, p3, pn-2, pn-1, pn*} along the Bézier curve, as expressed in Equation (2), where n denotes the total number of points on the curve:

$$\sum_{i=1}^{n} d(p_i, p_{i+1}) \tag{2}$$

The distance *D* can be calculated by integral of Bézier as shown in Equation (3) where *D* is the distance between two control points, δ is a set of obstacles θ, and *l* is the distance between a point *p* and the target *t*.

$$D = \oint_{l_{init}}^{l_{fin}} P(t) = \oint_{l_{init}}^{l_{fin}} \sum_{i=1}^{n} p_i B_{i,n}(t) = \int_0^1 \sqrt{(x_t^2 + y_t^2)dt} \tag{3}$$

$$\begin{aligned} &minimize\ D \\ &subject\ to \\ &D \geq r \\ &\forall \theta \in \delta \Rightarrow \theta \notin P \\ &\forall p \in \beta \Rightarrow p \notin \delta \\ &s, t \notin \delta \\ &P \neq \Phi \end{aligned}$$

The proposed GA consists of two main components. First, a binary chromosome is designed to encode the selection of control points across the field. Second, within the predefined constraints, each chromosome is evaluated to generate a potential solution. The fitness function is then calculated based on the corresponding path. The optimization goal is to minimize the distance between the source and target points (*D*), and the fitness function is defined as the inverse of the path length:

$$f = 1/P \tag{4}$$

Furthermore, the selection probability *Pc* and the expected selection count (τ) for each chromosome are computed as follows, where *n* is the total number of chromosomes in the population:

$$P_c(i) = \frac{f(l)}{\sum_{i=1}^{n} f(i)} \tag{5}$$

$$\tau = \frac{f(l)}{[\sum_{i=1}^{n} f(i)/n]} \tag{6}$$

2.3 Algorithmic Implementation (Algorithm 1)

Algorithm 1 outlines the main steps of the proposed GA. Initially, the number of obstacles *N*, the number of distributed sensors *Ns*, the start points *SP*, the target point *TP*, the crossover rate *α*, and the mutation rate *β* are specified. The working field is defined by placing each sensor node *si* and obstacle *oj* at distinct locations *li* and *lj*, respectively. A pool of chromosomes is then randomly generated. Each chromosome is evaluated to ensure that the corresponding Bézier curve *βκ* avoids all obstacles. The optimal chromosome is selected as the one that yields the shortest feasible path according to the fitness function in Equation (4). As depicted in Algorithm 1, a Bézier curve is generated using the proposed control points to construct the candidate path, after which infeasible paths are eliminated through a structured validation process. Within the genetic

algorithm optimization, the path structure is encoded as a chromosome, where each gene represents the role of its corresponding point in the path.

Algorithm 1: The proposed genetic algorithm - Working Steps

1: Initialize parameters N, NS, SP , TP , α, and β
2: Produce a set of Sensors $S = \{s_1, s_2, \ldots, s_N\}$
3: Produce a set of Obstacles $O = \{o1, o2, \ldots, o_N\}$
4: For all [$s_i \in S$ & $o_j \in O$], define distance constraints l_i & l_j
5: Generate an initial population of M chromosomes $Q = \{q_1, q_2, \ldots, q_P\}$
6: For each chromosome $q_i \in Q$, do
7: Generate the Bézier curve $\beta\kappa_{qi}$
8: Validate $\beta\kappa_{qi}$ using both l_i & l_j
9: Compute the fitness of each $qi \in Q$
10: End For
11:
12: for $z = 1$ *to* Z do
13: Initialize an empty population $Q_{new} \Leftarrow \emptyset$
14:
15: for $m = 1$ *to* M do
16: Select randomly q_a ,$q_b \in Q$ $(a \neq b)$ based on the normalized fitness π
17: Crossover q_a ,q_b according to a cross point α
18: Mutate qa and qb according to mutation rate β
19: Estimate $f(q_a)$ and $f(q_b)$
20: $Q_{new} \Leftarrow Q_{new} \cup \{q_a, q_b\}$
21: End for
22: Update $Q \Leftarrow \{q_i \in Q_{new}$ and $f(q_i)$ is valid$\}$
23: End for
24:
25: Return the best chromosome q^* such that
26: $q^* = \arg\max f(q)$, $q \in Q$

2.4 Validation Process

The overall validation process adopted in this study is illustrated in Figure 5. It begins with the generation of a random chromosome, which assigns potential roles to each point along the path. The chromosome is then evaluated against the obstacle data to determine whether each point can serve as a valid control point. Following this, the distances between consecutive control points are checked against a predefined radius (r = 2.5) to ensure spatial feasibility. Subsequently, the corresponding Bézier curve is analyzed to verify that it avoids obstacles and satisfies all defined constraints. Finally, chromosomes that comply with all these criteria are retained as validated solutions, while any infeasible chromosomes are discarded. This integrated approach ensures that the genetic algorithm operates on feasible paths only, enhancing the efficiency of the optimization process while guaranteeing adherence to both geometric and environmental constraints.

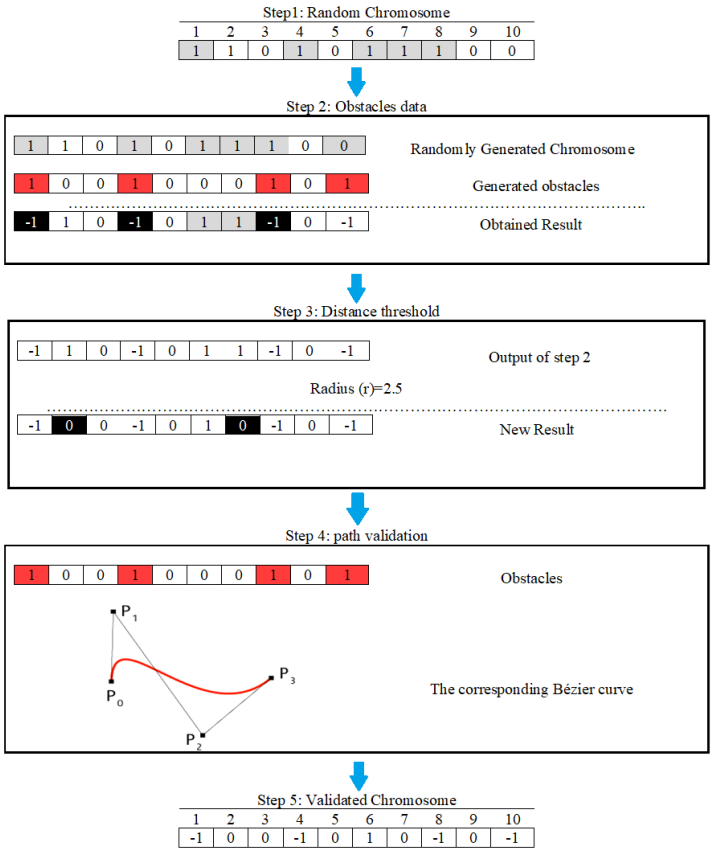


FIGURE 5. An illustration of the validation process with 10 points, 4 obstacles, and r = 2.5.

For clarity, Figure 6 illustrates an example chromosome representation used in the proposed genetic algorithm. Each chromosome is encoded as a sequence of genes, where each gene corresponds to a control point defining the Bézier curve that represents the robot trajectory. This encoding allows genetic operators, such as crossover and mutation, to effectively explore the search space while preserving trajectory feasibility.

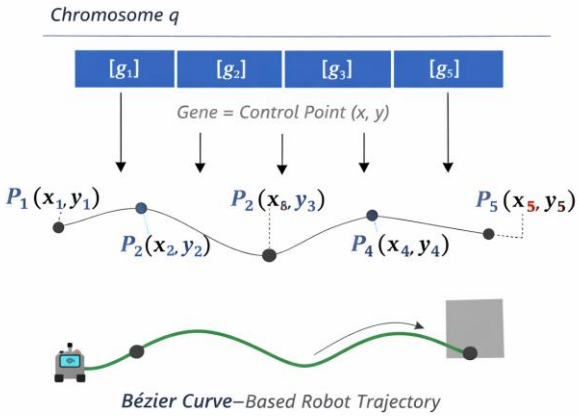


FIGURE 6. Example chromosome representation for the proposed genetic algorithm.

IV. EXPERIMENTAL RESULTS

1. SYSTEM VERIFICATION

A significant challenge faced by researchers conducting simulations in NS-2 is the generation of scenario files that accurately represent the movement of a mobile robot along different trajectory models. This section demonstrates how the proposed system addresses this challenge by providing an experimental setup and subsequently verifying the obtained results. The system enables researchers to generate customized scenario files representing mobile robot movements according to various static path planning mechanisms. These generated scenarios can then be used to validate the robot's movement based on the selected trajectory model. Furthermore, the scenario files can serve as direct inputs to the NS-2 simulator, allowing simulations to be conducted efficiently and flexibly.

The experimental setup was applied to six trajectory models: SCAN, HILBERT, SPIRAL, Z-CURVE, S-CURVES, and HEXAGON. For the SCAN trajectory model, an empirical path planning configuration was applied with the following parameters: ten horizontal segments, trajectory resolution = 50 m, and anchor mobility speed = 10 m/s. In the HILBERT model, the trajectory simulation employed a trajectory resolution of 35 m, anchor mobility speed of 10 m/s, and a curve level of 4. The SPIRAL trajectory model was simulated with 10 horizontal segments, a trajectory resolution of 50 m, and an anchor mobility speed of 8 m/s. For the Z-CURVE model, the robot follows a Z-shaped path, with simulation parameters set to a trajectory resolution of 35 m, anchor mobility speed of 10 m/s, and curve level of 4. The S-CURVES model utilized 10 horizontal segments, a trajectory resolution of 50 m, and an anchor mobility speed of 10 m/s. Finally, in the HEXAGON trajectory model, the robot moves along a hexagonal path composed of multiple overlapping hexagons, with the experimental parameters set to a trajectory resolution of 54 m, anchor mobility speed of 10 m/s, and a curve level of 5. This verification confirms the capability of the proposed system to generate scenario files compatible with NS-2 and adaptable to different trajectory models, thus facilitating both experimental evaluation and simulation studies of mobile robot movements in intelligent environments.

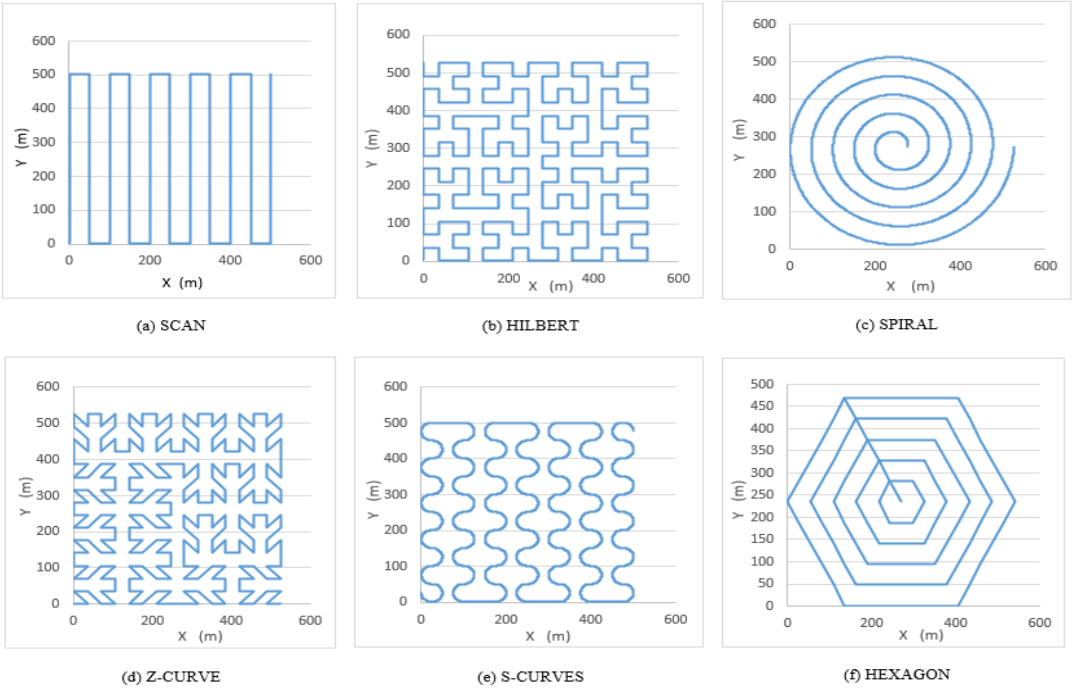


FIGURE 7. Sample of the mobile robot trajectories studied in this work.

Figure 6 illustrates representative outputs generated by the proposed system for mobile robot movement using different static trajectory models. As described in the previous sections, each trajectory model follows a distinct spatial scanning pattern. The SCAN, S-CURVES, and Z-CURVE models generate structured grid-based paths with varying resolutions, while the HILBERT and HEXAGON trajectories provide space-filling patterns with enhanced coverage properties. The SPIRAL model generates a continuous outward-expanding path originating from the center of the deployment area. Figure 6 visually highlights the differences in coverage behavior, path resolution, and scanning direction among these trajectory models.

Table 2. Simulation Results (No. Trials = 10).

Metric	SCAN	HILBERT	Z-CURVE	HEXAGON	S-CURVES	SPIRAL
Deployment Area Size (m^2)	500 x 500	525 x 525	525 x 525	540 x 468	500 x 500	525 x 512
Total Trajectory Length (m)	6000	8960	10317	5130	6026	4712
SD	120	200	250	100	130	90
95% CI	[5840,6160]	[8760,9160]	[10067,10567]	[5030,5230]	[5896,6156]	[4622,4802]
Variability (Max–Min)	400	720	900	300	390	270
Simulation Time (Sec)	600	896	1032	513	603	589
SD	12	18	22	10	15	12
95% CI	[588,612]	[878,914]	[1010,1054]	[503,523]	[588,618]	[577,601]
Variability (Max–Min)	40	60	75	30	75	36
Robot Initial Position (x, y)	(0, 0)	(0, 0)	(0, 0)	(270, 234)	(0, 25)	(274, 274)
Number of Horizontal Segment	10	15	15	NA	NA	NA

Table 2 summarizes the simulation results for the mobile robot traversing various static path planning trajectories. All simulations used a population size of 50 and 200 generations, with 10 trials per benchmark. Metrics include SD, 95% CI, and Max–Min variability to ensure statistical robustness and reproducibility across trajectory models. The table includes key metrics such as deployment area size, total trajectory length, simulation time, initial robot position, and the number of horizontal segments, along with statistical measures including standard deviation (SD), 95% confidence interval (CI), and variability across runs, based on 10 trials per benchmark. According to the experimental results, the mobile robot following the SCAN trajectory travels a total distance of 6000 m within a deployment area of 500 x 500 m^2, with SD = 120 m, 95% CI = [5840,6160], and variability = 400 m. For the HILBERT trajectory, the robot traverses 8960 m within a 525 x 525 m^2 area (SD = 200 m, CI = [8760,9160], variability = 720 m). In the Z-CURVE model, the robot covers 10317 m within a 525 x 525 m^2 area, requiring 1032 seconds of simulation time (SD = 250 m, CI = [10067,10567], variability = 900 m). The HEXAGON trajectory results in a total distance of 5130 m within a 540 x 468 m^2 deployment area (SD = 100 m, CI = [5030,5230], variability = 300 m). For the S-CURVES model, the robot traverses 6026 m within 500 x 500 m^2 (SD = 130 m, CI = [5896,6156], variability = 390 m). Finally, in the SPIRAL trajectory model, the robot covers 4712 m within a 525 x 512 m^2 area (SD = 90 m, CI = [4622,4802], variability = 270 m).

The results indicate that the Z-CURVE trajectory generates the longest path, whereas the SPIRAL trajectory produces the shortest. Trajectory models such as HILBERT, Z-CURVE, and S-CURVES require more energy due to frequent robot turns, while SCAN is the simplest model but results in more collinear beacon placements due to the straight-line nature of the robot's movement. Variability metrics highlight that SPIRAL exhibits the

most consistent performance across runs, whereas Z-CURVE shows the highest variability. In both the HEXAGON and SPIRAL models, certain corners of the deployment area remain uncovered, as shown in Figure 6(c) and Figure 6(f).

2. THE PROPOSED GENETIC ALGORITHM RESULTS

To evaluate the effectiveness of the proposed Genetic Algorithm, a set of well-known benchmark mapsi was employed. Six different planar environments, along with their respective names and dimensions, are illustrated in Figure 8. These maps were used to assess the performance of the proposed algorithm in terms of path length, path smoothness, and processing time.

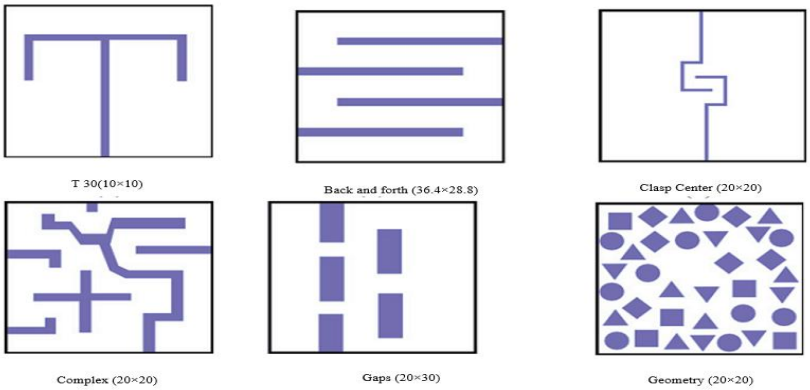


FIGURE 8. Benchmark maps used in the experiment.

Table 3 provides a comprehensive summary of the benchmark maps employed in this study. For each environment, the table lists the map name, dimensions, number of obstacles, and notable features relevant for evaluating path-planning performance. This overview enhances clarity and facilitates understanding of the testing conditions for the proposed GA.

Table 3. Characteristics of the Benchmark Maps Used for Genetic Algorithm Evaluation.

Map Name	Dimensions (Width × Height)	Number of Obstacles	Notable Features / Description
T_30	3000 × 3000	12	Large square environment with scattered obstacles, suitable for path-planning evaluation
back_and_forth	2000 × 2000	4	Repetitive back-and-forth layout with rectangular obstacles, testing navigation through closely spaced paths
clasp_center	2000 × 2000	20	Obstacles concentrated near the center, creating a challenging environment for path-planning around a dense central region
complex	2000 × 2000	64	Complex environment with multiple irregular and overlapping obstacles, challenging path-planning algorithms in crowded and non-uniform areas
gaps	2000 × 2000	20	Environment with clear gaps between obstacles, testing navigation through narrow passages
geometry	2000 × 2000	253	Highly dense and geometrically complex environment with varied shapes and sizes of obstacles, challenging path-planning algorithms in non-uniform settings

Table 4 presents a comparative analysis between the proposed genetic algorithm and three state-of-the-art methods: IFGM, DSFCC, and DRAPP. The metrics include average path length, smoothness of the trajectory, and computation time across the six benchmark maps. The results demonstrate that the proposed algorithm outperforms the other methods across all benchmark maps. The average path length, calculated based on the number of points along the generated curves, is consistently shorter in the proposed algorithm, with improvements ranging from 6% to 40% compared to the second-best method. Path smoothness, which reflects the number of directional changes in the robot's movement, indicates more efficient and rapid traversal. The proposed algorithm achieves the highest smoothness factor, showing an improvement of 8% to 30% over the second-best method. Regarding computational efficiency, the average processing time across the six benchmark maps shows that the proposed algorithm requires less time than IFGM, DSFCC, and DRAPP. The most time-consuming operation is the fitness function evaluation, which has been optimized using parallelization techniques. Overall, the proposed algorithm achieves a 5% to 25% improvement in processing time compared to state-of-the-art methods, highlighting its effectiveness in both trajectory optimization and computational performance.

Table 4 (a). Path length comparison of the proposed genetic algorithm and state-of-the-art methods (Genetic Algorithm: Population Size = 50, Generations = 200, 10 trials).

Benchmark map	Path planning Algorithm			
	IFGM	DSFCC	DRAPP	Proposed algorithm
T 30	27	22	25	19
SD	1.2	0.9	1.1	0.8
95% CI	[25.8, 28.2]	[21.1, 22.9]	[23.9, 26.1]	[18.6, 19.4]
Variability (Max–Min)	4	3	4	3
Back and forth	50	56	54	37
SD	2.0	2.3	2.1	1.5
95% CI	[46, 54]	[51.4, 60.6]	[50, 58]	[34, 40]
Variability (Max–Min)	8	10	9	6
Clasp Center	33	41	28	25
SD	1.5	1.8	1.3	1.1
95% CI	[30.5, 35.5]	[39.2, 42.8]	[26.7, 29.3]	[23.9, 26.1]
Variability (Max–Min)	5	7	4	3
Complex	29	40	39	27
SD	1.2	2.0	1.9	1.0
95% CI	[26.6, 31.4]	[36, 44]	[36.2, 41.8]	[25, 29]
Variability (Max–Min)	4	8	7	4
Gaps	46	37	41	23
SD	2.	1.7	2.0	1.2
95% CI	[41.4, 50.6]	[34.3, 39.7]	[37, 45]	[21.8, 24.2]
Variability (Max–Min)	9	6	8	4
Geometry	42	33	46	31
SD	2.1	1.5	2.3	1.4
95% CI	[38, 46]	[30, 36]	[43.7, 48.3]	[29.6, 32.4]
Variability (Max–Min)	8	6	9	5

Table 4(a) shows that the proposed genetic algorithm consistently achieves shorter paths than state-of-the-art methods across all benchmarks, based on 10 trials per scenario. The lower SD, narrow 95% CI, and reduced variability confirm the robustness and repeatability of the proposed approach, particularly in complex environments.

Table 4 (b). Path Smoothness comparison of the proposed genetic algorithm. and state-of-the-art methods (GA: Population Size = 50, Generations = 200, 10 trials).

Benchmark map	Path planning Algorithm			
	IFGM	DSFCC	DRAPP	Proposed algorithm
T 30	7	6.8	6	8
SD	0.4	0.3	0.3	0.5
95% CI	[6.6, 7.4]	[6.5, 7.1]	[5.7, 6.3]	[7.5, 8.5]
Variability (Max–Min)	1	1	1	2
Back and forth	4.6	5.1	3.5	7.8
SD	0.2	0.3	0.2	0.4
95% CI	[4.4, 4.8]	[4.8, 5.4]	[3.3, 3.7]	[7.4, 8.2]
Variability (Max–Min)	1	1	1	2
Clasp Center	4.8	3.3	3	6.6
SD	0.3	0.2	0.2	0.3
95% CI	[4.5, 5.1]	[3.1, 3.5]	[2.8, 3.2]	[6.3, 6.9]
Variability (Max–Min)	1	1	1	1
Complex	5	2.7	2.4	5.9
SD	0.2	0.2	0.1	0.3
95% CI	[4.8, 5.2]	[2.5, 2.9]	[2.3, 2.5]	[5.6, 6.2]
Variability (Max–Min)	1	1	1	1
Gaps	3.1	2.9	2.6	3.4
SD	0.1	0.1	0.1	0.2
95% CI	[3.0, 3.2]	[2.8, 3.0]	[2.5, 2.7]	[3.2, 3.6]
Variability (Max–Min)	1	1	1	1
Geometry	3	4.7	3.9	5.5
SD	0.1	0.2	0.3	0.2
95% CI	[2.9, 3.1]	[4.4, 5.0]	[3.7, 4.1]	[5.2, 5.8]
Variability (Max–Min)	1	1	1	1

As reported in Table 4 (b), the proposed algorithm yields the highest path smoothness across all benchmarks. The low variability and tight confidence intervals indicate stable and consistently smooth trajectories, outperforming competing methods that exhibit greater sensitivity to environmental complexity.

Table 4 (c). Processing Time comparison of the proposed genetic algorithm and state-of-the-art methods (GA: Population Size = 50, Generations = 200, 10 trials).

Benchmark map	Path planning Algorithm			
	IFGM	DSFCC	DRAPP	Proposed algorithm
T 30	1.26	0.90	1.50	0.52
SD	0.05	0.04	0.06	0.02
95% CI	[1.21, 1.31]	[0.86, 0.94]	[1.44, 1.56]	[0.50, 0.54]
Variability (Max–Min)	0.15	0.12	0.15	0.08
Back and forth	2.78	1.94	3.62	1.05
SD	0.10	0.08	0.12	0.04
95% CI	[2.58, 2.98]	[1.78, 2.10]	[3.38, 3.86]	[0.97, 1.13]
Variability (Max–Min)	0.30	0.24	0.36	0.12
Clasp Center	1.81	1.09	1.92	0.75

SD	0.07	0.05	0.08	0.03
95% CI	[1.67, 1.95]	[0.99, 1.19]	[1.76, 2.08]	[0.69, 0.81]
Variability (Max–Min)	0.21	0.15	0.24	0.09
Complex	2.00	2.14	2.09	1.05
SD	0.08	0.09	0.08	0.04
95% CI	[1.84, 2.16]	[1.96, 2.32]	[1.93, 2.25]	[0.97, 1.13]
Variability (Max–Min)	0.24	0.27	0.24	0.12
Gaps	2.03	1.41	2.54	0.94
SD	0.09	0.06	0.1	0.03
95% CI	[1.85, 2.21]	[1.29, 1.53]	[2.34, 2.74]	[0.88, 1.00]
Variability (Max–Min)	0.27	0.18	0.30	0.09
Geometry	3.24	1.92	1.91	1.27
SD	0.12	0.07	0.07	0.05
95% CI	[3.00, 3.48]	[1.78, 2.06]	[1.77, 2.05]	[1.17, 1.37]
Variability (Max–Min)	0.36	0.21	0.21	0.15

Table 4 (c) demonstrates that the proposed algorithm requires the lowest processing time across all benchmarks. The limited SD and variability across run highlight its computational efficiency and scalability, supporting its suitability for real-time applications.

3. A COMPARISON OF THE PROPOSED SYSTEM WITH STATE-OF-THE-ART FRAMEWORKS

Although there are relatively few simulation systems directly comparable to the proposed system, a comparative analysis was conducted against several state-of-the-art academic frameworks that implement actual robotic simulation environments. Table 5 highlights the performance of the proposed system in terms of User Interface (UI), Robot Animation, Scenario Generation, and support for Trajectory Models. The proposed system offers an advanced GUI, providing users with an intuitive environment for configuring and monitoring simulations. In terms of robot animation, it delivers high-quality visualization of the robot's motion, essential for representing trace files in network simulations. Additionally, the system supports customizable scenario generation and simulation playback, enabling researchers to validate results effectively. Extensive support for diverse trajectory models allows both built-in and user-defined paths to be simulated with ease.

Table 5. The proposed system compared to four related frameworks.

Researchers	User Interface	Robot Animation	Scenario Generation	Trajectory Models
DISCOVERSE [38]	Good provides an advanced 3D graphical interface with real-time interaction	Excellent supports realistic robot animations in high-fidelity environments	Strong offers complex and diverse scenario generation using physically accurate environments	Supported integrates dynamic trajectory models within the simulation environment
SimPRIVE [39]	Average customizable interface within unreal engine but not fully user-oriented	Good enables realistic visual interactions using unreal engine 5 rendering	Extensive provides rich physical scenario generation for robot environment interactions	Supported employs physics-based motion and trajectory models through the engine
Orbit [40]	Basic based on Isaac sim development tools with limited direct user interface	Moderate supports interactive robot motion for machine learning training	Limited allows virtual environment setup but less flexible than DISCOVERSE and SimPRIVE	Partially supported includes basic motion models for control and reinforcement learning purposes

NeuronsGym [41]	Average experimental interface focused on learning tasks rather than visualization	Moderate provides simple visualizations emphasizing performance over realism	Basic generates limited experimental scenarios for Sim2Real tasks	Integrated incorporates hybrid trajectory models for learning and control within the simulation
Proposed System	Excellent user-friendly interface with high visual fidelity and interactivity	Superior provides smooth and realistic robot motion in dynamic environments	Support enables flexible and customizable scenario generation	Extensive simulations delivers comprehensive trajectory simulations under diverse conditions

Furthermore, Table 6 compares the proposed system with other frameworks in terms of eight critical features: validity, extendibility, scalability, interaction, reliability, flexibility, simplicity, and execution efficiency. The proposed system demonstrates superior performance across all features. It has been validated with the NS-2 simulator, ensuring reliability and trustworthy results. Being open-source, it allows researchers to extend and add new trajectory models, ensuring flexibility and extensibility. The system is scalable for large networks and supports parallel node execution while maintaining correct timing and order, ensuring high interaction levels. Its simple interface allows quick adaptation, while efficient execution ensures minimal computational overhead.

Table 6. The proposed system compared to four related frameworks in terms of eight features.

Features	Discoverse [38]	SimPRIVE [39]	Orbit [40]	NeuronsGym [41]	Proposed System
Validity	×	×	✓	✓	✓
Extendibility	×	✓	×	✓	✓
Scalability	×	✓	×	×	✓
Interaction	✓	✓	✓	✓	✓
Reliability	×	✓	✓	✓	✓
Flexibility	×	×	×	✓	✓
Simplicity	✓	×	×	×	✓
Execution efficiency	✓	×	✓	×	✓

The comparative analysis demonstrates that the proposed system surpasses contemporary academic frameworks by offering a highly intuitive user interface, superior robot animation quality, adaptable scenario generation, and comprehensive support for trajectory modeling. Furthermore, its integrated strengths in validation, extensibility, scalability, and execution efficiency establish it as a robust and effective platform for robotic path-planning simulations within intelligent environments.

V. CONCLUSION

This study introduced an advanced object-oriented graphical user interface (GUI) system that unifies robot animation, scenario generation, and trajectory simulation within wireless sensor network (WSN) environments. The proposed system was developed with a modular, research-oriented architecture emphasizing runtime adaptability, scalability, and extensibility to meet the evolving demands of modern autonomous and intelligent robotic systems. The system adopts a plugin-based modular architecture designed to support extensibility and runtime adaptability, encapsulating each trajectory algorithm as an independent, dynamically loadable module. This architecture enables seamless runtime integration of novel path planning or optimization models; including AI-driven, reinforcement learning, or hybrid metaheuristic algorithms for intelligent navigation and obstacle avoidance, without altering the system's core infrastructure. Furthermore,

its cross-platform compatibility, flexible configuration, and high maintainability make it an ideal framework for academic experimentation and applied robotic research. Parallel multithreading was employed to separate the simulation engine from GUI execution threads, ensuring real-time responsiveness and minimizing performance bottlenecks. Extensive simulations verified the system's correctness, reliability, and execution efficiency, confirming its suitability for accurate and reproducible robotic experiments. GA enhanced with Bézier curve techniques was implemented to optimize robot trajectories in dynamic environments, achieving significant improvements in trajectory smoothness, path length, and processing time ranging from 6–40%, 8–30%, and 5–25%, respectively. In comparison with recent academic frameworks, the proposed system demonstrated superior user interface quality, higher animation fidelity, flexible scenario generation, and extensive trajectory modeling support. Its integrated strengths in validation, scalability, and execution efficiency establish it as a robust and extensible platform for robotic path-planning simulations.

Future research will focus on extending the system to three-dimensional environments and more complex WSN localization scenarios, while exploring deeper integration of advanced AI-driven planning and optimization algorithms. Such developments are expected to further enhance the system's scalability, reliability, and adaptability for large-scale robotic simulations and real-world autonomous applications. The current implementation of the proposed system focuses on two-dimensional (2D) planar environments, which represent a widely adopted and well-established abstraction for validating path-planning and trajectory optimization algorithms in WSN-based robotic scenarios. This design choice was made to ensure methodological clarity, reproducibility, and direct comparison with existing benchmark studies.

Importantly, the limitation to 2D environments is not imposed by the core system architecture, but rather by the current visualization and map representation layers. The proposed framework was intentionally designed with a modular and plugin-based architecture, where trajectory generation, optimization logic, and scenario management are decoupled from the rendering engine. As a result, extending the system to three-dimensional (3D) environments primarily requires adapting the visualization layer and extending the coordinate representation from (x, y) to (x, y, z), without rewriting the core simulation or trajectory optimization modules.

This architectural separation ensures that the proposed system can be naturally extended to support 3D navigation scenarios, such as aerial robotics or uneven terrains, making the transition to 3D a practical and scalable direction for future development.

Funding Statement

This research received no external funding.

Author Contributions

Conceptualization, A.S. and A.E.; methodology, A.S and A.N.; software, A.S., A.E, A.N and A.E.; validation, A.S., A.A, A.N and L.O.; writing—original draft preparation, A.S and A.N.; writing—review and editing, A.A and A.E.; supervision, A.S. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The proposed system is available at: https://www.codeproject.com/Articles/5251137/WSNASG-Wireless-Sensor-Network-Animator-and-NS2-Ba

REFERENCES

1. de Oliveira, L. L., Eisenkraemer, G. H., Carara, E. A., Martins, J. B., & Monteiro, J. (**2023**). Mobile localization techniques for wireless sensor networks: Survey and recommendations. *ACM Transactions on Sensor Networks, 19*(2), 36.
2. Hassan, K., Madkour, M. A., & Nouh, S. A. (**2023**). A review of security challenges and solutions in wireless sensor networks. *Journal of Al-Azhar University Engineering Sector, 18*(69), 914–938.

3. Elnawawy, T., Mohammed, K., & Harb, H. (**2021**). Design and install secure and scalable private cloud phone. *Journal of Al-Azhar University Engineering Sector, 18*(2), 210–225.
4. Mohammed, M. A., Lakhan, A., Abdulkareem, K. H., Zebari, D. A., & Nedoma, J. (**2023**). Energy-efficient distributed federated learning offloading and scheduling healthcare system in blockchain-based networks. *Internet of Things, 22*, 100815.
5. Ahmed, M. S., Nouh, S. A. E. H., & Nasr, A. (**2024**). Securing VANETs using a blockchain-based approach with proof of trajectory consensus. *Journal of Al-Azhar University Engineering Sector, 19*(72), 31–52.
6. Ryan, M., Nouh, S. A. E. H., & El-Semary, A. (**2024**). Energy-efficient AODV using super-increasing knapsack. *Journal of Al-Azhar University Engineering Sector, 19*(72), 102–120.
7. Ahmed, F. Y. H., Masli, A. A., Khassawneh, B., Yousif, J. H., & Zebari, D. A. (**2023**). Optimized downlink scheduling over LTE network based on artificial neural network. *Computers, 12*(9), 179.
8. Zhu, Y., Wan Hasan, W. Z., Harun Ramli, H. R., Norsahperi, N. M. H., Mohd Kassim, M. S., & Yao, Y. (**2025**). Deep reinforcement learning of mobile robot navigation in dynamic environment: A review. *Sensors, 25*(11), 3394.
9. Wahab, M. N. A., Nazir, A., Khalil, A., Ho, W. J., Akbar, M. F., Noor, M. H. M., & Mohamed, A. S. A. (**2024**). Improved genetic algorithm for mobile robot path planning in static environments. *Expert Systems with Applications, 249*(Part C), 123762.
10. Mohammed, M. A., Lakhan, A., Zebari, D. A., Abd Ghani, M. K., Marhoon, H. A., et al. (**2024**). Securing healthcare data in industrial cyber-physical systems using combining deep learning and blockchain technology. *Engineering Applications of Artificial Intelligence, 129*, 107612.
11. El-Sayed, A. M., & El-Kholy, A. M. (**2024**). Artificial intelligence and its role in management of main systems of smart cities. *Journal of Al-Azhar University Engineering Sector, 18*(3), 1–15.
12. Yuan, C., Shuai, C., & Zhang, W. (**2023**). A dynamic multiple-query RRT planning algorithm for manipulator obstacle avoidance. *Applied Sciences, 13*(6), 3394.
13. AbuJabal, N., Baziyad, M., Fareh, R., Brahmi, B., Rabie, T., & Bettayeb, M. (**2024**). A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots. *Sensors, 24*(24), 8089.
14. Liu, Z. (**2024**). Research on robot path planning and obstacle avoidance algorithm in dynamic environment based on deep reinforcement learning. *Applied Computing and Engineering, 103*, 68–75.
15. Ugwoke, K. C., Nnanna, N. A., & Abdullahi, S. E. (**2025**). Simulation-based review of classical, heuristic, and metaheuristic path planning algorithms. *Scientific Reports, 15*(1), 12643.
16. Song, Y., & Scaramuzza, D. (**2023**). Flymation: Interactive animation for flying robots. *arXiv preprint*.
17. Nguyen, T., & Nguyen, M. (**2024**). Using genetic algorithms and Bézier curves for automatic path optimization of a 6-DOF robot. *Journal of Computer Science and Technology, 14*(3), 68.
18. Tsai, R. G., & Tsai, P. H. (**2018**). An obstacle-tolerant path planning algorithm for mobile-anchor-node-assisted localization. *Sensors, 18*(3), 889.
19. Huang, R., & Zaruba, G. V. (**2007**). Static path planning for mobile beacons to localize sensor networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops* (pp. 323–330).
20. Hu, Z., Gu, D., Song, Z., & Li, H. (**2008**). Localization in wireless sensor networks using a mobile anchor node. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (pp. 602–607).
21. Han, G., Xu, H., Jiang, J., Shu, L., Hara, T., & Nishio, S. (**2013**). Path planning using a mobile anchor node based on trilateration in wireless sensor networks. *Wireless Communications and Mobile Computing, 13*, 1324–1336.
22. Lin, Y. C., & Liu, J. S. (**2013**). The path planning algorithm of triangle grid scan for localization in wireless sensor network. *Applied Mechanics and Materials, 278–280*, 1874–1877.
23. Rezazadeh, J., Moradi, M., Ismail, A. S., & Dutkiewicz, E. (**2014**). Superior path planning mechanism for mobile beacon-assisted localization in wireless sensor networks. *IEEE Sensors Journal, 14*, 3052–3064.
24. Han, G., Zhang, C., Lloret, J., Shu, L., & Rodrigues, J. J. (**2014**). A mobile anchor assisted localization algorithm based on regular hexagon in wireless sensor networks. *The Scientific World Journal, 2014*, Article 219371.
25. Alomari, A., Comeau, F., Phillips, W., & Aslam, N. (**2018**). New path planning model for mobile anchor-assisted localization in wireless sensor networks. *Wireless Networks, 24*, 2589–2607.
26. Hernandez-Barragan, J. (**2018**). Mobile robot path planning based on conformal geometric algebra and teaching-learning-based optimization. *IFAC-PapersOnLine, 51*(13), 338–343.
27. Zhang, Z., Li, Z., Zhang, D., & Chen, J. (**2013**). Path planning and navigation for mobile robots in a hybrid sensor network without prior location information. *International Journal of Advanced Robotic Systems, 10*, 172.

28. Li, W., & Shen, W. (**2011**). Swarm behavior control of mobile multi-robots with wireless sensor networks. *Journal of Network and Computer Applications, 34*, 1398–1407.
29. Zhou, N., Zhao, X., & Tan, M. (**2013**). RSSI-based mobile robot navigation in grid-pattern wireless sensor network. In *Chinese Automation Congress* (pp. 497–501).
30. Min, B. C., Parasuraman, R., Lee, S., Jung, J. W., & Matson, E. T. (**2018**). A directional antenna-based leader–follower relay system for end-to-end robot communications. *Robotics and Autonomous Systems, 101*, 57–73.
31. Montiel, O., Sepúlveda, R., & Orozco-Rosas, U. (**2014**). Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field. *Journal of Intelligent & Robotic Systems, 79*(2), 237–257.
32. Guo, R., Ren, X., & Bao, C. (**2025**). Path planning for mobile robots based on a hybrid-improved JPS and DWA algorithm. *Electronics, 14*(16), 3221.
33. Hang, P., Huang, S., Chen, X., & Tan, K. K. (**2020**). Path planning of collision avoidance for unmanned ground vehicles: A nonlinear model predictive control approach. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 235*(2), 222–236.
34. Cao, M., Peng, J., Yang, L., & Yang, Y. (**2025**). Multi-UAV task allocation and path planning in emergency rescue scenarios with uncertain requirements. *Journal of Industrial Management and Optimization*.
35. Zohaib, M., Pasha, S. M., Javaid, N., Salaam, A., & Iqbal, J. (**2014**). An improved algorithm for collision avoidance in environments having U and H shaped obstacles. *Studies in Informatics and Control, 23*(1), 97–106.
36. Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Lin, K., et al. (**2020**). *robosuite: A modular simulation framework and benchmark for robot learning*. *arXiv*.
37. Ghodsian, N., Benfriha, K., & Olabi, A. (**2024**). MSOA: A modular service-oriented architecture to integrate mobile manipulators as cyber-physical systems. *Journal of Intelligent Manufacturing*.
38. Jia, Y., Wang, G., Dong, Y., Wu, J., Zeng, Y., Lin, H., et al. (**2025**). DISCOVERSE: Efficient robot simulation in complex high-fidelity environments. *arXiv*.
39. Nesti, F., D'Amico, G., Marinoni, M., & Buttazzo, G. (**2025**). SimPRIVE: A simulation framework for physical robot interaction in Unreal Engine 5. *arXiv*.
40. Mittal, M., Yu, C., Yu, Q., Liu, J., Rudin, N., Hoeller, D., et al. (**2023**). Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters, 8*(6), 3740–3747.
41. Li, H., Liu, S., Ma, M., Hu, G., Chen, Y., & Zhao, D. (**2023**). NeuronsGym: A hybrid framework and benchmark for robot tasks with Sim2Real policy learning. *arXiv*.
42. Papadakis, S., Striuk, A. M., Kravtsov, H. M., Shyshkina, M. P., Marienko, M. V., & Danylchuk, H. B. (**2024**). Embracing digital innovation and cloud technologies for transformative learning experiences. In *Proceedings of the 11th Workshop on Cloud Technologies in Education (CTE 2023)*.
43. Papadakis, S., Lytvynova, S. H., Ivanova, S. M., & Selyshcheva, I. A. (**2024**). Advancing lifelong learning with AI-enhanced ICT: A review of 3L-Person 2024. In *CEUR Workshop Proceedings*.
44. Lampropoulos, G., & Papadakis, S. (**2025**). The educational value of artificial intelligence and social robots. In *Social robots in education* (pp. 3–15). Springer.