République Algérienne Démocratique et populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid -El Taref
Faculté des Sciences el da le Technologie
Département Informatique

الجمهورية الديمقراطية الشعبية الجزائرية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد - الطارف
كلية العلوم والتكنولوجيا
قسم اعلام الالي

MASTER THESIS IN

COMPUTER SCIENCE

Presented By

INES BOUTABIA

SPECIALITY: Intelligent Computer Science Systems

THEME

A Trajectory Planning using the Particle Swarm Optimization for an Autonomous Mobile Robot in an Environment with Obstacles

Presented on: June 2022

	Last and first name	*Grade:*	University
President	**Maatallah M**	**MCB**	**El-Tarf, University**
Supervisor	**Benmachiche A**	**MCA**	**El-Tarf, University**
Examiner	**Chemame C**	**MAA**	**El-Tarf, University**

University Year: 2021/2022

Thanks

My Thanks Go First And Foremost To God Almighty For The Will, Health, And Patience He Bestowed Upon Me During All These Years Of Study.

Special Gratitude Goes To Mr.Benmachiche Abdelmadjid, Senior Lecturer In The Department Of Computer Science At Chadli Ben Djedid University, Who Has Mentored And Put Up With Me Throughout This Thesis And All These Years, For His Valuable Advice And Support Throughout Difficult Phases. I Am Grateful For His Availability, Always Being There, And The Trust He Has Placed In Me. My Gratitude Also Goes To The Faculty And Staff Of The Department Of Computer Science.

I Sincerely Thank:

-Dr. Maatallah Majda- For Having Honored Me To Chair The Jury

-Mr. Chemmam chaouki-, For Agreeing To Be An Examiner.

A Huge Thank You To My Parents & My Family For Their Consideration And Concern For Me, Their Encouragement And Patience During The Course Of My Studies. Special Thanks Go To My Sister Abir For Taking Care Of Me When All Turned Their Backs On Me. My Thanks Also Go To My Best Friends, Houssem And Aya, For The Good Times And Bad We Passed Together.

Finally, I Thank All Those Who Helped Me.

I am Truly Grateful With The Bottom Of The Heart

Dedication

I Dedicate This Work To:

My Father May He Rest In Peace

My Mother

My Siblings

The Rest Of My Family

To All My Friends

Sincerely

Ines

TABLE OF CONTENTS

III.Chapter: Realization and Implementation

TABLE OF FIGURES

Chapter I

Chapter II

Chapter III

LISTE OF TABLES

ABBREVIATION LISTE

PSO: *Particle Swarm Optimization*

AMR: *Autonomous Mobile Robots*

FL : *Fuzzy Logic*

ACO: *Ant Colony Optimization Algorithm*

SA: *Simulated Annealing Algorithm*

NN: *Neural Network*

GA: *Genetic Algorithm*

BFO: *Bacterial Foraging Optimization*

gbest: *global best*

pbest : *personal best*

Abstract

In our day-to-day lives, we have difficulties functioning perfectly without the interference of robots for they have become an inseparable part of any daily aspect that we have seen for the past two decades. Most of us identify a robot as an entity or object pre-programmed by humans to accomplish a single function. Nonetheless, the researchers are putting more effort into developing an autonomous robot that can absorb knowledge from its surroundings and use it to complete given missions successfully. Like everything in existence, autonomous mobile robots too suffer from problems that lessen their efficiency and performance, the major being navigation, finding the shortest path, and avoiding obstacles. These issues have caused a setback for the evolution and progress of autonomous robots. In this thesis, we addressed these problems with particle swarm optimization, a meta-heuristic bio-inspired approach. We aim to develop a robot able to navigate from a given start point to a given endpoint while avoiding obstacles both stable and especially the moving, with the path traveled being the shortest possible. The PSO functions by analyzing the search environment through consecutive attempts. Then, the location of each particle in the search environment denotes a possible solution to the optimization problem. We applied the PSO and experimented with our algorithm in multiple conditions to view its efficiency and performance.

Keywords— PSO, Navigation, Autonomous mobile robot, Obstacles avoidance

ملخص

في حياتنا اليومية، نواجه صعوبات في العمل بشكل مثالي دون تدخل الروبوتات لأنها أصبحت جزءًا لا يتجزأ من أي جانب يومي رأيناه على مدار العقدين الماضيين. يعرف معظمنا الروبوت ككيان أو كائن مبرمج مسبقًا من قبل البشر لإنجاز وظيفة واحدة. ومع ذلك، يبذل الباحثون مزيدًا من الجهد في تطوير روبوت مستقل يمكنه استيعاب المعرفة من محيطه واستخدامها لإكمال المهام المحددة بنجاح. مثل كل شيء موجود، تعاني الروبوتات المتنقلة المستقلة أيضًا من مشاكل تقلل من كفاءتها وأدائها، وأهمها التنقل وإيجاد أقصر طريق وتجنب العقبات. تسببت هذه المشكلات في انتكاسة لتطور وتقدم الروبوتات المستقلة. في هذه الأطروحة، عالجنا هذه المشكلات من خلال تحسين سرب الجسيمات، وهو نهج تلوي مستوحى من الاستدلال البيولوجي. نحن نهدف إلى تطوير روبوت قادر على التنقل من نقطة بداية معينة إلى نقطة نهاية معينة مع تجنب العقبات المستقرة وخاصةً المتحركة، مع المسار الذي يتم قطعه هو أقصر مسار ممكن. يعمل PSO من خلال تحليل بيئة البحث من خلال محاولات متتالية. بعد ذلك، يشير موقع كل جزء في بيئة البحث إلى حل ممكن لمشكلة التحسين. قمنا بتطبيق PSO وجربنا مع خوارزمية لدينا في ظروف متعددة لعرض كفاءتها وأدائها.

الكلمات الرئيسية -تحسين سرب الجسيمات، التنقل، الروبوت المتحرك المستقل، تجنب العوائق

RÉSUMÉ

Dans notre vie quotidienne, nous avons des difficultés à fonctionner parfaitement sans l'interférence des robots car ils sont devenus une partie inséparable de tout aspect quotidien que nous avons vu au cours des deux dernières décennies. La plupart d'entre nous identifient un robot comme une entité ou un objet préprogrammé par l'homme pour accomplir une seule fonction. Néanmoins, les chercheurs s'efforcent davantage de développer un robot autonome capable d'absorber les connaissances de son environnement et de les utiliser pour mener à bien des missions données. Comme tout ce qui existe, les robots mobiles autonomes souffrent également de problèmes qui réduisent leur efficacité et leurs performances, les principaux étant la navigation, la recherche du chemin le plus court et l'évitement des obstacles. Ces problèmes ont causé un revers pour l'évolution et le progrès des robots autonomes. Dans cette thèse, nous avons abordé ces problèmes avec l'optimisation des essaims de particules, une approche méta-heuristique bio-inspirée. Nous visons à développer un robot capable de naviguer d'un point de départ donné à un point final donné tout en évitant les obstacles à la fois stables et surtout en mouvement, avec le chemin parcouru le plus court possible. Le PSO fonctionne en analysant l'environnement de recherche à travers des tentatives consécutives. Ensuite, l'emplacement de chaque particule dans l'environnement de recherche dénote une solution possible au problème d'optimisation. Nous avons appliqué le PSO et expérimenté notre algorithme dans de multiples conditions pour voir son efficacité et ses performances.

Mots-clés— Optimisation des essaims de particules, Navigation, Robot mobile autonome, Évitement d'obstacles

GENERAL INTRODUCTION

Nowadays, multiple domains are all aspiring to find new and ingenious means to enhance functional efficiency, improve speed, guarantee precision, and raise safety. Several have found an answer to their needs in robotics and embraced it for help.

The theme of our thesis is robotics which is the engineering and function of machines able to autonomously or semi-autonomously accomplish physical assignments on behalf of a human. Commonly robots execute tasks that are either to a great extent repetitive or overly dangerous for a person to conduct without harm. The Two types of mobile robotics are autonomous and non-autonomous mobile robots. Autonomous mobile robots (AMRs) can explore their environment without external guidance, while guided robots use some type of guidance system to move. Other semi-stationary robots have a small range of movement. Typically Mechanical robots use sensors, actuators, and data processing to interact with the physical world. However, in recent years, the field of robotics has begun to overlap with machine learning and artificial intelligence.

The question is what connects the physical side of the robot to its intelligent side? Well, that is where the robot controller comes in handy. The essential piece of a robot is the controller which is a computer system that connects to the robot to manage its movements. In addition to the physical form, the controller also has the responsibility of the end-effectors and averting interference that might occur within the work area of the robots. The robotic controller is usually known as the “brain” of a robot. This is due to the coding interpretation that acts as the program for a specific robotic application. The controller deciphers the code into instructions for the robot to operate and execute the steps of the application. Robotic programs are coded into the controller via the controller’s teach pendant.

When it comes to a mobile robot, the precise element of comprehension instantly linked to strong mobility is navigation competence. Given limited information about the surrounding environment and a goal position or series of positions, navigation encompasses the capacity of the robot to operate based on its understanding and sensor values to reach its goal as efficiently and as reliably as possible. Navigation has always been a problem for robotics that researchers sought to solve.

And even then navigation is not enough, for it must be the shortest path between the points. Of course, by that, we are not talking about a straight path regardless of the obstacles, we are referring to the shortest trajectory the robot can take while avoiding obstacles both fixed and moving.

Here is where our problem unfolds, the intelligent side of the robot. Artificial intelligence happens to be one of the sophisticated methods used by the AMRs for path planning to diagnose and navigate through their environment. AI is the simulation of human intelligence functions by machines, to be precise computer systems.

The main problem is how to create an AMR able to find the shortest path while avoiding obstacles?

As a solution, we propose the particle swarm optimization approach which is a method inspired by biology to solve optimization problems. Proposed by Kennedy and Eberhart in 1995 [1], PSO is a bio-inspired algorithm.

Several attempts have been made by researchers to solve this problem using several techniques like simulated annealing and fuzzy logic etc. However, we are interested in methods that hold similarities to our proposed algorithm like ant colony optimization and Bacterial Foraging Optimization.

How does this algorithm work? PSO analyzes the search area through consecutive attempts at particle positions, their movements are handled by simple equations. Thus, the location of each particle in the search environment denotes a possible solution to the optimization problem. And the “quality” linked with each solution is quantified by the objective function, optimized little by little according to the positions, more or less optimal.

In the PSO model, each particle moves unsystematically obeying three simple rules:

- Cohesion: the particles are drawn to the average position of the group;
- Alignment: the particles follow the exact course as their neighbors;
- Separation: to avoid collisions, the particles keep a specific space between them.

The most exciting part of PSO is there is a stable topology where particles can communicate with each other and increase the learning rate to achieve global optimum. The metaheuristic nature of this optimization algorithm gives us lots of opportunities as it optimizes a problem by iteratively trying to improve a candidate solution. Applicability of it will increase more with the ongoing research work. Potential applications include intelligent service robots for offices, hospitals, and factory floors; maintenance robots operating in hazardous or hardly accessible areas; domestic robots for cleaning or entertainment; semi-autonomous vehicles for help to disabled people; and many more.

So essentially our system operates as follows: the robot must find the shortest path from the starting point to the endpoint in an environment with moving obstacles it must evade while maintaining the shortest path possible using particle swarm optimization methods.

This document is divided into three sections as follows:

The first chapter is dedicated to navigation techniques. A general review in the field of AMRs and navigation where we present the classification of the methods used to solve the problem at hand. We provide a comprehensive state-of-the-art, elaborating on similar methods and displaying the previous attempts to solve this navigation in dynamic environment predicament. Then a comparison was conducted between the chosen methods, presenting the advantages and disadvantages of each, to better know the points of strength and weakness.

The second chapter is the system design. In this section, we have a better understanding of our problem and more so of the solution. We give a comprehensive explanation of the algorithm we chose -the Particle Swarm Optimization- and see how it functions theoretically. For a better understanding, we placed the conceptual model into use, by illustrating diagrams and flow charts to make it easier to grasp the internal design of the system and see step-by-step how it executes to provide the necessary results.

The final chapter is the realization and Implementation. We commence with the programming language used, where we provide an extensive view of PYTHON and the libraries that we applied to facilitate our experiments. Several implementations have been conducted with the python based development environment to test and evaluate the performance of the PSO. We provided illustrated examples to display the result of the shortest path finding of an autonomous mobile robot.

Finally, as a conclusion, we summarize our thesis by giving a general view of what we discussed thus far. We also provide a glimpse of the future work of this thesis

I. Chapter: Navigation Systems in Dynamic Environments

I.1. INTRODUCTION

The previous section was the general introduction, where we gave a basic image of our context for this thesis. We discussed the context of our work, which is robotics. Then, we dug deeper into its main problem, which is navigation. From that, we established that the shortest path was always the predicament sought to be solved. Before we display how the thesis is organized, we proposed a solution by applying particle swarm optimization.

However, in this chapter, we will review a few methods the researchers applied to solve the navigation while focusing on those handling the local navigation problem for the AMRs. Now, navigation is an important assignment in the domain of mobile robotics, which can be categorized into 2 types: global navigation and local navigation [2]. In global navigation, preliminary knowledge of the territory should be available. Numerous techniques have been designed for global navigation, i.e. Cell Decomposition Method [3], Grids [4], Dijkstra Algorithm [5], Artificial Potential Field Method [6, 7], Visibility Graph [8], And Voronoi Graph [9, 10], and so on. In the local navigation, the robot can determine or maintain its movement and orientation autonomously utilizing equipped sensors such as ultrasonic range finder detectors, sharp infrared range sensors, vision (camera) sensors, etc. Ant Colony Optimization Algorithm (ACO) [11], Simulated Annealing Algorithm (SA) [12], Neural Network (NN) [13], Genetic Algorithm (GA) [14], Fuzzy Logic (FL) [15], Bacterial Foraging Optimization (BFO) [16], Particle Swarm Optimization Algorithm (PSO) [1], etc. are used successfully by diverse researchers to solve the local navigation problem. Figure I.1:1 illustrates the general categorization of the Deterministic, Nondeterministic (Stochastic), and Evolutionary algorithms, which miscellaneous authors have implemented for mobile robot navigation.

Later in the chapter, we conducted a comparison between the selected methods where we listed the advantages and disadvantages of each to better see the strengths and weaknesses

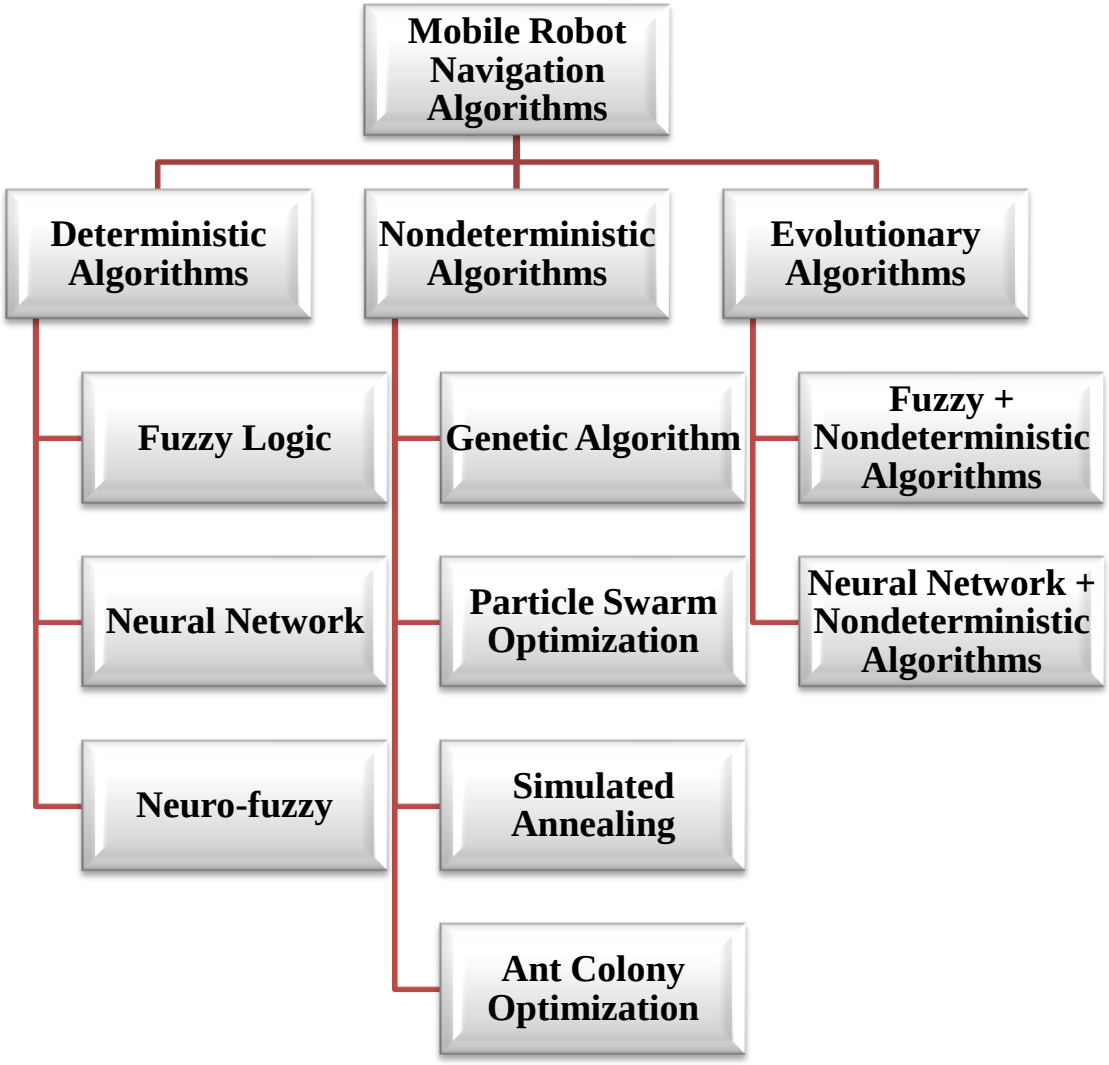

Figure I.1:1: Mobile Robot Navigation Algorithms [ref]

I.2. RELATED WORK

In this section, we will be reviewing methods used by researchers while searching for an answer to the problem of navigation of an AMR.

A. Fuzzy Logic Technique for Mobile Robot Navigation

The notion of fuzzy logic was introduced by Zadeh [17], which is broadly applied in multiple engineering applications such as mobile robotics, image processing, etc. This technique has a vital role in the domain of mobile robots. Fuzzy control systems are rule-based or knowledge-based systems having a set of fuzzy IF-THEN rules established from the domain knowledge of human experts.

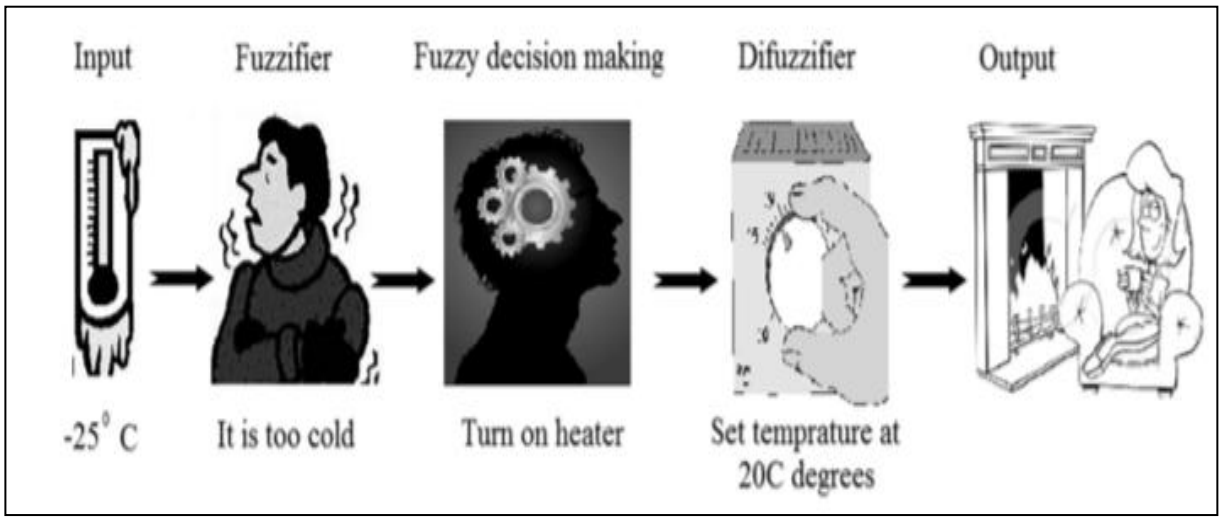

Figure I.2:1: Example of the Fuzzy Logic Process

The fuzzy logic approach has been successfully applied by numerous researchers to maintain the position and direction of mobile robots in the environment.

Al-Mutib et al. [18] proposed a stereo vision-based mechanism with Fuzzy logic. Abadi et al. [19] developed the Mamdani-based FL controller for a wheeled mobile robot to follow the mobile entity. They utilized the PSO algorithm with FL as a hybrid system to determine the most suitable parameters. The adequate functioning of FL was given by Castillo et al. [20] to sustain the variety rule in ACO and to bypass early convergence. Al-Jarrah et al. [21] came up with the path planning method for multiple mobile robot systems and active motion coordination between them by using a probabilistic fuzzy controller [22] with the NN. In this process, a leader robot position will be pursued by a follower robot. The initial order Sugeno fuzzy system was used for the head robot in an attempt to gain a high-level controller whereas the fellow robot has a low-level controller. The knowledge strategy is created by utilizing the NN and efficient fuzzy rules are adjusted by ANFIS. The fuzzy-based process was used for the navigation of a humanoid automaton in the 2D surroundings by Rath et al. [23]. Navigation in a 3D area is a challenging task that is handled with the fuzzy logic for path planning of robots by Abbasi et al. [24] and Xiang et al. [25]. The FL approach has been used in the defense field for controlling and guiding missiles, drones, mobile, and underwater robots.

Kashyap & Parhi [26] came up with Obstacle avoidance and path planning of humanoid robots using fuzzy logic controller aided owl search algorithm in complicated workspaces. They sought to draft and enforce a brand new hybrid controller in humanoid robots to map an optimal path. The hybrid controller is created using the Owl search algorithm (OSA) and Fuzzy logic

B. Neural Network Technique for Mobile Robot Navigation

The neural network is one of the significant methods for mobile robot navigation. This neural network technique is inspired by the human brain. Neural networks with several process layers are known as "deep" networks and are used for deep learning algorithms, which are being applied by multiple researchers in various domains to name a few signal and image processing, pattern recognition, mobile robot path planning, business, etc.

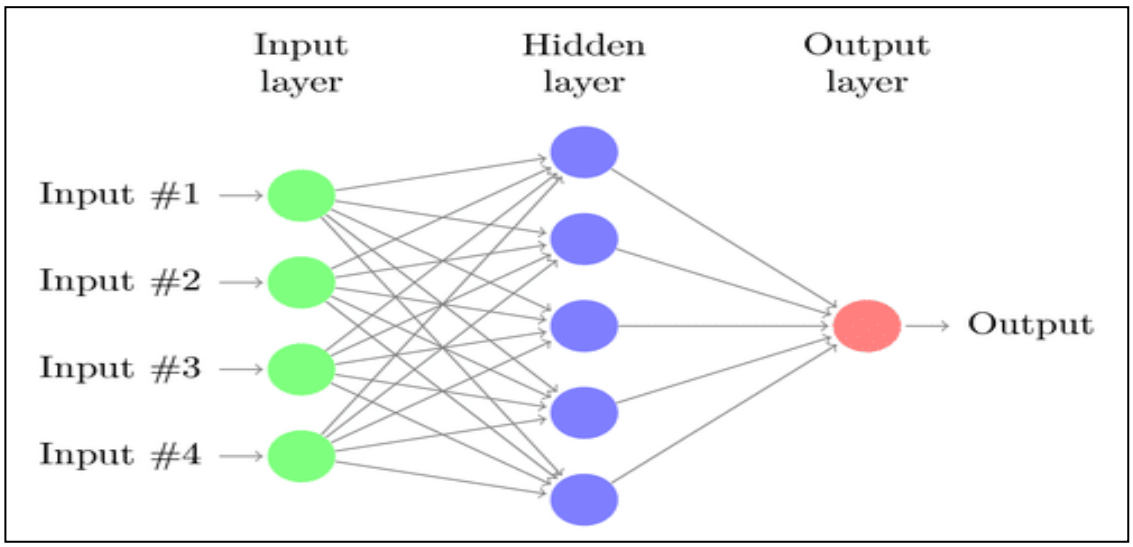

Figure I.2:2: Example of the neural network process

The main problem in the mobile robot is the navigation, characterized as how to guarantee the robot can complete the task safely without running into any obstacles. Yudha & al [27] studied the appliance of FLC and NN in robot navigation and approximates the performance in navigating the robot to the target. Ruan & al [28] suggested an end-to-end process utilizing deep reinforcement knowledge for the navigation of mobile robots in an unknown environment. Based on dueling network architectures for deep reinforcement learning (Dueling DQN) and deep reinforcement learning with double q learning (Double DQN), they adopted a dueling architecture-based double deep q network (D3QN). Cheng & Chen [29] employed the intellectual strategy of reinforcement knowledge to research a solution to handle the problem that requires a robot to discover an appropriate trail and move from its current status to a goal position without crashing into any obstacles. It considers the laser beam detected spaces and the relative motion angle as the input of the neural network model and the robot's movement stance is represented as the output. This neural network model is trained by a deep Q-learning network (DQN) algorithm via positive and negative feedback compensations determined by task-specific learning goals.

C. Genetic Algorithm for Mobile Robot Navigation

Genetic algorithms (GAs) are problem-solving techniques (or heuristics) that imitate the operation of natural evolution, discovered first by Bremermann [30] in 1958. These algorithms use the notions of natural selection to select the best solution for a problem.

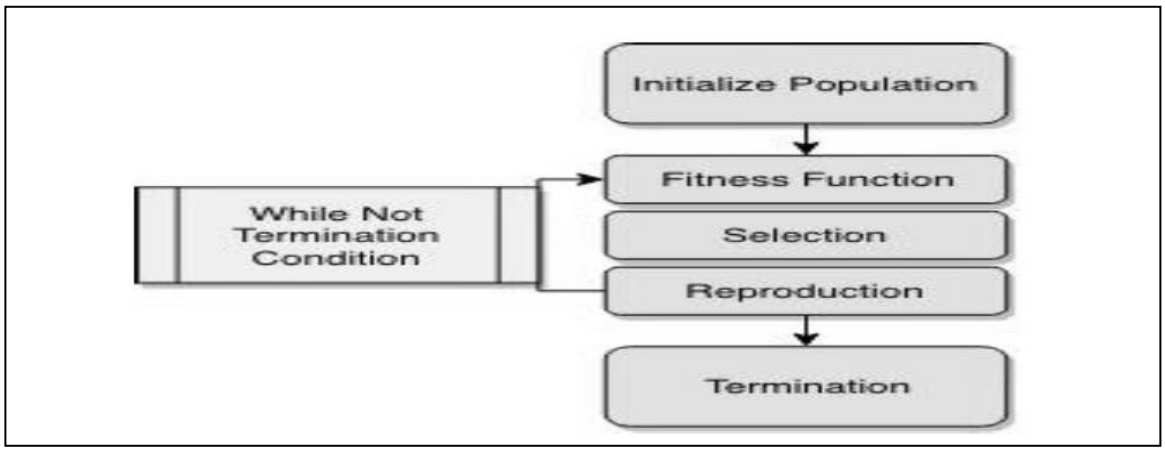

Figure I.2:3: Flow Chart of the Genetic algorithm

In [31], the researchers have enforced Genetic-Fuzzy Controller (GAFLC) to optimize and tune the Gaussian membership process parameters for mobile robot motion control. Arora et al. [32] have offered a single fitness-based genetic algorithm for solving the navigation problem in dynamic environments. Nazarahari & al [33] designed a Multi-objective multi-robot path planning in the continuous environment using an enhanced genetic algorithm.

For path optimization, an altered structure of GA is offered by Jianjun et al. [34]. In their technique, the length of the chromosome is adjusted to obtain the best outcome. The GA method reacts to the surroundings (known and unknown) efficiently; thus, it is adopted in the 3D path planning problem of the submarine robot [35] and aerial robot [[36], [37]], and 2D path planning of a humanoid robot [38]. To handle the problem of a mobile target, Patle et al. [39] have delivered the matrix binary code-based genetic algorithm (MGA) in the complicated environment for single and multi-robot approaches. In this method, the robot can efficiently follow the moving obstacle and moving goal and gets to the destination in a short period of time.

D. Ant Colony Optimization Algorithm for Mobile Robot Navigation

The Ant Colony Optimization (ACO) algorithm is a probabilistic algorithm proposed by Dorigo et al. [40] in 1999, which originated from bionics. ACO takes inspiration from the

foraging behavior of some ant species. The ants leave pheromone on the ground to keep some advantageous path that should be followed by other members of the colony.

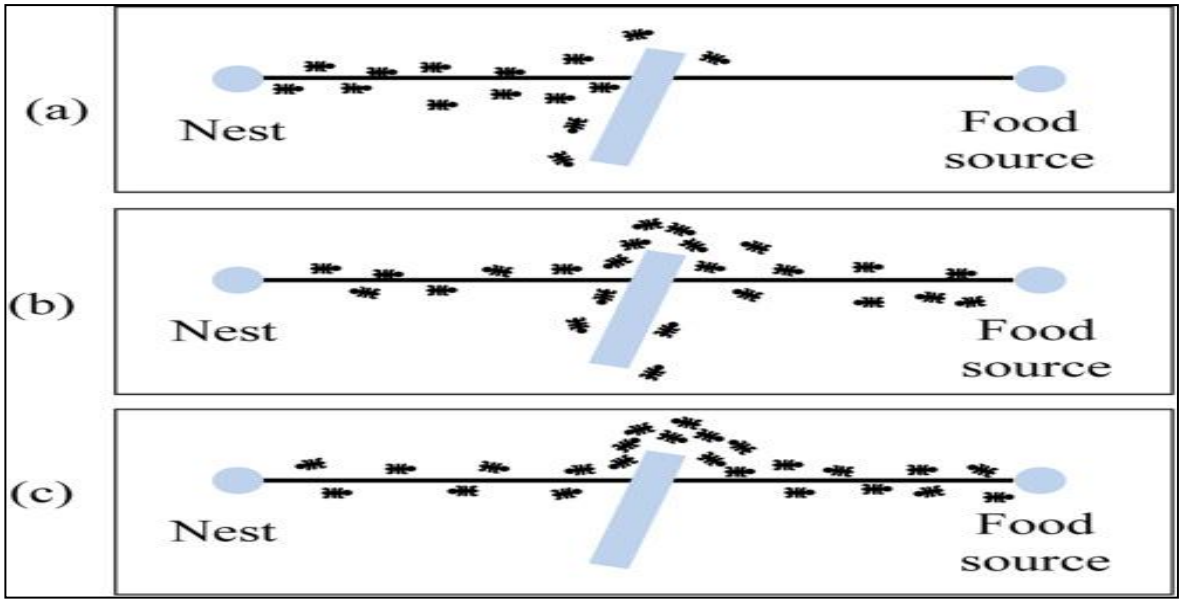

Figure I.2:4: Explanation of the Ant Colony Optimization

The Ant Colony Optimization (ACO) algorithm is utilized by numerous researchers for mobile robot navigation and obstacle avoidance in various environments. An ACO-fuzzy-based hybrid system for mobile robot navigation is offered by Castillo et al. [41] for navigation in static surroundings. An RA-ACO-based method for navigation of a humanoid robot is proposed for chaotic conditions by Kumar et al. [42]. They attempted the suggested method for navigation of numerous humanoid robots in a real-time environment using Petri-Net and received satisfactory affirmation in simulation outcomes and real-time results. To improve the performance of the present ACO technique in the static environment, modifications have been proposed by Liu et al. [43]. According to them, the convergence rate is the central contributor to performance. They incorporated pheromone dispersal and geometric local optimization for locating for the optimal path that ultimately results in the existing path pheromone diffusing in the direction of the possible area force during the searching - ants tend to explore for a more elevated fitness subspace, and the search-space of the pattern evolves more diminutive. An additional change for the dynamic environment is offered by Rajput et al. [44]. They also offered a novel pheromone technique that update to bypass disproportionate looping and to accomplish faster convergence. In [45], and for fuzzified evolved ant colony optimization (FAACO), the path pheromone update plan is split into two classifications like favorable and unfavorable trajectories. Applying these, path pheromone has been sorted out as the difficulties

of customary ACO like slow convergence. The evolved FACO enhances the evaporation speed of the pheromone to accelerate the convergence speed.

E. Bacterial Foraging Optimization for Mobile Robot Navigation

In 2002, Passino [16] introduced the new nature-inspired optimization algorithm which originated from the behavior of E. coli and M. Xanthus bacteria. These bacteria seek nutrients by making the most profitable use of energy attained per unit of time.

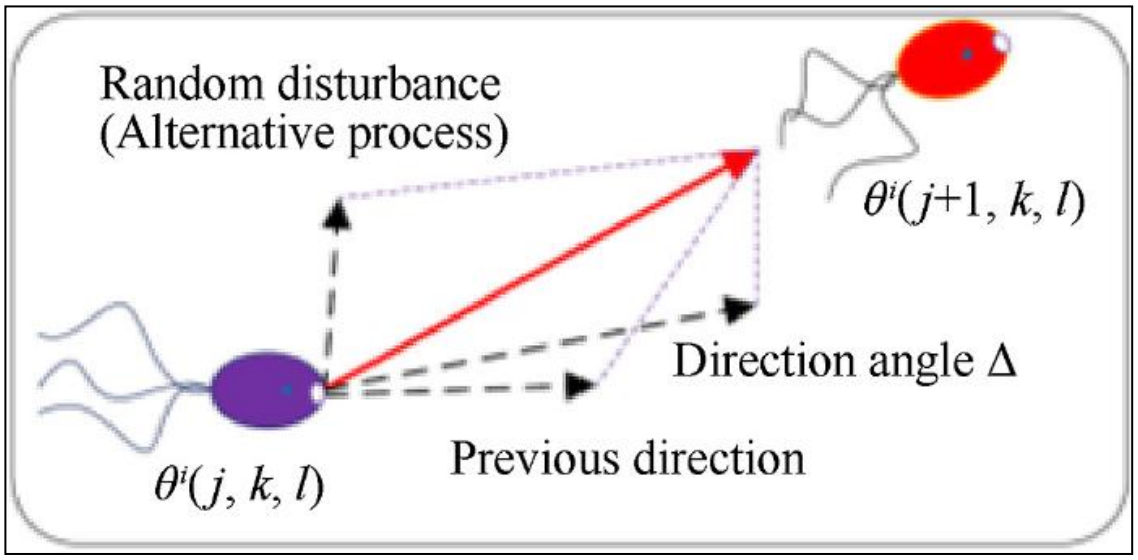

Figure I.2:5: Visual Representation of how the Bacterial Foraging Optimization Functions

Hossain & Ferdousand [46] attempted to solve the mobile robot navigation problem with the Bacterial Foraging Optimization (BFO) approach, the idea was to find the shortest achievable path within the minimum time from the start position to the goal position among moving obstacles. To enhance the execution of a wheeled robot in path planning, an improved BFO algorithm is designed by Abbas et al. [47]. the environment is modeled by the developed approach using an APF method over two opposite forces i.e. the goal has an attractive force and the obstacles have the repulsive force; the technique examines negative feedback from the algorithm to pick the right direction vectors that operate as guidance to the search operation to the profitable zone with more acceptable local search. Brand & Yu [48] have designed the Firefly Algorithm (Glow-worm swarm optimization) to find a collision-free most straightforward path in the two-dimensional static and dynamic environment for MR. this algorithm was compared to the ACO algorithm and stated that the suggested algorithm delivers a more reasonable results (when it comes to path length and computational cost).

F. Simulated Annealing Algorithm for Mobile Robot Navigation

The notion of a simulated annealing algorithm has come from statistical mechanics [49]. SA is a useful and known form of optimization. It is practical in locating global optima in the existence of large numbers of local optima. "Annealing" is a connection to an analogy with thermodynamics, particularly, how the metals cool and anneal. Simulated annealing utilizes the objective function of an optimization issue rather than the energy of a material. SA is an iterative search algorithm encouraged by the annealing of metals [50].

Miao & Tian [12] used the heuristic method-based simulated annealing algorithm for robot path planning in dynamic environments. The writers conducted a comparison of this suggested algorithm to the Dijkstra algorithm and declared that the proposed algorithm processing time for a solution is significantly less than that of the Dijkstra algorithm.

Tavares et al. [51] discussed the use of SAA in the off-line path planning problem of a mobile robot. They have prepared several adaptive tuning parameters to alter the conduct of that algorithm. Zhang et al. [52] have incorporated the simulated annealing algorithm and ant colony optimization (ACO) algorithm to raise the navigation speed of the mobile robot.

G. Particle Swarm Optimization Algorithm for Mobile Robot Navigation

Particle swarm optimization (PSO) is a population-based stochastic algorithm, which takes motivation from the social conduct of bird flocks. PSO algorithm is used to find an optimal or near-optimal solution to the problem using the fitness function f(x) = f(x1,x2,x3,……..,xi) where xi is a population of the particles.

Figure I.2:6: The Inspiration of the Swarm Particle Optimization

Ahmadzadeh & Ghanavati [53] have demonstrated a navigation method for numerous mobile robots based on the PSO algorithm. The main idea is that at first, the robot navigation problem is remade into an optimization problem. Then PSO process explores the solution space to find the appropriate minimum value. Based on the position of goal, they calculated an evaluation function for each particle in PSO. In each iteration of the algorithm, the global best position of a particle is established and the robot proceeds to the following calculated point to reach the goal. The environment is believed to be dynamic and obstacles are either fixed or movable.

Li & Chou [54] proposed a self-adaptive learning particle swarm optimization (SLPSO) with different learning strategies to address the path optimization problem. First, they convert the path planning problem into a minimization multi-objective optimization problem and formulate the objective function by carefully putting under consideration the path length, collision risk degree, and smoothness. Then, a new self-adaptive learning mechanism is designed to adaptively select the most suited search techniques at various phases of the optimization process, which can enhance the search-ability of particle swarm optimization (PSO).

Song & al [55] addressed the global smooth path planning for mobile robots by developing a novel multimodal delayed particle swarm optimization (MDPSO). Huang [56] worked on the designing the Parallel Met heuristic Particle Swarm Optimization (PPSO) algorithm to resolve the global path planning problem of an autonomous mobile robot. The author has executed this PPSO algorithm in real-time using the field-programmable gate array (FPGA) chip.

To set an optimal intelligent regulator for a self-reliant wheeled mobile robot, Castillo et al [57] have developed the hybridization of an Ant Colony Optimization (ACO) algorithm and the Particle Swarm Optimization (PSO) algorithm to optimize the membership function of a fuzzy controller.

Chung et al. [58] have worked on PSO and fuzzy-based combinatorial algorithms for a mobile robot's intelligent navigation architecture. The PSO algorithm is applied so that the robot escapes from dead-end situations, and the fuzzy algorithm controls the turning angle of a wheeled mobile robot during navigation and obstacle avoidance. Shiltagh & Jalal [59] have researched the application of Modified Particle Swarm Optimization (MPSO) in the domain of mobile robotics to define the shortest possible route from the starting point to the ending in an

environment filled with obstacles. The advanced altered PSO improves the intersection rate of the algorithms.

Allawi & Abdalla [60] have submitted the sensor-based PSO-fuzzy type-2 model for the navigation of numerous mobile robots. They have utilized the PSO algorithm to select the optimal input/output membership function parameters and rules for the fuzzy type-2 controller.

Wang & al [61], to discover the optimal path planning of mobile robots in a strange environment, submitted a particle swarm optimization algorithm based on trajectory length as the fitness function. The place of the global optimal particle is specified by the lowest fitness value, and the robot advances along with the points of the optimal particles to the target position. The robot path planning with obstacle avoidance using particle swarm optimization was proposed by Bartecki et al [62] which helped with the problem of unfinished convergence.

Gul & al [63] addressed this problem by adapting a multi-objective path planning algorithm by the hybridization of the Grey Wolf optimizer-particle swarm optimization algorithm. Said process follows these three steps: (1) the first is path optimization by the hybridization of the Grey Wolf optimizer-particle swarm optimization algorithm, which smoothes the path and lessens its distance. (2) In the second step, all optimal and viable points rendered by PSO–GWO algorithm are merged with the Local Search technique to convert points deemed infeasible into feasible solutions, the last step (3) is based on collision avoidance and detection algorithm, where mobile robot discovers the existence of an obstacle in its sensing perimeter, and sees to evade them using the appointed algorithm.

I.3. COMPARISON Between THE NAVIGATION METHODS

Approach	Advantages	Disadvantages
Fuzzy Logic	• FL is highly to reflect real-world problems more than classical logic. • FL algorithms have fewer hardware necessities than classical Boolean logic. • Fuzzy algorithms can deliver more true-to-life results with ambiguous or imprecise data.	• Fuzzy algorithms need comprehensive guarantee and verification. • Fuzzy control systems depend on human expertise and knowledge.
Neural Network	• Problems in ANN are defined by attribute-value duos. • ANNs are utilized for issues regarding the target function, the outcome may be discrete-valued, real-valued, or a vector of numerous real or discrete-valued attributes. • It is used where a rapid evaluation of the learned target function is demanded.	• Hardware Dependence NN requires processors with parallel processing power, by their structure. • The duration of the network is unknown • Unexplained functioning of the network
Genetic Algorithm	• The concept is straightforward to comprehend • GA search from a population of points, not just a single point • GA supports multi-objective optimization • GA is effortlessly parallelized • GA is stochastic	• GA execution is still an art • GA requires fewer data regarding the problem but creating an objective function and obtaining the proper representation and operators can be challenging • GA is time-consuming

Ant Colony Optimization	• ACO could be applied in dynamic applications • Positive Feedback directs to the quick discovery of acceptable solutions • Distributed computation prevents premature convergence	• While the convergence is ensured, the time to convergence is uncertain • Coding is not straightforward
Bacterial Foraging Optimization	• BFO enriches the velocity-swim operator which enhances its convergence behavior • A low computational cost may be obtained • The performance and the simplicity of the linear decreasing mechanism	• The extremely precise type of constraint-handling mechanism which is challenging to generalize • The high number of parameter values that is necessary to be specified by the user
Simulated Annealing	• SA is simple to code and use. • SA does not depend on the restrictive properties of the model and therefore is versatile. • SA deals with noisy information and highly non-linear models. • Supplies an optimal solution for multiple problems and is robust.	• SA is metaheuristic so a lot of parameters have to be adjusted. • The accuracy of the digits used in its execution has an important impact on the quality of results. • There is a tradeoff between the quality of the result and the time taken for the algorithm to run.
Particle Swarm Optimization	• Insensitive To Scaling Of Design Variables. • Easily Parallelized For Concurrent Processing • Very Few Algorithm Parameters. • A Very Efficient Global Search Algorithm	• Likely to fall into local optimum in high-dimensional space

Table I.3-1: Comparison Table of the Methods

I.4. CONCLUSION

In this chapter, we have presented the diverse methods employed for mobile robot navigation. We have seen how various researchers throughout the past 2 decades have put efforts into solving the most important problems in mobile robotics; Navigation and Obstacle avoidance. We displayed those efforts and then conducted a comparison between the mentioned methods, highlighting the advantages, and disadvantages of each technique. Our aim of this chapter is to demonstrate how most of the researchers have used these soft computing techniques for mobile robot navigation and obstacle avoidance in only static environments. However, several researchers have thought of dynamic environments for mobile robot navigation. Also focuses on nature-inspired algorithm-based mobile robot navigation and obstacle avoidance and how it is an important topic for the research.

We covered all that in this chapter. However, in the next chapter, we will be viewing how our system is designed, a detailed explanation of our chosen methods along with the Conceptual models that describe the system flow by creating interfaces of existing knowledge and frameworks, making it more effortless to comprehend.

II. Chapter: System Design

II.1. INTRODUCTION

In the previous chapter, we view the state of the art. We took a deeper look at the attempts accomplished by researchers in the past two decades to solve the navigation problem of an autonomous mobile robot. We viewed the deterministic, nondeterministic (stochastic), and evolutionary algorithms, and concentrated on nature-inspired algorithms considering they are significant subjects due to their resemblance to the topic we chose for the research.

Nonetheless, the main object of this chapter is our approach on the navigation and obstacles avoidance problem with the Particle Swarm Optimization method. The PSO technique is motivated by the conduct of a swarm to locate food. Our solution begins with creating the particles where each particle has its own coordinate and speed values. Every particle attempts a function to discover the best solution, which is known as the fitness function. After evaluating this function every particle has a new fitness value, in each iteration, this fitness function value is examined and particles obtain the best fitness value as the local best. This continues till the goal is reached.

We also illustrate the Conceptual models which are abstract, psychological representations of how tasks should be performed. They are commonly used subconsciously and intuitively as a way of systematizing processes. Which is exactly the goal of this chapter to break down how the PSO in particular and the application in general function, by creating interfaces of existing knowledge and frameworks, making it easier to comprehend.

II.2. Our PROBLEM

All roads lead to Rome, is a famous quote known for centuries, but unlike our problem, we do not want all roads we only want the shortest road to Rome. Our problem lies in finding the shortest and optimal path between two points in an environment rich with fixed and changing obstacles. On a larger scale, we examined the use of PSO in optimization problems to see its ability to find suitable solutions and then attempt it in dynamic environments with much more complicated conditions to better test its efficiency and performance. Once that is completed, we were able to ask our main question:

How could the PSO algorithm make an autonomous mobile robot capable of searching and finding the shortest path, and navigating through it, all while avoiding obstacles in real-time?

II.3. THE PROPOSED SOLUTION

As an answer to the search question, the study focuses on adjusting that which already exists and developing further efficient PSOs to solve large-scale optimization problems of assignments given to a mobile robot. however, our focus is on how successfully adapt the PSO and apply it to solve the problems regarding massive search space and find one specific solution among many that perfectly answers the dilemma.

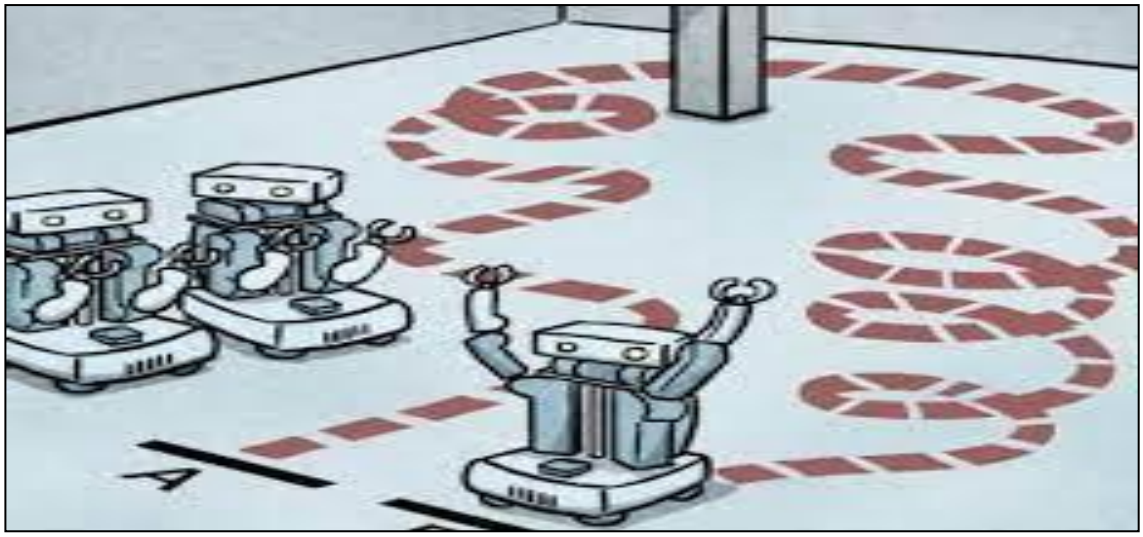

Figure II.3:1: Finding the Optimal Trajectory for a Mobile Robot

II.4. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is an optimization method that is motivated by the behavior of a swarm. To solve a problem all the particles of the swarm are created. All particles try a fitness function and get a score after trying the fitness function. If the score is more promising than the pbest (best score of the particle), the pBest is updated. After all the particles attempt the fitness function, the best record of the swarm is selected, it is referred to as the local best. After selecting the local best value, it is approximated with a different value which is called the global best. Global best is the results of all particles throughout all iterations. The gbest is updated at the end of every run by comparing it with the pbest. If the latter is better than the gbest value then the gbest is replaced by pbest which has a finer value. At the end of each iteration, the particle with the finest fitness value controls the others in the swarm to alter their paces according to its coordinates and speed. Every particle adjusts its speed value to match the leader particle's coordinates.

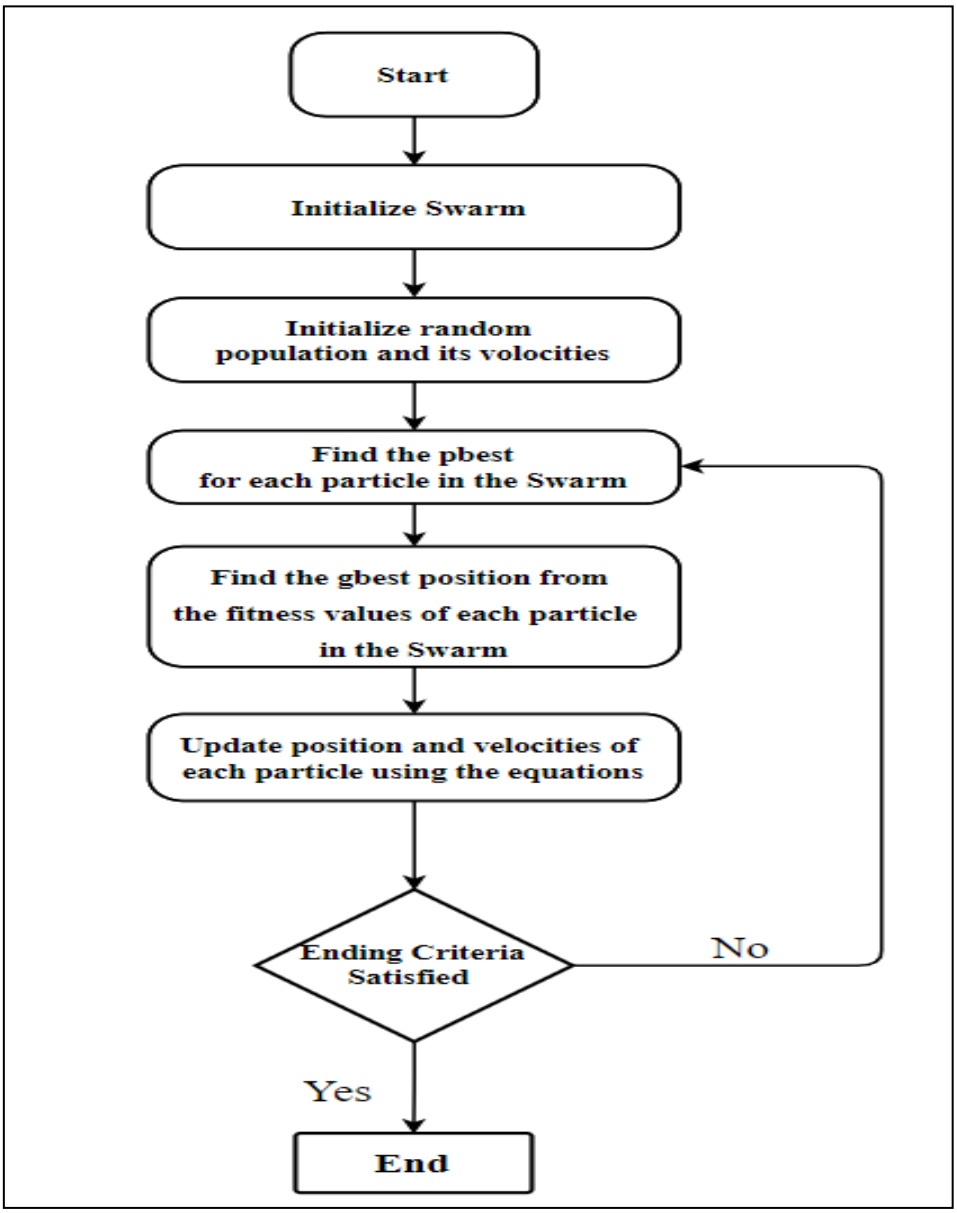

Figure II.4:1: Flow Chart of the Particle Swarm Optimization

Figure II.4:1 gives a fairly comprehensive representation of the PSO algorithm. In the continuous space coordinate system, mathematically, the PSO can be expressed as follows consider:

- The swarm size is N.
- Each particle's position vector in D-dimensional space is $X_i = (x_{i1}, x_{i2}, \dots, x_{id}, \dots, x_{i\,D})$.
- The velocity vector is $V_i = (v_{i1}, v_{i2}, \dots, v_{id}, \dots, v_{i\,D})$.

- Individual's optimal position (i.e., the optimal position that the particle has experienced) is $P_i = (p_{i1}, p_{i2}, \dots, p_{id}, \dots, p_{iD})$.
- Swarm's optimal position (i.e., the optimal position that any individual in this swarm has experienced) is represented as $P_g = (p_{g1}, p_{g2}, \dots, p_{gd}, \dots, p_{gd})$.

To solve our shortest path problem, we will consider it as a minimization problem which we will seek to solve with the PSO, the Update formula of the individual's optimal position in **Equation II.4-1**

$$p_{i,t+1}^{d} = \begin{cases} x_{i,t+1}^{d}, & \text{if } f(X_{i,t+1}) < f(P_{i,t}) \\ p_{i,t}^{d}, & \text{otherwise} \end{cases}$$

Equation II.4-2

The swarm's optimal position is selected from all the individual optimal positions. Update formula of position is presented in **Equation II.4-2** and velocity is represented in **Equation II.4-3** as follows:

$$x_{i,t+1}^{d} = x_{i,t}^{d} + v_{i,t+1}^{d}$$

Equation II.4-3

$$v_{i,t+1}^{d} = \omega * v_{i,t}^{d} + c_1 * rand * (p_{i,t}^{d} - x_{i,t}^{d}) + c_2 * rand * (p_{g,t}^{d} - x_{i,t}^{d})$$

Equation II.4-4

Figure II.4:2 is a representation of the iteration procedure of a particle. Every particle in our swarm alters its movement depending on this. We analyzed the velocity update formula from a sociological perspective; we can see that the first part affects the particle's prior velocity in this update formula. It signifies that the particle is confident in its current moving condition and conducts inertial moving based on its own velocity, so parameter ω is dubbed inertia weight. the inertia weight is a positive constant that is important for balancing the global search also known as exploration (when higher values are set) and local search known as exploitation (when lower values are set).

The second part relies on the distance separating the current position of the particle and its own optimal position, called the "cognitive" item. It means a particle's own thinking or

Personal Influence, i.e., a particle's move resulting from its own experience. Hence, parameter c1 is called the cognitive learning factor (also called the cognitive acceleration factor).

The third part counts on the length between the current position of the particle and the global (or local) optimal position in the swarm, called the "social" factor. It signifies the knowledge share and cooperation among the particles. Essentially, the movement of each particle is undeniably connected to the experiences of the other particles in the swarm.

It mimics the motion of a good particle through cognition, so the c2 setting is referred to as the social learning factor (also called the social acceleration factor).

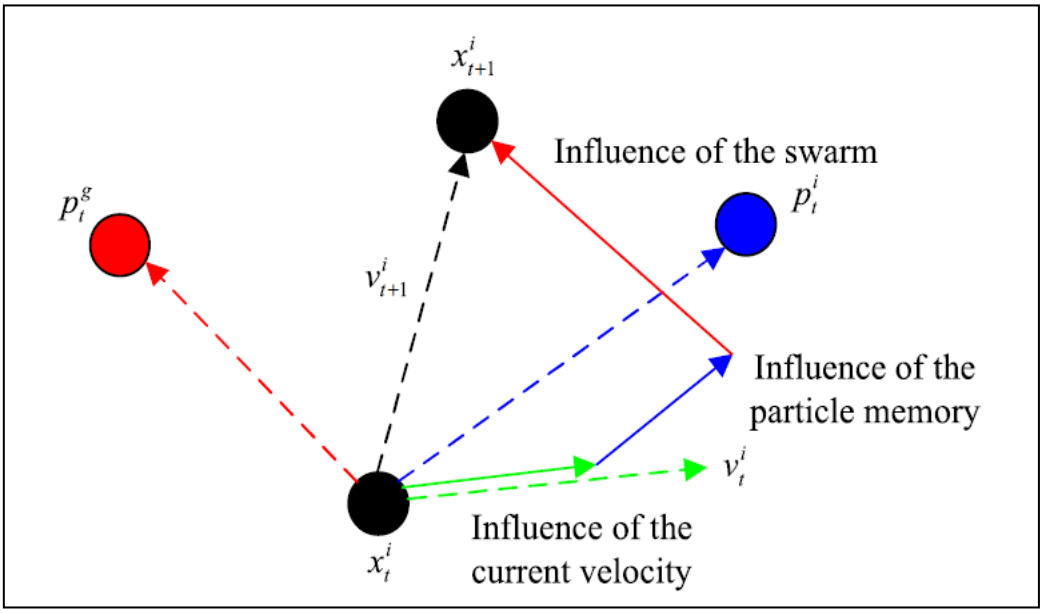

Figure II.4:2: Iteration Scheme of the Particles

What we established is that each particle alter its position according to these primary criteria

- Its current position
- Its current velocity
- The distance between its current position and pbest.
- The distance between its current position and gbest.

At the end of the next iteration, the particles' chances of finding a more suitable solution while evaluating fitness function is higher as every agent attempts to find a better gbest.

The major problem in using the PSO algorithm to the shortest path problem is encoding the nodes to the functioning of the particles composing the swarm. For accomplishing this, an encoding technique must be applied.

1. PSO PARAMETERS

PSO Algorithm Parameters	
Number of birds	Obtained by the user
Number of iterations	Obtained by the user
Speed	Obtained by the user between-0.5 and 0.5
Position	Vector presenting nodes
c1	Random number between 0 and 1
c2	Random number between 0 and 1
rand	Random number between 0 and 1

Table II.4-1: Table of PSO Parameters

2. ENCODING TECHNIQUE

To locate the shortest path problem to particle swarm optimization, every particle has a priority array. The priority array holds the priority values of the nodes to be utilized for uncovering the shortest path. Initially, we stochastically select the values of the priority array. When a particle tests a haphazardly created path, according to the score of the particle, the priority array is updated. The precedence of the nodes depicting the shortest path solution is improved. The increment value is accomplished with the speed value of the particle. The best trajectory of the particle is described by the coordinate value of the value particle. To find the most straightforward path, every particle attempts the following possible node which has the highest priority. The speed value and the priority values of the priority array are stochastically created. Initially, the coordinate value (the array that represents the path) has but the starting node.

3. ENDING CRITERIA

Ending criteria could be several various criteria. For instance, if we desire to terminate the loop in a specific step, the ending criteria can be iteration size. But in our case, to end the loop, the ending criterion is composed of three conditions that must be fulfilled once these conditions are satisfied the operation terminates as follows

Condition	Answer
Is the goal reached?	**Yes**
The path found is it the shortest?	**Yes**
Are any obstacles violated?	**No**

Table II.4-2: Ending Criteria

II.5. PROBLEM DEFINITION

The situation is as follows: initially we have a square matrix with X and Y coordinates that represents the environment in which we will be working. random obstacles are placed both fixed and moving.The objective is for the robot to reach the goal point from a start point. The robot has the start position (Xs, Ys) and the goal (Xg,Yg), noting that the robot is infimiliar with the surrounding environment.

What we are seeking to solve are the following problems: Finding the optimal trajectory (the shortest path) and avoiding obstacles

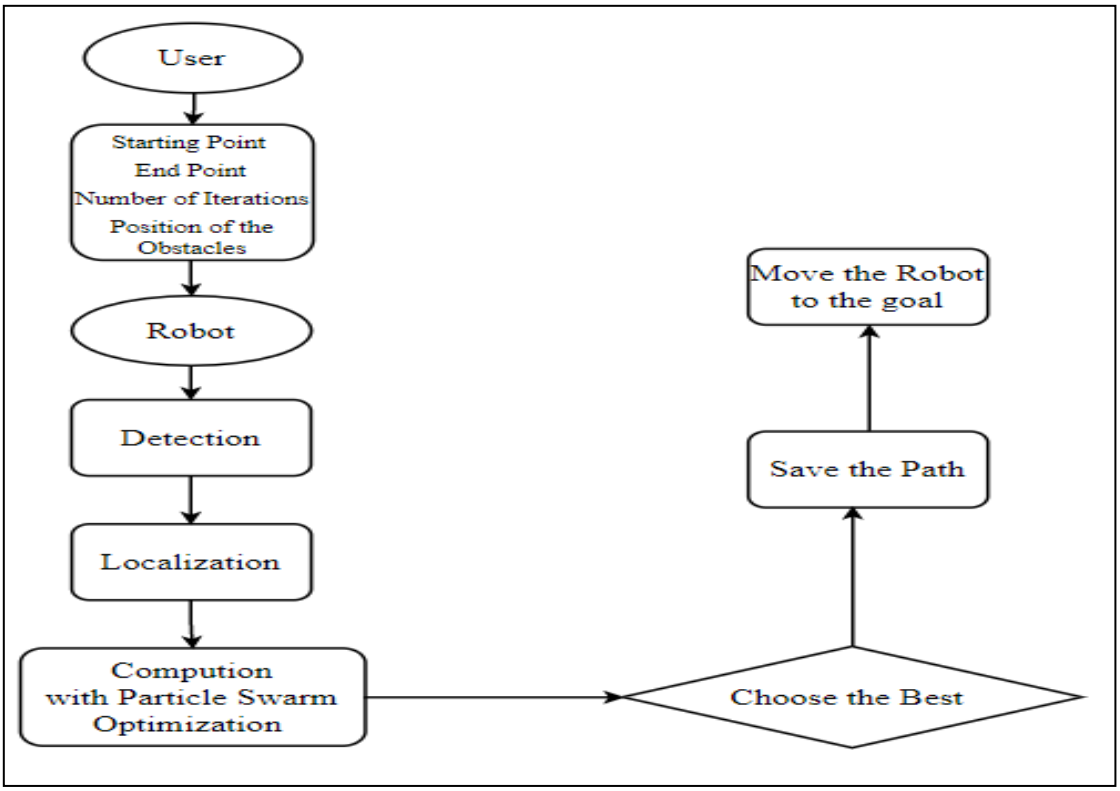

Figure II.5:1: General Architecture of Our Proposed System

1. USE CASE DIAGRAM

In the Unified Modeling Language (UML), a use case diagram could summarize the specifics of a system's users (also known as actors) and their interactions with the system. An effective use case diagram can help consult and represent the Scenarios in which a system or application interacts with people, organizations, or external systems, the Goals that the system or application helps those entities (known as actors) obtain, and basically the scope of the system:

- UML use case diagrams are ideal for:
- Illustrating the purposes of system-user interactions
- Determining and managing functional necessities in a system
- Defining the context and requirements of a system
- Modeling the fundamental flow of events in a use case.

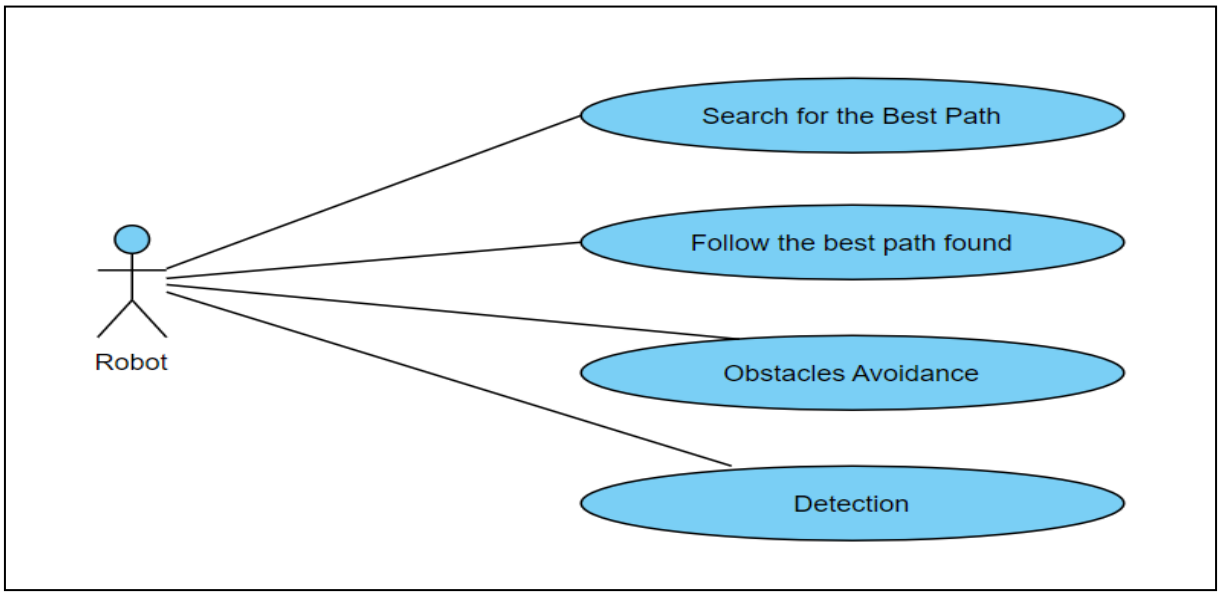

Figure II.5:2: Robot Use Case Diagram

Figure II.5:2is a visual depiction of the intercommunications between the user and the system. It represents the methodology used in analyzing the system to define, clarify, and organize system requirements. The main actor of this use case diagram is the robot, which performs different types of use cases such as search and avoidance.

With our robot being the single entity and the main actor, it mainly has four fundamental operations to accomplish:

- Search the best path: the robot is positioned at the starting point in an environment with a target to reach. It is entirely up to it to discover the shortest path to its designated goal;
- Follow the best path found: once the target is acquired, the robot begins moving toward it with each iteration getting him closer, and with each iteration, it recalculates to find the next closest point to the target until it reaches;
- Obstacles detection and avoidance: now our robot functions in an environment rich with obstacles. The fixed are simple and easy to handle, however the moving obstacles are a bit trickier and require a bit more attention.

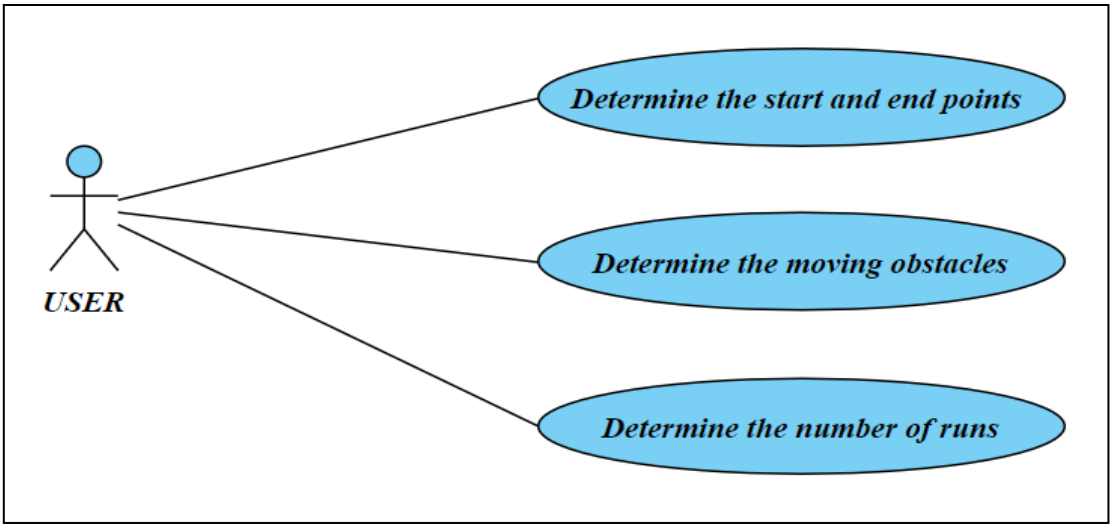

Figure II.5:3: User Use Case Diagram

In this use case diagram, the user is the only entity and the sole actor. The user has three major operations (**Figure II.5:3**):

- Determine the starting and arrival points: before launching, the user must provide the starting and endpoint for the robot
- Determine the type of the obstacles whether they be fixed or moving: the obstacles are set in the environment but their condition whether they are stable or moving is up to the user
- Determine the number of runs: the number of iterations the robot attempts to find the solution.

2. CLASS DIAGRAM

Class diagrams are a kind of structure diagram because they define what must be present in the system being modeled. UML was developed as a standardized model to represent an object-oriented programming method. Since classes are the building block of objects, class diagrams are the building blocks of UML. The different elements in a class diagram can describe the classes that will be programmed, the main objects, or the interactions between classes and objects.

The benefits of the UML class diagrams are:

- Demonstrate data models for information systems, whether they are simple or complex.
- Better comprehend the general synopsis of the schematics of an application.
- Visually express any distinct essentials of a system and disseminate that information.

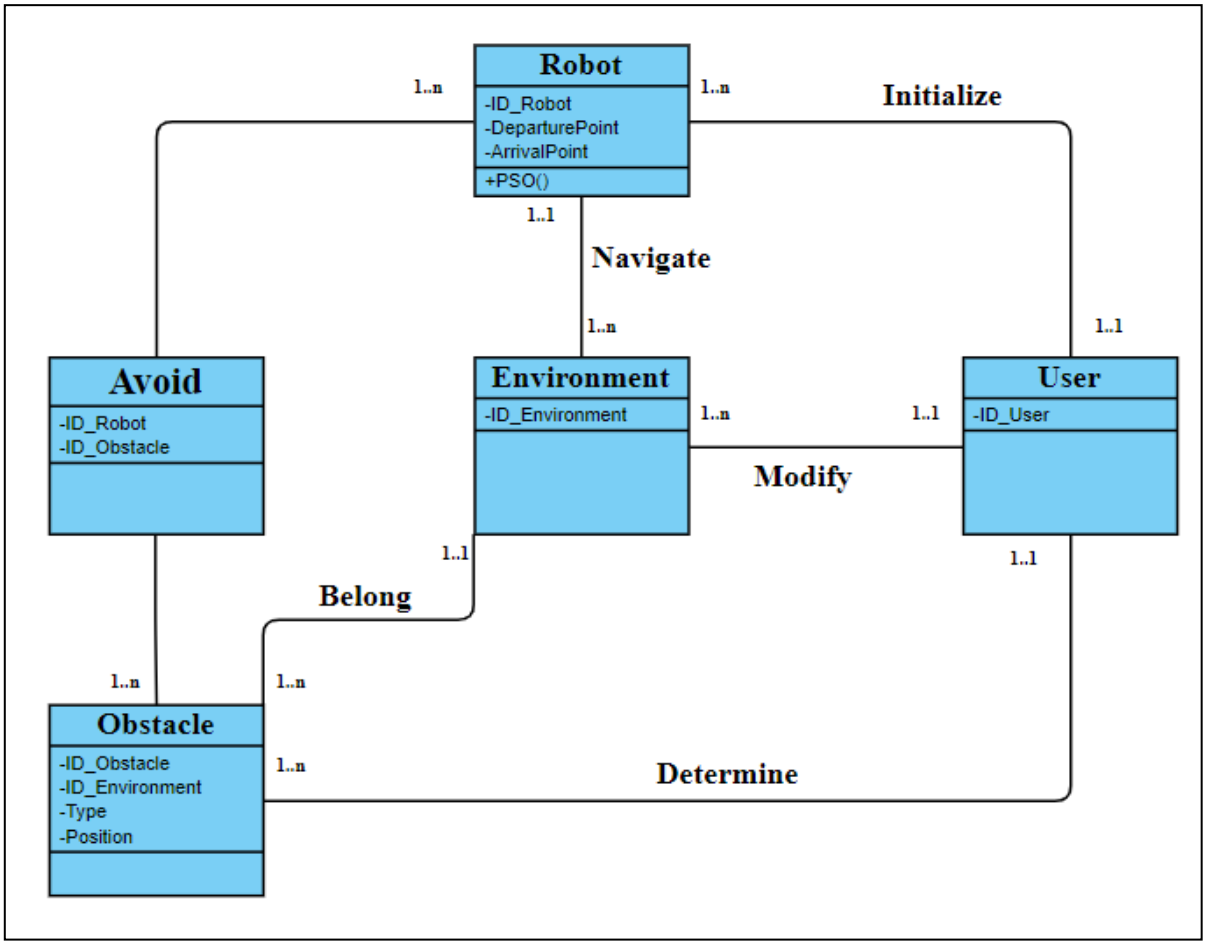

Figure II.5:4: Class Diagram of the System

Figure II.5:4 is Class diagrams used for conceptual modeling of the static view of the application, and for translating models into programming code in a detailed manner. This diagram utterly represents the arrangement of the classes, attributes, methods or operations, and associations amongst objects. The classes of our system are:

- Robot class: possess the robot ID, the departure point, and the arrival point as attributes. It manages all the Particle Swarm Optimization algorithm operations, which is controlling the sequence of the trajectory generation and commands to avoid errors and confusion such as malfunctioning and sudden halt.
- Environment Class: possess the environment ID as an attribute. It is designed and modified by the user so the robot can navigate
- Obstacles Class: contains the obstacle and environment ID as attributes and oversees all operations of the obstacles including positions and type whether they are fixed or moving.
- User class: possess the user ID as an attribute. Handles all operations performed by the user.
- Avoid class: contains the robot and obstacles ID as attributes. Avoided is originally considered a relation between the robot and obstacles classes however given the fact that the robot can avoid one or multiple obstacles at a time. And an obstacle can be avoided by one or many robots at a time.

3. SEQUENCE DIAGRAM

Sequence diagrams are a favored dynamic modeling solution in UML because they specifically concentrate on lifelines, or the processes and objects that exist simultaneously, and the messages exchanged between them to accomplish a function before the lifeline completes.

The benefits of a sequence diagram are to:

- Illustrate the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- Display how objects and components interact with each other to achieve a process.
- Plan and apprehend the detailed functionality of an existing or forthcoming scenario

Figure II.5:5 represent the sequence diagram of the system it shows the objects and classes involved and the communication between them. The user provides the start and end points' coordination for the robot. The PathPlanning class optimizes the path using the PSO class and sends the best solution back to the robot.

Once the update is complete and the obstacles reviewed, the particles begin the search for the goal. With each iteration, the PSO compares the personal best (pbest) obtained by the particle with the global best (gbest) achieved by the swarm. If the new pbest is better than the gbest then that pbest is the new gbest. The operation continues until the goal is reached and the path is the shortest.

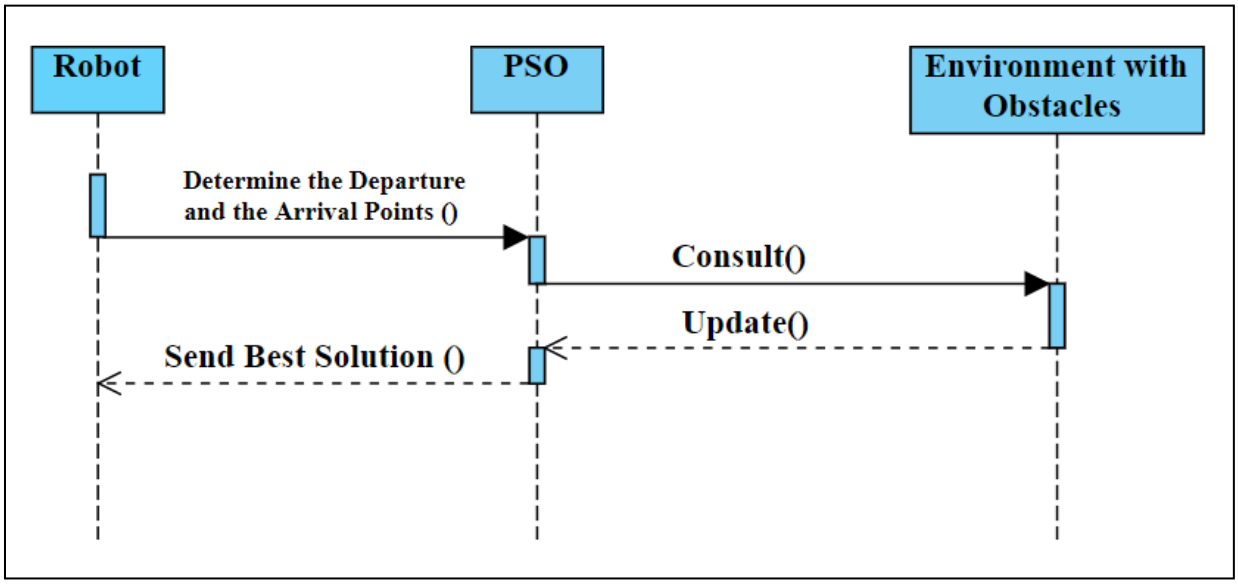

Figure II.5:5: Sequence diagram of the System

4. ACTIVITY DIAGRAM

Activity diagrams are behavior diagrams because they express what must occur in the system that is modeled.

An activity diagram visually displays a series of actions or flow of control in a system comparable to a flowchart or a data flow diagram. Activity diagrams are usually applied in business strategy modeling. They can also represent the steps in a use case diagram. Activities modeled can be sequential and simultaneous. In either case, an activity diagram has a start (an initial state), and an end (a final state).

In between there are forms to illustrate activities, flows, decisions, guards, merge and time events, and more.

Activity diagrams have several advantages to users:

- Exemplify the logic of an algorithm.
- Define the steps conducted in a UML use case.
- Represent a business approach or workflow between users and the system.
- Facilitate and enhance any procedure by clarifying complex use cases.

Figure II.5:6 is an activity diagram, it describes the dynamic aspects of the system. The diagram begins with the user, who is responsible for launching the process by initializing: the start and goal points, the obstacles, and the number of runs. Once that is terminated, the PSO takes over by randomly initializing the population and velocities and commences in determining the best of each particle. We consider the starting point is the gbest of the swarm. From that point, the particles search the solutions surrounding them, each presenting its own pbest. At each run, the PSO compares the gbest and the new pbest obtained by the particles. The pbest of the particle closest to the goal is the new gbest. From that, the robot moves, and the gbest is updated.

This process continues until the goal is reached. If no, it repeats the process. If yes, it continues to confirm if it is the shortest path. If no, it repeats, else it verifies if any obstacles violate. If yes, repeat the process otherwise end the algorithm with the most optimal trajectory.

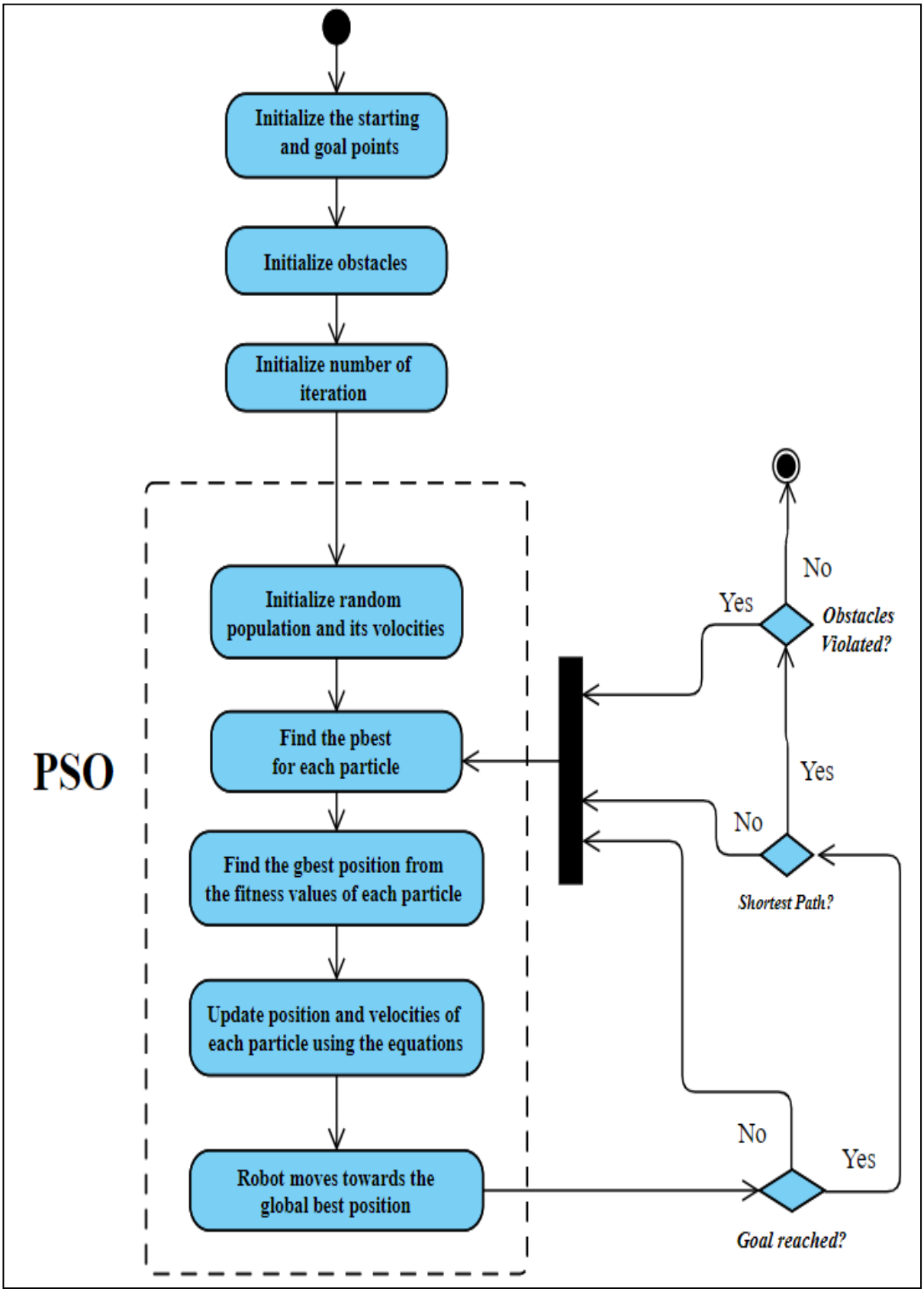

Figure II.5:6: Activity Diagram of our System

II.6. CONCLUSION

In this chapter, we understood our problem and what we see as an optimal solution. After the introduction, we explained the problem of the autonomous mobile navigation through an environment with obstacles. We proceed to propose a solution for it and exactly how a solution is most helpful. As a solution, we proposed a PSO approach and how it uses the personal and the global best, which are the collective results of all the swarm members, to discover the appropriate outcomes for the optimization problem. We presented the equations necessary for calculating and updating the global and personal position of the particles, encoding techniques in which we explained how the data is kept and selected to display the results, parameters of the algorithm, and the ending criteria necessary for the successful results. For further details, we built a conceptual model of the model to illustrate the knowledge and information for better comprehension.

In the next chapter, it is realization and implementation. We will display the results of our work. We tested our algorithms, and it is essential to present the result. We will offer an illustrated representation of our outcome. After, we show the tools, programming languages, and libraries we used to achieve these results.

III. Chapter: Realization and Implementation

III.1. INTRODUCTION

In the previous section, we comprehended our problem better. We view where our problem lies and proposed a solution, the PSO. Afterward, we did an extensive review of our chosen method where we understood exactly how it functions and on which bases. We used the conceptual model for further explanation of the techniques and how it operates. Diagrams like the activity and sequence were used to help explain more efficiently.

In this chapter, however, we get a more practical view. We start by the display of the tools and equipment used in the designing and implementation of the PSO-based path-finding robot.

We have chosen PYTHON as the primary programming language. We have chosen PYTHON as the primary programming language. Python is one of the most mention-worthy programming languages in the modern world. It is high among the fastest-growing programming language worldwide. It is adaptable, flexible, extremely efficacious, and straightforward to employ and develop. It has an extremely active society as well. It is utilized in countless organizations due to its numerous programming paradigm asset and its implementation of automatic memory management. Due to its extensive standard library, Python is also often referred to as a battery-included language.

Once that is clear, we move on to the implementation part of our thesis in which we display the result of our work. We provide a detailed execution of our program with illustrations of different phases of execution with a special case example. We conclude with a discussion section in which we compare how different settings affect the functionality of the robot in its process of finding the shortest path possible between two points in an environment with obstacles.

III.2. TOOLS :

1. *Software Environment:*

Python is easily the most well-known and used open-source programming language for IT professionals. Python is a diagnosed, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, mixed with dynamic typing and dynamic binding, render it extremely appealing for Rapid Application Development and for use as a scripting or glue language to connect existing components. Python is a

straightforward, easy-to-learn syntax that highlights readability and, thus, lessens program maintenance expenses. Python supports modules and packages, which boost program modularity and the reuse of the code [64].

Python is a renowned programming language. It was developed in 1991 by Guido van Rossum [65]. It is utilized for web development (server-side), software development, mathematics, and system scripting. Python can be used:

- On a server to construct web applications.
- Alongside software to build workflows.
- To manage big data and conduct complicated mathematics.
- For fast prototyping, or for production-ready software development.
- Python can link to database systems. It can also read and alter files.

Python Syntax compared to other programming languages:

- Python was developed for readability and resemblances to the English language with influence from mathematics.
- Python operates new lines to achieve a command, compared to other programming languages which usually use semicolons or parentheses.
- Python relies on indentation, applying whitespace, to specify a scope; such as the scope of loops, functions, and classes. While other programming languages apply curly brackets for this purpose.

Figure III.2:1: Python Official Logo [64]

2. *Libraries Used*

Python's standard library is very comprehensive, presenting a broad range of structures. The library includes built-in modules (written in C) that supply admission to system functionality such as file I/O that would otherwise be unreachable to Python programmers, as well as modules composed in Python that furnish standardized solutions for numerous difficulties that arise in everyday programming. Few of these modules are explicitly developed to encourage and improve the portability of Python programs [66].

- ***NumPy***

NumPy is the essential package for scientific computing in Python. It is a Python library that supplies a multidimensional array object, diverse derived objects (such as masked arrays and matrices), and a variety of patterns for swift procedures on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [67].

Figure III.2:2: NumPy Official Logo [67]

- ***SciPy***

SciPy, a scientific library for Python is an open-source, BSD-licensed library for mathematics, science, and engineering. The SciPy library relies on NumPy, which supplies suitable and fast N-dimensional array manipulation. The primary motivation for creating the SciPy library is functions with NumPy arrays. It delivers numerous user-friendly and efficient numerical practices such as patterns for numerical integration and optimization. Interpolation is the methodology of locating a value between two points on a line or a curve [68].

1-D Interpolation: The interp1d class in the scipy.interpolate is a convenient technique to make a function founded on fixed data points, which can be assessed anywhere within the environment represented by the provided data using linear interpolation.

Figure III.2:3: SciPy Official Logo [68]

- ***Matplotlib***

Matplotlib is a comprehensive library for constructing static, animated, and interactive visualizations in Python. Matplotlib delivers publication-quality figures in a mixture of hardcopy formats and interactive environments across platforms [69].

Figure III.2:4: Matplotlib Official Logo [69]

- ***Pillow***

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It includes weightless image processing means which helps in editing, developing, and preserving images. Pillow was declared as a substitute for PIL for future use. Pillow supports an extensive number of image file formats like BMP, PNG, JPEG, and TIFF. it was designed for quick access to data kept in a few basic pixel formats. It ultimately provides a substantial basis for general image processing equipment [70].

Figure III.2:5: Pillow Official Logo [70]

- ***Copy***

Copy is occasionally required so one can modify one copy without the other. A user might want copies that he can alter without automatically affecting the original at the same time, to achieve that we create copies of objects. In Python, there are two kinds of copies: Deep copy and Shallow copy

In our algorithm, we use the deep copy. Deep copy is an operation in which the copying procedure occur recursively. It means first creating a new collection object and then recursively occupying it with copies of the child objects located in the original [71].

```
import numpy as np
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse, Circle, Polygon
from pso import PSO
```

Figure III.2:6: Libraries used in the PathPlanning

```
import sys
from copy import deepcopy
import numpy as np
from matplotlib.animation import FuncAnimation
```

Figure III.2:7: Libraries Used in the Test

III.3. WORKING ENVIRONMENT :

Before thinking of placing a robot to search and navigate, we first need to consider where to place it. The search space is essential in the navigation problem especially if it is considered dynamic and the robot has no prior knowledge of it. Nevertheless, the search space in one way or another requires a computer representation. The environment we choose for our robot to search in is a 2-dimensional XY coordination, where x denotes the position of the point concerning the x-axis, and y represents the position of the point on the y-axis. The limits of the environment are one of the parameters that the user has control over and has the liberty in setting.

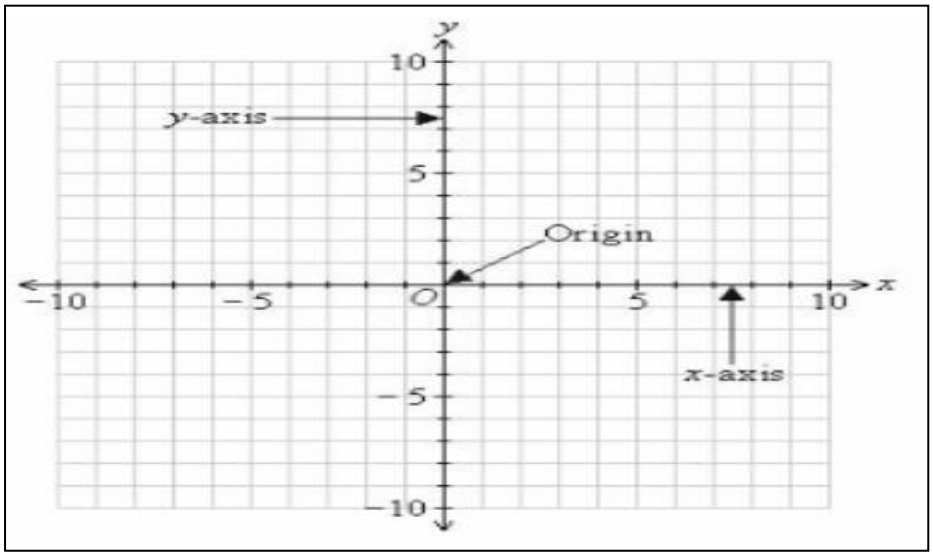

Figure III.3:1: Two Dimensional (X, Y) Representation of the Environment

1. Initialization of Obstacles

Once we initialize a 2-dimensional environment, and before obtaining the start and goal point, we initialize the obstacles. Now the obstacle from their initial position which the user allocates, the chosen commence to stochastically move at a speed determined by the user in the range between -0.5 and 0.5.

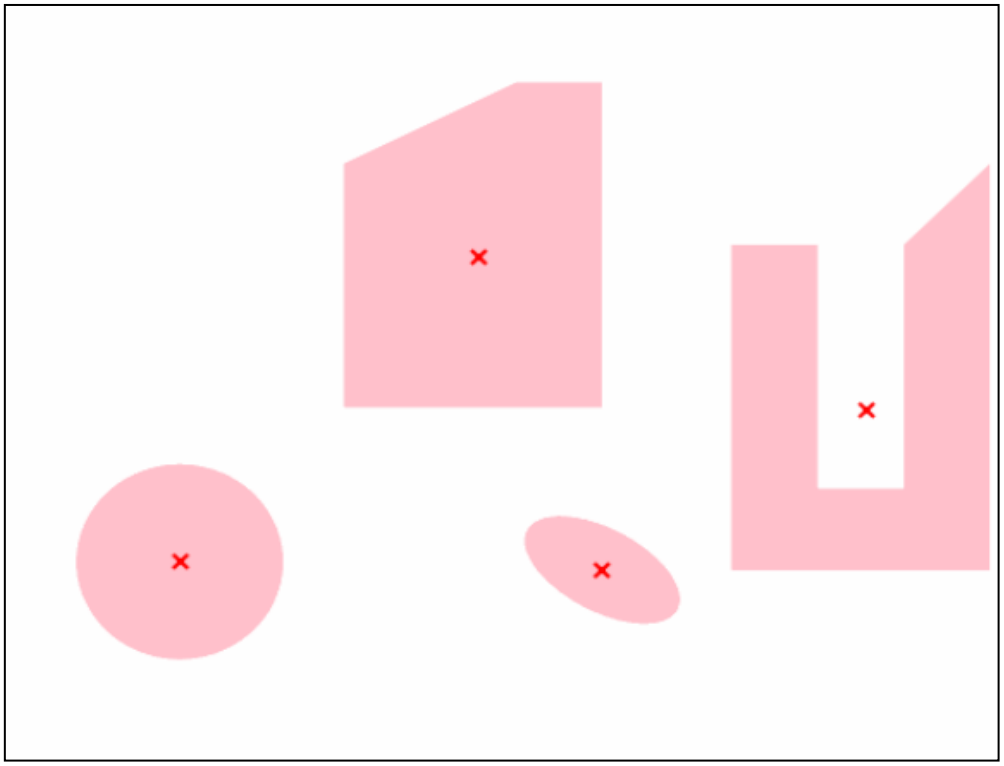

Figure III.3:2: The Environment in which Our Robot Will Function

Figure III.3:2 is the initial position of the obstacles. In our implementation, we chose the circle to be the moving. The circle starts moving in one direction at a speed determined by the user. Once the robot conducts a certain number of runs, also determined by the user, the obstacle changes its direction and so on until the number of runs is completed. Points on the obstacle borders/edges are not considered inside the obstacle therefore the robot can use them to advance along those borders. Basically, the user is in charge of the following:

- Number of swarms: the number of agents
- The start and goal points: the initial point that the robot starts its search from and the goal which must be found
- Number of runs: number of iterations the robot conducts to locate the goal
- The speed of the robot: the speed at which the robot navigates once it discovers a better position closer to the goal
- The speed of the obstacle: the speed that the obstacle moves in, which is important as it affects the path of the robot. If the robot and the obstacle have different speeds it can cause further effect on the robot forcing it to alter its original path or it can collide with the robot and that contact in a way pushes the robot off its course and forces changes on the robot while it moving.
- Invert direction: the number of runs before the obstacle reverses its direction. This can make the robot recalculates and finds another path to the goal than the one it initially found.

Python Code

```
# Move the circular obstacle along a straight line with a "speed" of 0.1
vel_o = 0.1

while run < nRun:
    if run % 15 == 0 :
        vel_o = -vel_o          # Invert direction after 15 runs
    run+=1
theta_o = +150.0 * np.pi / 180.0
x_o = layout.obs[3][1] + vel_o * np.cos(theta_o)
y_o = layout.obs[3][2] + vel_o * np.sin(theta_o)
layout.remove_obs(3)
layout.add_circle(x_o, y_o, r=1.2, Kv=100)
```

Figure III.3:3: Python code of Obstacles Speed and Direction Inversion

2. Initialization of Points

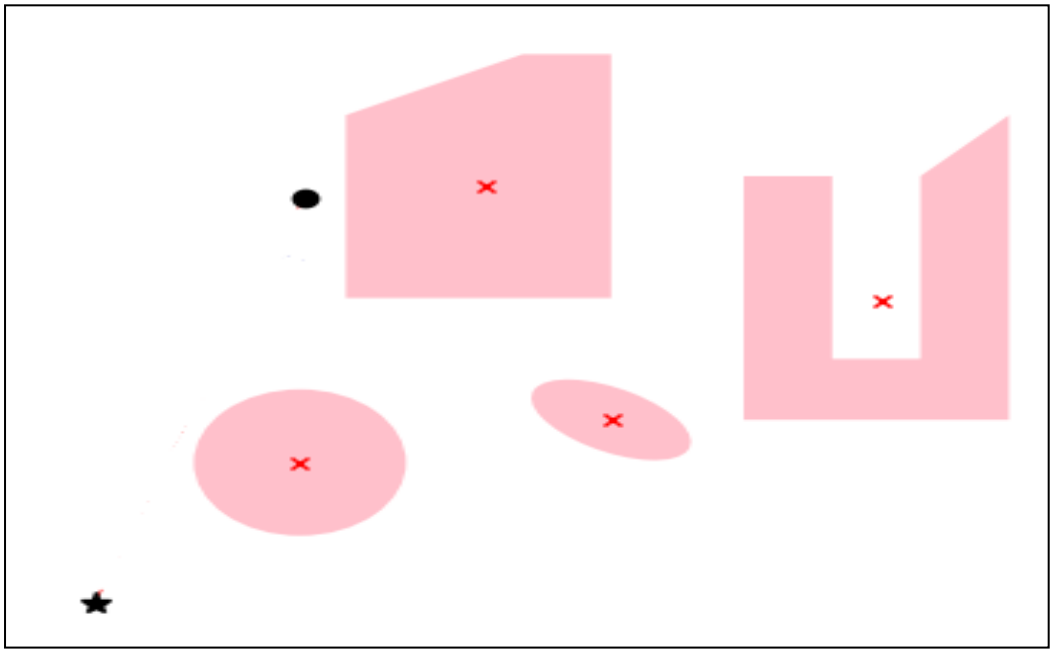

Figure III.3:4: the start and goal point representation

Figure III.3:4 is the visual manifestation of our algorithm. The user enters the X and Y coordinates of the start and goal points. These points each need a distinct representation for a better view and comprehension. The black dot denotes the start point where our robot will begin the search, and the black star denotes the goal to which we search for the shortest path.

III.4. IMPLEMENTATION

After setting the necessary point and establishing the obstacles, it is time to display the results. As mentioned earlier in this chapter, we used scipy.interpolation to represent data, or in our case the results we established from the PSO algorithms search. With a considerable number of data points or in our case solutions, it is more suitable to keep the order of interpolation low. This can be done with piecewise interpolation utilizing splines. Splines are polynomials on subintervals that are linked with a certain degree of continuity.

Initially, our PSO finds all the possible solutions or paths from the starting point to the goal regardless of their correctness or the obstacles then the filtering begins. We used the Pillow package to obtain these results in an image form as shown below:

1. Path Finding

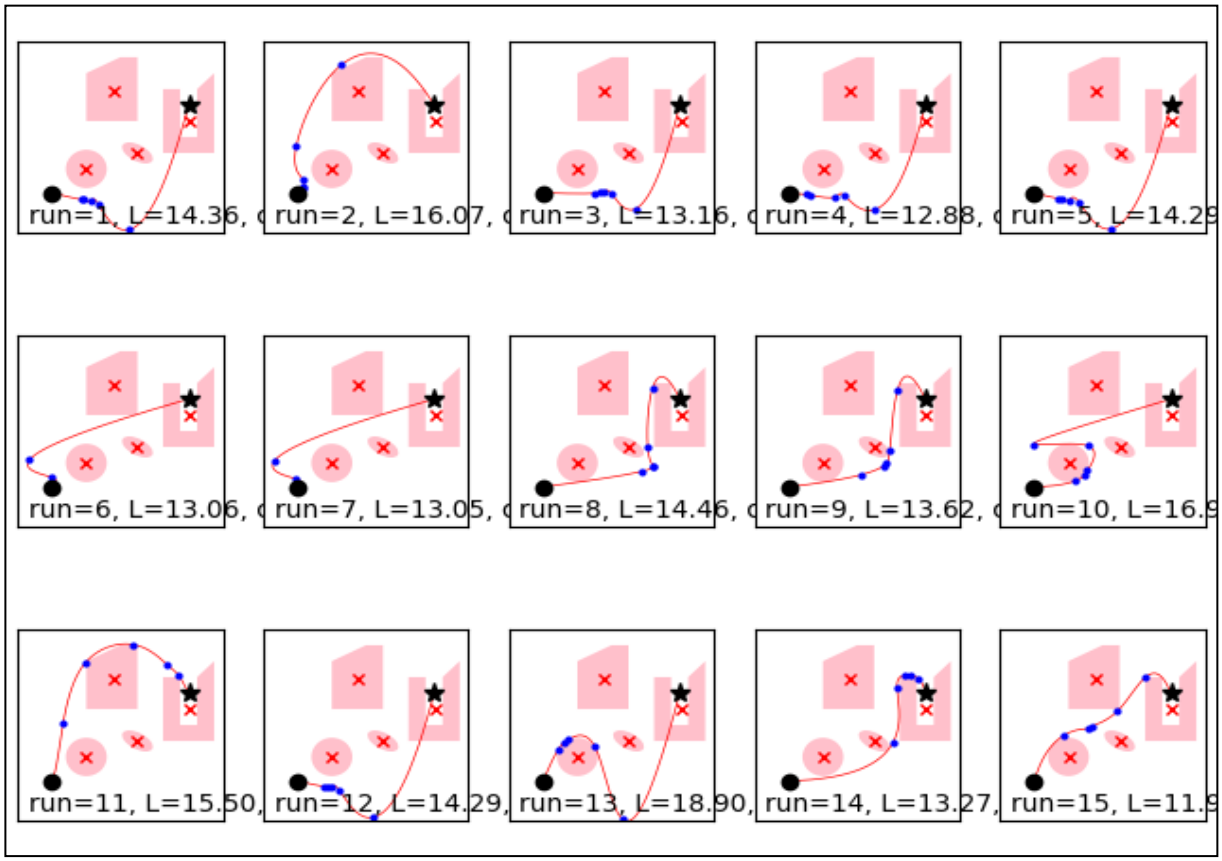

Figure III.4:1: Implementation example of path finding using PSO

```
ns =   301

run=1,  L=14.36,  count=1
run=2,  L=16.07,  count=0
run=3,  L=13.16,  count=1
run=4,  L=12.88,  count=1
run=5,  L=14.29,  count=1
run=6,  L=13.06,  count=1
run=7,  L=13.05,  count=1
run=8,  L=14.46,  count=0
run=9,  L=13.62,  count=0
run=10,  L=16.99,  count=1
run=11,  L=15.50,  count=0
run=12,  L=14.29,  count=1
run=13,  L=18.90,  count=1
run=14,  L=13.27,  count=0
run=15,  L=11.91,  count=0

Best:  run=15,  L=11.91,  count=0
```

Figure III.4:2: System Execution

Figure III.4:1 above displays the possible solutions for the path planning between the start and goal point, found by the PSO without an initial solution is available. As shown, it initially finds all possible solutions regardless of the obstacles which are then considered by the system as a violation of the obstacle. **Figure III.4:2** show the system while executing. It displays:

- The run: The number of iteration currently being executed
- L: signifies the length which is the new updated distance between the start and goal point at that exact moment
- Count: by count, it refers to the number of obstacles violated by the robot in order to reach the goal in the shortest distance for example take run 1

In the end, the system filters the result found according to the ending criteria:

- Is the goal reached? YES
- Is it the shortest path? YES
- Are any obstacles violated? NO

Once the system filters the runs achieved, it obtains all the runs that fulfill the conditions then it conducts a comparison among them based on the length the run with the smallest length is displayed as the most optimal solution as shown above.

2. Navigating

Next, we have an implementation of our robot navigating through the environment from a starting point to an end while avoiding obstacles.

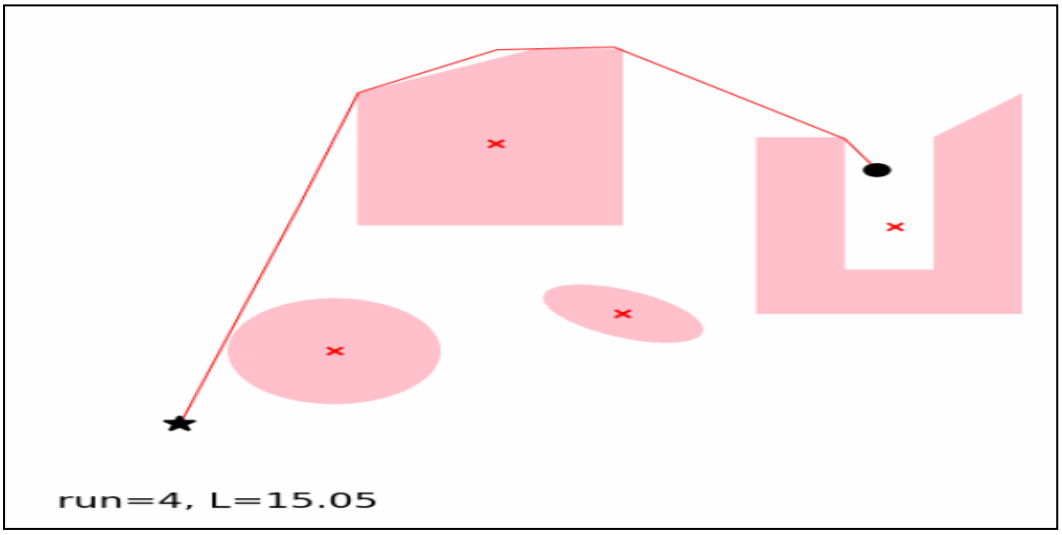

Figure III.4:3: A

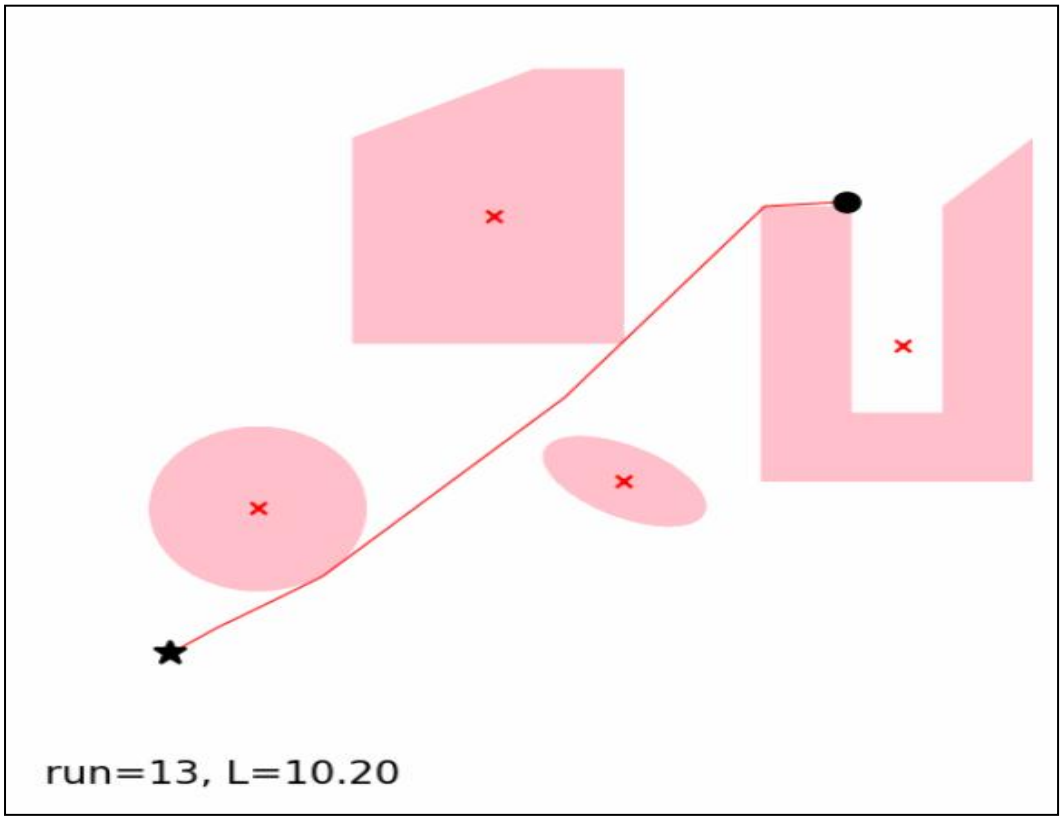

Figure III.4:4: B

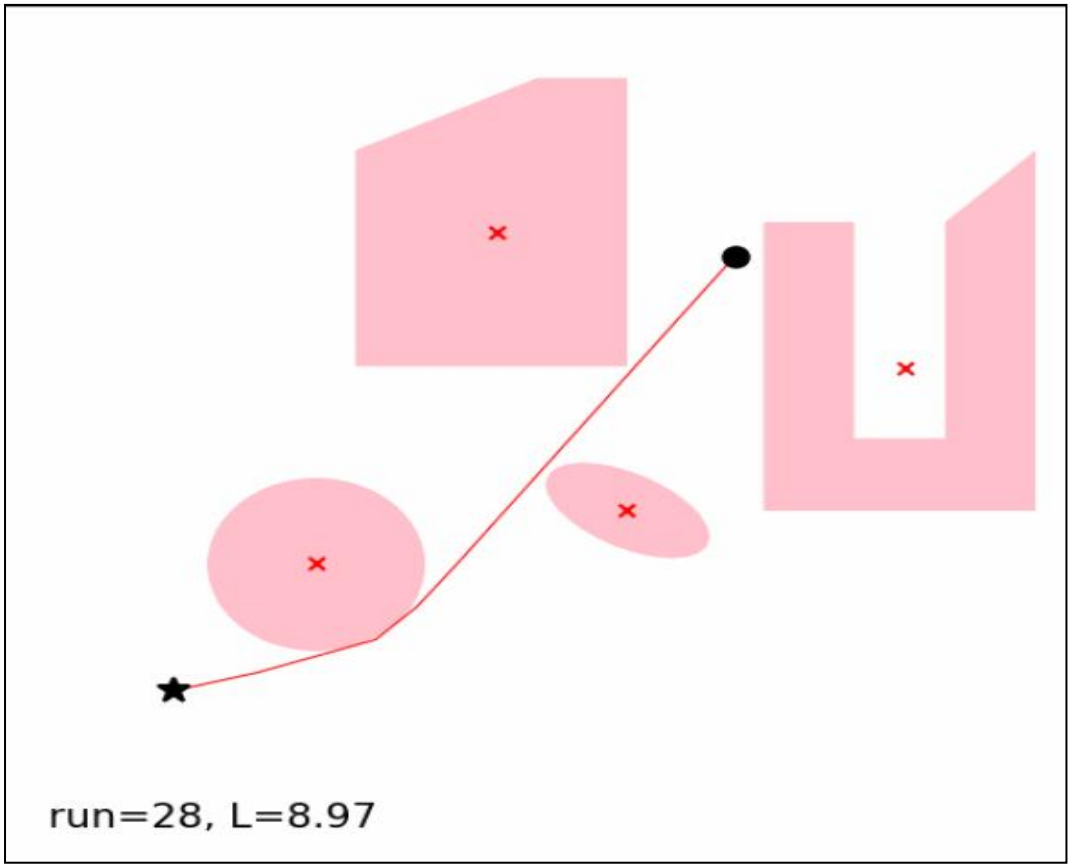

Figure III.4:5: C

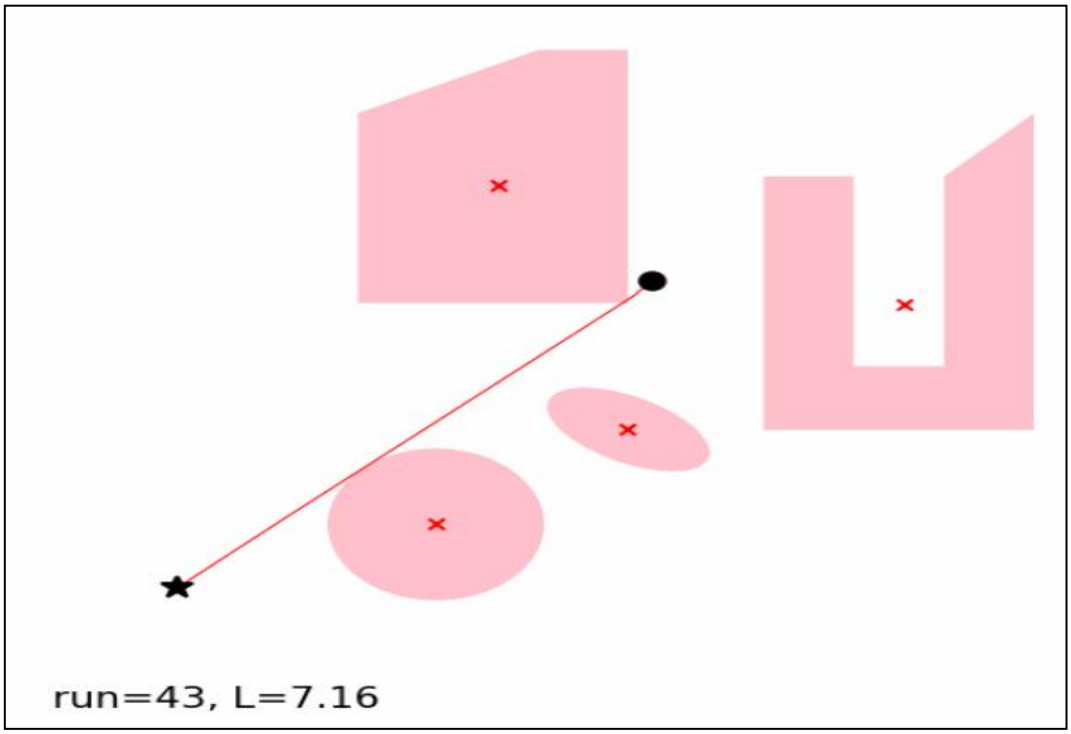

Figure III.4:6: D

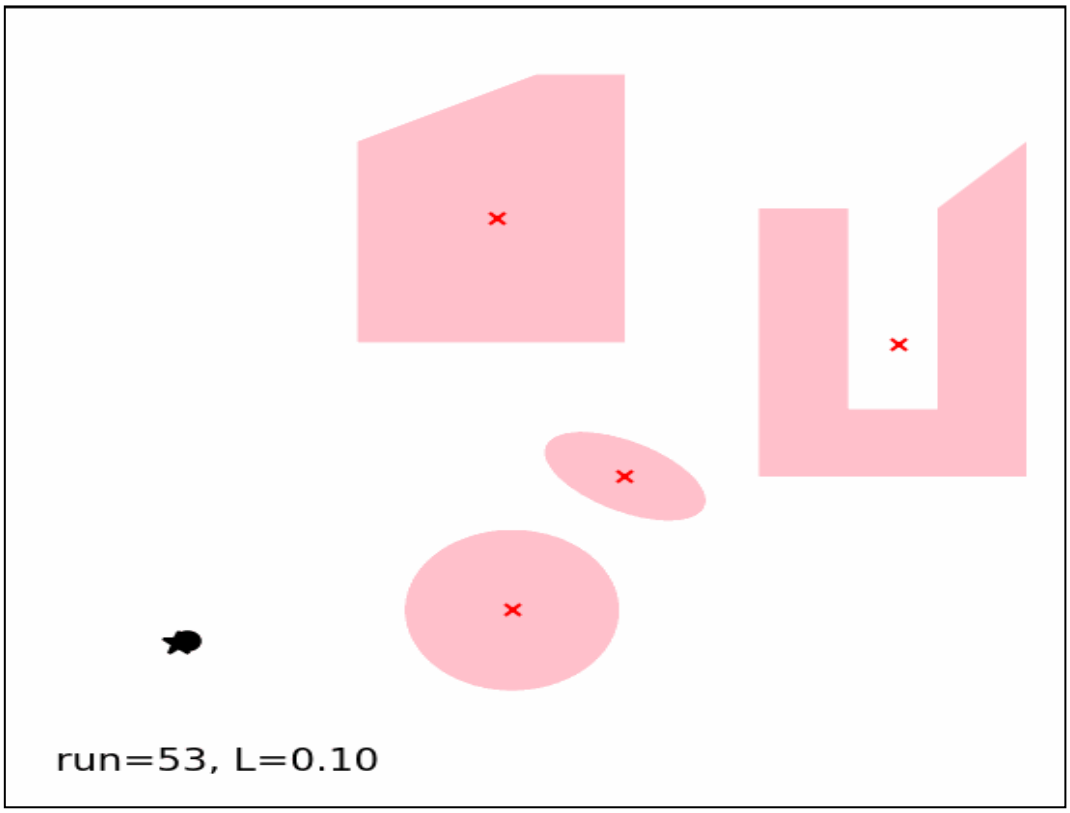

Figure III.4:7: E

Figure III.4 from 3 to 8 represent our robot conducting its journey from a starting point to an endpoint in an environment with obstacles, we will be reviewing each stage:

As shown in A: the beginning of the execution, the robot selects a path, preferably one with the least amount of obstacles, as an initial solution, and proceeds on that path. Now, according to the robot, that path is the most optimal until a moving obstacle intervenes.

In B, we notice how the moving obstacle (the circle) intervened with the path the robot had in mind, making it longer. Therefore, the robot is obligated to find an alternate way, leading it to discover a much more optimal path than the original, and once more proceeds on it.

In C & D, we see the path is altered by the moving obstacle once again, and our robot, in both cases, finds the shortest trajectory to the target around said obstacle

In E, the robot has reached the target through the most optimal path without violating any obstacles and successfully evading them.

```
ns =  301

run=1, L=15.35, count=0, s=[8.00,2.00], g=[0.00,-3.50], c=[2.00,-2.00]
run=2, L=15.25, count=0, s=[7.95,2.09], g=[0.00,-3.50], c=[1.91,-1.95]
run=3, L=15.15, count=0, s=[7.91,2.18], g=[0.00,-3.50], c=[1.83,-1.90]
run=4, L=15.05, count=0, s=[7.86,2.27], g=[0.00,-3.50], c=[1.74,-1.85]
run=5, L=15.07, count=0, s=[7.81,2.35], g=[0.00,-3.50], c=[1.65,-1.80]
run=6, L=15.04, count=0, s=[7.77,2.44], g=[0.00,-3.50], c=[1.57,-1.75]
run=7, L=15.00, count=0, s=[7.73,2.53], g=[0.00,-3.50], c=[1.48,-1.70]
run=8, L=13.11, count=0, s=[7.69,2.63], g=[0.00,-3.50], c=[1.39,-1.65]
run=9, L=12.57, count=0, s=[7.64,2.72], g=[0.00,-3.50], c=[1.31,-1.60]
run=10, L=11.35, count=0, s=[7.60,2.80], g=[0.00,-3.50], c=[1.22,-1.55]
run=11, L=11.20, count=0, s=[7.55,2.89], g=[0.00,-3.50], c=[1.13,-1.50]
run=12, L=11.33, count=0, s=[7.50,2.98], g=[0.00,-3.50], c=[1.05,-1.45]
run=13, L=10.20, count=0, s=[7.45,3.07], g=[0.00,-3.50], c=[0.96,-1.40]
run=14, L=10.08, count=0, s=[7.35,3.06], g=[0.00,-3.50], c=[0.87,-1.35]
run=15, L=9.98, count=0, s=[7.25,3.05], g=[0.00,-3.50], c=[0.79,-1.30]
run=16, L=9.88, count=0, s=[7.15,3.04], g=[0.00,-3.50], c=[0.70,-1.25]
run=17, L=9.78, count=0, s=[7.05,3.04], g=[0.00,-3.50], c=[0.61,-1.20]
run=18, L=9.68, count=0, s=[6.95,3.03], g=[0.00,-3.50], c=[0.70,-1.25]
run=19, L=9.58, count=0, s=[6.85,3.02], g=[0.00,-3.50], c=[0.79,-1.30]
run=20, L=9.48, count=0, s=[6.75,3.02], g=[0.00,-3.50], c=[0.87,-1.35]
```

Figure III.4:8: System Outcome of the Iterations

As the figure illustrates and as we aforementioned, the system displayed with each run its number, the length, and the count. In addition, considering that our robot is now navigating and moving its way in the environment it constantly shows:

- g: The coordinates of the goal which is stable for the moment.
- s: The coordinates of the start which change with each iteration given the fact that the PSO works by finding all the points surrounding the start point and then selecting that which is closest to the goal. Therefore, each run has a new starting coordination.
- c: the coordination of the center of the moving obstacle that also changes with each iteration.

3. Special Case

Our aim is a realistic presentation of navigation usable in our daily lives. Therefore, we mimic a real-life situation in **Figure III.4:8**. The figure illustrates the case when our robot crosses paths with a moving obstacle, as it is far too late to evade the obstruction, and the contact is inevitable.

When the robot is too far into a path that it is not possible to recalculate around an obstacle mainly because the robot is already in the closest position to the goal, the robot then falls under the influence of the moving object, meaning the obstacle continues its current path unaffected while the robot gets pushed along the current direction of the obstacle. When that occurs, that force influences the calculation conducted by the robot. Under that impact, the robot is driven to find a path that might appear stochastic until the force placed upon it is gone, then it returns to its original state.

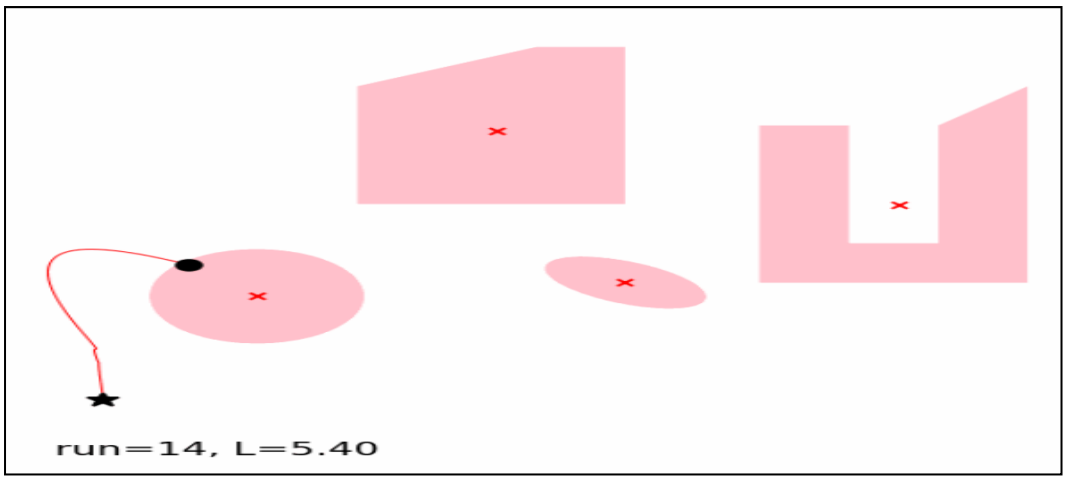

Figure III.4:9: The Moving Obstacle and the Robot Cross Paths

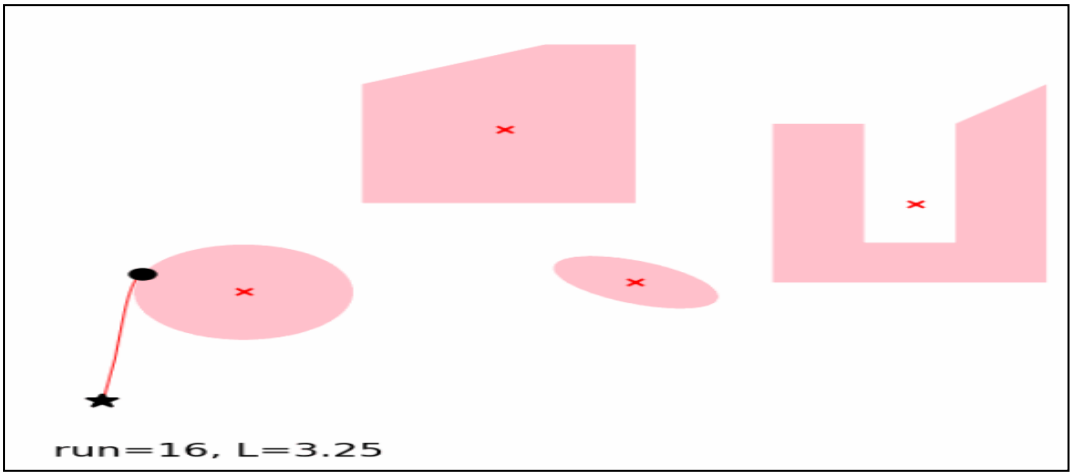

Figure III.4:10: The Robot Regain State after the Contact with the Obstacle is Gone

III.5. DISCUSSION

	Environment	Starting	Ending	Obstacles	Robot speed	Obstacle speed	Number of Runs
1	[-2, 10, -6, 6]	8, 2	0, -3.5	1 MOVING	0.1	0.1	103
2	[-2, 10, -6, 6]	8, 2	0, -3.5	1 MOVING	0.5	0.1	29

Table III.5-1: Comparison between Different Speeds

When it comes to the navigation domain we cannot consider just one scenario. Diversity is an important factor because it mirrors the real world which is where we like our work to be executed. This makes it possible to identify a set of criteria to ensure the quality of such a solution. To best test our PSO algorithm, we alter a few of the input parameters and compared the results. The main aspect we are focusing on is the speed, both the robot and the obstacles. The speed the robot is navigating in can either cause it to rapidly evade the obstacle or crash into them.

As shown in the table, all the experiments have the same setting except for the speed of the robot and the obstacle, for example, notice how with the same circumstances, the same starting point, and the same goal, the robot with the speed of 0.1 completed the journey in 103 iterations while the robot with 0.5 speed completed in 29.

	Environment	Arrival point	Departure point	Euclidien distance	Manhattan distance	Obstacles	Path length (with obstacles)	Convergence rate
BFO/APF	[10X 10]	9, 9	0, 0	12.72	18	10	22	**81.81%**
GA							22	**81.81%**
ANN & QL							23	**78.26%**
A*							22	**81.81%**
BFO/APF	[100X 100]	99, 99	0, 0	140.00	198	100	212	**93.33%**
GA							220	**90%**
ANN & QL							235	**84.25**
A*							212	**93.33%**
BFO/APF	[1000X 1000]	999, 999	0, 0	1412.79	1998	1000	2155	**92.71%**
GA							2269	**88.05%**
ANN & QL							2705	**73.86%**
A*							/	**Stack overflow**

III.6. CONCLUSION

In this chapter, we displayed the implementation part of our thesis. We placed our algorithms in function and brought those results for discussion. First, we presented the tools we used in creating the algorithm. Python is our principal programming language, along with the libraries applied to make the execution easier such as Numpy, Scipy, copy, etc. Then we began showing the initialization of our workspace, the points obstacles, etc. In the actual implementation, we presented illustrated explanation, all the path the robot finds, how it selects the shortest path, and how it navigates through the environment to get to the goal. We also presented the special case on when the robot makes inevitable contact with the robot. In the end, we discussed the result, compared the execution of the algorithms under different parameters, and observed the results.

In the next section, we will review what we have did thus far, the chapters, the implementation, the result, etc., and share our future plans and perspectives.

GENERAL CONCLUSION

In conclusion, we defined a robot as any automatically operated device that serves as a substitute for human effort. Even though it might lack the human physical appearance or the humanlike manner, in which it performs those functions. By extension, robotics is the engineering specialization handling the design, construction, and operation of robots.

The main problem the robotics faces is the navigation problem to be precise the shortest path problem. So the predicament is to make AMRs navigate and find the shortest path in an environment with obstacles? As a solution, we proposed the PSO. We explained our choice in three chapters: the first is where we presented a review of the efforts of the researchers to solve this same problem with similar methods. Afterward, we conducted a comparison between the chosen techniques.

The second chapter is where we provided an illustrated explanation of our system and how it functions, with a comprehensive explanation of the chosen technique.

The last chapter is where we discuss the implementation part. We presented an illustrated model step by step of our system in action. We also provided a special case scenario, where our robot comes to inevitable contact with the obstacle.

To make this possible we used PYTHON as the primary programming language for its straightforward, easy-to-learn syntax that highlights readability and, therefore, reduces program maintenance expenses. Several libraries were used for instance the Pillow was to display the results in an image form; NumPy is in charge of arrays which we used for the solution, etc.

We have several future perspectives on our work. An example is the notion of a moving destination where the robot chases a moving endpoint. Our constant object is always to ameliorate the PSO by lowering its complexity and even attempts hybrid the PSO with other algorithms to solve this and other problems in the future.

BIBLIOGRAPHY

[1] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th international symposium on micro machine and human science*, Nagoya,Japan, 1995.

[2] J. Ni, L. Wu, X. Fan and S. Yang, "Bioinspired Intelligent Algorithm and Its Applications for Mobile Robot Control: A Survey," in *Comput Intell Neurosci*, 2016.

[3] K. Park, Y. Kim and J. Kim, "Modular Q-Learning based Multi-Agent Cooperation for Robot Soccer," *ELSEVIER Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 109-122, 2001.

[4] M. Weigl, B. Siemiaatkkowska, K. Sikorski and A. Borkowski, "Grid-Based Mapping for Autonomous Mobile Robot.," *ELSEVIER Robotics and Autonomous Systems*, vol. 11, no. 1, pp. 13-21, 1993.

[5] A. Soltani, H. Tawfik, J. Goulermas and T. Fernando, "Path Planning in Construction Sites: Performance Evaluation of the Dijkstra, A∗, and GA Search Algorithms," *ELSEVIER Advanced Engineering Informatics*, vol. 16, no. 4, pp. 291-303, 2002.

[6] E. Gomez, F. Martinez Santa and F. Martinez Sarmiento, "Comparative Study of Geometric Path Planning Methods for a Mobile Robot: Potential Field and Voronoi Diagrams.," in *IEEE International Congress of Engineering Mechatronic and Automation (CIIMA)*, Colombia, 2013.

[7] R. Abiyev, D. Ibrahim and B. Erin, "Navigation of Mobile Robots in the Presence of Obstacles," *Advances in Engineering Software*, vol. 41, no. 10, pp. 1179-1186, 2010.

[8] E. Masehian and M. Amin-Naseri, "A Voronoi Diagram–Visibility Graph–Potential Field Compound Algorithm for Robot Path Planning.," *Journal of Robotic System*, vol. 21, no. 6, pp. 275-300, 2004.

[9] O. Takahashi and R. Schilling, "Motion Planning in a Plane Using Generalized Voronoi Diagrams," *IEEE Robotics and Automation*, vol. 5, no. 2, pp. 143-150, 1989.

[10] P. Bhattacharya and M. Gavrilova, "Roadmap-Based Path Planning-Using the Voronoi Diagram for a Clearance-Based Shortest Path," *IEEE Robotics and Automation*, vol. 15, no. 2, pp. 58-66, 2008.

[11] M. Garcia, O. Montiel, O. Castillo, R. Sepulveda and P. Melin, "Path Planning for Autonomous Mobile Robot Navigation with Ant Colony Optimization and Fuzzy Cost Function Evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102-1110, 2009.

[12] H. Miao and Y. Tian, "Dynamic Robot Path Planning using an Enhanced Simulated Annealing Approach," *ELSEVIER Applied Mathematics and Computation*, vol. 222, pp. 420-437, 2013.

[13] I. Engedy and G. Horvath, "Artificial Neural Network based Local Motion Planning of a Wheeled Mobile Robot.," in *IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, Hungary, 2010.

[14] A. Ghorbani, S. Shiry and A. Nodehi, "Using Genetic Algorithm for a Mobile Robot Path Planning.," in *In IEEE International Conference on Future Computer and Communication*, Malaysia, 2009.

[15] M. Montaner and A. Ramirez-Serrano, "Fuzzy Knowledge-Based Controller Design for Autonomous Robot Navigation.," *ELSEVIER Expert Systems with Applications*, vol. 14, no. 1, pp. 179-186, 1998.

[16] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *EEE Control Syst*, vol. 22, no. 3, pp. 52-67, 2002.

[17] L. Zadeh, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning-I.," *ELSEVIER Information*, vol. 8, no. 3, pp. 199-249, 1975.

[18] A.-M. Khalid, M. Ebrahim and A. Mansour, "Implementation of fuzzy decision based mobile robot navigation using stereo vision," *Procedia Computer Sciences*, vol. 62, pp. 143-150, 2015.

[19] N. M. A. Davood and K. Mohammad Hassan, "Design of optimal Mamdani-type fuzzy controller for nonholonomic wheeled mobile robots," *Journal of King Saud–Engineering Sciences*, vol. 27, pp. 92-100, 2015.

[20] O. Castillo, H. Neyoy, J. Soria, P. Melin and F. Valdez, "A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot," *Appl Soft Comput*, vol. 28, pp. 150-159, 2015.

[21] A.-J. Rami, S. Amir and R. Huber, "Path planning and motion coordination for multi-robot's systems using probabilistic neuro fuzzy," *IFAC-papers on line*, vol. 48, no. 10, pp. 046-051, 2015.

[22] B. Patle, D. Parhi, A. Jagadeesh and K. Sunil Kumar, "Probabilistic fuzzy controller based robotics path decision theory," *World Journal of Engineering*, vol. 13, no. 2, pp. 181-192, 2016.

[23] A. Rath, D. Parhi, H. Das, M. Muni and P. Kumar, "Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone," *Defence Technology*, vol. 14, no. 6, pp. 677-682, 2018.

[24] A. Y, M. S.A.A and N. A.B, "Formation control of aerial robots using virtual structure and new fuzzy-based self-tuning synchronization," *Transactions of the Institute of Measurement and Control* , vol. 39, no. 12, pp. 1906-1919, 2017.

[25] X. Xianbo and e. al., "Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles," *International Journal of Fuzzy Systems* , vol. 20, p. 572–586 , 2018.

[26] A. Kashyap and D. Parhi, "Obstacle avoidance and path planning of humanoid robot using fuzzy logic controller aided owl search algorithm in complicated workspaces," *Industrial Robot*, vol. 49, no. 2, pp. 280-288, 2022.

[27] Y. H. M, D. T, H. N, R. P and O. a. S. K. Y, "Performance Comparison of Fuzzy Logic and Neural Network Design for Mobile Robot Navigation," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2019.

[28] R. X, R. D, Z. X and H. and J, "Mobile Robot Navigation based on Deep Reinforcement Learning," in *2019 Chinese Control And Decision Conference (CCDC)*, 2019.

[29] C. C and C. Y, "A Neural Network based Mobile Robot Navigation Approach using Reinforcement Learning Parameter Tuning Mechanism," in *2021 China Automation Congress (CAC)*, 2021.

[30] B. H.J, "The evolution of intelligence. The Nervous system as a model of its environment," Washington, Seattle, 1958.

[31] M. Algabri, H. Mathkour, R. Hedjar, M. Alsulaiman and K. Al Mutib, "Self-learning Mobile Robot Navigation in Unknown Environment Using Evolutionary Learning," *Journal of Universal Computer Science*, vol. 20, no. 10, pp. 1459-1468, 2014.

[32] T. Arora, Y. Gigras and V. and Arora, "Robotic Path Planning using Genetic Algorithm in Dynamic Environment," *International Journal of Computer Application*, vol. 89, no. 11, pp. 8-12, 2014.

[33] M. Nazarahari, E. Khanmirzab and S. Doostiea, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Systems with Applications*, vol. 115, pp. 106-120, 2019.

[34] N. Jianjun, W. Kang, H. Haohao, W. Liuying and L. Chengming, "Robot path planning based on an improved genetic algorithm with variable length chromosome," in *12th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, 2016.

[35] C. JiaWang and e. al, "Research on fuzzy control of path tracking for underwater vehicle based on genetic algorithm optimization," *Ocean Eng*, vol. 156, pp. 217-223, 2018.

[36] R. Vincent, T. Mohammed and L. Gilles, "Fast genetic algorithm path planner for fixed-wing military UAV using GPU," in *IEEE Trans Aerosp Electron Syst* , 2018.

[37] R. M. T. Vincent, "Massively parallel hybrid algorithm on embedded graphics processing unit for unmanned aerial vehicle path planning," *International Journal of Digital Signals and Smart Systems*, vol. 2, no. 1, pp. 68-93, 2018.

[38] K. A, B. K. Priyadarshi and P. Dayal R, "Intelligent navigation of humanoids in cluttered environments using regression analysis and genetic algorithm," *Arabian Journal for Science and Engineering* , vol. 43, p. 7655–7678, 2018.

[39] P. B.K, P. D.R. K, J. A and K. Sunil Kumar, "Matrix-binary codes based genetic algorithm for path planning of mobile robot," *Computers & Electrical Engineering*, vol. 67, pp. 708-728, 2018.

[40] M. Dorigo and G. Caro, "Ant Colony Optimization: A New Meta-Heuristic.," in *IEEE International Congress on Evolutionary Computation*, Serbia, 1999.

[41] C. O, N. H, S. J, M. P and V. F, "A New approach for dynamic fuzzy logic parameter tuning in ant colony optimisation and its application in fuzzy control of a mobile robot," *Applied Soft Computing*, vol. 28, pp.

150-159, 2015.

[42] K. P.B, S. C and P. D.R, "A hybridized regression-adaptive ant colony optimization approach for navigation of humanoids in a cluttered environment," *Applied Soft Computing*, vol. 68, pp. 565-585, 2018.

[43] L. J, Y. J, L. H, T. X and G. M, "An improved ant colony algorithm for robot path planning," *Soft Computing*, vol. 21, no. 19, pp. 5829-5839, 2017.

[44] R. U and K. M, "Mobile robot path planning with modified ant colony optimisation," *International Journal of Bio-Inspired Computation*, vol. 9, no. 2, pp. 106-113, 2017.

[45] K. Saroj, P. Krishna Kant, M. Manoj Kumar and P. Dayal R, "Path Planning of the Mobile Robot Using Fuzzified Advanced Ant Colony Optimization," in *Innovative Product Design and Intelligent Manufacturing Systems*, Singapore, 2020.

[46] M. Hossain and I. Ferdousand, "Autonomous Robot Path Planning in Dynamic Environment using a New Optimization Technique Inspired by Bacterial Foraging Technique," *Robotics and Autonomous Systems 64:* ., vol. 64, pp. 137-141, 2015.

[47] A. N.H and A. F.M, "Path planning of an autonomous mobile robot using enhanced bacterial foraging optimization algorithm," *Al-Khwarizmi Engineering Journal*, vol. 12, no. 4, pp. 26-35, 2017.

[48] M. Brand and X. Yu, "Autonomous Robot Path Optimization using Firefly Algorithm.," in *IEEE International Conference on Machine Learning and Cybernetics (ICMLC)*, China, 2013.

[49] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087-1092, 1953.

[50] T. Yanar and Z. Akyurek, "Fuzzy Model Tuning Using Simulated Annealing.," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8159-8169, 2011.

[51] R. Tavares, T. Martins and M. and Tsuzuki, "Simulated Annealing with Adaptive Neighbourhood: A Case Study in Off-Line Robot Path Planning.," *Expert Systems with Applications*, vol. 38, no. 4, pp. 2951-2965, 2011.

[52] Q. Zhang, J. Ma and Q. Liu, "Path Planning based Quad tree Representation for Mobile Robot Using Hybrid-Simulated Annealing and Ant Colony Optimization Algorithm.," in *IEEE International World Congress on Intelligent Control and Automation*, China, 2012.

[53] S. Ahmadzadeh and M. Ghanavati, "Navigation of Mobile Robot Using the PSO Particle Swarm Optimization.," *Journal of Academic and Applied Studies (JAAS)*, vol. 2, no. 1, pp. 32-38, 2012.

[54] G. Li and W. Chou, "Path planning for mobile robot using self-adaptive learning particle swarm optimization.," *Science China Information Sciences*, p. 61, 2018.

[55] B. Song, Z. Wang and L. Zou, "On Global Smooth Path Planning for Mobile Robots using a Novel Multimodal Delayed PSO Algorithm.," *Cognitive Computation volume*, vol. 9, p. 5–17 , 2017.

[56] H. Huang, "FPGA-Based Parallel Metaheuristic PSO Algorithm and its Application to Global Path Planning for Autonomous Robot Navigation," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3-4, pp. 475-488, 2014.

[57] O. Castillo, R. Martinez-Marroquin, P. Melin, F. Valdez and J. Soria, "Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot," *ELSEVIER Information sciences*, vol. 192, pp. 19-38, 2012.

[58] H. Chung, C. Hou and S. Liu, "Automatic Navigation of a Wheeled Mobile Robot using Particle Swarm Optimization and Fuzzy Control," in *IEEE International Symposium on Industrial Electronics (ISIE)*, Taiwan, 2013.

[59] N. Shiltagh and L. Jalal, "Optimal Path Planning for Intelligent Mobile Robot Navigation using Modified Particle Swarm Optimization," *International Journal of Engineering and Advanced Technology*, vol. 2, no. 4, pp. 260-267, 2013.

[60] Z. Allawi and T. Abdalla, "PSO-Optimized Type-2 Fuzzy Logic Controller for Navigation of Multiple Mobile Robots.," in *IEEE International Conference on Methods and Models in Automation and Robotics(MMAR)*, Poland, 2014.

[61] W. Yong, C. Feng and W. Ying, "Application of particle swarm optimization in path planning of mobile robot," in *AIP Conference Proceedings*, 2017.

[62] K. Bartecki, D. Król and J. Skowroński, "Wyznaczanie kluczowych wskaźników wydajności procesu produkcyjnego - część I: Badania teoretyczne.," *Primary Automation on Robotics*, vol. 22, no. 3, p. 5–13, 2018.

[63] F. Gul, W. Rahiman, S. Alhady and a. al, "Meta-heuristic approach for solving multi-objective path

planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming," in *Journal of Ambient Intelligence and Humanized Computing*, 2021.

[64] "Python".

[65] "Guido van Rossum".

[66] "Python Libraries".

[67] "NumPy".

[68] "SciPy".

[69] "Matplotlib".

[70] "Pillow".

[71] "Copy".

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
CHADLI BENDJEDID EL-TARF UNIVERSITY
SCIENCES AND TECHNOLOGY FACULTY
COMPUTER SCIENCE DEPARTMENT

CERTIFICATE OF PARTICIPATION

Awarded to:

Ines Boutabia

For presenting the paper entitled:

A particle swarm optimization approach for a robot in an environment with obstacles

At the 1st International Conference on Autonomous Systems and their Applications (ICASA'22).

Author(s): Ines Boutabia, Abdelmadjid Benmachiche and Maroua Hammadi.

Dr Abdelmadjid Benmachiche

Conference On Autonomous Systems & Their Applications * First International
ICASA'2022
General Chair

UCBET / **FST** / **INF** - P.O. Box 73, El-Tarf 36000, Algeria - : http://univ-eltarf.dz - : cniati.info@gmail.com

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
ChadliBendjedid El-Tarf University
Sciences and Technology Faculty
Computer Science Department

CERTIFICATE OF PARTICIPATION

Awarded To:

Boutabia Ines

for presenting the paper entitled :

« Recommendation system based on collaborative filtering in the library of El Tarf University »

At the First **N**ational **C**onference on **A**rtificial **I**ntelligence and **I**nformation **T**echnologies (**CNIATI'20**).

Co-Author(s): Benmachiche Abdelmadjid, Maatallah Majda and Makhlouf Amina,

El-Tarf on May 24th, 2021
Conference Chair

جامعة الشاذلي بن جديد - الطارف
كلية العلوم والتكنولوجيا
الملتقى الوطني الأول حول "الذكاء الإصطناعي وتكنولوجيات المعلوماتية" "CNIATI 20"
رئيس الملتقى الدكتور
عبد المجيد بن مشيش

UCBET / **FST**/ **INF**- P.O. Box 73, El-Tarf 36000, Algeria
: http://univ-eltarf.dz
: +213 (0) 38 88 09 87