



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E
DE COMPUTAÇÃO



A Real-Time Meta-Heuristic-Based Safe Navigation Approach for Mobile Robots in Unknown Environments

Micael Balza

Advisor: Prof. Dr. Marcelo Augusto Costa Fernandes

Co-advisor: Prof. Dr. Sérgio Natan Silva

Master's Dissertation submitted to the Graduate Program in Electrical and Computer Engineering at UFRN (field of concentration: Computer Engineering) as part of the requirements for obtaining the degree of Master of Science.

Order number PPgEEC: M684
Natal, RN, December 2024

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Balza, Micael.

A Real-Time meta-heuristic-based safe navigation approach for mobile robots in unknown environments / Micael Balza. - 2025.
74f.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-graduação em Engenharia Elétrica e de Computação, Natal, 2025.

Orientação: Prof. Dr. Marcelo Augusto Costa Fernandes.

Coorientação: Prof. Dr. Sérgio Natan Silva.

1. Autonomous navigation - Dissertação. 2. Metaheuristic - Dissertação. 3. Mobile robots - Dissertação. 4. Path planning - Dissertação. 5. Unknown environment - Dissertação. I. Fernandes, Marcelo Augusto Costa. II. Silva, Sérgio Natan. III. Título.

RN/UF/BCZM

CDU 004

Elaborado por Jackeline dos Santos Pinheiro da Silva Maia
Cavalcanti - CRB-15/317

A Real-Time Meta-Heuristic-Based Safe Navigation Approach for Mobile Robots in Unknown Environments

Micael Balza

Master's Dissertation approved on February 19, 2025, by the examination committee composed of the following members:

Prof. Dr. Marcelo Augusto Costa Fernandes (Advisor) DCA/UFRN

Prof. Dr. Sérgio Natan Silva (Co-advisor) InovAI Lab/Lance/nPiti/UFRN

Dr. Fábio Fonseca de Oliveira External to the Institution

Prof. Dr. Diogo Pinheiro Fernandes Pedrosa DCA/UFRN

Acknowledgments

I would like to express my deep gratitude to my family, especially to my parents, Romualdo and Márcia Balza, for all these years of dedication and unconditional support. To my sisters, Mayra and Milena, who encouraged and assisted me throughout this journey. I am also grateful to Thiago for his valuable conversations about graduate studies. I cannot fail to mention my nephews, Moara, Lui, and Ian, who are a source of inspiration to me. My gratitude also extends to my wife's family for their support and understanding during this period.

To my wife, Maria Helena, who has been by my side in every stage of this journey, always supporting and motivating me to pursue new paths. And, with immense joy, to our little one, who will soon be joining us.

To my advisor, Marcelo, whose patience and dedication have been fundamental since my undergraduate research. I am grateful for his generous knowledge sharing and the countless hours of discussions and guidance over the years.

To my co-advisor, Sérgio, for his continuous support, knowledge exchange, and fundamental contributions in the laboratory.

To my colleagues and friends from the laboratory, who share this academic journey, enriching the environment with productive discussions, collaboration, and mutual learning.

To the Federal University of Rio Grande do Norte (UFRN) for providing a conducive environment for learning, research, and academic development.

To the National Council for Scientific and Technological Development (CNPq) for the financial support, which was essential for the realization of this work.

Abstract

Autonomous navigation in mobile robots is a complex challenge, particularly in unknown and dynamic environments where obstacle avoidance and real-time trajectory optimization are crucial. This work introduces the MetaHeuristic Real-Time Safe Navigation (MHRTSN) strategy, which integrates potential fields with population-based metaheuristics to enhance trajectory planning and navigation efficiency. The proposed strategy was evaluated through a series of simulations in different static and dynamic scenarios, comparing the performance of two versions: MetaHeuristic Real-Time Safe Navigation with Genetic Algorithm (MHRTSN-GA) and MetaHeuristic Real-Time Safe Navigation with Particle Swarm Optimization (MHRTSN-PSO). The evaluation considered key metrics such as displacement, distance traveled, CPU time, and clock time. The results indicate that both versions provide sub-optimal solutions, with MHRTSN-PSO demonstrating superior performance in terms of computational efficiency and convergence when using a small population size. Comparisons with existing approaches in the literature revealed that MHRTSN generated paths of similar length while maintaining a safer distance from obstacles. Thus, the proposed approach offers an efficient and safe solution for autonomous navigation in mobile robots, contributing to advancements in real-world robotic applications.

Keywords: Autonomous navigation; Metaheuristic; Mobile robots; Path planning; Unknown environment.

Contents

Table of Contents	i
List of Figures	iii
List of Tables	vii
1 Introduction	1
1.1 State of the art	3
1.2 Objectives and contributions	6
1.3 Organization of the work	7
1.4 Published Works	7
List of Symbols and Abbreviations	1
2 MHRTSN Strategy	9
2.1 Data Collection	13
2.2 Preprocessing	15
2.3 Metaheuristic Evaluation	19
2.4 Displacement	24
2.5 Algorithm Complexity	25
3 Methodology	28
3.1 Simulation Platform	28
3.1.1 Methodology for Performance Evaluation	29
4 Results	34
4.1 Static Environments	34
4.1.1 Results for Env1	36
4.1.2 Results for Env2	39
4.1.3 Results for Env3	41
4.1.4 Results for Env4	43
4.2 Dynamic Environments	45
4.2.1 Results for Env5	47
4.2.2 Results for Env6	49
4.3 Comparison with Other Approaches	51
4.4 Analysis of Results	55
4.5 Demonstration video	56

5 Conclusion	57
5.1 Future Work	58
References	59

List of Figures

2.1	Flowchart illustrating the MHRTSN algorithm's input-output relationships and processing steps.	10
2.2	Illustration of the conversion of distances into spatial points based on the robot's current location and angle.	13
2.3	Illustration of the data collection stage with $P = 60$ and $\Phi = 3$ meters for the n -th displacement. The robot is illustrated by a blue circle with its center marked by a blue square. Obstacles are illustrated by black-outlined objects. The information contained in $\mathbf{dp}_n$ is represented by red points. The polygon $\mathbf{dp}_n$ is illustrated by a red dashed line that connects these points and represents the free space around the robot.	15
2.4	Illustration of the obstacle point recognition stage (Equation 2.8), with $P = 60$, and $\Phi = 3$ meters.	16
2.5	Illustration of the generation of the polygon $\mathbf{dp}_n^{space}$ from the polygon $\mathbf{dp}_n$. In this illustration, $\mathbf{dp}_n$ is represented by a red dashed line. $\mathbf{dp}_n^{space}$ is depicted by a blue dashed line, marking the safe area for the movement of a point.	17
2.6	Illustration of the possible values of R based on the robot's turning capability. The robot is represented as an orange object with gray wheels. For Robot 1, which rotates around its own axis, R corresponds to the radius of the object that delimits it. For Robot 2, which has minimum turning restrictions, R represents the minimum turning radius.	18
2.7	Illustration of the calculation of the points of $\mathbf{dp}_n^{space}$ from the polygon $\mathbf{dp}_n$. $\mathbf{dp}_n^{space}$ is shown in blue, and $\mathbf{dp}_n$ in red. The vector $\mathbf{v}_{n,p}$ is represented by a black arrow, indicating the direction of the displacement.	19
2.8	Representation of the polygons $\mathbf{dp}_n$ and $\mathbf{dp}_n^{space}$ in an environment with obstacles. Configuration: $N = 60$, $R = 0.15$ m, $\alpha = 1.4$, and $\Phi = 3$ m. The robot is illustrated by a blue circle with its center marked by a blue square. Obstacles are represented by black outlines. The red dashed line indicates $\mathbf{dp}_n$, while the blue dashed line represents $\mathbf{dp}_n^{space}$. Obstacle points in $\mathbf{op}_n$ are marked as purple circles.	19

2.9	Illustration of the creation of individuals and the three possible situations for the individuals with $N = 60$, $R^{radius} = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, $R^{mh} = 1m$, $K = 10$, and $G = 1$. In Situations 1 and 2, the individuals are not penalized because they are located in a valid region for the robot. However, the individuals in Situation 3 are located outside the polygon $\mathbf{dp}_n^{space}$ and are therefore penalized. In this illustration, the robot is represented by a blue square. The polygon $\mathbf{dp}_n$ is represented by a red dashed line. The polygon $\mathbf{dp}_n^{space}$ is represented by a blue dashed line. The generation radius R^{mh} is illustrated by a yellow dashed line. The obstacle is represented by a rectangle with a black outline.	21
2.10	Illustration of the evaluation of individuals using the fitness function with $N = 60$, $R = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, $R^{mh} = 1m$, $K = 10$, $\varepsilon = 0.05m$. In the illustration, $\mathbf{dp}_n$ is shown as a red dashed line, while $\mathbf{dp}_n^{space}$ is represented by a blue dashed line. The yellow dashed line indicates the circle with radius R^{mh} from the constraint equation. The robot's position, sp^R , is marked by a blue square, while the individuals are represented by the symbol $\times$. Obstacles are depicted as black outlined objects. The blue and red lines represent the attractive and repulsive forces of the fittest individual in each generation, respectively.	24
2.11	3D plot of the fitness function with $\Phi = 3m$, $R^{mh} = 1m$, $N = 3$, and $\tau = 1000$ in an obstacle-filled environment. In this illustration, obstacles are depicted as black outlined objects on the x,y plane. $\mathbf{dp}_n$ is illustrated in blue. The penalty factor $A(n,k,g)$ is represented as a green region. The goal point is marked by a green circle on the x,y plane.	25
2.12	Example of MHRTSN movement in an obstacle-filled environment with $N = 60$, $R^{radius} = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, and $\varepsilon = 0.05m$. In this illustration, the polygon $\mathbf{dp}_N(P)$ is represented by a red dashed line, the polygon $\mathbf{dp}_N^{space}(P)$ by a blue dashed line, and sp_R is represented by a blue square. The robot is illustrated as a blue circle. The global target point, sp_G , is represented by a green circle. Obstacles are depicted as black outlined objects.	26
3.1	Flow diagram of the steps associated with the simulation platform, illustrating the interaction between the steps of Algorithm 1 and the Webots and MATLAB platforms.	28
3.2	Flow diagram illustrating the relationships between the simulation platform and the orchestrator program.	30
3.3	Example of navigation in an environment filled with obstacles.	31
4.1	Representation of the static environments used in the simulations with MHRTSN-GA and MHRTSN-PSO. The robot is indicated by a blue circle at the initial position, while the final navigation point is shown by a black circle. Obstacles are represented by brown and white boxes. In these environments, the floor is marked with 1 by 1 meter squares in a checkerboard pattern.	35

4.2	Mean and Standard Deviation of Indicator <i>I1</i> in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.	36
4.3	Mean and Standard Deviation of Indicator <i>I2</i> in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.	36
4.4	Mean processing time, Indicator <i>I3</i> , in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.	38
4.5	Mean and Standard Deviation of <i>I1</i> in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.	40
4.6	Mean and Standard Deviation of <i>I2</i> in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.	40
4.7	Mean processing time, indicator <i>I3</i> , in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.	41
4.8	Mean and Standard Deviation of <i>I1</i> in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.	42
4.9	Mean and Standard Deviation of <i>I2</i> in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.	42
4.10	Mean processing time, indicator <i>I3</i> , in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.	43
4.11	Mean and Standard Deviation of Indicator <i>I1</i> in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.	44
4.12	Mean and Standard Deviation of Indicator <i>I2</i> in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.	44
4.13	Mean Clock Time and CPU Time per Displacement Event, Indicator <i>I3</i> , in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.	45
4.14	Representation of the dynamic environments used in the simulations with MHRTSN-GA and MHRTSN-PSO. The robot is indicated by a blue circle at the initial position, while the final navigation point is shown by a black circle. Static obstacles are represented by brown and white boxes, and dynamic obstacles are shown as red boxes. In these environments, the floor is marked with 1 by 1 meter squares in a checkerboard pattern.	46
4.15	Mean and Standard Deviation of Indicator <i>I1</i> in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.	48
4.16	Mean and Standard Deviation of Indicator <i>I2</i> in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.	48
4.17	Mean Clock Time and CPU Time per Displacement Event, Indicator <i>I3</i> , in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.	49
4.18	Mean and Standard Deviation of Indicator <i>I1</i> in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.	49
4.19	Mean and Standard Deviation of Indicator <i>I2</i> in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.	50
4.20	Mean Clock Time and CPU Time per Displacement Event, Indicator <i>I3</i> , in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.	50

4.21	Comparison of different paths in the environment proposed in (Yan et al. 2022). The trajectories are indicated by the following colors: ACO (green), SSA (red), ACO+GA (purple), MSSA (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the bottom-left corner, while the endpoint is located in the top-right corner.	52
4.22	Comparison of different paths in the environment proposed in (Zheng et al. 2023). The trajectories are indicated by the following colors: rPSO (green), apfrPSO (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the far-left point, while the endpoint is located in the top-right corner.	52
4.23	Comparison of different paths in the environment proposed in (Silva et al. 2018). The trajectories are indicated by the following colors: DPNA-GA (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the top-left corner, while the endpoint is located in the bottom-right corner.	53

List of Tables

1.1	Categories and characteristics of Path Planning Algorithms.	2
2.1	Detailed explanation of Algorithm 1 steps.	11
2.2	Summary of symbols used in the paper.	12
3.1	Number of samples according to statistical parameters.	31
3.2	Common parameters used in MHRTSN simulations.	32
3.3	Specific parameters used in MHRTSN-GA simulations.	32
3.4	Specific parameters used in PSO simulations.	33
4.1	Initial and final points for each static environment.	35
4.2	Proposed parameter configuration for simulations in the static environments using MHRTSN-GA and MHRTSN-PSO techniques.	37
4.3	Simulation results for Env1: Clock times, CPU times, Displacements, and Distances.	38
4.4	Statistical test results for indicators I_1 and I_2 in Env1 for Groups G_1 and G_2	39
4.5	Simulation results for Env2: Clock times, CPU times, Displacements, and Distances.	39
4.6	Statistical test results for indicators I_1 and I_2 in Env2 for Groups G_3 and G_4	41
4.7	Simulation results for Env3: Clock times, CPU times, Displacements, and Distances.	42
4.8	Statistical test results for indicators I_1 and I_2 in Env3 for Groups G_5 and G_6	43
4.9	Simulation results for Env4: Clock times, CPU times, Displacements, and Distances.	44
4.10	Statistical test results for indicators I_1 and I_2 in Env4 for Groups G_7 and G_8	45
4.11	Initial and final points for each dynamic environment.	46
4.12	Dimensions, locations, and times for adding boxes in Env5.	47
4.13	Dimensions, locations, and times for adding boxes in Env6.	47
4.14	Proposed parameter configuration for simulations in the dynamic environments using MHRTSN-GA and MHRTSN-PSO techniques.	47
4.15	Simulation results for Env5: Clock times, CPU times, Displacements, and Distances.	48
4.16	MWU test results for indicators I_1 and I_2 in Env5 for Group G_9	49
4.17	Simulation results for Env6: Clock times, CPU times, Displacements, and Distances.	49
4.18	Statistical test results for indicators I_1 and I_2 in Env6 for Group G_{10}	51

4.19 Environments used for comparisons with other techniques from the literature. 51

4.20 Results of the simulations of environments. 53

4.21 Comparison of navigation techniques across different environments. . . . 54

Chapter 1

Introduction

In recent decades, robotic applications have expanded into sectors such as industry, agriculture, surveillance, environmental monitoring, search and rescue, playing an increasingly significant role in society. With the advancement of technology, mobile robots are employed in various environments, including aerial, terrestrial, and aquatic, performing a wide range of tasks and proving indispensable in complex operations (Lin et al. 2022), so that the development of these applications is directly related to an increase in productivity and a decrease in the risk factor in performing tasks. The practicality of using robotic applications is due to the autonomous capability of some systems, which are able to understand the environment around them and make intelligent decisions to achieve their objectives (Qin et al. 2023). The autonomy of mobile robots depends on several integrated stages, including data extraction by sensors, environment modeling and representation, movement planning, and a control system that regulates these movements. These stages allow an autonomous mobile robotic system to map, construct, and follow a viable path between initial and final points, optimizing criteria such as distance, energy, safety, and time. In this context, the path planning stage is responsible for finding a viable route avoiding obstacles, in a timely processing time, that respects the characteristics of the robotic project (Lamini et al. 2018).

However, path planning presents several challenges, such as constraints in environment representation, computational power limitations, and sensor sampling rate restrictions, among others. These challenges become even more pronounced in complex and dynamic environments. Consequently, the path generation problem, classified as NP-hard, demands novel approaches and significant computational resources to be effectively addressed (Zhu et al. 2015, Xu et al. 2023). Various strategies have been proposed in the literature, typically categorized based on their underlying methodology, algorithmic characteristics, and sources of inspiration. Approaches within the same category share fundamental traits that offer both advantages and drawbacks, as shown in Table 1.1 (Lin et al. 2022, Sánchez-Ibáñez et al. 2021, Xu et al. 2023, Ait Saadi et al. 2022). Different types of environments or specific constraints often require tailored or adapted path planning techniques. Machine learning-based and evolutionary algorithm-based methods have demonstrated strong adaptability in dynamic and uncertain environments, while structured and static scenarios can be more effectively handled using classical techniques. Systems with stringent computational constraints benefit from the low processing demands of reactive approaches (Sánchez-Ibáñez et al. 2021). Hybrid solutions that inte-

grate multiple techniques are also widely explored, as they aim to mitigate the limitations of individual methods and enhance overall performance (Chakraborty et al. 2022).

Table 1.1: Categories and characteristics of Path Planning Algorithms.

Category	Characteristics	Advantages	Disadvantages	Examples
Reactive Computing (Sánchez-Ibáñez et al. 2021)	Real-time reaction to local obstacles	Quick adaptation to new obstacles	Prone to local minima	Artificial Potential Fields (APF), Bug Algorithms
	Local sensor information	Simple to implement	Does not guarantee global optimality	
Artificial Intelligence (Lin et al. 2022, Sánchez-Ibáñez et al. 2021, Xu et al. 2023, Ait Saadi et al. 2022)	Uses machine learning, evolutionary algorithms, or fuzzy logic	Suitable for dynamic and complex environments	High computational resource demand	Neural Networks, Genetic Algorithms (GA), Particle Swarm Optimization (PSO)
	Nature-inspired techniques	Handles complex problems with many degrees of freedom	Difficult to tune and train	
Classical Methods (Lin et al. 2022, Sánchez-Ibáñez et al. 2021, Xu et al. 2023, Ait Saadi et al. 2022)	Graph-based and sampling algorithms	Well-established and studied solutions	Can be computationally intensive	A*, Dijkstra, Rapidly-exploring Random Trees (RRT), Probabilistic Roadmaps (PRM)
	Optimized path search	Good for global planning	Generally require prior knowledge of the environment	

Table 1.1 summarizes the main categories of path planning approaches, highlighting their characteristics, advantages, and limitations. Reactive computing methods, such as APF and Bug Algorithms, excel in real-time obstacle avoidance but may struggle with local minima, making them less suitable for global navigation in complex environments. Artificial intelligence-based approaches, including Neural Networks, GA, and PSO, offer high adaptability to dynamic scenarios but require significant computational resources and careful parameter tuning. Classical methods, such as A*, Dijkstra, and sampling-based techniques like RRT and PRM, provide well-established solutions that ensure global path optimization but can be computationally demanding and often require prior knowledge of the environment.

The choice of an appropriate path planning strategy depends on the trade-offs between computational efficiency, adaptability, and the complexity of the operational scenario. For instance, in structured and well-defined environments, classical algorithms can efficiently

determine optimal paths, whereas in highly dynamic and uncertain environments, AI-based approaches provide a more flexible solution. In contrast, reactive computing methods are advantageous in real-time applications with strict processing constraints, where immediate responses are more critical than global optimality. Understanding these trade-offs is essential for selecting the most appropriate navigation strategy for autonomous robotic systems.

Given the challenges outlined above, this work proposes a novel hybrid approach for real-time navigation in unknown environments. The Meta-Heuristic Real-Time Safe Navigation (MHRTSN) strategy integrates the adaptability of population-based meta-heuristics with the local responsiveness of artificial potential fields to generate safe and efficient trajectories in static and dynamic environments. Unlike conventional methods, MHRTSN does not require prior knowledge of the environment and dynamically adjusts its trajectory based on sensor data. By leveraging meta-heuristic search mechanisms, MHRTSN aims to mitigate issues such as local minima and oscillatory navigation, ensuring that the planned trajectory remains safe and efficient.

The following sections of this document will present the state-of-the-art approaches in more detail, discuss their limitations, and further elaborate on the proposed methodology and contributions of this work.

1.1 State of the art

Several works in the literature utilize solutions based on Classical Methods (Bulut 2023, Qi et al. 2021, Vasconcelos et al. 2020). These solutions combine offline and online processing to deal with unmapped obstacles, allowing them to work in both static and dynamic environments by adjusting the trajectory in real time. However, these approaches require prior knowledge of the environment, modeled by a map. For example, the work in (Bulut 2023) models obstacles using geometric shapes and applies analytical geometry to detect collisions and calculate free paths, employing cubic Bézier curves with three parameters to adjust the path smoothly and optimally. The work in (Qi et al. 2021), on the other hand, presents the multiobjective dynamic rapidly exploring random (MOD-RRT*) strategy, which combines offline planning based on an Optimal Rapidly-exploring Random Tree (RRT*) tree with an online path re-planner that selects the best alternative node when obstacles block the previously planned path, modeling the environment as a configuration space updated in real time by sensors. Finally, the Lifelong Planning A* (LPA*) algorithm (Vasconcelos et al. 2020) guides a mobile robot in a grid-discretized environment, integrating the LPA* algorithm with Dijkstra's algorithm, enabling adaptive trajectory planning.

Furthermore, the literature also includes works that use a reactive computing approach to solve the navigation problem in robotic systems (Roy et al. 2021, Sivaranjani et al. 2021). These solutions are based on sensor data captured from the environment, adjusting the trajectory in real time. However, the effectiveness of this approach is negatively impacted as the quality of the data collected by the sensors decreases. The work in (Roy et al. 2021) presents a navigation and control methodology based on geometric regions to guide a swarm of robots in occluded regions to the target point. In this study, a

hierarchical approach is adopted, where a controller robot detects obstacles and adjusts a virtual region for the other robots to follow, adapting this region according to the environment's configuration. Additionally, (Sivaranjani et al. 2021) implements and compares the Bug0, Bug1, and Bug2 algorithms, evaluating their effectiveness in terms of path cost and computational time. In this solution, the environment is described as a grid-shaped map, where obstacles are detected as they enter the robot's perception field, and path processing occurs online, being effective in static environments. However, these algorithms can become computationally expensive in complex environments.

Several studies in the literature present solutions based on artificial intelligence algorithms (Josef & Degani 2020, Yu et al. 2020, Chen et al. 2020, Samadi Gharajeh & Jond 2022, Yan & Li 2016, Teli & Wani 2021, Pattnaik, S. K. et al. 2022, Yan et al. 2022, Zheng et al. 2023, Wang et al. 2022, Cardoza Plata et al. 2022, Shi et al. 2023, Montiel et al. 2015, Silva et al. 2018). Among these, some present hybrid solutions that combine one or more approaches to achieve better results. By categorizing these works concerning the most influential approach, it is observed that artificial neural networks are frequently used (Josef & Degani 2020, Yu et al. 2020, Chen et al. 2020) to solve the navigation problem in static and dynamic environments. Specifically, the work (Josef & Degani 2020) proposes a hybrid Deep Reinforcement Learning approach with LSTM layers, optimized for perception and navigation in rugged terrain. This approach operates without prior knowledge of the environment, which is represented as a continuous elevation map, with perception limited to the sensor's range. Another approach combines artificial neural networks with hierarchical reinforcement learning (Yu et al. 2020) to plan the path in an unknown environment. Sensors capture information about the environment, represented by a grid-shaped map, and the reward function is designed to avoid obstacles and follow a smooth and efficient path. However, the approaches in (Josef & Degani 2020, Yu et al. 2020) may become impractical in real-world applications due to the high computational power required for the trial-and-error processes. Lastly, the work in (Chen et al. 2020) presents the Padding Mean Neural Dynamic Model for online trajectory planning in robotic systems, utilizing a topologically organized network so that the propagation of neural stimuli benefits path selection. In this solution, the environment is perceived by sensors providing the spatial position of obstacles, allowing the algorithm to build a discrete representation of the environment. Although the solutions in (Josef & Degani 2020, Yu et al. 2020, Chen et al. 2020) work in dynamic environments, they require high computational power, which may limit the models' adaptation rate to sudden changes in the environment.

Among the mentioned works, some present solutions based on Fuzzy systems (Samadi Gharajeh & Jond 2022, Yan & Li 2016, Teli & Wani 2021) for unknown dynamic and static environments. In these approaches, the environment is sampled by sensors that inform the positions of obstacles to the algorithm in real time, with online processing. In (Samadi Gharajeh & Jond 2022), navigation for robots with hardware limitations is proposed, based on an adaptive neuro-fuzzy inference system. A set of ultrasonic sensors collects data from the environment, allowing the robot to move based on a function weighted by the distance between the robot, the obstacles, and the target. Another solution uses a hybrid strategy combining a sensor data filtering technique with a fuzzy system

to improve movement smoothness (Yan & Li 2016). In this approach, the sensor-sampled environment is represented by a set of points processed online to define the path. The work in (Teli & Wani 2021) presents a hybrid strategy combining potential fields with fuzzy logic to enhance navigation, avoiding local minima. The algorithm monitors the robot's speed and traveled distance to identify local minima situations and assigns a virtual map to the region, calculating an intermediate goal to help repel the robot out of the local minimum. However, the solutions presented in (Samadi Gharajeh & Jond 2022, Yan & Li 2016, Teli & Wani 2021) may face challenges if the fuzzy rule system is not well-tuned to the environment's possibilities.

Several works (Pattnaik, S. K. et al. 2022, Yan et al. 2022, Zheng et al. 2023, Wang et al. 2022, Cardoza Plata et al. 2022, Shi et al. 2023, Montiel et al. 2015, Silva et al. 2018) utilize meta-heuristics to plan paths in environments with obstacles. Among them, the solutions proposed in (Yan et al. 2022, Zheng et al. 2023, Cardoza Plata et al. 2022) work exclusively in static environments. In the works (Yan et al. 2022, Zheng et al. 2023), the planning is global and requires prior knowledge of the environment, which is modeled as a grid. In (Yan et al. 2022), the Sparrow Search algorithm is modified with an opposition-based learning approach to improve the quality of the initial population, increasing search precision and accelerating convergence. In (Zheng et al. 2023), the combination of the PSO algorithm with artificial potential fields allows for dynamic adjustment of the inertia weight and the classification of particle position vectors, guiding the robot to the goal while avoiding obstacles. Initially, this work introduces the Ranking-based Particle Swarm Optimization (rPSO), a modified version of PSO that dynamically adjusts the inertia weight to balance local and global exploration. Subsequently, the work presents the Artificial Potential Field-based PSO (apfrPSO), which combines rPSO with potential fields to plan paths while avoiding obstacles. The solution presented in (Cardoza Plata et al. 2022) differs in that it does not require prior knowledge of the environment. In this case, data are collected as the robot moves, configuring an online planning process, calculated by a variation of the GA, the micro-genetic algorithm, adapted to operate with a reduced population, where each individual is a potential solution to the problem.

Conversely, the works (Pattnaik, S. K. et al. 2022, Wang et al. 2022, Shi et al. 2023, Montiel et al. 2015, Silva et al. 2018) can operate in both static and dynamic environments. In the solutions presented in (Wang et al. 2022, Shi et al. 2023), a hybrid planning approach is applied, combining initial offline planning with local adjustments that allow the route to be adapted to possible changes in the environment. In these approaches, prior knowledge of the environment, discretized into a grid, is required to define the global route in the offline phase. In (Wang et al. 2022), the Ant Colony Optimization (ACO) algorithm is employed to calculate the global trajectory, while the Dynamic Window Approach method adjusts the trajectory locally. In (Shi et al. 2023), a dynamic path planner for mobile robots, based on a modified version of the Simulated Annealing algorithm, is proposed. The modification includes an initial path selection method and a deletion operation, aiming to reduce the algorithm's computational effort.

The solutions described in (Pattnaik, S. K. et al. 2022, Montiel et al. 2015, Silva et al. 2018) operate differently, as they do not require prior knowledge of the environment. In these works, processing occurs locally as sensors collect data from the environment. In

(Pattnaik, S. K. et al. 2022), the Hybrid Particle Swarm and Chemical Reaction Optimization (HPCRO) algorithm is presented, designed to navigate unknown environments with circular and polygonal obstacles, maintaining a smooth trajectory and ensuring a safe distance from obstacles while the robot moves toward the target. In (Montiel et al. 2015), the Bacterial Potential Field method is proposed, which combines the artificial potential field method with the Bacterial Evolutionary Algorithm (BEA), optimized to avoid local minima. In this approach, attractive and repulsive gains are adjusted by the BEA to efficiently calculate the robot's trajectory. In (de Oliveira & Fernandes 2016, Silva et al. 2018), the dynamic navigation algorithm called Dynamic Planning Navigation Algorithm optimized with Genetic Algorithm (DPNA-GA) is tested across various environments and parameters. This algorithm recalculates the robot's movement in real time based on new local goals, considering the current position and obstacles detected by distance sensors. In DPNA-GA, each individual generated by the GA represents a candidate position (x, y) for the robot's next step, with these individuals being evaluated and selected according to the evaluation function, resulting in iterative movement.

In the works (Zheng et al. 2023, Montiel et al. 2015, de Oliveira & Fernandes 2016, Silva et al. 2018), the individuals share a common characteristic: the amount of stored information remains constant, regardless of the environment's complexity. On the other hand, in the works (Pattnaik, S. K. et al. 2022, Yan et al. 2022, Wang et al. 2022, Cardoza Plata et al. 2022, Shi et al. 2023), each individual represents a sequence of coordinates that describes the robot's entire trajectory, from the start point to the end. Thus, in more complex environments, the sequence of steps becomes longer, resulting in an increase in the amount of processed information.

Thus, this work is set in the context of path planning and robotic navigation, proposing a hybrid approach for simultaneous planning and navigation in unknown static and dynamic environments. The strategy, named MetaHeuristic Real-Time Safe Navigation (MHRTSN), combines artificial potential fields with metaheuristic evaluation, utilizing the strengths and addressing the limitations of each technique. In contrast to traditional approaches, MHRTSN quickly adapts to new obstacles, handles navigation oscillations, and overcomes local minima that would affect a purely reactive technique. Additionally, MHRTSN generates trajectories and performs navigation considering the robot's constraints and dimensions, using a population-based metaheuristic evaluation. The approach uses different types of sensors to map the environment, generating samples that, combined with other data, are evaluated by a function. This function analyzes the individuals in the meta-heuristic population, where each individual represents points (x, y) as possible solutions to the navigation problem. This method ensures that, even as the environment becomes more complex, the process of generating and evaluating individuals remains efficient and consistent, allowing the system to handle more challenging scenarios.

1.2 Objectives and contributions

Given these challenges, this research proposes a novel hybrid navigation strategy to address the limitations of existing solutions, particularly in unknown and dynamic environments. Thus, this research aims to develop and validate the MHRTSN strategy,

designed to address challenges in path planning and navigation in unknown static and dynamic environments. In contrast to traditional approaches, MHRTSN quickly adapts to new obstacles, handles navigation oscillations, and overcomes local minima that would affect a purely reactive technique. The tests showed that MHRTSN, unlike the approaches found in the literature (Yan et al. 2022, Zheng et al. 2023, Silva et al. 2018), generated a safe path, avoiding obstacles and reaching the goal. The path taken by MHRTSN was very close to those of the other approaches, however, generating a path that respects the safety and kinematic constraints of the robot. Tests with the MHRTSN-GA and MHRTSN-PSO versions also demonstrated the viability of the technique with different meta-heuristic approaches. These tests also showed a superiority related to processing time (on average 31.85%) and generation of longer displacements (on average 21.90%) of MHRTSN-PSO compared to MHRTSN-GA. The cognitive capacity of the swarm, which generated updates in individuals through the social factor, allowed for greater efficiency in selecting target points. Thus, the MHRTSN dynamic navigation strategy aims to be an effective tool for autonomous robotic systems in complex environments.

The expected contributions of this work include:

- The development and implementation of the MHRTSN strategy, which combines potential fields and population-based meta-heuristics, optimized for unknown static and dynamic environments.
- Presentation of the results of the MHRTSN technique in unknown static and dynamic environments, using the GA (MHRTSN-GA) and PSO (MHRTSN-PSO) meta-heuristics as evaluators, as well as the comparison of these solutions with those present in the literature.
- The design and deployment of an optimized simulation platform for data generation and performance evaluation of robotic navigation strategies.

1.3 Organization of the work

This document is structured as follows: Section 1 introduces and contextualizes the challenges associated with robotic navigation. Additionally, this section presents a literature review highlighting key solutions to the path generation problem proposed in recent years. Section 2 details the MHRTSN strategy and its theoretical foundations, explaining how it integrates these techniques to address the identified challenges. Section 3 describes the developed simulation platform and the methodologies employed for testing and evaluation. Section 4 presents the results obtained from the experiments and discusses their implications. Finally, Section 5 provides the conclusions of this study and suggests directions for future research.

1.4 Published Works

1. BALZA, Micael; FERNANDES, Marcelo A. C. “Proposta de Planejamento Dinâmico de Navegação para Robôs Móveis Terrestres baseada em Otimização por Enxame de

Partículas”. Anais do XX Congresso Brasileiro de Automática, SBAI-SBSE, 2023. DOI: <https://doi.org/10.20906/SBAI-SBSE-2023/4014>.

2. BALZA, M. ; GOLDBARG, M. A. ; SILVA, S. N. ; SILVA, L. M. D. ; FERNANDES, M. A. C. . "A Real-Time Safe Navigation Proposal for Mobile Robots in Unknown Environments Using Meta-Heuristics". IEEE Access, 2025. DOI: 10.1109/ACCESS.2025.3536081.

3. JOSÉ C. V. S. JUNIOR ; BALZA, M. ; SILVA, S. N. ; SILVA, L. M. D. ; FERNANDES, M. A. C. . "Optimizing Tactile Feedback Devices with Fixed-Point Computation on FPGA". Circuits Systems And Signal Processing, 2025. DOI: <https://doi.org/10.1007/s00034-025-02994-1>

Chapter 2

MHRTSN Strategy

This section describes the MHRTSN navigation strategy proposed in this work for autonomous robotic systems. Understanding the navigation process and the steps involved in this approach is essential for interpreting the results. The MHRTSN strategy uses a metaheuristic method to evaluate an objective function that assigns a value to each spatial point in the environment. This function incorporates terms representing potential fields, penalty factors, and optimization factors, aiming to guide the robot toward the target point by minimizing the function. The proposed approach builds upon the strategies outlined in (de Oliveira & Fernandes 2016, Silva et al. 2018).

The MHRTSN strategy divides the navigation process into N displacements. At each n -th displacement, the environment is sampled by P points uniformly distributed around the robot. In this approach, prior knowledge of the environment is not required, as the environment data is collected during the navigation process. The MHRTSN strategy requires three types of spatial points (x, y) . The first is related to the robot's spatial position during the n -th displacement, defined as $sp_n^R = (x_n^R, y_n^R)$. The second is related to the location of the target point, expressed as $sp^G = (x^G, y^G)$. The third is related to the points sampled from the environment. Thus, each p -th point sampled from the environment is expressed as $sp^{OP} = (x^{OP}, y^{OP})$.

At each n -th displacement in the MHRTSN, a new value of sp_n^R is stored in the vector that keeps track of the robot's previously visited positions. This vector is expressed as

$$\mathbf{pv}^R = \begin{bmatrix} sp_0^R \\ \vdots \\ sp_n^R \\ \vdots \\ sp_{N-1}^R \end{bmatrix} = \begin{bmatrix} (x_0^R, y_0^R) \\ \vdots \\ (x_n^R, y_n^R) \\ \vdots \\ (x_{N-1}^R, y_{N-1}^R) \end{bmatrix}. \quad (2.1)$$

The record of the robot's previously visited positions is later consulted by the MHRTSN strategy to increase the probability of selecting non-visited areas, thereby improving navigation efficiency. $\mathbf{pv}^R$ is the only data generated from past displacements that is used in the MHRTSN strategy, all other data utilized comes from the current displacement. Thus, the MHRTSN strategy processes one portion of the environment at a time, denoted as a displacement.

As mentioned, the MHRTSN divides the navigation process into N displacements, making navigation occur iteratively. The transition from the initial point to the final goal, sp^G , is carried out through a path composed of these N displacements. At each n -th displacement, a local target point is defined, expressed as $sp_n^{LG} = (x_n^{LG}, y_n^{LG})$, where x_n^{LG} and y_n^{LG} represent the x and y components of the point, respectively.

Thus, MHRTSN navigation occurs through the movement of the robot across the generated sp_n^{LG} points. The MHRTSN iterates until the robot's position (sp_n^R) is within a tolerance radius specified by $sp_n^R = (sp^G \pm \epsilon)$, where ϵ is the tolerance factor that creates a tolerance area around the target point. Alternatively, the algorithm stops when the maximum number of iterations, defined as N_{Max} , is reached. The MHRTSN strategy can be divided into four main sub-steps: data collection, preprocessing, metaheuristic evaluation, and displacement. These steps are presented in Algorithm 1 and are illustrated in Figure 2.1. Each step is described in detail in the following sections.

Algoritmo 1: MHRTSN

```

1:  $n = 0$ 
2:  $sp_n^R = [sp_0^R]$ 
3: while  $sp_n^R \neq (sp^{LG} \pm \epsilon)$  and  $n < N_{Max}$  do
4:    $sp_n^R, dp_n = \text{DataCollection}$ 
5:    $op_n, dp_n^{space} = \text{PreProcessing}(dp_n)$ 
6:    $sp_n^{LG} = \text{MetaHeuristicEvaluation}(sp_n^R, op_n, dp_n^{space}, sp^G)$ 
7:    $sp_{n+1}^R = \text{Displacement}(sp_n^{LG})$ 
8:    $pv^R = \begin{bmatrix} pv^R \\ sp_{n+1}^R \end{bmatrix}$ 
9:    $n = n + 1$ 
10: end while

```

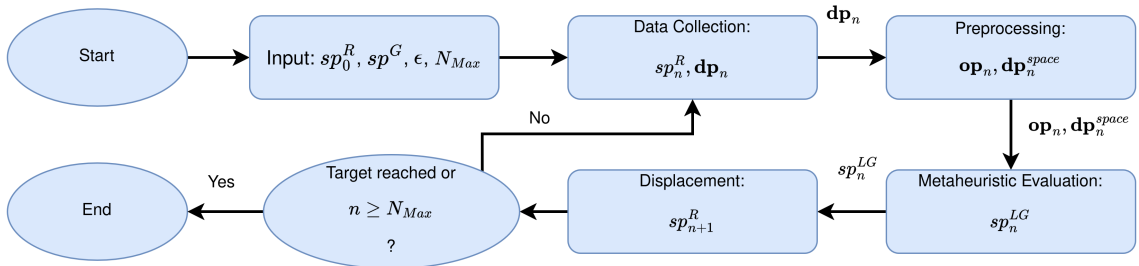


Figure 2.1: Flowchart illustrating the MHRTSN algorithm's input-output relationships and processing steps.

Table 2.1 provides a detailed summary of the steps of Algorithm 1, highlighting each algorithm line with its respective title and a comprehensive description. This table aims to clarify the main stages of the algorithm's operation, including variable initialization, stopping conditions, sensory data processing, application of metaheuristics, and updating information related to the robot's displacement. Each line is organized to offer a struc-

tured and educational view, facilitating the understanding of the algorithm's flow and the operations performed at each step.

Table 2.1: Detailed explanation of Algorithm 1 steps.

Line	Title	Description
1	Counter initialization	The displacement counter n is initialized to 0.
2	Initial position definition	The robot's initial position sp_0^R is stored as the first element of the position history sp_n^R .
3	Stopping condition	The algorithm checks whether the robot is within the tolerance area around the global target sp^G or if the maximum number of iterations N_{Max} has been reached. If neither condition is met, the algorithm continues.
4	Data collection	The robot scans its surroundings, gathering sensor data about its current position sp_n^R and the environment $\mathbf{dp}_n$. These data describe the environment's configuration at the n -th displacement (see subsection 2.1).
5	Preprocessing	The collected data $\mathbf{dp}_n$ is processed to identify obstacles, generating $\mathbf{op}_n$, and to create the navigable space polygon ($\mathbf{dp}_n^{space}$). This step allows the MHRTSN algorithm to identify suitable areas for navigation (see subsection 2.2).
6	Metaheuristic evaluation	The metaheuristic evaluates multiple candidate solutions to determine the best local target point sp_n^{LG} . This evaluation considers the robot's current position sp_n^R , obstacle information $\mathbf{op}_n$, the navigable polygon $\mathbf{dp}_n^{space}$, and the global target sp^G (see subsection 2.3).
7	Displacement	The robot moves towards the local target sp_n^{LG} generated in the previous step. After displacement, the robot's position is updated to sp_{n+1}^R (see subsection 2.4).
8	Position history update	The robot's new position sp_{n+1}^R is added to the trajectory history $\mathbf{pv}^R$, which records the path taken.
9	Counter increment	The displacement counter n is incremented by 1.

Table 2.2: Summary of symbols used in the paper.

Symbol	Explanation
n	n -th displacement.
sp_n^R	Robot's spatial position during the n -th displacement.
sp^{LG}	Local target point in the environment.
sp^G	Global target point in the environment.
sp^{OP}	Sampled points from the environment during navigation.
dp_n	Distance polygon representing free space around the robot.
dp_n^{space}	Polygon representing navigable space.
P	Number of samples taken from the environment.
Φ	Maximum sensor range.
ε	Tolerance factor for target approximation.
δ	Distance between the robot's center and the sensor.
α	Angle between the robot's reference and the sensor.
β	Angle between the robot's and the environment's reference.
R	Robot radius or minimum turning radius.
τ	Penalty factor for revisiting areas.
$F_{ed}^G(k, g)$	Euclidean distance to the goal point.
$F_{ed}^{OP}(n, p)$	Repulsive force from obstacles.
ρ	Adjustment factor for obstacle presence.
pv^R	History of the robot's visited positions.
$mhn(k, g)$	Position of the k -th individual in the g -th generation.
C	Path connecting two points.
$f_{res}(C)$	Feasibility function of a path C .
θ	Angular resolution for environment sampling.
$\mathbf{v}_{n,p}$	Unit orthogonal normal vector for displacement.
$f_{mr}(n, k, g)$	Function verifying valid movement region.
op_n	Points associated with obstacles during the n -th displacement.
$m_n(p)$	Slope of the line formed by two consecutive points.
$b_n(p)$	Intercept of the line formed by two consecutive points.
$f_n(k, g)$	Fitness function evaluating the k -th individual in the g -th generation.
$F_{ed}(A, B)$	Euclidean distance between two points A and B .
$B(n, k, g, f_{mr}(n, k, g))$	Penalty term based on path validity and location.
$A(n, k, g)$	Penalty factor for revisiting explored areas.
R^{mh}	Radius used to generate the initial population in the metaheuristic.
$D_n(k, g)$	Euclidean distance between an individual and the robot's history.
x, y	Cartesian coordinates of points.

2.1 Data Collection

This stage is responsible for collecting and updating the environmental data (line 4 of Algorithm 1). The process begins with the update of the robot's spatial position, sp_n^R , and its storage in the vector $\mathbf{p}^R$ (see Equation 2.1). Next, the environment around the robot is scanned, generating P equally spaced samples by converting the distance returned by the sensor, defined by ϕ , into spatial points, sp^{OP} . This conversion is expressed as

$$sp^{OP} = (x^{OP}, y^{OP}) = (x^R + (\delta + \phi) \cos(\alpha + \beta), y^R + (\delta + \phi) \sin(\alpha + \beta)), \quad (2.2)$$

where ϕ ranges from the shortest detected distance, $\phi = 0$, to the sensor's maximum range, defined as Φ . When no obstacles are detected, the sensor returns the maximum value Φ , which is then converted into Cartesian coordinates. The terms x^{OP} and y^{OP} represent the spatial components of the point, respectively. The terms x^R and y^R represent the spatial components of the robot's current location. The terms α and β represent the angle between the robot's reference and the sensor, and the angle between the environment's reference and the robot's reference, respectively, both expressed in radians. The distance between the robot's center and the distance sensor is denoted by δ . The terms representing distances, ϕ and δ , are expressed in meters. Figure 2.2 illustrates in detail each term defined in Equation 2.2.

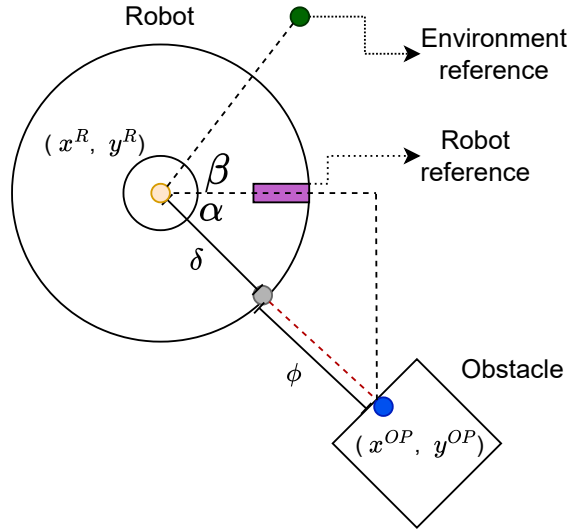


Figure 2.2: Illustration of the conversion of distances into spatial points based on the robot's current location and angle.

In Figure 2.2, the robot is represented by a circle, with the spatial localization device sp^R located at its center, denoted by a smaller circle within the robot's boundary (yellow). The distance sensor, illustrated as a small circle offset by δ meters from the robot's center, measures a distance of ϕ meters to the obstacle (gray). The robot's reference is depicted by a rectangle positioned on the right side of the robot's edge (purple). In the illustration, the distance sensor is offset by α degrees relative to the robot's reference, while the robot

itself is rotated by β degrees with respect to the environmental reference. Based on this configuration, the measured distance is converted into points sp^{OP} using Equation 2.2.

In the RTSMHN algorithm, at every n -th displacement, the environment is sampled by P points sp^{OP} , uniformly distributed around the robot. Thus, the environment's resolution can be defined as

$$\theta = \frac{360^\circ}{P}. \quad (2.3)$$

Which means that a new environment sample is captured every θ degrees. By adjusting the value of P , it is possible to balance memory constraints with the resolution required for the robotic application.

The P environment samples captured at each n -th displacement, equally spaced by θ degrees, are allocated in an n -th vector representing the free space around the robot. This vector is described as

$$\mathbf{dp}_n = \begin{bmatrix} sp_n^{OP}(0) \\ \vdots \\ sp_n^{OP}(p) \\ \vdots \\ sp_n^{OP}(P-1) \end{bmatrix} = \begin{bmatrix} (x_n^{OP}(0), y_n^{OP}(0)) \\ \vdots \\ (x_n^{OP}(p), y_n^{OP}(p)) \\ \vdots \\ (x_n^{OP}(P-1), y_n^{OP}(P-1)) \end{bmatrix}, \quad (2.4)$$

where the term $sp_n^{OP}(p)$ represents the value returned by the distance sensor for the p -th point during the n -th displacement, converted into spatial points (see Equation 2.2). In other words, the vector $\mathbf{dp}_n$ is composed of points that form a polygon representing the free space around the robot. Figure 2.3 illustrates the environment scanning process and its representation in $\mathbf{dp}_n$. In this image, $\mathbf{dp}_n$ is represented by a set of red points, where each point corresponds to a sample $sp_n^{OP}(p)$ collected from the environment. The set of these P points generates a polygon that represents the free space around the robot. The robot is represented in the illustration by a circle with a square at its center. Although the sampled points from the environment form a polygon representing the space around the robot, the MHRTSN strategy employs a point-based representation.

The MHTRSN strategy is compatible with various types of distance sensors, either directional or with 360° coverage, without depending on specific technology. Sensor selection can be based on criteria such as cost, technology, and accuracy, adapting to the specific needs of each project. For sensors with 360° coverage, the method of converting obstacle distances into Cartesian points can be directly applied to the sampled distances. For directional (fixed) sensors, it is necessary to rotate around the robot's axis, by a value expressed as

$$\gamma = \frac{360^\circ}{s}, \quad (2.5)$$

where γ represents the rotation required to scan the environment, in degrees, and s corresponds to the number of sensors equally distributed in the robotic application. Thus, every θ degrees during the rotation, the set of s sensors obtains a sample of the environment.

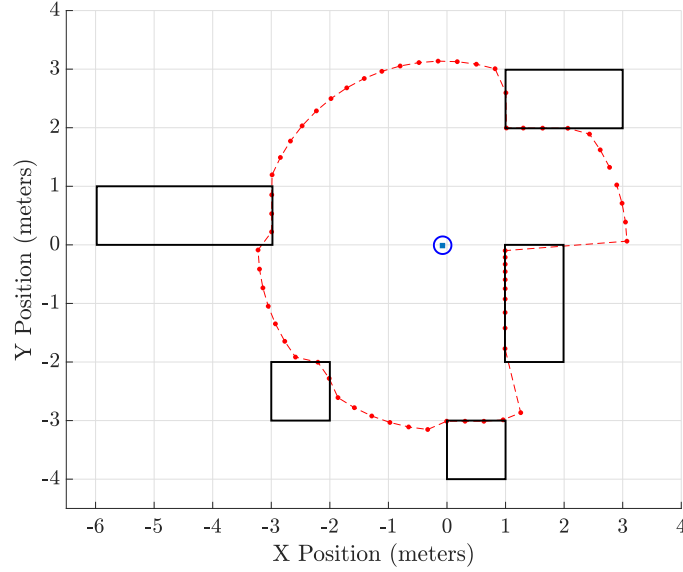


Figure 2.3: Illustration of the data collection stage with $P = 60$ and $\Phi = 3$ meters for the n -th displacement. The robot is illustrated by a blue circle with its center marked by a blue square. Obstacles are illustrated by black-outlined objects. The information contained in $\mathbf{dp}_n$ is represented by red points. The polygon $\mathbf{dp}_n$ is illustrated by a red dashed line that connects these points and represents the free space around the robot.

2.2 Preprocessing

The preprocessing step, as described in line 5 of Algorithm 1, generates two new vectors at every n -th displacement, which are expressed as

$$\mathbf{op}_n = \begin{bmatrix} op_n(0) \\ \vdots \\ op_n(p) \\ \vdots \\ op_n(P - \kappa - 1) \end{bmatrix} = \begin{bmatrix} (x_n^{OP}(0), y_n^{OP}(0)) \\ \vdots \\ (x_n^{OP}(p), y_n^{OP}(p)) \\ \vdots \\ (x_n^{OP}(P - \kappa - 1), y_n^{OP}(P - \kappa - 1)) \end{bmatrix}, \quad (2.6)$$

and

$$\mathbf{dp}_n^{space} = \begin{bmatrix} dp_n^{space}(0) \\ \vdots \\ dp_n^{space}(p) \\ \vdots \\ dp_n^{space}(P - 1) \end{bmatrix} = \begin{bmatrix} (x_n^{space}(0), y_n^{space}(0)) \\ \vdots \\ (x_n^{space}(p), y_n^{space}(p)) \\ \vdots \\ (x_n^{space}(P - 1), y_n^{space}(P - 1)) \end{bmatrix}. \quad (2.7)$$

The vector $\mathbf{op}_n$ stores the points associated with obstacles. This vector consists of $P - \kappa - 1$ points, where κ defines the number of points not associated with obstacles. Figure 2.4 illustrates a case with $P = 60$ and $\kappa = 37$. In this illustration, the robot is represented by

a blue circle, with its center marked by a blue square, while obstacles in the environment are depicted as objects with black outlines. The environment around the robot, as shown in Figure 2.4, presents several samples detected as obstacles, indicating that most of the polygon provides a free path.

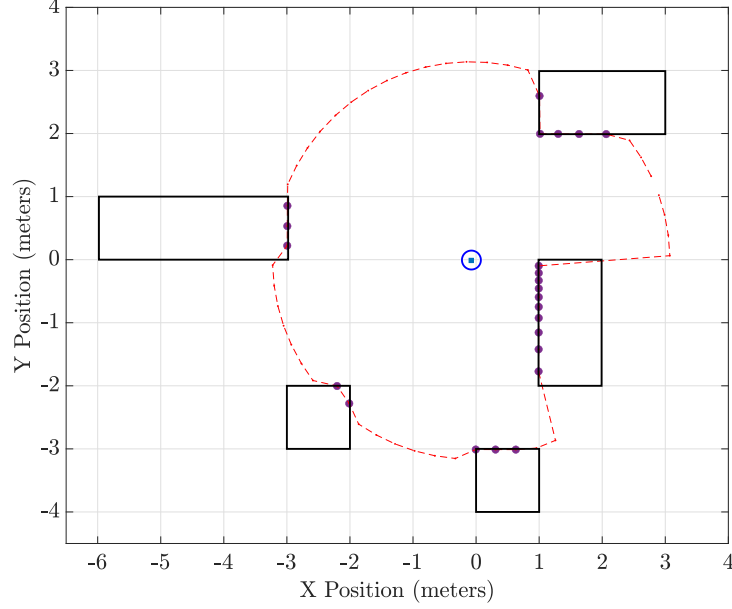


Figure 2.4: Illustration of the obstacle point recognition stage (Equation 2.8), with $P = 60$, and $\Phi = 3$ meters.

The vector $\mathbf{op}_n$ is populated through a comparison operation applied to each p -th point of the vector $\mathbf{dp}_n$ during the n -th displacement. The comparison operation is expressed as

$$op_n(p) = \begin{cases} sp_n^{OP}(p) & \text{if } F_{ed}(sp_n^R, sp_n^{OP}(p)) < (\Phi \cdot (1 - \eta) + \delta) \\ \# & \text{otherwise} \end{cases}, \quad (2.8)$$

where the term η represents a safety factor applied to modify the obstacle detection threshold, and the function $F_{ed}(\cdot, \cdot)$ is expressed as

$$F_{ed}(A, B) = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}. \quad (2.9)$$

Thus, Equation 2.8 compares the distance of the p -th sample from the vector $\mathbf{dp}_n$ with the limit distance, the term $(\Phi \cdot (1 - \eta) + \delta)$, and assigns it or not to the vector $\mathbf{op}_n$ during the n -th displacement.

The second transformation performed during the preprocessing step aims to convert the distance polygon, $\mathbf{dp}_n$, into a new polygon also composed of points, $\mathbf{dp}_n^{space}$, which represents the free and navigable space of the environment during the n -th displacement. In other words, this transformation maps the environment to ensure that any movement made by a point within this new polygon does not result in entering a restricted region in the workspace, thus avoiding collisions.

The conversion of the original polygon into a polygon representing the navigable space is based on the concept of mapping the workspace into a configuration space (Latombe 1991). In this mapping, the robot's characteristics are integrated into the environment. The MHTRSN strategy simplifies this process by generating the polygon $\mathbf{dp}_n^{space}$, as during each n -th displacement, the relevant environment is always contained within the polygon $\mathbf{dp}_n$ with the robot at its center. Thus, only the parts of the obstacles facing the robot are mapped, eliminating the need to map the entire environment at once. Figure 2.4 shows an example of the generation of the polygon $\mathbf{dp}_n^{space}$ from $\mathbf{dp}_n$ with $P = 60$. In this illustration, the polygon $\mathbf{dp}_n$ is represented by an external dashed line (red), while the polygon $\mathbf{dp}_n^{space}$ is depicted by an internal dashed line (blue). The robot is illustrated as a circle located within the polygons (blue).

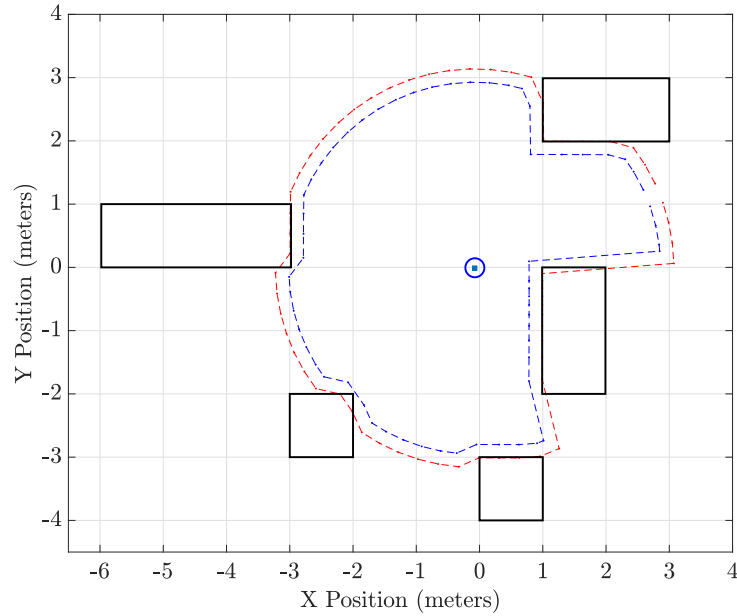


Figure 2.5: Illustration of the generation of the polygon $\mathbf{dp}_n^{space}$ from the polygon $\mathbf{dp}_n$. In this illustration, $\mathbf{dp}_n$ is represented by a red dashed line. $\mathbf{dp}_n^{space}$ is depicted by a blue dashed line, marking the safe area for the movement of a point.

Each p -th point of the vector $\mathbf{dp}_n^{space}$ can be expressed as

$$\begin{aligned} x_n^{space}(p) &= \frac{b_n(p+1) - b_n(p)}{m_n(p) - m_n(p+1)}, \\ y_n^{space}(p) &= m_n(p)x_n^{space}(p) + b_n(p) \end{aligned} \quad (2.10)$$

where

$$m_n(p) = \frac{(y_n^{OP}(p+1) + \alpha R v_{n,p+1}^y) - (y_n^{OP}(p) + \alpha R v_{n,p}^y)}{(x_n^{OP}(p+1) + \alpha R v_{n,p+1}^x) - (x_n^{OP}(p) + \alpha R v_{n,p}^x)}, \quad (2.11)$$

and

$$b_n(p) = (y_n^{OP}(p) + \alpha R v_{n,p}^y) - m_n(p)(x_n^{OP}(p) + \alpha R v_{n,p}^x). \quad (2.12)$$

The terms $m_n(p)$ and $b_n(p)$ calculate, respectively, the slope and intercept of the line formed by two consecutive points. Thus, the polygon $\mathbf{dp}_n^{space}$ is composed of the intersection points between two consecutive lines. The term R represents the radius of the object delimited by the robot after a full rotation. Figure 2.6 details the possible values of R . For Robot 1, which can rotate around its own axis, R corresponds to the radius of the circle formed by its rotational movement. For Robot 2, which has a minimum turning restriction, R represents the minimum turning radius allowed by the robot.

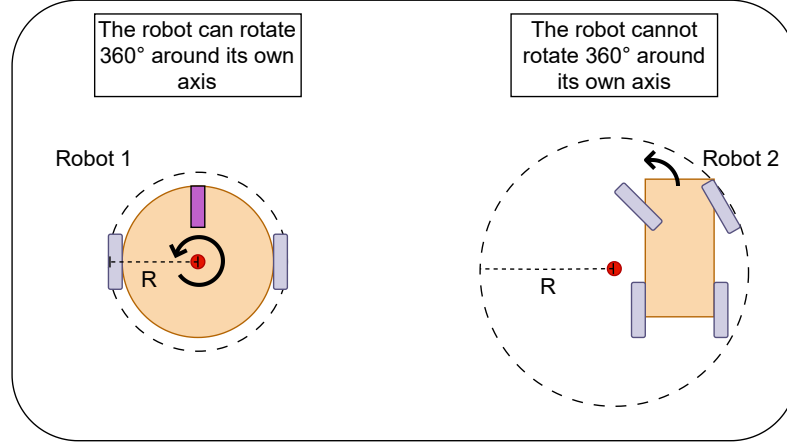


Figure 2.6: Illustration of the possible values of R based on the robot's turning capability. The robot is represented as an orange object with gray wheels. For Robot 1, which rotates around its own axis, R corresponds to the radius of the object that delimits it. For Robot 2, which has minimum turning restrictions, R represents the minimum turning radius.

The terms $v_{n,p}^x$ and $v_{n,p}^y$, as presented in Equations 2.11 and 2.12, represent the x and y components of the unit orthogonal normal vector calculated from two consecutive points. This vector is expressed as

$$\mathbf{v}_{n,p} = [v_{n,p}^x, v_{n,p}^y] = \left[-\frac{y_n^{OP}(p+1) - y_n^{OP}(p)}{\|\mathbf{d}\|}, \frac{x_n^{OP}(p+1) - x_n^{OP}(p)}{\|\mathbf{d}\|} \right], \quad (2.13)$$

where,

$$\|\mathbf{d}\| = \sqrt{(x_n^{OP}(p+1) - x_n^{OP}(p))^2 + (y_n^{OP}(p+1) - y_n^{OP}(p))^2}. \quad (2.14)$$

The term $\|\mathbf{d}\|$ represents the magnitude of the vector connecting two consecutive points.

Figure 2.7 illustrates the process of obtaining the p -th point of $\mathbf{dp}_n^{space}$ from the points of $\mathbf{dp}_n$. In the figure, $\mathbf{dp}_n$ is represented by the upper dashed line with points marked by circles (red), while $\mathbf{dp}_n^{space}$ is represented by the lower dashed line with points also marked by circles (blue). The p -th point of $\mathbf{dp}_n^{space}$ is calculated by displacing the p -th point of $\mathbf{dp}_n$ by a factor αR in the direction of the vector $\mathbf{v}_{n,p}$.

Figure 2.8 shows the final result of the preprocessing step for a scenario with $P = 60$, $\kappa = 37$, $R = 0.15$ m, and $\alpha = 1.4$. In the figure, $\mathbf{dp}_n$ is represented by the outer dashed line (red), while $\mathbf{dp}_n^{space}$ is illustrated by the inner dashed line (blue). The points of $\mathbf{op}_n$ are depicted as circles along the outer dashed line (purple). In this illustration, the obstacle on the far left is partially represented due to the low value of P . The MHTRSN strategy

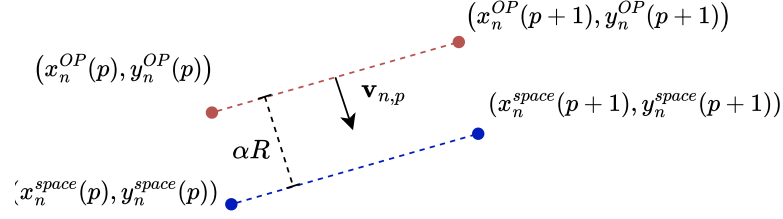


Figure 2.7: Illustration of the calculation of the points of $\mathbf{dp}_n^{space}$ from the polygon $\mathbf{dp}_n$. $\mathbf{dp}_n^{space}$ is shown in blue, and $\mathbf{dp}_n$ in red. The vector $\mathbf{v}_{n,p}$ is represented by a black arrow, indicating the direction of the displacement.

allows for the adjustment of the environment representation through the parameters P and α .

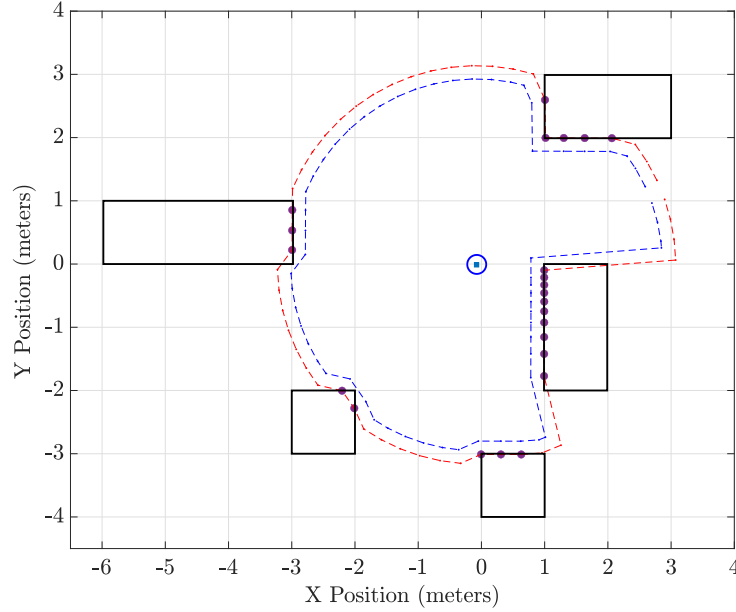


Figure 2.8: Representation of the polygons $\mathbf{dp}_n$ and $\mathbf{dp}_n^{space}$ in an environment with obstacles. Configuration: $N = 60$, $R = 0.15$ m, $\alpha = 1.4$, and $\Phi = 3$ m. The robot is illustrated by a blue circle with its center marked by a blue square. Obstacles are represented by black outlines. The red dashed line indicates $\mathbf{dp}_n$, while the blue dashed line represents $\mathbf{dp}_n^{space}$. Obstacle points in $\mathbf{op}_n$ are marked as purple circles.

2.3 Metaheuristic Evaluation

The metaheuristic evaluation step, as described in line 6 of Algorithm 1, generates the local goal point, sp_n^{LG} , at each n -th displacement. The selection of sp_n^{LG} occurs through the minimization of an evaluation function that describes the environment using a population-based metaheuristic algorithm. At each n -th displacement, a vector $\mathbf{mh}_{n,g}$ is generated,

containing a set of K individuals associated with the metaheuristic algorithm, which can be expressed as

$$\mathbf{mh}_n = \begin{bmatrix} mh_n(0,0) \\ \vdots \\ mh_n(k,g) \\ \vdots \\ mh_n(K-1,g-1) \end{bmatrix} = \begin{bmatrix} (x_n^{mh}(0,0), y_n^{mh}(0,0)) \\ \vdots \\ (x_n^{mh}(k,g), y_n^{mh}(k,g)) \\ \vdots \\ (x_n^{mh}(K-1,g-1), y_n^{mh}(K-1,g-1)) \end{bmatrix}, \quad (2.15)$$

where G represents the number of generations of the algorithm. Each individual, denoted by $mh_n(k,g)$, represents a spatial point that is a potential candidate for solving the problem.

The first generation of individuals, $g = 0$, is generated using a constraint equation defined as

$$R^{mh} \geq \sqrt{(x_n^{mh}(k,g) - x_n^R)^2 + (y_n^{mh}(k,g) - y_n^R)^2}, \quad (2.16)$$

where the individuals are created within a circle of radius R^{mh} , centered at the robot's current location, (x_n^R, y_n^R) . The solution of creating individuals as described in Equation 2.16 is adopted to simplify the generation of initial individuals.

Figure 2.9 illustrates how the initial individuals are generated according to Equation 2.16 in an environment with obstacles. The generated individuals may fall into one of three possible situations. Situation 1 occurs when the individual is generated within the radius R^{mh} and inside the polygon $\mathbf{dp}_n^{space}$; in this case, the individual is considered valid and is not penalized. In Situation 2, the individual is generated outside the radius R^{mh} but inside the polygon $\mathbf{dp}_n^{space}$; this individual is not penalized because it is within the polygon representing the valid space. In Situation 3, the individual is penalized because, even if it is within the radius R^{mh} , its location is outside the polygon $\mathbf{dp}_n^{space}$. Therefore, the individual is in a forbidden location.

In this illustration, the polygon $\mathbf{dp}_n$ is represented by a red dashed line, while the polygon $\mathbf{dp}_n^{space}$ is represented by a blue dashed line. The radius R^{mh} is illustrated as a yellow dashed line. The robot's center is represented by a blue square. In this illustration, the individuals $mh_n(k,g)$ are represented by the symbol $\times$.

Each k -th individual of the g -th generation at the n -th displacement is evaluated by a fitness function expressed as

$$f_n(k,g) = F_{ed}^G(k,g) + \rho_n \frac{1}{F_{ed}^{op}(n,p)} + \rho_n A(n,k,g) + B(n,k,g), \quad (2.17)$$

where $F_{ed}^G(k,g)$ represents the Euclidean distance between the goal point and the k -th individual in the g -th generation, $\frac{1}{F_{ed}^{op}(n,p)}$ represents the repulsion force of the individual from an obstacle, ρ_n is an adjustment factor, and $A(n,k,g)$ and $B(n,k,g, f_{mr}(n,k,g))$ are penalty factors.

The terms $F_{ed}^G(k,g)$ and $\frac{1}{F_{ed}^{op}(n,p)}$ are related to the concept of potential fields, which

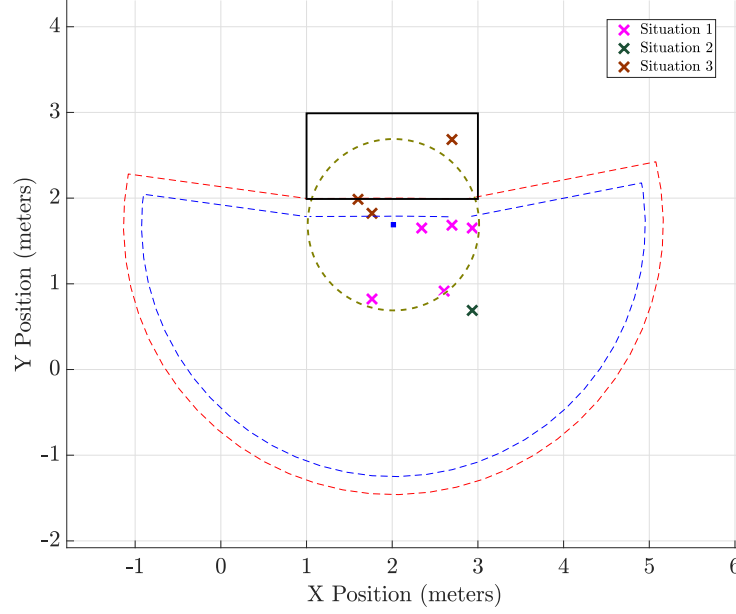


Figure 2.9: Illustration of the creation of individuals and the three possible situations for the individuals with $N = 60$, $R^{radius} = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, $R^{mh} = 1m$, $K = 10$, and $G = 1$. In Situations 1 and 2, the individuals are not penalized because they are located in a valid region for the robot. However, the individuals in Situation 3 are located outside the polygon $\mathbf{dp}_n^{space}$ and are therefore penalized. In this illustration, the robot is represented by a blue square. The polygon $\mathbf{dp}_n$ is represented by a red dashed line. The polygon $\mathbf{dp}_n^{space}$ is represented by a blue dashed line. The generation radius R^{mh} is illustrated by a yellow dashed line. The obstacle is represented by a rectangle with a black outline.

convey information about the environment to the individual. The term $F_{ed}^G(k, g)$ in Equation 2.17 represents the Euclidean distance between the goal point and the k -th individual in the g -th generation, calculated as

$$F_{ed}^G(k, g) = \sqrt{(x^G - x_n^{mh}(k, g))^2 + (y^G - y_n^{mh}(k, g))^2}. \quad (2.18)$$

This term acts as an attractive force, whose value decreases as the individual approaches the goal point. On the other hand, the term $\frac{1}{F_{ed}^{op}(n, p)}$ represents the repulsive force. The repulsive force is calculated based on the shortest Euclidean distance between the k -th individual of the g -th generation in $\mathbf{mh}_n$ and $\mathbf{op}_n$. This distance is determined by

$$F_{ed}^{op}(n, p) = \min \left(\sqrt{(x_n^{OP}(p) - x_n^{mh}(k, g))^2 + (y_n^{OP}(p) - y_n^{mh}(k, g))^2} \right) \quad (2.19)$$

for $p = 0$ to $(P - \kappa - 1)$.

As the individual gets closer to the obstacle, the term $\frac{1}{F_{ed}^{op}(n, p)}$ increases significantly, overpowering the attractive force and preventing collisions.

The term ρ_n , present in Equation 2.17, is an adjustment factor defined as

$$\rho_n = \begin{cases} 1 & \text{if } P > \kappa \\ 0 & \text{if } P = \kappa. \end{cases} \quad (2.20)$$

This term adjusts the calculation of the fitness of individuals by nullifying the terms multiplied by ρ_n in Equation 2.17 when $\rho_n = 0$. This occurs when P and κ are equal, indicating the absence of obstacles during the preprocessing step. Under this condition, the calculations of the terms $\frac{1}{F_{ed}^{op}(n,p)}$ and $A(n,k,g)$ become unnecessary.

The term $A(n,k,g)$, present in Equation 2.17, is defined as

$$A(n,k,g) = \begin{cases} 1 & \text{if } \nexists m \in \{0, \dots, N-1\} : F_{ed}(\mathbf{mh}_n(k,g), \mathbf{pv}^R) < R_{mh} \\ \tau & \text{if } \exists m \in \{0, \dots, N-1\} : F_{ed}(\mathbf{mh}_n(k,g), \mathbf{pv}^R) < R_{mh}, \end{cases} \quad (2.21)$$

where τ is a high penalty applied to individuals generated in previously visited areas, encouraging the algorithm to explore new regions. The penalty is determined by the Euclidean distance between the individual $\mathbf{mh}_n(k,g)$ and the robot's position history $\mathbf{pv}^R$. If this distance is smaller than R_{mh} , the individual is penalized with τ . Otherwise, the penalty is equal to 1, indicating the individual is in an unvisited area. The second penalty term, $B(n,k,g)$, is defined as

$$B(n,k,g) = \begin{cases} 0 & \text{if } mh_n(k,g) \in \mathbf{dp}_n^{space} \text{ and } f_{mr}(sp_n^R, mh_n(k,g)) = 0 \\ \infty & \text{otherwise,} \end{cases} \quad (2.22)$$

Each k -th individual in the g -th generation is evaluated based on whether it is inside the polygon $\mathbf{dp}_n^{space}$ and if there is a valid path between the robot and the individual, as determined by the function $f_{mr}(n,k,g)$. If the individual is inside the polygon (situations 1 and 2 in Figure 2.9) and the path is valid ($f_{mr}(n,k,g) = 0$), it is not penalized ($B = 0$). Otherwise, it gets an infinite penalty ($B = \infty$), greatly lowering its fitness.

The term $f_{mr}(n,k,g)$ in Equation 2.22 verifies if a viable path exists between the robot's position and the individual. This is expressed as

$$f_{mr}(n,k,g) = \begin{cases} 0 & \text{if } \exists \text{ a path } C \text{ from } sp_n^R \text{ to } mh_n(k,g) \text{ that satisfies } f_{res}(C) = 1 \\ 1 & \text{otherwise,} \end{cases} \quad (2.23)$$

where the term C represents a path connecting the points sp_n^R and $mh_n(k,g)$. The term $f_{res}(C)$ represents a function that checks whether the path C meets all the robot's constraints. If a path C exists that connects sp_n^R to $mh_n(k,g)$ and satisfies the system's constraints, the individual is not penalized. Otherwise, $f_{mr}(n,k,g)$ returns 1, indicating that no viable path exists.

The definition of the function $f_{res}(C)$ in Equation 2.23 varies according to the characteristics of the robotic system. For example, for a robot that rotates around its own axis

and moves only in a straight line, the constraint can be defined as

$$f_{\text{res}}(C) = \begin{cases} 1 & \text{if } C \text{ is a straight line between } sp_n^R \text{ and } mh_n(k, g) \\ 0 & \text{otherwise.} \end{cases} \quad (2.24)$$

For a robot that moves along a circular arc, the constraint can be defined as

$$f_{\text{res}}(C) = \begin{cases} 1 & \text{if } C \text{ is an arc of a circle that connects } sp_n^R \text{ to } mh_n(k, g) \text{ with radius } R, \\ 0 & \text{otherwise.} \end{cases} \quad (2.25)$$

Thus, the term f_{mr} is adjusted according to the specific constraints of the project to determine the penalty for individuals based on the feasibility of the path.

In the final generation, the fittest k -th individual, based on the evaluation function (see Equation 2.17), is selected. This selection is expressed as

$$k = \arg \min_k \{f_n(0, G-1), \dots, f_n(k, G-1), \dots, f_n(K-1, G-1)\}, \quad (2.26)$$

and,

$$sp_n^{LG} = mh_n(k, G-1). \quad (2.27)$$

Where the fittest individual from the final generation is assigned to the n -th local goal point, sp_n^{LG} .

Figure 2.10 provides a detailed illustration of the metaheuristic evaluation step and the convergence of individuals over generations. In the illustration, the repulsive and attractive components are represented by red and blue lines, respectively, only for the fittest individual of each generation, to facilitate visualization. The polygon $\mathbf{dp}_n$ is illustrated by a outer dashed line, while the polygon $\mathbf{dp}_n^{space}$ is represented by a inner dashed line. The inner dashed line circle indicates the circle with radius R^{mh} from Equation 2.16. The robot is represented by a blue circle, and obstacles are depicted as black objects. The individuals, illustrated by the symbol $\times$, converge toward the region with the lowest objective function value as generations progress. Figures 2.10(a), 2.10(b), 2.10(c), and 2.10(d) show the behavior of individuals across generations. These figures correspond to the second generation, $g = 1$, the fifth generation, $g = 4$, the ninth generation, $g = 8$, and the eleventh generation, $g = 10$, respectively. It is observed that as the number of generations increases, the individuals converge to the point with the lowest evaluation function value.

Figure 2.11 shows the fitness function for the environment illustrated in Figure 2.10, considering a total of 3 displacements. The z -axis of the 3D plot represents the function value for each point, highlighting the discontinuities and jumps present in the fitness function. In this illustration, obstacles are depicted as black outlined objects on the x, y plane. $\mathbf{dp}_n$ is illustrated in blue. The penalty factor $A(n, k, g)$ is represented as a green region. The goal point is marked by a green circle on the x, y plane. As the environment samples approach the obstacle, there is a sharp increase in the value of the repulsive charges (z -axis), contrasting with the decrease in the attractive component (purple region on the x, y plane). Due to the complexity of the fitness function, which includes jumps and discontinuities, some approaches may struggle to find a solution. However, metaheuristic

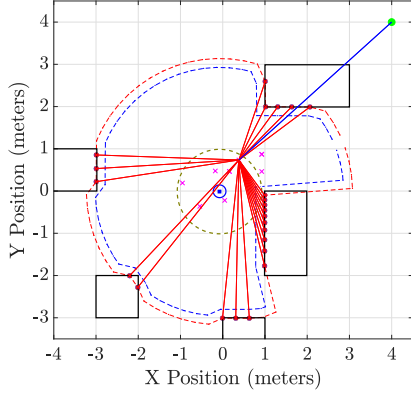
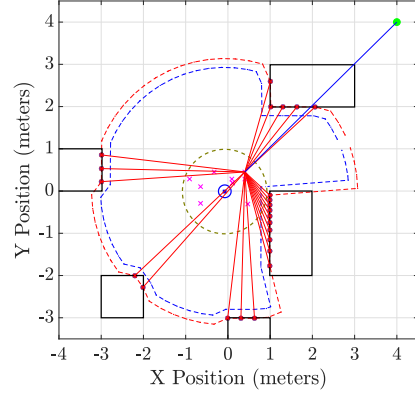
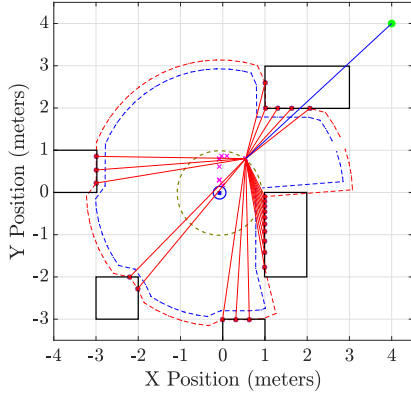
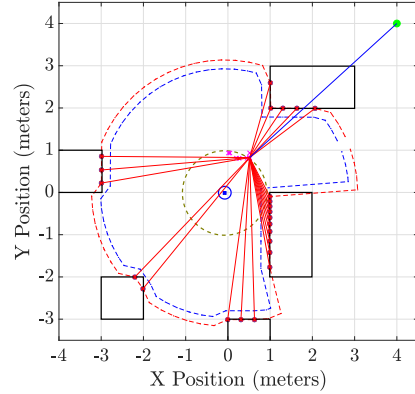
(a) Second generation ($g = 1$).(b) Fifth generation ($g = 4$).(c) Ninth generation ($g = 8$).(d) Eleventh generation ($g = 10$).

Figure 2.10: Illustration of the evaluation of individuals using the fitness function with $N = 60$, $R = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, $R^{mh} = 1m$, $K = 10$, $\varepsilon = 0.05m$. In the illustration, $\mathbf{dp}_n$ is shown as a red dashed line, while $\mathbf{dp}_n^{space}$ is represented by a blue dashed line. The yellow dashed line indicates the circle with radius R^{mh} from the constraint equation. The robot's position, sp^R , is marked by a blue square, while the individuals are represented by the symbol $\times$. Obstacles are depicted as black outlined objects. The blue and red lines represent the attractive and repulsive forces of the fittest individual in each generation, respectively.

algorithms have proven effective in solving this type of problem, providing suboptimal solutions with lower computational requirements compared to optimal solutions (Rajwar et al. 2023).

2.4 Displacement

In the displacement step, line 7 of Algorithm 1, the robot moves to the local goal point selected by the metaheuristic function. As mentioned earlier, a region defined by $sp_n^{LG} \pm \varepsilon$ is created around sp_n^{LG} . Any position reached by sp^R within this region is considered as goal achieved. This ensures that minor setbacks are absorbed by the technique, allowing

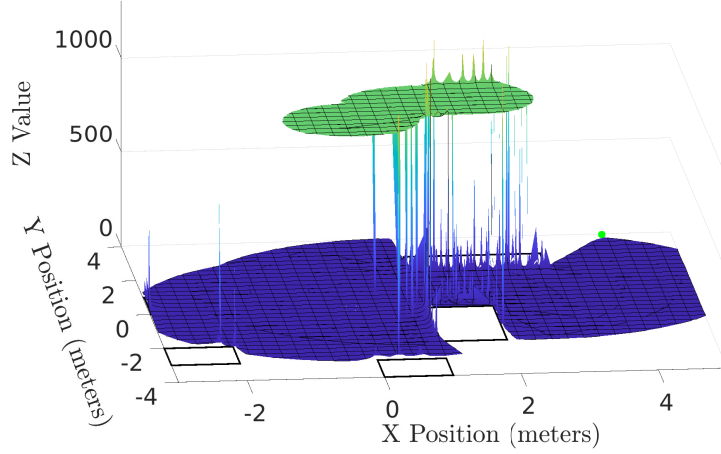


Figure 2.11: 3D plot of the fitness function with $\Phi = 3\text{ m}$, $R^{mh} = 1\text{ m}$, $N = 3$, and $\tau = 1000$ in an obstacle-filled environment. In this illustration, obstacles are depicted as black outlined objects on the x, y plane. $\mathbf{dp}_n$ is illustrated in blue. The penalty factor $A(n, k, g)$ is represented as a green region. The goal point is marked by a green circle on the x, y plane.

the robot to reach its local goal.

After moving to sp_n^{LG} , a new robot position, sp_{n+1}^R , is generated, as expressed by:

$$sp^R(n+1) = (sp_n^{LG} \pm \epsilon), \quad (2.28)$$

where the robot's position at iteration $n+1$ is equal to the location of the local goal point, plus or minus the factor ϵ . Equation 2.28 is linked to Equation 2.1, as the robot's position history is related to the local goal points chosen during the metaheuristic evaluation step (see line 6 of Algorithm 1).

Figure 2.12 provides a detailed illustration of how displacement is performed in each iteration, as well as how the polygons $\mathbf{dp}_n$ and $\mathbf{dp}_n^{space}$ function during each iteration until reaching the global target. In this illustration, the polygon $\mathbf{dp}_N(P)$ is represented by the outer dashed line (red), and the polygon $\mathbf{dp}_N^{space}(P)$ by the inner dashed line (blue). The robot's reference point, sp_R , is represented by a square at the robot's center (blue), while the robot itself is illustrated as a circle (blue). The global target point, sp_G , is depicted as a circle at the final destination (green). Obstacles are shown as objects with black outlines. In this example, the robotic system using MHRTSN performs a total of 6 displacements to move from the starting point (0.0, 0.0) to the target point (4.0, 4.0).

2.5 Algorithm Complexity

The computational complexity of the MHRTSN algorithm is analyzed considering its main steps: data collection, preprocessing, metaheuristic evaluation, and displacement. This section presents the complexity of these steps using Big-O notation.

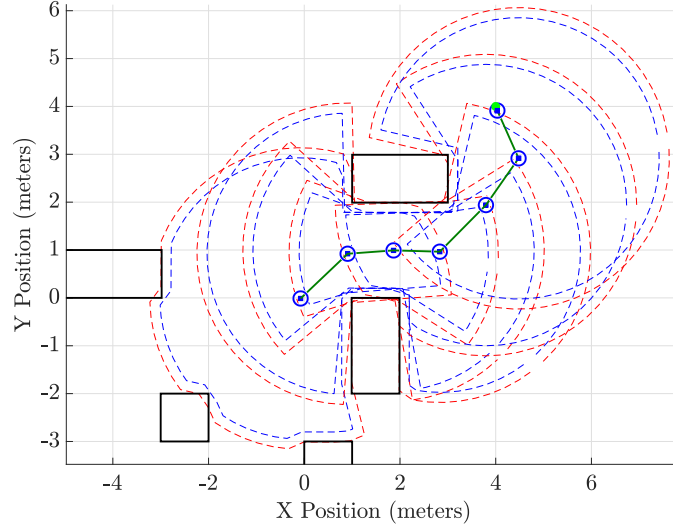


Figure 2.12: Example of MHRTSN movement in an obstacle-filled environment with $N = 60$, $R^{radius} = 0.15m$, $\alpha = 1.4$, $\Phi = 3m$, and $\epsilon = 0.05m$. In this illustration, the polygon $\mathbf{dp}_N(P)$ is represented by a red dashed line, the polygon $\mathbf{dp}_N^{space}(P)$ by a blue dashed line, and sp_R is represented by a blue square. The robot is illustrated as a blue circle. The global target point, sp_G , is represented by a green circle. Obstacles are depicted as black outlined objects.

The Data Collection step scans the environment and collects P samples that represent the environment in $\mathbf{dp}_n$ (see section 2.1). This process has a complexity of $O(P)$.

In the Preprocessing step, the collected samples are processed to identify obstacles and construct the navigable space polygon, filling the vectors $\mathbf{op}_n$ and $\mathbf{dp}_n^{space}$ (see section 2.2). Both operations depend linearly on the number of samples, resulting in a complexity of $O(P)$.

The Metaheuristic Evaluation step, described in section 2.3, consists of three sub-steps: generation of the initial population, fitness evaluation of each individual, and evolution over G generations. The generation of the initial population occurs with a complexity of $O(K)$. Each individual is evaluated based on multiple operations, such as Euclidean distance calculations, the search for the shortest distance to obstacles, and the application of penalties related to the position history and path validity. Evaluating a single individual has a complexity of $O(P + N)$, and when applied to K individuals in a single generation, the total complexity per generation becomes $O(K \cdot (P + N))$. Over G generations, the total complexity of the Metaheuristic Evaluation step is $O(G \cdot K \cdot (P + N))$.

In the Displacement step, the robot moves to the point sp_n^{LG} and updates its position to $sp^R(n + 1)$ (see section 2.4). This step involves only simple calculations and has a constant complexity of $O(1)$.

The total complexity of the MHRTSN algorithm is determined by the sum of the complexities of all steps over N_{Max} iterations. Since the Metaheuristic Evaluation step domi-

nates the computational cost, the total complexity is expressed as $O(N_{Max} \cdot G \cdot K \cdot (P + N))$. As N_{Max} is a constant used to limit the maximum number of iterations, the complexity of the MHRTSN algorithm is $O(G \cdot K \cdot (P + N))$. This result reflects the combined impact of the algorithm's main parameters, with the Metaheuristic Evaluation being the predominant factor in the asymptotic behavior.

Chapter 3

Methodology

3.1 Simulation Platform

The experiments and data were generated using the Webots robotic simulation platform, in conjunction with MATLAB software (Inc. 2022). These tools were integrated through a custom solution, enabling dynamic and optimized interaction to track the robot's navigation movements. This simulation platform links the steps of Algorithm 1 (MHRTSN) to the environment, as illustrated in Figure 3.1.

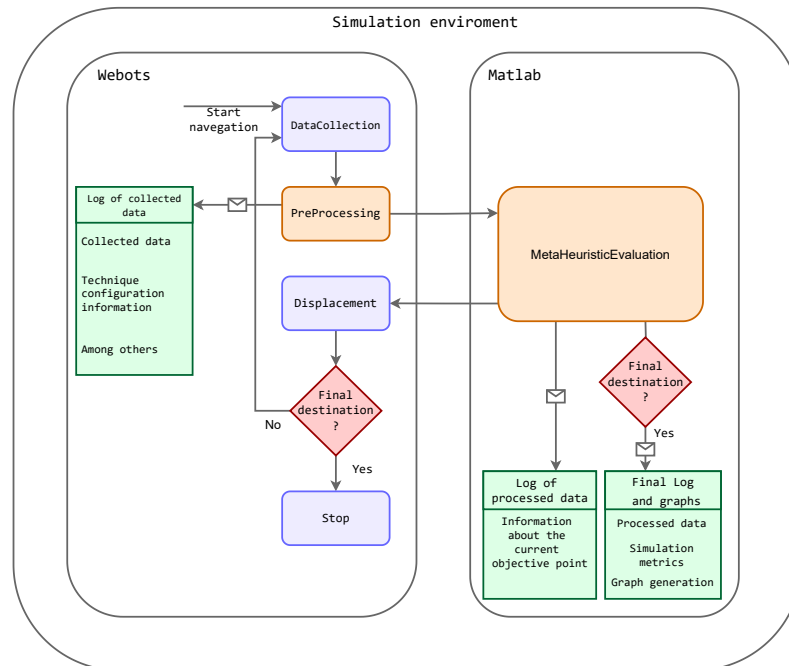


Figure 3.1: Flow diagram of the steps associated with the simulation platform, illustrating the interaction between the steps of Algorithm 1 and the Webots and MATLAB platforms.

Figure 3.1 illustrates the integration between MATLAB and Webots, forming the simulation environment. The steps on the left-hand side are performed within the Webots environment, while those on the right-hand side are executed in MATLAB. Communication between these tools is carried out through message exchange. In the illustration,

steps in blue correspond to physical operations on the robot, such as data collection and movement, as detailed in Sections 2.1 and 2.4, respectively. Steps highlighted in orange correspond to data processing stages, such as preprocessing (described in Section 2.2), which transforms raw collected data for metaheuristic evaluation.

All processed and raw data are saved as logs within the platform. The metaheuristic evaluation step, described in Section 2.3, uses data from previous stages of the MHRTSN algorithm to evaluate the optimal target point. After this step, logs are generated containing information about the target point, simulation metrics, and visualizations of the collected and generated data. These logs provide a complete history of all collected and generated data during each displacement, enabling further analysis beyond what is required by MHRTSN.

Conditional tests, represented by red diamonds in the diagrams, determine whether the target point has been reached. Final evaluation metrics, including traveled distance, metaheuristic processing time (measured using MATLAB's `tic/toc` and `cputime` functions), and the number of displacements required, are computed within the MATLAB environment.

An orchestrator program manages the simulation process via system calls, as shown in Figure 3.2. This program does not interfere with the simulation process itself but enables scalability across various configuration patterns. It automates the creation of the simulation environment for each desired configuration, avoiding manual setup and execution of simulations.

The orchestrator program begins by reading a JSON file containing simulation configuration details, such as the environment to be simulated, metaheuristic processing type, algorithm parameters, and the number of repetitions. After defining the simulation configurations, the program enters a loop that autonomously executes the platform, as shown in the simulation step. At the end of all simulations, a log with simulation status information is produced.

As illustrated in Figure 3.2, the orchestrator program creates and manages simulations within the developed ecosystem, ensuring synchronization between multiple platforms and programming languages. This framework enhances the efficiency of data collection and scalability of simulations tailored to robotic navigation requirements. To run the simulations, the environments are built as 3D models, as shown in Figure 3.3.

3.1.1 Methodology for Performance Evaluation

To evaluate the performance of the MHRTSN navigation strategy, different static and dynamic environments were proposed. These environments allowed the MHRTSN technique to be tested under various scenarios. Population-based metaheuristics, GA and PSO, were used, forming the MHRTSN-GA and MHRTSN-PSO versions of the technique. These versions were evaluated in terms of displacement quality, convergence, processing time, among other metrics. The MHRTSN-GA and MHRTSN-PSO versions were tested under the same conditions. During the simulations, certain parameters were varied to analyze the impact of these changes on the technique's performance. All simulations in this study were conducted on a computer with a 64-bit processor (i5-10500T), containing

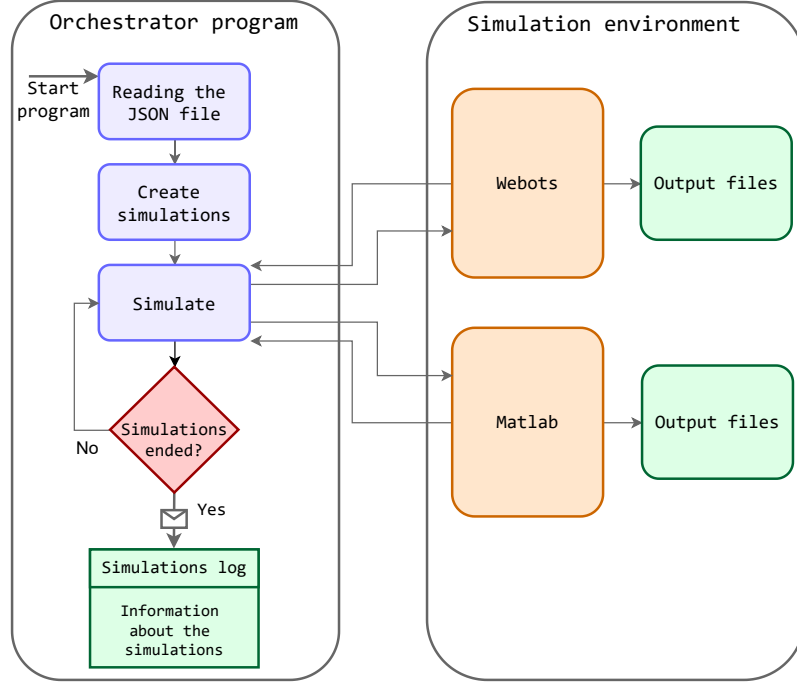


Figure 3.2: Flow diagram illustrating the relationships between the simulation platform and the orchestrator program.

6 cores and 12 threads, operating at 2.30GHz, and equipped with 20GB of RAM.

Each simulation generated a dataset that included detected points, traveled distances, processing times, and other relevant metrics. The processing time and CPU time data were obtained using specific MATLAB functions designed for this purpose (see section 3.1). To evaluate the performance of the techniques, the following key metrics were considered: number of displacements, traveled distances, clock times, and CPU times spent during metaheuristic processing. Since the values varied across trials, the minimum number of samples required to represent the population was calculated using the following statistical equation

$$s = \left(\frac{Z \cdot \sigma^s}{E} \right)^2 \quad (3.1)$$

where s is the minimum number of samples, Z represents the confidence level, σ^s is the sample standard deviation, and E is the margin of error. Thus, the minimum number of samples needed to represent the population was defined by considering the confidence level, margin of error, and variance of an initial dataset. Preliminary trials were conducted to estimate the required number of simulations, using a confidence interval of 95% ($Z = 1.96$). Table 3.1 shows the highest values of n calculated from 20 trials. Based on this estimate, 60 simulations were performed for each configuration pattern where the algorithm reached the minimum. All data related to the mean (μ), standard deviation (σ), and other metrics presented throughout the work were derived from these 60 simulations.

For each proposed environment, one table was generated containing the values of μ and σ for each of the base indicators (see, for example, Table 4.3). Additionally, three

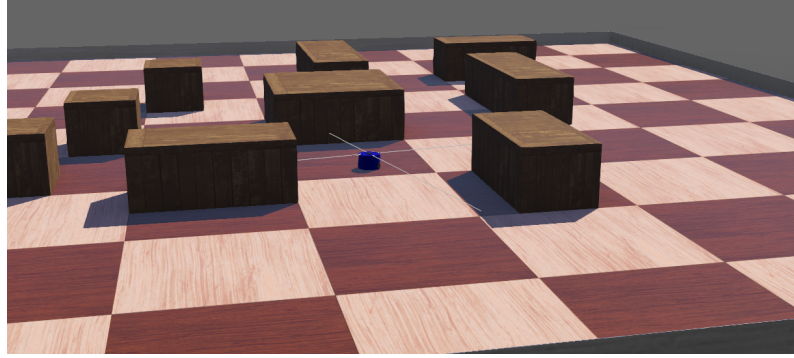


Figure 3.3: Example of navigation in an environment filled with obstacles.

Table 3.1: Number of samples according to statistical parameters.

Parameter	σ^s	E	s (Number of Samples)
Displacement	3.4479	1	46
Path Distance	2.9430	1	33
CPU Time	0.3611	0.1	50
Clock Time	0.1969	0.1	15

new indicators were generated for each environment. The first is the processing time per displacement event ($I1$). The second is the average distance traveled per displacement event ($I2$). The third is the sum of CPU time and clock time spent per displacement event ($I3$). These measures are more robust against variations in path length or the total number of displacements.

To assess the significance of differences between the analyzed indicators, statistical tests were performed on the samples of indicators $I1$ and $I2$. Initially, the data were subjected to the Kolmogorov-Smirnov (KS) test to verify the normality of the distribution. In cases where the data followed a normal distribution, an Analysis of Variance (ANOVA) was conducted for each simulation pattern, considering a 95% confidence level ($\alpha = 0.05$). In this case, the null hypothesis (H_0) assumes that there is no significant difference between the groups, while the alternative hypothesis (H_A) indicates that at least one group exhibits a significant difference.

For cases where the KS test indicated that the data did not follow a normal distribution, non-parametric statistical tests were applied. As in the ANOVA, a confidence level of 95% ($\alpha = 0.05$) was considered. When only two groups were compared, the Mann-Whitney U (MWU) test was used, where the null hypothesis (H_0) assumes that both groups have identical distributions, and the alternative hypothesis (H_A) suggests a significant difference between them. For cases involving three or more groups, the Kruskal-Wallis (KW) test was applied, where the null hypothesis (H_0) considers that all samples come from the same distribution, while the alternative hypothesis (H_A) indicates that at least one group differs from the others.

To evaluate the MHRTSN strategy, the simulation platform described in Section 3.1

was used. The simulations were configured based on a non-holonomic¹ circular robot with a radius of 0.15 m and a height of 0.2 m, equipped with differential drive and fixed distance sensors. The common parameters used in the simulations for both the MHRTSN-GA and MHRTSN-PSO versions are listed in Table 3.2.

Table 3.2: Common parameters used in MHRTSN simulations.

Parameter	Value
Number of sensors (s)	4
Maximum sensor range (Φ)	3 m
Angular displacement (θ)	6°
Metaheuristic creation radius (R^{mh})	1 m
τ (Equation 2.21)	1000
Tolerance factor ϵ	0.2
Codification of individuals	Real number

The specific parameters of the MHRTSN-GA strategy, related to GA, were defined as shown in Table 3.3. In the stochastic uniform selection method, individuals are chosen proportionally to their fitness, increasing the probability of selecting the fittest individuals. The elitism parameter ensures that the fittest individual from each generation is preserved for the next generation, maintaining its genes. The crossover operator randomly selects, for each gene, whether it will be inherited from the first parent (0) or the second parent (1), with equal probability for both, using a random distribution. The chosen mutation operator was the Gaussian random mutation, which introduces random variations into the genes of individuals, with values drawn from a normal distribution with mean 0 and standard deviation 1.

Table 3.3: Specific parameters used in MHRTSN-GA simulations.

Parameter	Value
Selection method	Stochastic uniform
Elitism	1 individual
Crossover operator	Uniform random [0, 1]
Mutation operator	Gaussian random [mean = 0, standard deviation = 1]

The specific parameters of the MHRTSN-PSO strategy, related to PSO, were defined as shown in Table 3.4. The initial swarm is created by uniformly distributing particles within a rectangular range centered on the robot, `pswcreationuniform` function, with a side length equal to $2 \cdot R^{mh}$ (which is later penalized according to Equation 2.16). The velocities of the particles are uniformly generated within the range $[-1, 1]$ for the x and y coordinates, allowing the particles to initially move in any direction with random velocities. The inertia parameter is adaptively adjusted throughout the iterations, starting at 1.1 to promote the exploration of new regions and gradually decreasing to 0.1, favoring solution refinement in the final iterations.

¹A non-holonomic robot has movement constraints that prevent it from moving freely in all directions. For example, a differential drive robot cannot move sideways and must rotate to change direction.

Table 3.4: Specific parameters used in PSO simulations.

Parameter	Value
Swarm creation function	pswcreationuniform
Initial velocity range	Uniform random $[-1, 1]$
Inertia range	Uniform decreasing $[0.1, 1.1]$

Chapter 4

Results

This section presents the results obtained from evaluating the MHRTSN strategy in different simulated environments. The experiments were designed to assess the approach in static and dynamic scenarios, as well as to compare its performance with other techniques found in the literature. The evaluation followed the methodology described in Section 3, maintaining a consistent set of parameters across all simulations.

First, the effectiveness of the MHRTSN strategy was evaluated in four static environments, representing indoor and outdoor scenarios with varying obstacles. The configurations and results for these environments are detailed in Section 4.1. Following this, two dynamic environments were introduced to analyze the adaptability of the strategy in the presence of moving obstacles, as discussed in Section 4.2. In these scenarios, the robot had to adjust its trajectory in real time to avoid collisions and navigate safely toward the goal.

Finally, a comparative analysis was conducted with other approaches from the literature to highlight differences in performance, as presented in Section 4.3. The paths generated by MHRTSN were compared to those produced by alternative methods in similar environments, evaluating aspects such as trajectory efficiency, computational cost, and safety in navigation.

4.1 Static Environments

Four static environments were proposed to evaluate the effectiveness of the MHRTSN strategy. These environments represent indoor and outdoor scenarios with varying obstacles. Figure 4.1 illustrates each of these environments. In the figure, the robot, positioned at the initial navigation point, is represented by a blue circle, while the final navigation point is indicated by a black circle. The robot's objective is to navigate from the starting point to the target in an unknown environment, avoiding obstacles represented by brown and white boxes. Figure 4.1(a) shows Environment 1 (Env1), a mixed scenario. The robot starts in an indoor area and must navigate against the flow to exit and then traverse an outdoor environment with obstacles. Figure 4.1(b) illustrates Environment 2 (Env2), a fully indoor scenario characterized by 90° turns and zigzagging obstacles. Figure 4.1(c) presents Environment 3 (Env3), an outdoor scenario where the robot needs to navigate around an obstacle and park inside it. Figure 4.1(d) illustrates Environment 4 (Env4), an

outdoor scenario densely populated with obstacles. The initial and final points for each environment are defined in Table 4.1.

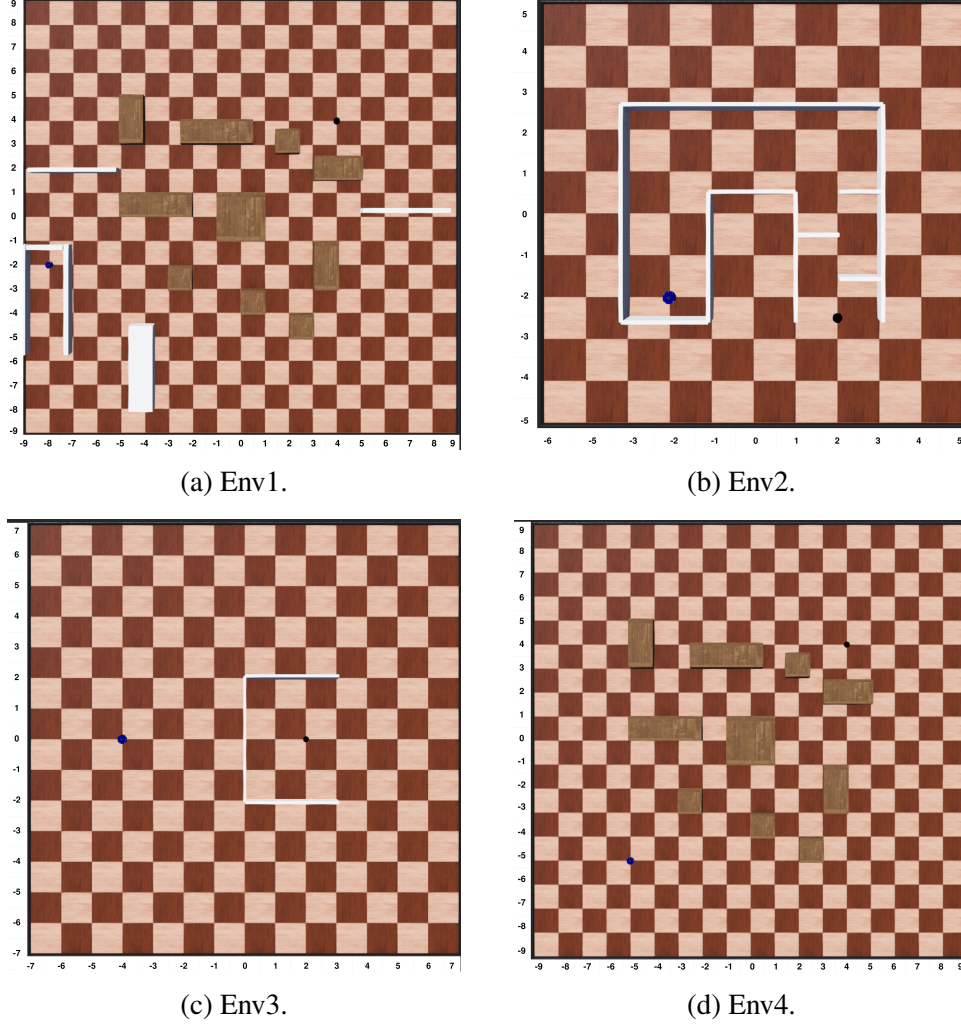


Figure 4.1: Representation of the static environments used in the simulations with MHRTSN-GA and MHRTSN-PSO. The robot is indicated by a blue circle at the initial position, while the final navigation point is shown by a black circle. Obstacles are represented by brown and white boxes. In these environments, the floor is marked with 1 by 1 meter squares in a checkerboard pattern.

Table 4.1: Initial and final points for each static environment.

Environment	Initial Point	Final Point
Env1	$(-8.0, -2.0)$	$(4.0, 4.0)$
Env2	$(-2.0, -2.0)$	$(2.0, -2.5)$
Env3	$(-4.0, 0.0)$	$(2.0, 0.0)$
Env4	$(-5.0, -5.0)$	$(4.0, 4.0)$

In each environment, the MHRTSN-GA and MHRTSN-PSO strategies were evaluated with four parameter variations. The parameters related to the number of individuals and generations in the metaheuristic algorithms were kept constant and grouped, allowing a direct comparison between the approaches. Table 4.2 presents the configurations used in the static environments and the groupings performed according to the parameters.

4.1.1 Results for Env1

Table 4.3 presents the results obtained from the simulations in Env1. Analyzing these results, it was observed that within the same group, simulations using MHRTSN-PSO required less average processing time and exhibited lower variance compared to MHRTSN-GA.

Figures 4.2 and 4.3 present indicators $I1$ and $I2$, respectively.

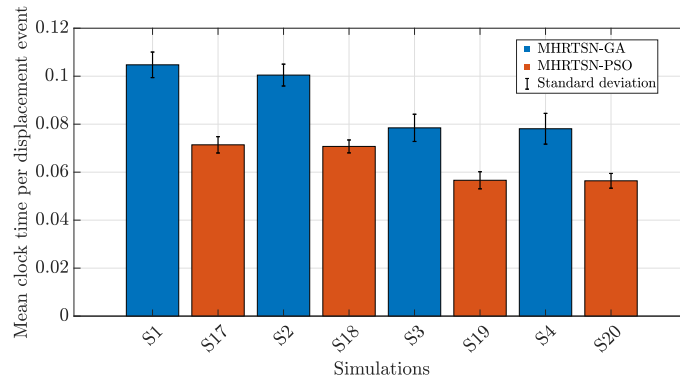


Figure 4.2: Mean and Standard Deviation of Indicator $I1$ in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.

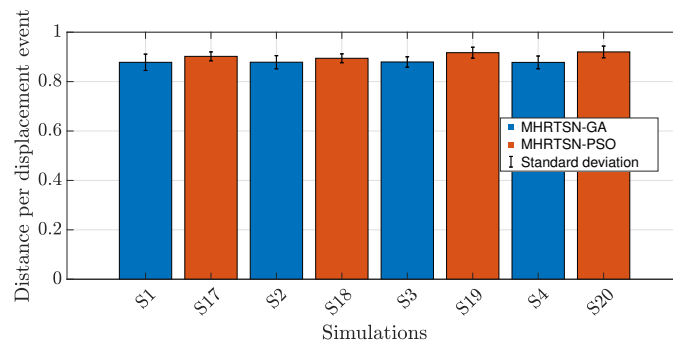


Figure 4.3: Mean and Standard Deviation of Indicator $I2$ in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.

In Figures 4.2 and 4.3, simulations with MHRTSN-GA are represented in blue, while those with MHRTSN-PSO appear in orange. Bars represent the mean of each simulation, with the upper bars indicating the standard deviation. Simulations were grouped according to their configurations. Figure 4.2 reveals reduced processing times per displacement

Table 4.2: Proposed parameter configuration for simulations in the static environments using MHRTSN-GA and MHRTSN-PSO techniques.

Simu- lation	Group	Environ- ment	Meta- heuristic	Popu- lation	Max. Generations	Cross- over
S1	G1	Env1	GA	20	30	50%
S2	G1	Env1	GA	20	30	70%
S3	G2	Env1	GA	30	10	50%
S4	G2	Env1	GA	30	10	70%
S5	G3	Env2	GA	20	30	50%
S6	G3	Env2	GA	20	30	70%
S7	G4	Env2	GA	30	10	50%
S8	G4	Env2	GA	30	10	70%
S9	G5	Env3	GA	20	30	50%
S10	G5	Env3	GA	20	30	70%
S11	G6	Env3	GA	30	10	50%
S12	G6	Env3	GA	30	10	70%
S13	G7	Env4	GA	20	30	50%
S14	G7	Env4	GA	20	30	70%
S15	G8	Env4	GA	30	10	50%
S16	G8	Env4	GA	30	10	70%

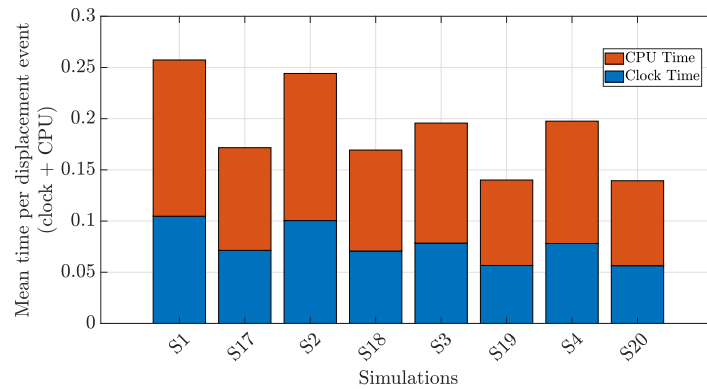
Simu- lation	Group	Environ- ment	Meta- heuristic	Popu- lation	Max. Generations	Cogni- tive	Social
S17	G1	Env1	PSO	20	30	1	1.5
S18	G1	Env1	PSO	20	30	1.5	1
S19	G2	Env1	PSO	30	10	1	1.5
S20	G2	Env1	PSO	30	10	1.5	1
S21	G3	Env2	PSO	20	30	1	1.5
S22	G3	Env2	PSO	20	30	1.5	1
S23	G4	Env2	PSO	30	10	1	1.5
S24	G4	Env2	PSO	30	10	1.5	1
S25	G5	Env3	PSO	20	30	1	1.5
S26	G5	Env3	PSO	20	30	1.5	1
S27	G6	Env3	PSO	30	10	1	1.5
S28	G6	Env3	PSO	30	10	1.5	1
S29	G7	Env4	PSO	20	30	1	1.5
S30	G7	Env4	PSO	20	30	1.5	1
S31	G8	Env4	PSO	30	10	1	1.5
S32	G8	Env4	PSO	30	10	1.5	1

Table 4.3: Simulation results for Env1: Clock times, CPU times, Displacements, and Distances.

Simu- lation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S1	G1	3.0156	0.2034	4.3943	0.3266	28.9333	3.2087	25.3632	2.5197
S2	G1	2.9312	0.1830	4.1977	0.3293	29.3000	2.9682	25.7195	2.5127
S3	G2	2.1947	0.0959	3.2733	0.1269	28.1833	3.0795	24.7751	2.6665
S4	G2	2.2218	0.1276	3.3962	0.2346	28.7000	3.4220	25.1656	2.8514
S17	G1	2.0065	0.1369	2.8166	0.1992	28.2143	2.9137	25.4554	2.6798
S18	G1	1.9722	0.1070	2.7486	0.1398	27.9661	2.4144	25.0109	2.1882
S19	G2	1.4153	0.0731	2.0848	0.1015	25.1667	2.8877	23.0519	2.4214
S20	G2	1.4241	0.0662	2.0938	0.0958	25.3833	2.5696	23.3478	2.3484

event for MHRTSN-PSO compared to MHRTSN-GA. No significant differences were observed between simulations using the same meta-heuristic within the same group, such as S1 and S2. Figure 4.3 shows indicator I_2 , where MHRTSN-PSO traveled greater average distances than MHRTSN-GA, suggesting higher efficiency in displacement.

Figure 4.4, illustrating indicator I_3 , shows that simulations with MHRTSN-GA required more clock and CPU time per displacement than MHRTSN-PSO. Additionally, it was observed that PSO evaluations did not benefit from parallelism, resulting in increased CPU time.

Figure 4.4: Mean processing time, Indicator I_3 , in Env1 During MHRTSN-GA and MHRTSN-PSO Simulations.

The results of the statistical tests performed on the simulations illustrated in Figures 4.2-4.4 are summarized in Table 4.4. The KS test was first applied to verify the normality of the data. For cases where the data did not follow a normal distribution, the KW test was applied. When the data were normally distributed, the ANOVA test was used. In all cases, the null hypothesis (H_0) was rejected, as the p -values obtained were below the significance level ($\alpha = 0.05$). These results indicate statistically significant differences among one or more groups, suggesting performance differences among the evaluated techniques

for indicators I_1 and I_2 (Mean Distance per Displacement).

Table 4.4: Statistical test results for indicators I_1 and I_2 in Env1 for Groups G_1 and G_2 .

Group	Indicator	Test Applied	p -value
G_1	I_1	KW	5.7679×10^{-39}
G_1	I_2	KW	1.0778×10^{-7}
G_2	I_1	KW	1.1762×10^{-38}
G_2	I_2	ANOVA	1.9229×10^{-28}

4.1.2 Results for Env2

Table 4.5 presents the results obtained from the simulations in Env2. A significant reduction in both μ and σ for displacements was observed when using MHRTSN-PSO compared to MHRTSN-GA. This reduction was approximately 31.20% for simulations in G3 and around 28.05% for simulations in G4. For the same scenario, the variation in traveled distance was 13.42% for G3 and 14.21% for G4, indicating that the reduction in the number of displacements was greater than the reduction in distance traveled. In a closed environment with more obstacles detected and calculated, MHRTSN-PSO demonstrated an advantage of 42.16% and 38.53% in processing time for groups G3 and G4, respectively.

Table 4.5: Simulation results for Env2: Clock times, CPU times, Displacements, and Distances.

Simulation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S5	G3	3.0156	0.2034	4.3943	0.3266	28.9333	3.2087	25.3632	2.5197
S6	G3	2.9312	0.1830	4.1977	0.3293	29.3000	2.9682	25.7195	2.5127
S7	G4	2.1947	0.0959	3.2733	0.1269	28.1833	3.0795	24.7751	2.6665
S8	G4	2.2218	0.1276	3.3962	0.2346	28.7000	3.4220	25.1656	2.8514
S21	G3	2.0065	0.1369	2.8166	0.1992	28.2143	2.9137	25.4554	2.6798
S22	G3	1.9722	0.1070	2.7486	0.1398	27.9661	2.4144	25.0109	2.1882
S23	G4	1.4153	0.0731	2.0848	0.1015	25.1667	2.8877	23.0519	2.4214
S24	G4	1.4241	0.0662	2.0938	0.0958	25.3833	2.5696	23.3478	2.3484

Figures 4.5 and 4.6 illustrate the mean and standard deviation of indicators I_1 and I_2 , as well as the mean for I_3 , for groups G3 and G4.

As shown in Figure 4.5, the MHRTSN-PSO strategy demonstrated a reduction in average processing time compared to MHRTSN-GA for both G3 and G4. The variance of results was also lower for MHRTSN-PSO, indicating greater consistency in performance.

Figure 4.6 illustrates indicator I_2 for Env2. In this environment, the difference in average distances traveled between the MHRTSN-GA and MHRTSN-PSO strategies was more pronounced. On average, MHRTSN-PSO covered longer distances with lower variation compared to the GA-based approach.

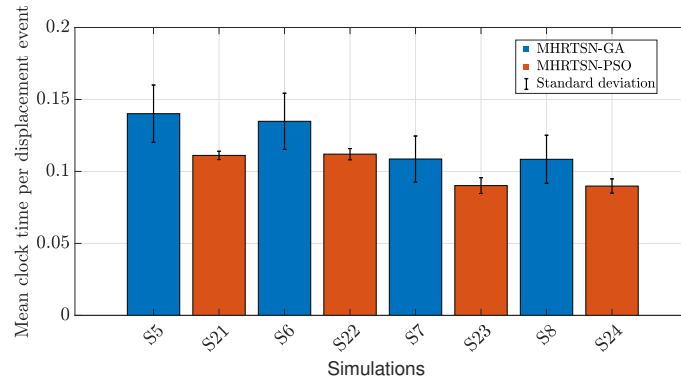


Figure 4.5: Mean and Standard Deviation of $I1$ in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.

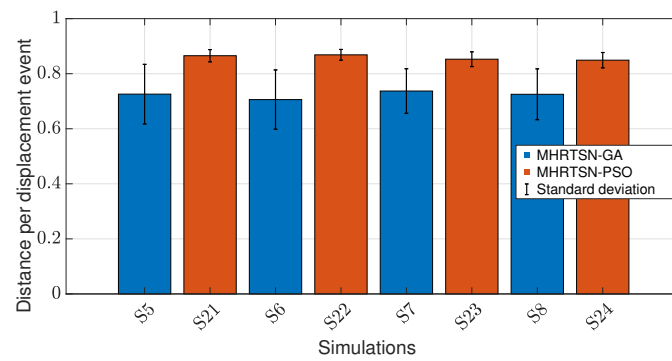


Figure 4.6: Mean and Standard Deviation of $I2$ in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.

As shown in Figure 4.7, the behavior observed for indicator I_3 in Env2 followed a pattern similar to that in Env1, where the total processing time was higher for the GA-based approach than for PSO.

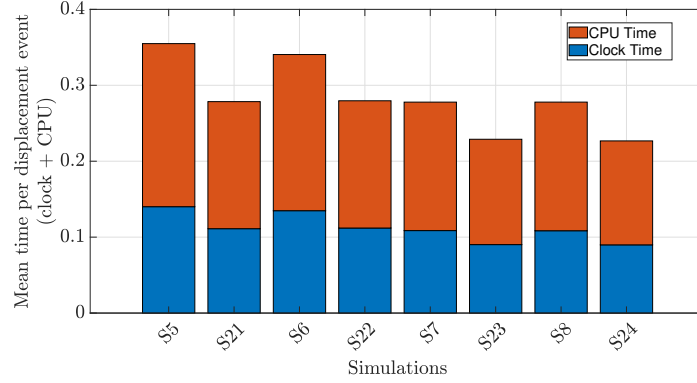


Figure 4.7: Mean processing time, indicator I_3 , in Env2 During MHRTSN-GA and MHRTSN-PSO Simulations.

The statistical test results obtained from the simulations illustrated in Figures 4.5-4.7 are presented in Table 4.6. Initially, the KS test was performed to assess the normality of the data. Since none of the indicators exhibited a normal distribution, the KW test was subsequently applied to all cases. In both groups, the null hypothesis (H_0) was rejected, as all p -values were below the significance threshold ($\alpha = 0.05$). These findings reveal significant differences among one or more groups, highlighting variations in the performance of the evaluated techniques with respect to indicators I_1 and I_2 (Mean Distance per Displacement).

Table 4.6: Statistical test results for indicators I_1 and I_2 in Env2 for Groups G_3 and G_4 .

Group	Indicator	Test Applied	p -value
G_3	I_1	KW	1.4718×10^{-27}
G_3	I_2	KW	2.4485×10^{-33}
G_4	I_1	KW	5.9719×10^{-18}
G_4	I_2	KW	3.2763×10^{-34}

4.1.3 Results for Env3

Table 4.7 presents the results obtained from simulations in Env3. No significant differences were observed in the number of displacements or the distance traveled between the configurations within each group. However, a reduction of approximately 44.55% and 41.21% in the clock time for meta-heuristic processing was observed for groups G_5 and G_6 , respectively.

Figures 4.8 and 4.9 show indicators I_1 and I_2 for Env3, respectively. Similar to previous results, MHRTSN-PSO demonstrated lower mean and variance for indicator I_1 compared to MHRTSN-GA, and a slightly higher average distance (I_2) than MHRTSN-GA.

Table 4.7: Simulation results for Env3: Clock times, CPU times, Displacements, and Distances.

Simu- lation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S9	G5	2.4682	0.1934	3.6928	0.3121	17.8333	2.5441	14.5864	1.7039
S10	G5	2.3475	0.1688	3.4955	0.2221	17.9000	2.9704	14.3906	1.7777
S11	G6	1.9194	0.0870	2.8827	0.1254	17.7667	2.5972	14.3946	1.7677
S12	G6	1.8866	0.0845	2.8323	0.1587	17.5500	2.1168	14.3109	1.4890
S25	G5	1.6647	0.0781	2.3643	0.1160	18.1167	1.7425	15.2469	1.6034
S26	G5	1.6668	0.0770	2.3522	0.1153	18.0000	1.4142	15.0099	1.3620
S27	G6	1.3459	0.0631	2.0075	0.0931	17.1667	2.0344	14.4135	1.6265
S28	G6	1.3494	0.0548	2.0123	0.0879	17.1833	1.5863	14.4383	1.4592

However, in this environment, with fewer obstacles, the differences between the two approaches for $I2$ were less pronounced.

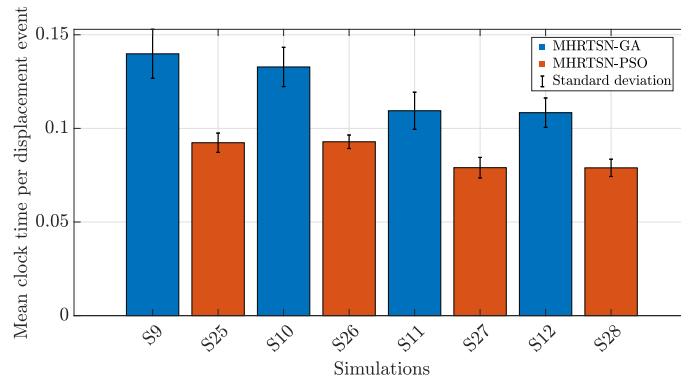
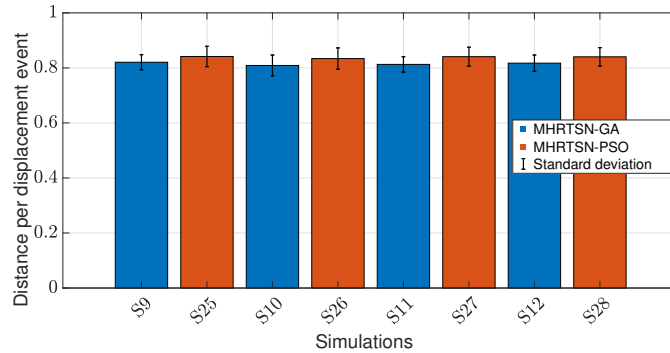
Figure 4.8: Mean and Standard Deviation of $I1$ in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.Figure 4.9: Mean and Standard Deviation of $I2$ in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.

Figure 4.10 illustrates indicator I_3 obtained from simulations in Env3. This figure shows that simulations using MHRTSN-GA required more processing time compared to MHRTSN-PSO in this environment.

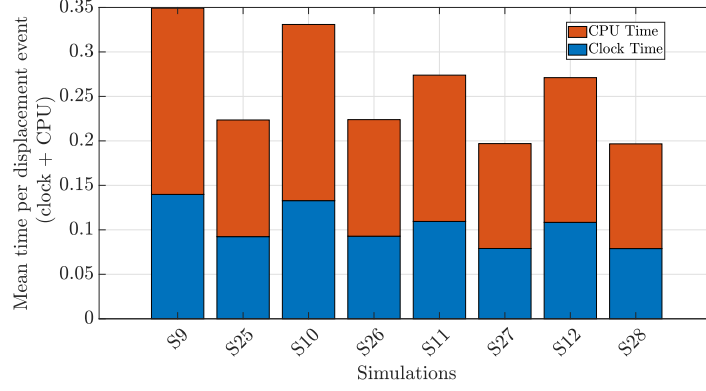


Figure 4.10: Mean processing time, indicator I_3 , in Env3 During MHRTSN-GA and MHRTSN-PSO Simulations.

The statistical test results for the simulations illustrated in Figures 4.8-4.10 are summarized in Table 4.8. The KS test was first applied to check data normality. Since the data for I_1 did not follow a normal distribution, the KW test was applied. For I_2 , where normality was confirmed, the ANOVA test was used. In all cases, the null hypothesis (H_0) was rejected, as the p -values were below the significance level ($\alpha = 0.05$). These results demonstrate statistically significant differences among the groups, highlighting variations in the performance of the evaluated techniques based on indicators I_1 and I_2 .

Table 4.8: Statistical test results for indicators I_1 and I_2 in Env3 for Groups G_5 and G_6 .

Group	Indicator	Test Applied	p -value
G_5	I_1	KW	3.7278×10^{-39}
G_5	I_2	ANOVA	4.2872×10^{-6}
G_6	I_1	KW	5.3287×10^{-38}
G_6	I_2	ANOVA	4.9678×10^{-8}

4.1.4 Results for Env4

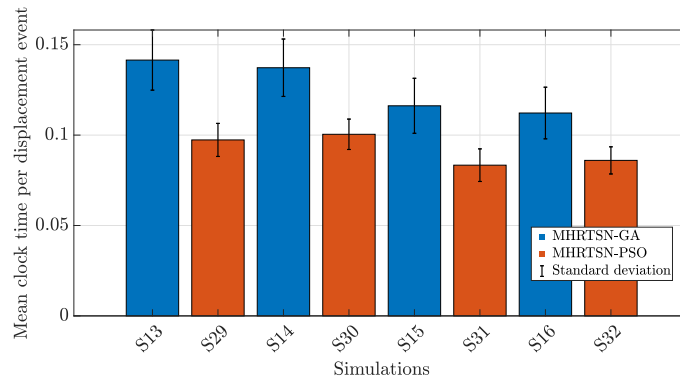
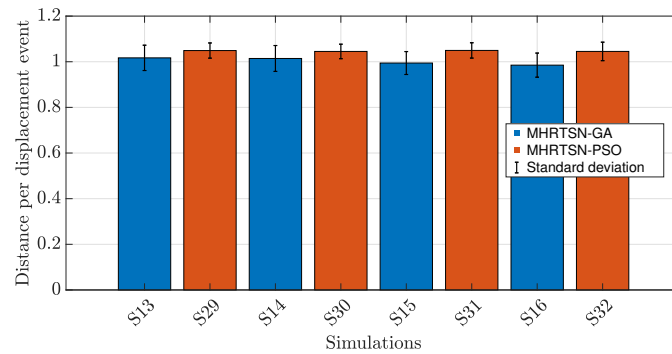
Table 4.9 presents the results obtained from simulations in Env4. The results indicate that processing time was lower for PSO-based simulations within the same group compared to GA. This reduction was approximately 41.21% for G_7 and 48.47% for G_8 .

Figures 4.11, 4.12, and 4.13 illustrate the mean and standard deviation of indicators I_1 and I_2 , as well as the mean of indicator I_3 for groups G_7 and G_8 .

Figures 4.11, 4.12, and 4.13 show that indicators I_1 , I_2 , and I_3 followed a pattern similar to other environments. Simulations using GA required more processing time and traveled slightly shorter distances compared to the PSO-based solution.

Table 4.9: Simulation results for Env4: Clock times, CPU times, Displacements, and Distances.

Simu- lation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S13	G7	2.4447	0.2422	3.6858	0.3146	17.7000	3.8397	17.8308	2.9924
S14	G7	2.4410	0.2504	3.6603	0.3497	18.2167	4.0294	18.3114	3.2256
S15	G8	1.9430	0.1434	2.9067	0.1882	17.2500	4.2096	17.0096	3.3815
S16	G8	1.9263	0.1338	2.8662	0.1969	17.6500	3.9405	17.2540	3.1600
S29	G7	1.6056	0.1245	2.3090	0.1608	16.7500	2.7966	17.5086	2.5588
S30	G7	1.5645	0.1171	2.2458	0.1511	15.7833	2.5955	16.4461	2.3858
S31	G8	1.3158	0.0723	1.9653	0.1021	16.0667	2.6887	16.8075	2.4839
S32	G8	1.2903	0.0634	1.9290	0.0999	15.1833	2.1870	15.8221	1.9710

Figure 4.11: Mean and Standard Deviation of Indicator $I1$ in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.Figure 4.12: Mean and Standard Deviation of Indicator $I2$ in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.

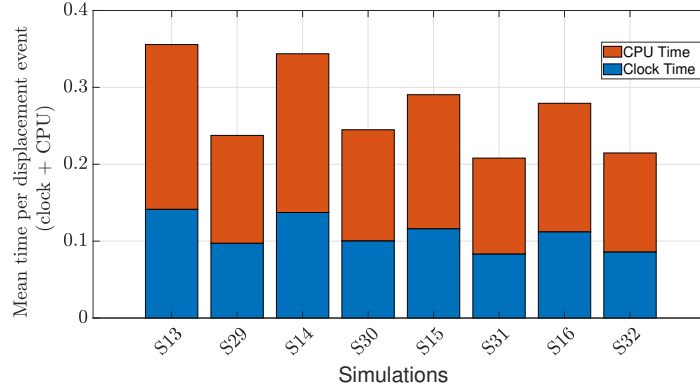


Figure 4.13: Mean Clock Time and CPU Time per Displacement Event, Indicator I_3 , in Env4 During MHRTSN-GA and MHRTSN-PSO Simulations.

The statistical analysis results for the simulations in Figures 4.11-4.13 are presented in Table 4.10. Based on the KS test, the data did not meet the normality assumption for any indicator. Consequently, the KW test was employed in all cases. The null hypothesis (H_0) was rejected for both groups since all p -values were below the significance level ($\alpha = 0.05$). These outcomes demonstrate statistically significant differences between one or more groups, reflecting performance variations among the evaluated techniques for indicators I_1 and I_2 .

Table 4.10: Statistical test results for indicators I_1 and I_2 in Env4 for Groups G_7 and G_8 .

Group	Indicator	Test Applied	p -value
G_7	I_1	KW	3.4529×10^{-37}
G_7	I_2	KW	9.2636×10^{-5}
G_8	I_1	KW	4.3202×10^{-31}
G_8	I_2	KW	8.6461×10^{-16}

4.2 Dynamic Environments

Two dynamic environments were proposed to evaluate the MHRTSN strategy under these conditions. Figure 4.14 illustrates each of these environments. In the figure, the robot, positioned at the initial navigation point, is represented by a blue circle, while the final navigation point is indicated by a black circle. The robot's objective is to navigate from the starting point to the target in an unknown environment, avoiding static obstacles represented by brown and white boxes and dynamic obstacles represented by red boxes. Figure 4.14(a) illustrates Environment 5 (Env5), while Figure 4.14(b) illustrates Environment 6 (Env6). The initial and final points for the simulations in Env5 and Env6 are listed in Table 4.11.

In Figure 4.14(a), the robot navigates in an outdoor environment with obstacles, where three red boxes are dynamically added over time (see Table 4.12). These obstacles are

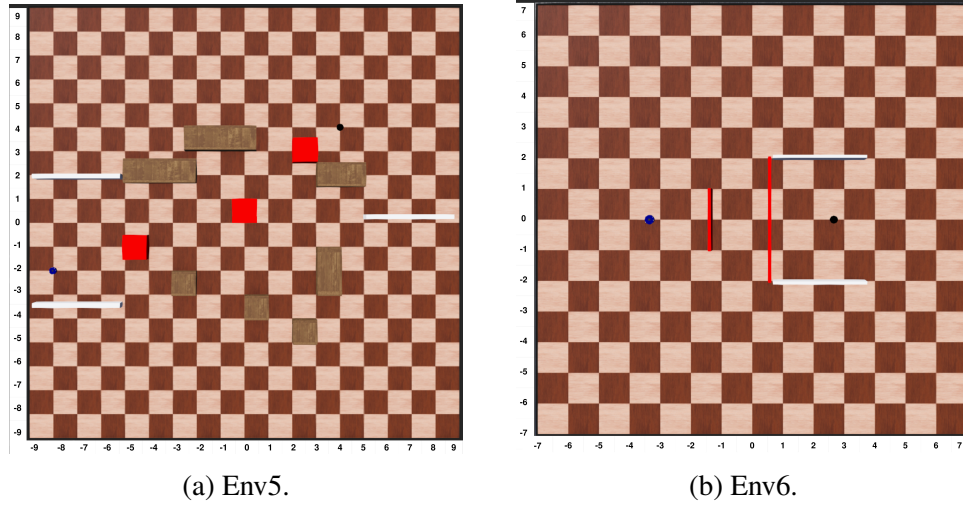


Figure 4.14: Representation of the dynamic environments used in the simulations with MHRTSN-GA and MHRTSN-PSO. The robot is indicated by a blue circle at the initial position, while the final navigation point is shown by a black circle. Static obstacles are represented by brown and white boxes, and dynamic obstacles are shown as red boxes. In these environments, the floor is marked with 1 by 1 meter squares in a checkerboard pattern.

Table 4.11: Initial and final points for each dynamic environment.

Environment	Initial Point	Final Point
Env5	$(-8.0, -2.0)$	$(4.0, 4.0)$
Env6	$(-4.0, 0.0)$	$(2.0, 0.0)$

strategically placed to block the most direct and intuitive paths, making the robot's trajectory more challenging. The timing for adding each obstacle was adjusted based on the robot's speed to interfere with the route without being detected in advance.

Table 4.12: Dimensions, locations, and times for adding boxes in Env5.

Box	Translation (x, y)	Size (x, y, z)	Time to Add (s)
Box 1	(-4.5, -1.0)	(1, 1, 1)	45
Box 2	(0.0, 0.5)	(1, 1, 1)	180
Box 3	(2.5, 3.0)	(1, 1, 1)	330

In Figure 4.14(b), the robot starts navigating in an environment with few obstacles. However, as the simulation progresses, two large dynamic obstacles, represented in red, are introduced into the environment, acting as barriers. The timing and locations of these dynamic obstacles are described in Table 4.13.

Table 4.13: Dimensions, locations, and times for adding boxes in Env6.

Box	Translation (x, y)	Size (x, y, z)	Time to Add (s)
Box 1	(-2.0, 0.0)	(0.1, 2, 1)	25
Box 2	(-0.1, 0.0)	(0.1, 4, 1)	180

The MHRTSN-GA and MHRTSN-PSO versions were tested in these dynamic environments. For each environment, a specific configuration pattern was applied to each approach, resulting in a simulation group per environment, as shown in Table 4.14.

Table 4.14: Proposed parameter configuration for simulations in the dynamic environments using MHRTSN-GA and MHRTSN-PSO techniques.

Simu- lation	Group	Environ- ment	Meta- heuristic	Popu- lation	Max. Generations	Cross- over
S33	G9	Env5	GA	30	10	70%
S35	G10	Env6	GA	30	10	70%

Simu- lation	Group	Environ- ment	Meta- heuristic	Popu- lation	Max. Generations	Cogni- tive	Social
S34	G9	Env5	PSO	30	10	1	1.5
S36	G10	Env6	PSO	30	10	1	1.5

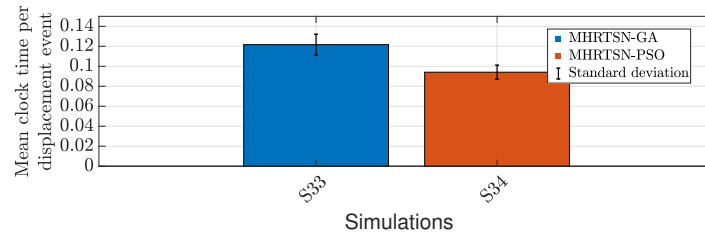
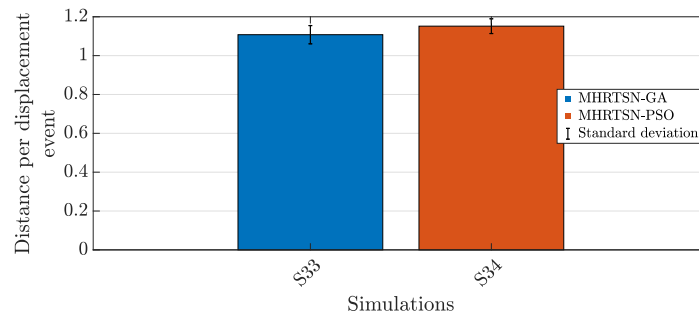
4.2.1 Results for Env5

Table 4.15 presents the results of the simulations conducted in Env5. The results show that the processing time required for the PSO-based solution was approximately 39.44% lower than that of the GA-based solution.

Table 4.15: Simulation results for Env5: Clock times, CPU times, Displacements, and Distances.

Simu- lation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S33	G9	1.7592	0.0722	2.8217	0.1348	14.6333	2.0653	16.1201	1.4959
S34	G9	1.2616	0.0468	1.9118	0.0752	13.5333	1.7172	15.5221	1.2627

Figures 4.15 and 4.16 illustrate the mean and standard deviation of indicators I_1 and I_2 for G_9 . It was observed that the simulations using MHRTSN-PSO presented lower variance and mean for I_1 and a higher mean for I_2 compared to MHRTSN-GA. Figure 4.17 illustrates the mean of indicator I_3 for G_9 . This figure shows that PSO processed target points faster than GA.

Figure 4.15: Mean and Standard Deviation of Indicator I_1 in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.Figure 4.16: Mean and Standard Deviation of Indicator I_2 in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.

The statistical analysis for Env5 is summarized in Table 4.16. According to the KS test, neither of the indicators followed a normal distribution. Since the comparison involved only two groups, the MWU test was applied. In both cases, the null hypothesis (H_0) was rejected, as the p -values were significantly lower than the significance level ($\alpha = 0.05$). These results confirm statistically significant differences between the groups for indicators I_1 and I_2 .

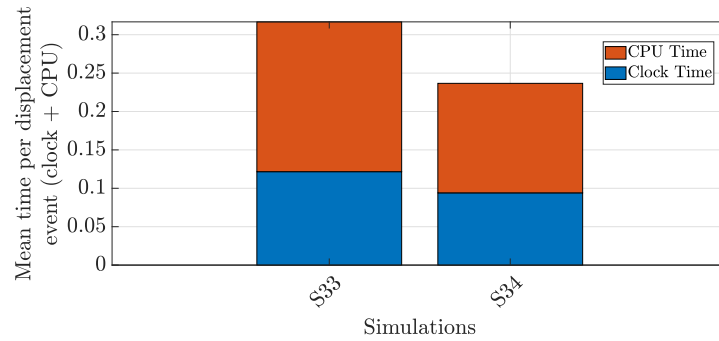


Figure 4.17: Mean Clock Time and CPU Time per Displacement Event, Indicator I_3 , in Env5 During MHRTSN-GA and MHRTSN-PSO Simulations.

Table 4.16: MWU test results for indicators I_1 and I_2 in Env5 for Group G_9 .

Indicator	Test Applied	p -value
I_1	MWU	3.8892×10^{-17}
I_2	MWU	7.8313×10^{-12}

4.2.2 Results for Env6

Table 4.17 presents the results obtained from simulations in Env6. The results indicate that processing time was lower for PSO-based simulations within the same group. This reduction was approximately 24.80% for G_{10} .

Table 4.17: Simulation results for Env6: Clock times, CPU times, Displacements, and Distances.

Simulation	Group	Clock times (s)		CPU times (s)		Displacements		Distances (m)	
		μ	σ	μ	σ	μ	σ	μ	σ
S35	G10	1.9279	0.1036	3.0060	0.1860	20.1166	0.9502	16.5257	0.4833
S36	G10	1.4497	0.0552	2.1653	0.0797	21.1167	2.0256	18.0422	1.8626

The behavior of indicators I_1 , I_2 , and I_3 followed the pattern previously observed. PSO-based simulations presented lower variance and mean for I_1 , higher mean value for I_2 , and lower mean time for I_3 (see Figures 4.18, 4.19, and 4.20).

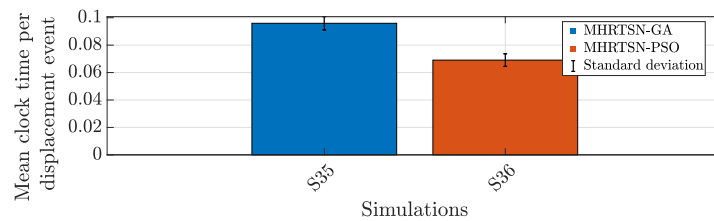


Figure 4.18: Mean and Standard Deviation of Indicator I_1 in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.

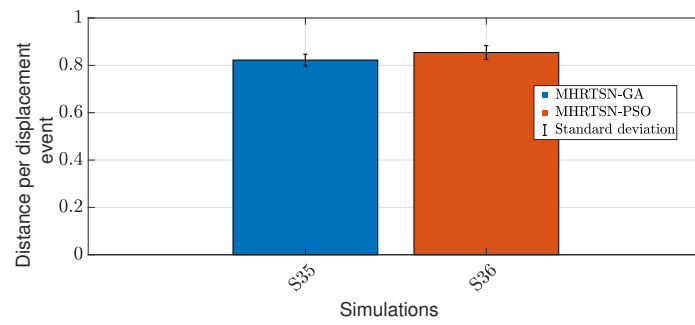


Figure 4.19: Mean and Standard Deviation of Indicator I_2 in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.

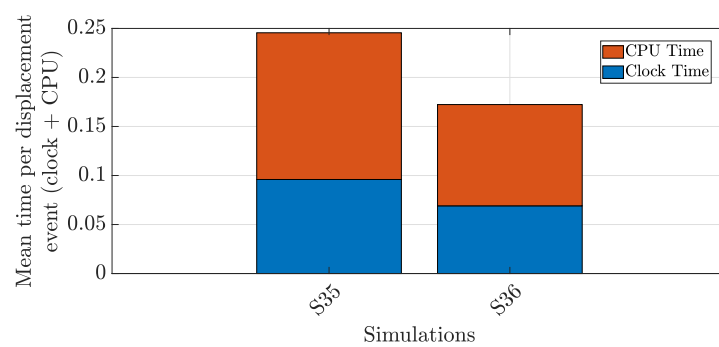


Figure 4.20: Mean Clock Time and CPU Time per Displacement Event, Indicator I_3 , in Env6 During MHRTSN-GA and MHRTSN-PSO Simulations.

The statistical analysis results for Env6 are summarized in Table 4.18. According to the KS test, the data for I_1 did not follow a normal distribution, while I_2 was normally distributed. Therefore, the Mann-Whitney U (MWU) test was applied for I_1 , and the ANOVA test was used for I_2 . In both cases, the null hypothesis (H_0) was rejected, as the p -values were significantly lower than the significance level ($\alpha = 0.05$). These results confirm statistically significant differences between the groups for both indicators.

Table 4.18: Statistical test results for indicators I_1 and I_2 in Env6 for Group G_{10} .

Indicator	Test Applied	p -value
I_1	MWU	3.5566×10^{-21}
I_2	ANOVA	2.5116×10^{-9}

4.3 Comparison with Other Approaches

The MHRTSN strategy was tested in three environments proposed in the referenced works (Yan et al. 2022, Zheng et al. 2023, Silva et al. 2018). Table 4.19 associates the information for each of these environments, linking the reference with the simulated environment. Figures 4.21, 4.22, and 4.23 illustrate these environments, named respectively as Env7, Env8, and Env9. In these illustrations, the environments and the paths taken by the techniques are presented.

Table 4.19: Environments used for comparisons with other techniques from the literature.

Environment	Reference	Figure
Env7	(Yan et al. 2022)	Figure 4.21
Env8	(Zheng et al. 2023)	Figure 4.22
Env9	(Silva et al. 2018)	Figure 4.23

The simulations followed the methodology described at the beginning of Section 3, using the predefined common parameters, as well as the number of simulations per environment, resulting in the average and standard deviation data. To ensure a fair comparison with other techniques, a safety factor of $\alpha = 1.0$ was adopted, allowing the robot to approach obstacles in a similar manner to the compared approaches. The meta-heuristic parameters used in the MHRTSN simulations were: $K = 30$, $N_{Max} = 10$, with crossover = 70% for MHRTSN-GA, and cognitive = 1 and social = 1.5 for MHRTSN-PSO. These values were defined based on results obtained during tests in static and dynamic environments, aiming for a balance between processing time, distance traveled, and navigation quality.

The paths of the other approaches, illustrated in Figures 4.21, 4.22, and 4.23, were reconstructed based on an approximate mapping of the points extracted from the images presented in their respective works. The path generated by the MHRTSN strategy, in turn, corresponds to the median value in terms of distance traveled, based on the 60 simulations conducted. It is observed that, unlike the compared approaches, the MHRTSN strategy

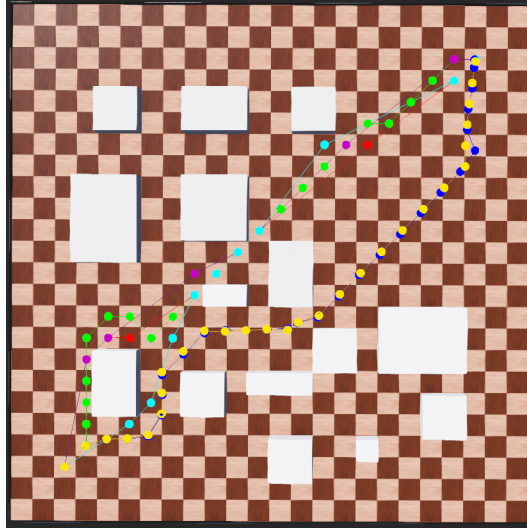


Figure 4.21: Comparison of different paths in the environment proposed in (Yan et al. 2022). The trajectories are indicated by the following colors: ACO (green), SSA (red), ACO+GA (purple), MSSA (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the bottom-left corner, while the endpoint is located in the top-right corner.

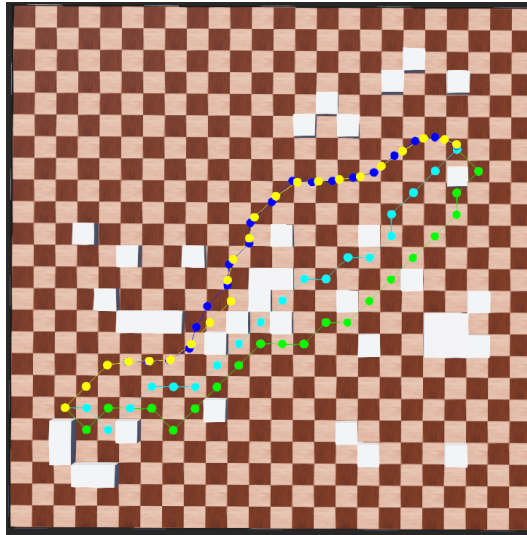


Figure 4.22: Comparison of different paths in the environment proposed in (Zheng et al. 2023). The trajectories are indicated by the following colors: rPSO (green), apfrPSO (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the far-left point, while the endpoint is located in the top-right corner.

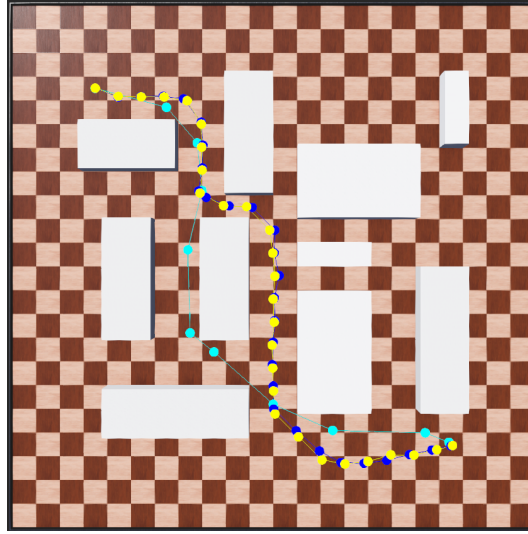


Figure 4.23: Comparison of different paths in the environment proposed in (Silva et al. 2018). The trajectories are indicated by the following colors: DPNA-GA (cyan), MHRTSN-GA (blue), and MHRTSN-PSO (yellow). Obstacles are represented by white areas. The robot's starting point is the top-left corner, while the endpoint is located in the bottom-right corner.

maintained a safe distance from obstacles during navigation. The other approaches do not consider the robot's dimensions and physical characteristics, which in some cases result in trajectories very close to obstacles.

Table 4.20 presents the results of simulations conducted in the environments shown in Figures 4.21-4.23. It was observed that the average processing time was lower in the simulations using MHRTSN-PSO compared to MHRTSN-GA.

Table 4.20: Results of the simulations of environments.

Environ- ment	MHRTSN Version	Displace- ment	Path Distance	Clock Time	Time per Displace- ment Event
Env7	GA	27.96	30.43 m	2.07 s	0.0741 s/disp
	PSO	27.18	30.27 m	1.64 s	0.0606 s/disp
Env8	GA	22.83	24.94 m	2.30 s	0.0851 s/disp
	PSO	22.20	24.49 m	1.57 s	0.0708 s/disp
Env9	GA	27.91	27.71 m	2.05 s	0.0734 s/disp
	PSO	27.43	27.69 m	1.57 s	0.0575 s/disp

Table 4.21 shows a comparison of the paths traversed by other approaches in relation to MHRTSN. The data related to the solutions in (Yan et al. 2022, Zheng et al. 2023) were obtained by calculating the approximate distance of the points mapped from the images in the respective studies. In the case of the solution presented in (Silva et al. 2018), the distance values refer to the average presented in the article itself.

Table 4.21: Comparison of navigation techniques across different environments.

Tech- nique	Meta- heuristic	Environ- ment	Route (m)	Prior Know- ledge of Environ- ment	Safe Dista- nce	Fixed Indi- vidual Size	Popu- lation Size
-	ACO	Env7	30.63	Yes	No	No	-
-	SSA ¹	Env7	28.18	Yes	No	No	-
-	ACO+GA	Env7	28.22	Yes	No	No	-
-	MSSA ²	Env7	27.78	Yes	No	No	-
MHRTSN	GA	Env7	30.43	No	Yes	Yes	30
MHRTSN	PSO	Env7	30.27	No	Yes	Yes	30
rPSO ³	PSO	Env8	27.21	Yes	No	No	-
apfrPSO ³	PSO	Env8	24.38	Yes	No	No	-
MHRTSN	GA	Env8	24.94	No	Yes	Yes	30
MHRTSN	PSO	Env8	24.49	No	Yes	Yes	30
DPNA-GA ⁴	GA	Env9	25.95	No	No	Yes	30
MHRTSN	GA	Env9	30.43	No	Yes	Yes	30
MHRTSN	PSO	Env9	30.27	No	Yes	Yes	30

¹ SSA technique referenced from (Xue & Shen 2020).² MSSA technique referenced from (Yan et al. 2022).³ rPSO and apfrPSO techniques referenced from (Zheng et al. 2023).⁴ DPNA-GA technique referenced from (Silva et al. 2018).

The data presented in Table 4.21 indicate that, in Env7 and Env8, the MHRTSN strategy generated paths comparable to those produced by other approaches, without requiring prior knowledge of the environment. In this case, both navigation and planning occurred fully online. A key aspect of MHRTSN is that, regardless of the environment's dimensions, the information contained in each individual remains constant, unlike the other approaches used in Env7 and Env8. This characteristic reduces computational complexity as the environment scales, mitigating the scalability challenges faced by the techniques applied in these scenarios.

In Env9, the DPNA-GA approach also operates without prior knowledge of the environment, allowing planning and navigation to be conducted online with fixed-size information in the individuals, similar to MHRTSN. However, DPNA-GA does not guarantee safe trajectories, resulting in movements in close proximity to obstacles (see Figures 4.21 - 4.23). This limitation is also observed in the approaches applied in Env7 and Env8. Unlike MHRTSN, these techniques do not explicitly consider constraints to prevent excessive proximity to obstacles during navigation. As a result, the paths generated often require post-processing adjustments to accommodate the robot's physical and dynamic constraints, introducing an additional step in the planning and navigation process.

Previous studies (Yan et al. 2022, Zheng et al. 2023) do not explicitly specify the population size used to generate their results. However, the study by (Silva et al. 2018) employed a population of 30 individuals, which is the same population size adopted in this work. Additionally, (Yan et al. 2022) does not describe the navigation strategy used for comparing the metaheuristics, merely stating that the environment is represented as a grid and that the metaheuristics were tested under the same conditions.

4.4 Analysis of Results

The results indicate that MHRTSN-PSO outperformed MHRTSN-GA in all tested scenarios, both static and dynamic, in terms of processing time and average displacement (see Section 4.1, 4.2). This behavior is due to the dynamics of PSO, where the social influence among particles allows for faster and more efficient convergence. On the other hand, GA exhibited longer processing times, reflecting the nature of its selection and recombination operators, which require more iterations to refine solutions.

Comparing with the results from (Yan et al. 2022, Zheng et al. 2023, Silva et al. 2018) (see Section 4.3), it is observed that MHRTSN-PSO generated trajectories with similar lengths to the approaches found in the literature while ensuring greater safety by maintaining a larger distance from obstacles and achieving shorter processing times. This outcome reinforces the effectiveness of combining metaheuristics with potential fields for safe trajectory planning, mitigating collisions without compromising navigation efficiency. Furthermore, the data suggest that hybrid approaches are promising for solving robotic navigation problems (Chakraborty et al. 2022).

Another relevant aspect is that, unlike the methodologies in (Yan et al. 2022, Zheng et al. 2023, Silva et al. 2018), MHRTSN explicitly considers the robot's dimensions and its safety during navigation, reducing risks associated with trajectories too close to obstacles. The tests also highlight the capability of MHRTSN-GA and MHRTSN-PSO to plan

and execute trajectories even in dynamic environments, demonstrating robustness against changes in the scenario. This behavior suggests that the model can be applied in real-world contexts, such as autonomous inspection robots and mobile vehicles in unknown environments.

4.5 Demonstration video

To better understand the results presented in this work, a demonstration video is available at the following link: https://drive.google.com/drive/folders/1N3gB53Zsn_EfBaTN23m5kr_lKEet95jp. The video illustrates the practical application of the MHRTSN strategy in different scenarios, highlighting mobile robots' safe and efficient navigation in static and dynamic environments. Watching the video is recommended to observe the effectiveness of the proposed solution visually.

Chapter 5

Conclusion

The objective of this work is to present a new proposal for mapping and navigation in autonomous robotic systems, called MHRTSN. This approach is designed for unknown static and dynamic environments, combining the use of potential fields with population-based meta-heuristics to navigate the environment safely and efficiently. Simulations conducted with the MHRTSN-GA and MHRTSN-PSO versions demonstrated that this approach can adapt to new obstacles, overcoming challenges and generating an optimized trajectory based on the robot's and the environment's constraints.

The simulations showed that both MHRTSN-GA and MHRTSN-PSO achieved feasible solutions, with MHRTSN-PSO presenting a clear advantage in terms of processing time and distance traveled per displacement event. Specifically, in Env3, MHRTSN-PSO achieved reductions of up to 44.55% in processing time compared to MHRTSN-GA in group *G5*, and up to 41.21% in group *G6*. In dynamic environments (Env4), the reduction reached 48.47% for group *G8*. Additionally, in constrained indoor scenarios (Env2), MHRTSN-PSO demonstrated a reduction of approximately 31.20% in the number of displacements for group *G3*, and 28.05% for group *G4*, while achieving a processing time reduction of 42.16% in group *G3* and 38.53% in group *G4*. These results indicate that MHRTSN-PSO provides more efficient navigation while maintaining similar or greater traveled distances per displacement event.

A comparison with other techniques from the literature was conducted, revealing that the MHRTSN strategy generates trajectories comparable to those found in the literature, without requiring prior knowledge of the environment. This comparison also highlighted the advantage of having fixed-size information in the individuals, maintaining a constant computational cost to process the individual as the environment's dimension increases. The comparison further illustrated differences in the trajectories generated by some literature approaches, which produce paths close to obstacles, compared to the safer paths generated by MHRTSN. Finally, the series of simulations performed in diverse environments successfully evaluated and validated the MHRTSN strategy for path generation and navigation, demonstrating its robustness and applicability in real-world scenarios.

5.1 Future Work

Future developments of the MHRTSN strategy aim to enhance its adaptability and efficiency in real-world applications. One key direction is the incremental construction of an environmental map using the information collected at each displacement event. By incorporating a memory mechanism, the system can refine its understanding of the environment over time, improving navigation in partially known or dynamic scenarios.

Another promising improvement is to make the individual creation radius dynamic. Instead of using a fixed radius, an adaptive strategy based on environmental complexity and obstacle density could be implemented. Similarly, dynamically adjusting the population size would be another enhancement, allowing the algorithm to regulate the number of individuals in the search process according to the characteristics of each displacement. These adaptations would enable the algorithm to refine its search space in real time, optimizing computational resources while ensuring more efficient path planning, particularly in cluttered environments.

Finally, a thorough experimental validation in physical robotic platforms should be conducted to bridge the gap between simulation and real-world deployment, ensuring that the method remains robust under real-world uncertainties, sensor noise, and system constraints.

Bibliography

- Ait Saadi, Amylia, Assia Soukane, Yassine Meraihi, Asma Benmessaoud Gabis, Seyedali Mirjalili & Amar Ramdane-Cherif (2022), ‘Uav path planning using optimization approaches: A survey’, *Archives of Computational Methods in Engineering* **29**(6), 4233–4284.
- Bulut, Vahide (2023), ‘Path planning of mobile robots in dynamic environment based on analytic geometry and cubic b zier curve with three shape parameters’, *Expert Systems with Applications* **233**, 120942.
- Cardoza Plata, Jose Eduardo, Mauricio Olgu n Carbajal, Juan Carlos Herrera Lozada, Jacobo Sandoval Gutierrez, Israel Rivera Zarate & Jose Felix Serrano Talamantes (2022), ‘Simulation and implementation of a mobile robot trajectory planning solution by using a genetic micro-algorithm’, *Applied Sciences* **12**(21).
- Chakraborty, Suprava, Devaraj Elangovan, Padma Lakshmi Govindarajan, Mohamed F. ELnaggar, Mohammed M. Alrashed & Salah Kamel (2022), ‘A comprehensive review of path planning for agricultural ground robots’, *Sustainability* **14**(15).
- Chen, Yanjie, Jiacheng Liang, Yaonan Wang, Qi Pan, Jianhao Tan & Jianxu Mao (2020), ‘Autonomous mobile robot path planning in unknown dynamic environments using neural dynamics’, *Soft Computing* **24**(18), 13979–13995.
- de Oliveira,  t la V. F. M. & Marcelo A. C. Fernandes (2016), ‘Dynamic planning navigation strategy for mobile terrestrial robots’, *Robotica* **34**(3), 568–583.
- Inc., The MathWorks (2022), ‘Matlab version: 9.12.0 (r2022a) license 596681’.
- Josef, Shirel & Amir Degani (2020), ‘Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain’, *IEEE Robotics and Automation Letters* **5**(4), 6748–6755.
- Lamini, Chaymaa, Said Benhl ma & Ali Elbekri (2018), ‘Genetic algorithm based approach for autonomous mobile robot path planning’, *Procedia Computer Science* **127**, 180–189.
- Latombe, Jean Claude (1991), *Robot Motion Planning*, Springer Science+Business Media.
- Lin, Shiwei, Ang Liu, Jianguo Wang & Xiaoying Kong (2022), ‘A review of path-planning approaches for multiple mobile robots’, *Machines* **10**(9).

- Montiel, Oscar, Ulises Orozco-Rosas & Roberto Sepúlveda (2015), 'Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles', *Expert Systems with Applications* **42**(12), 5177–5191.
- Pattnaik, S. K., Mishra, D. & Panda, S. (2022), 'A comparative study of meta-heuristics for local path planning of a mobile robot', *Engineering Optimization* **54**(1), 134–152.
- Qi, Jie, Hui Yang & Haixin Sun (2021), 'Mod-rrt*: A sampling-based algorithm for robot path planning in dynamic environment', *IEEE Transactions on Industrial Electronics* **68**(8), 7244–7251.
- Qin, Hongwei, Shiliang Shao, Ting Wang, Xiaotian Yu, Yi Jiang & Zonghan Cao (2023), 'Review of autonomous path planning algorithms for mobile robots', *Drones* **7**(3).
- Rajwar, Kanchan, Kusum Deep & Swagatam Das (2023), 'An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges', *Artificial Intelligence Review* **56**(11), 13187–13257.
- Roy, Dibyendu, Arijit Chowdhury, Madhubanti Maitra & Samar Bhattacharya (2021), 'Geometric region-based swarm robotics path planning in an unknown occluded environment', *IEEE Transactions on Industrial Electronics* **68**(7), 6053–6063.
- Samadi Gharajeh, Mohammad & Hossein B. Jond (2022), 'An intelligent approach for autonomous mobile robots path planning based on adaptive neuro-fuzzy inference system', *Ain Shams Engineering Journal* **13**(1), 101491.
- Shi, Kun, Zhengtian Wu, Baoping Jiang & Hamid Reza Karimi (2023), 'Dynamic path planning of mobile robot based on improved simulated annealing algorithm', *Journal of the Franklin Institute* **360**(6), 4378–4398.
- Silva, Caroline A. D., Átila V. F. M. De Oliveira & Marcelo A. C. Fernandes (2018), 'Validation of a dynamic planning navigation strategy applied to mobile terrestrial robots', *Sensors* **18**(12).
- Sivaranjani, S, Divya A Nandesh, K Gayathri, R Ramanathan et al. (2021), An investigation of bug algorithms for mobile robot navigation and obstacle avoidance in two-dimensional unknown static environments, *em* '2021 International Conference on Communication information and Computing Technology (ICCICT)', IEEE, pp. 1–6.
- Sánchez-Ibáñez, José Ricardo, Carlos J. Pérez-del Pulgar & Alfonso García-Cerezo (2021), 'Path planning for autonomous mobile robots: A review', *Sensors* **21**(23).
- Teli, Tawseef Ahmed & M Arif Wani (2021), 'A fuzzy based local minima avoidance path planning in autonomous robots', *International Journal of Information Technology* **13**(1), 33–40.

- Vasconcelos, João Vítor R., Alexandre S. Brandão & Mário Sarcinelli-Filho (2020), ‘Real-time path planning for strategic missions’, *Applied Sciences* **10**(21).
- Wang, Qian, Junli Li, Liwei Yang, Zhen Yang, Ping Li & Guofeng Xia (2022), ‘Distributed multi-mobile robot path planning and obstacle avoidance based on aco-dwa in unknown complex terrain’, *Electronics* **11**(14).
- Xu, Yiqi, Qiongqiong Li, Xuan Xu, Jiafu Yang & Yong Chen (2023), ‘Research progress of nature-inspired metaheuristic algorithms in mobile robot path planning’, *Electronics* **12**(15).
- Xue, Jiankai & Bo Shen (2020), ‘A novel swarm intelligence optimization approach: sparrow search algorithm’, *Systems Science & Control Engineering* **8**(1), 22–34.
- Yan, Yupei & Yangmin Li (2016), Mobile robot autonomous path planning based on fuzzy logic and filter smoothing in dynamic environment, *em* ‘2016 12th World Congress on Intelligent Control and Automation (WCICA)’, pp. 1479–1484.
- Yan, Zijian, Wenhui Luo, Lijuan Yu, Danhong Zhang, Bo Wang & Qi Li (2022), A mobile robot path planning based on modified sparrow search algorithm, *em* ‘2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC)’, pp. 1537–1541.
- Yu, Jinglun, Yuancheng Su & Yifan Liao (2020), ‘The path planning of mobile robot by neural networks and hierarchical reinforcement learning’, *Frontiers in Neuro-robotics* **14**, 63.
- Zheng, Li, Wenjie Yu, Guangxu Li, Guangxu Qin & Yunchuan Luo (2023), ‘Particle swarm algorithm path-planning method for mobile robots based on artificial potential fields’, *Sensors* **23**(13).
- Zhu, Zexuan, Fangxiao Wang, Shan He & Yiwen Sun (2015), ‘Global path planning of mobile robots using a memetic algorithm’, *International Journal of Systems Science* **46**(11), 1982–1993.