M. ABDI

ATILIM UNIVERSITY

AN EFFECTIVE PATH PLANNING ALGORITHM FOR SWARM OF ROBOTS IN AN UN-KNOWN ENVIRONMENT

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

MOHAMMED ISAM ISMAEL ABDI

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF MECHATRONICS ENGINEERING

January 2020

AN EFFECTIVE PATH PLANNING ALGORITHM FOR SWARM OF ROBOTS IN AN UN-KNOWN ENVIRONMENT

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

MOHAMMED ISAM ISMAEL ABDI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF MASTER OF SCIENCE

IN

THE DEPARTMENT OF MECHATRONICS ENGINEERING

January 2020

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science Engineering in Mechatronics Atilim University.**

_____

Prof. Dr. Hulusi Bülent ERTAN
Head of Department

This is to certify that we have read the thesis AN EFFECTIVE PATH PLANNING ALGORITHM FOR SWARM OF ROBOTS IN AN UN-KNOWN ENVIRONMENT submitted by Mohammed Isam Ismael, Abdi and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Muhammad Umer KHAN
Supervisor

Examining Committee Members:

Asst. Prof. Dr. Bülent İRFANOĞLU
Mechatronics Eng. Department, Atilim University _____

Asst. Prof. Dr. Muhammad Umer KHAN
Mechatronics Eng. Department, Atilim University _____

Asst. Prof. Dr. Habib GHANBARPOURASL
Mechatronics Eng. Department, THK University _____

Date: *22.01.2020*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name, Last Name: Mohammed Isam Ismael, Abdi

Signature:

ABSTRACT

AN EFFECTIVE PATH PLANNING ALGORITHM FOR SWARM OF ROBOTS IN AN UN-KNOWN ENVIRONMENT

Abdi, Mohammed Isam Ismael
M.S., Department of Mechatronics Engineering
Supervisor:Asst.Prof. Dr. M. Umer Khan
January 2020, 54 pages

In many circumstances, several mobile robots —independent agents— team up to achieve goals that are hard or impossible for an individual robot. These mobile robots cooperate to perform any particular task, complexity of this cooperation is correlated with the swarm size. Each individual robot is to interact with the local environment using sensors. The primary concern for the swarm is to define and follow a safe route from initial to target location. To achieve this task, many algorithms exist in the literature, namely, Neural Network, Genetic Algorithms, Bacterial Foraging Optimization, Ant Colony Optimization, Artificial Potential Field etc. Among these, Bacterial Foraging Optimization (BFO) algorithm has attracted the attention of many scientists due to its effectiveness of finding the destination with the consideration of all known obstacles in the work space ensuring safety. Furthermore, it discovers and always follows the determined path correctly. BFO is a straightforward but strong bioinspired method of optimization using the analogy of swarming principles and social behavior in nature.

The BFO successfully searches for an optimal path from start to goal point in the presence of obstacles over a flat surface map. However, the algorithm suffers from getting stuck in local minima whenever non-convex obstacles are encountered. In case of any of these robots from the swarm getting stuck is considered as the failure of the whole task.

This research proposes an improved version of BFO algorithm that can effectively avoid obstacles, both of convex and non-convex nature. The proposed algorithm helps the robot to recover from local minima by covering a certain distance in an opposite direction to the obstacle. The algorithm will start generating virtual obstacles to generate a safe path when facing acute angle. This information is then passed onto other robots, so that they can also avoid local minima.

To test the effectiveness of the proposed algorithm, a comparison is made against the existing BFO algorithm. The performance of both algorithms is tested in an unknown dynamic and static environments. Through results, it is witnessed that the proposed approach successfully recovers from the local minima, whereas, BFO gets stuck.

Keywords: Swarm Robots, Path Planning, Bacterial Foraging Optimization, BFO, Local Minima, Information Sharing, Dynamic Environment, and Static Environment

ÖZ

BİLİNMEYEN ORTAMLARDA ROBOT SÜRÜLERİ İÇİN ALGORİTMA PLANLAMADA ETKİN BİR YOL

Abdi, Mohammed Isam Ismael
Yüksek Lisans, Mekatronik Mühendisliği Bölümü
Danışman: Dr. M. Umer Khan

Ocak 2020, 54 Sayfa

Birçok durumda birkaç mobil robot —bağımsız ajan— tek bir robot için gerçekleştirilmesi zor veya imkânsız hedefleri elde etmek amacıyla ekip halinde bir araya gelebilirler. Bu mobil robotlar belli bir görevi yerine getirmek için iş birliği yapabilirler, bu, sürünün büyüklüğüyle tam bir karşılıklı ilişki halindedir. Tek tek her robot sensörlerini kullanarak yerel ortamla karşılıklı olarak etkileşir. Sürü açısından birincil endişe başlangıçtan hedef yere kadar güvenli bir yolun tanımlanması ve izlenmesidir. Literatürde bu hedefin gerçekleştirilmesiyle ilgili Neural Network (Sinir Ağları), Genetic Algorithms (Genetik Algoritmalar), Bacterial Foraging Optimization (Bakteriyel Besin Arama Optimizasyonu), Ant Colony Optimization (Karınca Kolonisi Optimizasyonu), Artificial Potential Field (Yapay Potansiyel Alan), v.b. gibi pek çok algoritma mevcuttur. Bunlar arasında Bacterial Foraging Optimization (BFO) algoritması çalışma ortamında bilinen tüm engelleri göz önüne alarak güvenliği ve hedefin bulunmasını sağlamaktaki etkinliği nedeniyle pek çok bilimcinin dikkatini çekmektedir. Ayrıca, belirlenen yolu keşfeder ve doğru olarak izler. BFO kümeleşme prensiplerini ve doğadaki sosyal davranışlar analojisini kullanan, ilhamını biyolojiden alan doğrudan yaklaşımlı ama güçlü bir optimizasyon yöntemidir.

BFO yassı bir yüzey haritası üzerinde engellerin varlığında başlangıçtan hedef noktaya kadar optimal yolu başarıyla araştırır. Ancak bu algoritma, konveks olmayan

engellerin işe karışması durumunda yerel asgari şartlara sıkışmak gibi bir zayıflığa sahiptir. Sürünün robotlarından herhangi birinin sıkışıp kalması durumu görevinin tamamının başarısızlığı olarak görülmektedir.

Bu araştırma BFO algoritmasının hem konveks olan hem de olmayan niteliklerdeki engellerden başarıyla kaçınılmasını sağlayan iyileştirilmiş bir versiyonunu önermektedir. Önerilen algoritma engele zıt yöndeki belli bir mesafeyi kapsayarak robotun yerel asgari değerlerden kurtulmasına yardım eder. Sert bir açıyla karşılaşıldığında algoritma güvenli bir yol oluşturmak için görsel engeller oluşturmaya başlar. Daha sonra bu bilgi diğer robotlara aktarılarak onların da yerel minimumlardan kaçınmaları sağlanır.

Önerilen algoritmanın etkinliğinin test edilmesi için mevcut BFO algoritmasıyla bir karşılaştırma yapılmıştır. Her iki algoritmanın performansı bilinmeyen dinamik ve statik ortamlarda test edilmiştir. Sonuçlara göre, önerilen algoritmanın yerel minimumlardan başarıyla kurtulduğu ve BFO'nun sıkışıp kaldığı gözlenmiştir.

Anahtar kelimeler: Sürü Robotları, Yol Planlama, Bakteriyel Besin Arama Optimizasyonu, BFO (Bacterial Foraging Optimization), Yerel Minimumlar, Enformasyon Paylaşımı, Dinamik Ortam ve Statik Ortam.

To my parents and friends

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Muhammed Umer KHAN for his continuous support, patience, guidance, and motivation throughout my studies.

I would like to thank my dear father, mother, brother and close relatives. I wish that, may Allah bless them all and reward them with the best of this world and hereafter. I really cherish the moments spent with my close friend, Ali Fadhil. He has always been a source of inspiration for me. I would also like to thank Muhammed T.Beider for the wonderful friendship that brought us 12 years ago. I really appreciate his immense support throughout this period.

Last, but not the least, I would like to thank all the friends who walked with me in this endeavor, their company made me feel like home, far from home. I would like to apologize if someone's name is skipped from the list:

Faisal Khalid, Mustafa Al-Halawachi, Meqdam Gassiy, Waqid Khudayer, Furat Turkan, Emrah Makina, Anas Shaban, Ahmed Majad, Othman Mahmoud, Nawaf Hani, Basim Ahmed, Mohammed Shakir, Ahmed Aljanabi, Ayman Tariq, Oguzhan Kose, Ahmed Mohsen, Omar Mohammed, Ahmed Al-Taie, Othman Al-hayali, Yousif Laith, Shuayip Zeydan, TahaYaseen, Hasan al-jawad, Mustafa Yaseen, Hasan al-jalal, Zaid Algargary, Mahir Al-Mashharwi, Hussein Zuhair, Mohammed Moayed, Ahmed Idris, and Feyzullah Yavuz. It must be rude not to mention the names of the girls who supported me in this journey: Gizem Magemizoglu, Sanarya Dizayie, Zainab Al-Attar, Gozde Yilmaz, Marwa al-bayati, and Melike Temizyurek.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

1.1. Motivation

The development in the field of robotics has a long history that dates to 1960s, when first industrial robot was developed. Many critics, who are against the advancement in the field of robotics are of the view that our world will be dominated by the robots and humans will be working in the tungsten mines under their supervision. In actual, robots have been extensively utilized to make humans' job easier, such as space craft dock at the International Space Station, vacuum the kitchen floor, mowing the loan, cleaning windows, and pools. These are all the promising effects on our lives made by robots. All the mobile robots are supposed to maneuver in its surroundings in the presence of obstacles. Therefore, path planning is considered the most critical task that defines the success or failure of the whole process.

Before embarking upon the history of robotics, the term 'robots' need to be defined first. In technical terms, a robot is defined as a machine capable to perform a certain task. In the context of this definition, the robot can achieve the goal using its decision-making capability as possessed by humans. For instance, if a human has to pick a coin, it can be done in three basic steps. First, recognition of the coin must be made, the brain then uses this information to decide whether to pick up or not. Once the decision is made, action command is generated for the limbs to perform pick up.

Almost in all cases, robots follow certain set of commands to perform some task. They are equipped with input sensors, decision-making control system, and end-effectors. These individual tasks can be quite challenging due to their complexity. The sensors must be able to detect images or sound accurately, the end-effectors need to be flexible and fast enough, and the control system ensures the coordination between sensors and end-effector to achieve the overall task. It can be said that

swarm robotics are useful due to parallelism, robustness, and flexibility. It is desired that robots should be capable enough to avoid complex obstacles both in on-line and off-line fashion.

1.2. Problem statement

In some cases, robots are assumed to perform more complex task than just wandering. It is assumed that in the presence of obstacles, the robots will have some decision-making capability to avoid them. So, every robot should be able to perform obstacle avoidance because goal-to-goal movement and obstacle avoidance are really the dynamic duo of mobile robotics.

The obstacles can go un-noticed if they are not obstructing the robots' path to the goal. Some strategy should be adopted to bypass the obstacles when encountered. The most common approach can be to control the heading angle of the mobile robot in order to steer away. There are many algorithms that help the robots to perform such actions, some are preferable over others based upon their performance. The main problem with bacterial foraging optimization (BFO) algorithm is its limitation to deal only with the circular obstacles, whereas, it fails to avoid non-circular obstacles.

1.3. Contribution

- This thesis proposes an improved version of BFO algorithm that can effectively avoid local minima.
- To avoid local minima, virtual obstacles are introduced that help the robot to retrieve from that location.
- The information about virtual obstacles is shared among the whole swarm that will be used by the swarm to prevent same local minima.

1.4. Organization of the Thesis

The thesis is divided into following chapters:

Chapter 1 introduces the mobile robot and discusses the motivation behind current research work, defines the problem statement and contributions of this research.

Chapter 2 presents the literature review of various approaches used in the field of mobile robot navigation.

Chapter 3 provides an overview about the artificial potential field, bacterial foraging optimization algorithm, and kinematic model of wheeled mobile robot.

Chapter 4 proposes the improved version of BFO and discusses the effectiveness and working of the proposed algorithm.

Chapter 5 presents two scenarios to test the effectiveness of the proposed approach.

Chapter 6 concludes the thesis and defines the potential research topics that can be explored based upon this work.

CHAPTER 2

2. LITERATURE REVIEW

2.1. Swarm Robotics

When the environment is filled with multiple mobile robots, it creates a swarm of robots. A required collective behavior is intended to emerge from the robots' interactions with the surroundings. This strategy is adapted in the field of artificial swarm intelligence as well as in the biological research of insects, ants, and other areas where swarm behavior takes place. Swarm robotics' research is to explore robots' design, physical body, and control behaviors. A key component is the communication between group members to build up a system of continual feedback. The swarm behavior includes continual individual changes in collaboration with others as well as the behavior of the entire group.

The general area of motion planning for group of robots has been explored much in the recent years. The pioneer work by Reynolds [1] dealt with the motion control of a number of agents for generating realistic computer animation of flock of birds (called boids). Essentially, local connections among adjoining agents create an evolving behavior for the whole flock. In robotics, such interactions form a special case of the potential field approach [2] where robots are attracted towards the goal while experiencing some resistance by obstacles and other robots. In swarm, attractive forces are generally modeled through the gradient descent of specific functions [3], [4]. Unfortunately, in regular potential field approaches, the presence of obstacles and local repulsion forces among the robots may cause convergence problems due to general gradient descent, as robots are required to synthesize shapes.

The work presented in [5], for example, models a group with a deformable shape and uses a probabilistic roadmap to plan for this shape. Belta et al.[6] shows how group of robots can be modeled as deformable ellipses and proposed the use of decentralized controllers to control the shape and position of the ellipses. This approach was extended in [7] with the development of a hierarchical framework for

trajectory planning and control of swarms. Similar approach was also used by [8] in which planning was simplified by dynamically grouping robots using a sphere tree structure. A related work that investigates coordination mechanisms for boundary coverage with swarm is presented in [9].

2.2. Robot's Trajectory

2.2.1. Path Planning Algorithms

In the field of path planning, some researchers focused on the development of intelligent mobility strategies that enable mobile robots to explore both static and dynamic environments, independently [10]. The path planning of the mobile robot is one of the interesting and most challenging issue. The goal is to approach the target from the start point in an optimal manner in a given environment. There can exist multiple paths that lead to the target. The selection of an optimal path is usually dependent upon the satisfaction of some constraints or rules, such as shortest distance, least energy consumption, and minimum time. Among these, the cost function based upon displacement is considered the most important due to its effectiveness in almost all situations [11]. Therefore, the overall optimization problem can be posed as the determination of the path that ensures the shortest distance from start to end point.

On the other hand, a robot can generate a cost map associated with an environment. The cost map comprises of a set of pixels, each corresponding to a location in the environment. Furthermore, the robot can generate a plurality of masks having projected path portions. Each mask represents a plurality of mask pixels that correspond to locations in the environment. The robot can then determine a mask cost associated with each mask. Based on the projected path portions within the selected mask, the robot can navigate in a space [17].

Rapidly-exploring Random Trees, Level set, and Linguistic Geometry lies under probabilistic roadmaps, whereas, Neural Networks, Genetic Algorithms (GAs) [12], [13], [14], Simulated Annealing [15], Ant Colony Optimization [16], Particle Swarm Optimization, Wavelets and Fuzzy Logic belongs to the wide class of heuristic and meta-heuristic. These methods have their own advantages and disadvantages.

2.2.2. Artificial Potential Field

Mobile robots can be engaged in various industrial sectors, a key area for human advancements in terms of science and technology. The background of robot movement planning can be traced back to the mid-1960s [18]. In old days, mathematical programming, cell decomposition, roadmap and artificial potential field (APF) were considered as a handy tool to perform motion planning. APF is still considered as one of the most common and reliable planning techniques [19], however, it provides only one trajectory that may not be optimal necessarily.

APF received much attention after the work of Khatib et al., who worked upon obstacle avoidance in an effective manner [2]. It has also been shown in [20] that potential fields can be utilized for controlling robot swarm formation, obstacles prevention, and swarm motion. Artificial field generated around the mobile robot comprises of two main forces: attractive potential field force that attracts the robot to the goal point, and the other one is known as repulsive potential field force that repels the robot away from the obstacle's field. The total potential field of APF is generated when both forces are combined (attractive + repulsive) [21].

2.2.3. Deterministic and Stochastic Classification

In the last decade, many optimization algorithms were proposed that grabbed the researchers' attention. A wider class of optimization methods is broadly classified based upon their nature as deterministic and stochastic [22]. Deterministic techniques are sometimes considered as in-efficient in solving complex and discrete problems. Stochastic techniques, on the other hand, do not depend on the mathematical properties of a given function and are considered more appropriate for finding optimal solutions for any type of objective function. They are also regarded as user friendly from implementation point of view. With the increase in complexity and magnitude of the problems, use of stochastic methods become inevitable. These techniques have performed better than the classical or gradient-based methods, especially in optimizing the non-differential and discrete complex functions.

Since 1980, some efficient stochastics methods that mimic certain animals' or insects' behavior (e.g., birds, ants, bees, flies and even germs) are proposed and regarded as nature-inspired algorithms. Presently, these paradigms—inspired by

nature—have already been commonly used in many areas [23]. The performance of these algorithms can get affected from the increase in search space that can even lead to failure. Due to the non-linear nature and computationally extensive requirements for path planning, the need of effective path planning techniques has emerged as a particular concern. Recently, the class of evolutionary algorithms and swarm intelligence is gaining popularity. The genetic algorithm and bacterial foraging optimization are preferably used in trajectory planning for mobile robots when the environment description is available.

2.2.4. Bacterial Foraging Optimization

From a wide class of path planning algorithms, bacterial foraging optimization (BFO) is relatively new and has a lot of potential [24] as it can find an optimal path quite effectively. Using this path, a robot can prevent all obstacles in the work space, hence ensure maximum safety. BFO is a simple yet effective bio-inspired method of optimization that has solved many of engineering and mobile robotics issues including problems of path planning [25], however, the solution is not always guaranteed. Ideally, BFO can encounter only single local minima, located at the desired configuration. In practice, it can run into situations where the attractive and repulsive forces can produce local minima at unknown locations. The robot cannot estimate local minima in an un-observed environment. In other words, the simple control strategy based upon gradient may or may not converge to the goal. The local minimas correspond to local valleys where the robot can get stuck. Avoidance of a local minima has been an active topic of research in the path planning algorithms. So, many solutions exist to overcome the local minima problem and few of them are discussed in the following section.

2.3. Background of Local Minima

Due to its simplicity, practicality and, moreover, the implicit capacity to collect worldwide data, the wall following technique has been commonly used. This is a primary approach when robot has to perform the navigation. Bug algorithms' family [26], [27] is well known to guarantee global convergence in unknown 2d environments with its proven terminating conditions. Therefore, these algorithms enable the robot to find a collision free path from the start towards the goal as long as

there exists one, otherwise it will terminate. Since Lumelsky and Stepanov proposed this concept [26], various efforts, e.g., DistBug [28], Tangent Bug, have been proposed to seek a smooth pathway from the initial to the target location, thereby preventing encounters with obstacles. A similar approach spreads set of points (cellular robots) over the whole path and in order to establish an ideal collision-free path following the local rules [29].

In recent years, another approach that has gained popularity due to its reliability and ability to avoid local minima is artificial potential field [30]. Artificial potential field (APF) often prevents local minima when spherically symmetric function is used as an attractive potential. The work done by [31] effectively avoids the obstacles using potential field approach. The standard APF relies upon the gap between the robot and the obstacles around it. It generates a parameter to evaluate the significance of every obstacle on the available path of the robot. The parameter is defined based upon the location of the obstacle; angle, and length of the robot [32]. Using harmonic methods in a cluttered area greatly reduces the local minima. The panel method is used to define random obstacles and also to obtain the ability across the entire region. Kim suggested an elegant control strategy for real-time robot control with a potential function [33]. An enhanced artificial potential field (E-APF) generates mobile robot trajectory while assuring the continuity without any disruption. The proposed approach–implemented on wheeled mobile robot–successfully avoids local minima when compared to the classical APF [34]. The repulsive potential is used for discretizing outline of an arbitrarily shaped obstacle with their boundary points.

Another way to get rid of the local minima is to introduce virtual obstacle near the robot. The virtual obstacle is located alongside the robot to push the robot away from the local minimum [35]. The proposed work in this thesis is actually inspired by this work. The existing method has few discrepancies that are listed below:

1. The robot doesn’t go back on the same path where it came from. Therefore, the robot may stuck in new local minima.
2. This strategy doesn’t work with wide-ranging area.
3. There is no sharing information between robots.

This research proposes an improvement to the existing strategy of bacterial foraging algorithm to get rid of local minima and improve the communication among swarm.

If any robot got stuck in local minima, then this strategy will force that robot to go back on the same path where it came from by covering a specific distance. Afterwards, it generates a virtual obstacle at local minima to prevent other robots from falling in it. This strategy builds a forbidden area after generating several virtual obstacles which are close to each other.

CHAPTER 3

3. MATHEMATICAL BACKGROUND

The path is to be identified using path planning algorithm, and this path provides the guideline (x, y) to move in the environment. There exist many algorithms in the literature like artificial potential field, artificial neural network, fuzzy logic control, genetic algorithms, bacterial foraging optimization that can help the robot to reach the target and avoid obstacles.

From the above stated optimization algorithms, Bacterial Foraging Optimization Algorithm (BFOA) is employed in this research to discover the optimal path for mobile robots from its initial to desired position. The idea behind the selection of BFO is due to its robustness against the size or non-linearity of the problem. This chapter is divided into three sections that discusses background information related to the proposed work. In the first section, a brief review about the artificial potential field and its related mathematical equations are provided. The second section highlights the general information about bacterium‘s behavior in the environment and its simple foraging strategies. Furthermore, information about bacterial foraging optimization algorithm is also provided. The third section discusses the kinematic model of differential drive mobile robot. The understanding of the kinematic model helps to effectively implement the proposed approach.

3.1. Artificial Potential Field Method

Path planning is vital for the mobile robot’s movement, and is explored extensively by many researchers. Artificial potential field has recognized itself as one of the most desirable approach due to its ease of understanding. The robot is identified as an object that is driven by the field of forces. An attractive force generated by the target will attract the robot and it will be repelled by a repulsive force generated by the obstacles. The navigation control of a mobile robot in a clouded environment is one

of the hot topics in the robotics community. Artificial potential field algorithm can effectively work for local and global path planning. This depends on the fact that whether the environment is fully observable or not. The environmental data is used for regional path planning through grid cells, maps, etc. In an un-observable environment, the robot must learn about it by means of the sensors.

Practically, the artificial potential field approach ensures easy and effective motion planning. The potential field method, one of the most preferable local path planning method, concurrently deals with the encountered obstacles. The main concept of this method is to view the robot plunged into an attractive potential provided by the target configuration and the repulsive potential produced by the obstacles. By following the negative gradient of the potential function, the path is generated online. When the robot is immersed in the potential field, attractive and repulsive forces guide the robot towards the target. This two-force combination is devoted to the control of robot's movement in a safer environment while avoiding obstacles. The attractive and repulsive functions are explained in detail below:

3.1.1. Attractive potential function

In this section, the attractive potential field is generated for driving the robot to the target location. The general idea is to increase the attractive $U_{att}(q)$ field as the robot goes away from the target. Among many mathematical functions that can serve this purpose, conic and quadratic forms are widely used. The quadratic potential (3.1) is used for this study because the conical forms can cause the problem of discontinuity in the target location.

$$U_{att}(q) = \frac{1}{2} K_{att} \left\| q_{goal} - q \right\|^2 \tag{3.1}$$

where, K_{att} is a positive parameter, $q_{goal} = (x_{goal}, y_{goal})^t$ is the position of the target, $q = (x, y)^t$ is the position of the robot and $\left\| q_{goal} - q \right\|$ is the euclidean distance between the robot and the target. Therefore, the force created by the goal is termed as the negative gradient of the attractive potential function while taking into consideration the position of the robot.

$$F_{att}(q) = -\nabla U_{att}(q) = -\frac{\partial U_{att}(q)}{\partial q} \tag{3.2}$$

$$F_{att}(q) = -K_{att}(q - q_{goal}) \tag{3.3}$$

$F_{att}(q)$ is a vector pointed toward q_{goal} with magnitude linearly related to the distance from q to q_{goal}. Its components can be written as:

$$F_{att_x}(q) = -K_{att}(x - x_{goal}) \tag{3.4}$$

$$F_{att_y}(q) = -K_{att}(y - y_{goal}) \tag{3.5}$$

Where, F_{att_x} is the attractive force along the $x-$axis and F_{att_y} is the attractive force along the $y-axis$.

3.1.2. Repulsive potential function

The repulsive field is generated to repel the robot from the annoying obstacles. However, if robot stays outside the safe region defined around the obstacles, robot's motion remains un-affected. The repulsive function has the following relation:

$$U_{repi}(q) = \frac{1}{2}K_{rep}(\frac{1}{d_i(q)} - \frac{1}{d_o})^2 \qquad if\ d_i(q) \le d_o \tag{3.6}$$

$$U_{repi}(q) = 0 \qquad if\ d_i(q) > d_0 \tag{3.7}$$

where

$U_{repi}(q)$ is the repulsive potential field of each individual obstacle;

K_{rep} is a positive parameter;

d_o is the influence distance of the obstacle; and

$d_i(q)$ is the distance from the robot to the closet point in each individual obstacle. The repulsive force is the negative gradient of repulsive potential function:

$$F_{repi}(q) = -\nabla U_{repi}(q) \tag{3.8}$$

$$F_{repi}(q) = K_{rep}\left(\frac{1}{d_i(q)} - \frac{1}{d_o}\right)\left(\frac{1}{d_i^2(q)}\right)\nabla d_i(q) \qquad if d_i(q) \le d_o \tag{3.9}$$

$$F_{repi}(q) = 0 \qquad if d_i(q) > d_o \tag{3.10}$$

or

$$\nabla d_i(q) = \frac{q-q_c}{\|q-qc\|} \tag{3.11}$$

where, $q_c = (x_c, y_c)^t$ is the position of the closest point in the obstacle. Thus, the repulsive potential force is:

$$F_{repi}(q) = K_{rep}\left(\frac{1}{d_i(q)} - \frac{1}{d_o}\right)\left(\frac{1}{{d_i}^2(q)}\right)\frac{q-q_c}{\|q-q_c\|} \quad if d_i(q) \le d_o \tag{3.12}$$

$$F_{repi}(q) = 0 \quad if\ d_i(q) > d_o \tag{3.13}$$

The Cartesian components $F_{repi-x}(q)$ and $F_{repi-y}(q)$ of the repulsive force can be written as:

$$F_{repi-x}(q) = K_{rep}\left(\frac{1}{d_i(q)} - \frac{1}{d_o}\right)\left(\frac{1}{{d_i}^2(q)}\right)\frac{x-x_c}{\|x-x_c\|}\ if d_i(q) \le d_o \tag{3.14}$$

$$F_{repi-x}(q) = 0 \quad if d_i(q) > d_o \tag{3.15}$$

$$F_{repi-y}(q) = K_{rep}\left(\frac{1}{d_i(q)} - \frac{1}{d_o}\right)\left(\frac{1}{{d_i}^2(q)}\right)\frac{y-y_c}{\|y-y_c\|} if d_i(q) \le d_o \tag{3.16}$$

$$F_{repi-x}(q) = 0 \quad if\ d_i(q) > d_o \tag{3.17}$$

Note that, if the environment contains *n* number of obstacles, then the total repulsive potential field is:

$$U_{rep}(q) = \sum_{i=1}^{n} U_{repi}(q) \tag{3.18}$$

3.1.3. Total potential function

After calculating the attractive and the repulsive potential forces individually, the total potential function is obtained by summing both forces:

$$U(q) = U_{att}(q) + \sum_{i=1}^{n} U_{repi}(q) \tag{3.19}$$

Thus, the resultant potential force is:

$$F(q) = -\nabla U(q) \tag{3.20}$$

$$F(q) = F_{att}(q) + \sum_{i=1}^{n} F_{repi}(q) \tag{3.21}$$

3.1.4. Gradient Descent Potential Guided Planning

Attractive potential function and repulsive potential function are additive and can be defined separately. The total potential is used to drive the robot towards the goal while avoiding the obstacles. Thus, the trajectory of the mobile robot is achieved by following the total force generated by the summation of attractive and potential forces. This total force is calculated as the negative gradient of the total potential function using gradient descent algorithm. It relies upon calculating the position of the robot at every step taken in the direction of the total force $F(q)$. Before calculating the total force, the attractive force, Euclidean distance between the current position of the robot, the closest obstacle ($d_i = \text{EuclideanDistance}(q, q_c)$), and the repulsive force must be calculated.

3.2. Information About Bacteria and Bacterial Foraging Optimization Algorithm

The brain is the most complicated organ of the human body that collects and analyses data; control and manages most of the organs of the body. The human brain is difficult to understand and it is very hard to build a machine of this complexity. In order to evaluate and understand its behavior, researchers turned their efforts to organisms and micro-organisms like bacteria. Once done with the behavior, the same knowledge is applied to the field of robotics. Bacteria is a very delicate organism consisting of only one cell, usually several micrometers in length. It has simple foraging strategies for locating, handling, and ingesting food. Therefore, the biology and physics underlying the foraging behavior of E. coli bacteria must be understood first. For this purpose, those who have successful foraging strategies are more likely to enjoy reproductive success, whereas poor foraging strategies are either eliminated or transformed into good ones. A foraging bacterium takes an action to maximize the energy obtained per unit time. Optimization models are also valid for social foraging where groups of animals cooperatively forage.

In bacteria, foraging involves finding such patches, deciding whether to enter a patch and search for food, or whether to continue searching for food in another patch that might have a higher quality and quantity of nutrients than the current patch. Patches

are generally encountered sequentially, and sometimes great effort and risk are needed to travel from one patch to another. Generally, in case of a nutritionally poor patch, the bacterium keeps looking for a better patch elsewhere (based on experience). It will also calculate the risks and efforts of finding another patch, and if the risk is minimum it will move to the new patch. Moreover, if a bacterium has been in a patch for some time, it might begin to deplete its resources, so there should be an optimal time to leave the patch and venture to find a richer one. Although, it might not want to waste resources that are readily available, it still wouldn't want to waste time in the face of diminishing energy.

Optimal foraging theory formulates the foraging problem as an optimization problem and specify as how foraging decisions are made (traditionally, dynamic programming formulations have been used). There are quantifications of how foraging decisions must be made, for instance, researchers have studied how to maximize long-term average rate of energy intake where only certain decisions and constraints are allowed. Constraints due to incomplete information (e.g., due to limited sensing capabilities) and risks (e.g., due to predators) have also been considered. Essentially, these optimization approaches seek to construct an optimal controller (policy) for making foraging decisions. However, the optimal foraging formulation is only meant to be a model that explains what the optimal behavior would be like.

In fact, understanding this type of social foraging behaviors enable the scientists to create adaptive controllers and cooperative control strategies for autonomous vehicles. They derive basic ideas from some interesting biological phenomenon that suggests new types of optimization to better understand the concepts related to control systems.

3.2.1. Overview of Chemotaxis

Chemotaxis is a process where a cell (such as a bacterium) migrates towards chemical concentration. In other words, there is a particular chemical concentration that can be perceived as either a positive or negative source for food. The migration of a cell is due to the chemical gradient and chemical molecule. It is assumed that these chemical molecules, shown in Fig. (3.1), are spread out there at some distance from the bacteria, hence the bacterium will go towards it because the bacterium can

sense the attraction coming from it. The movement of the bacterium is termed as chemotaxis.

There are two chemotaxis possibilities in this scenario: the bacterium would be attracted towards the chemical molecule (right). This kind of chemotaxis is called chemo attraction (positive kind of chemotaxis). Secondly, there is a chemical warning signal, which acts as a repellent from that chemical molecule. Consequently, the bacterium will start to move away from that chemical molecule, and this is called chemo repellent feature.

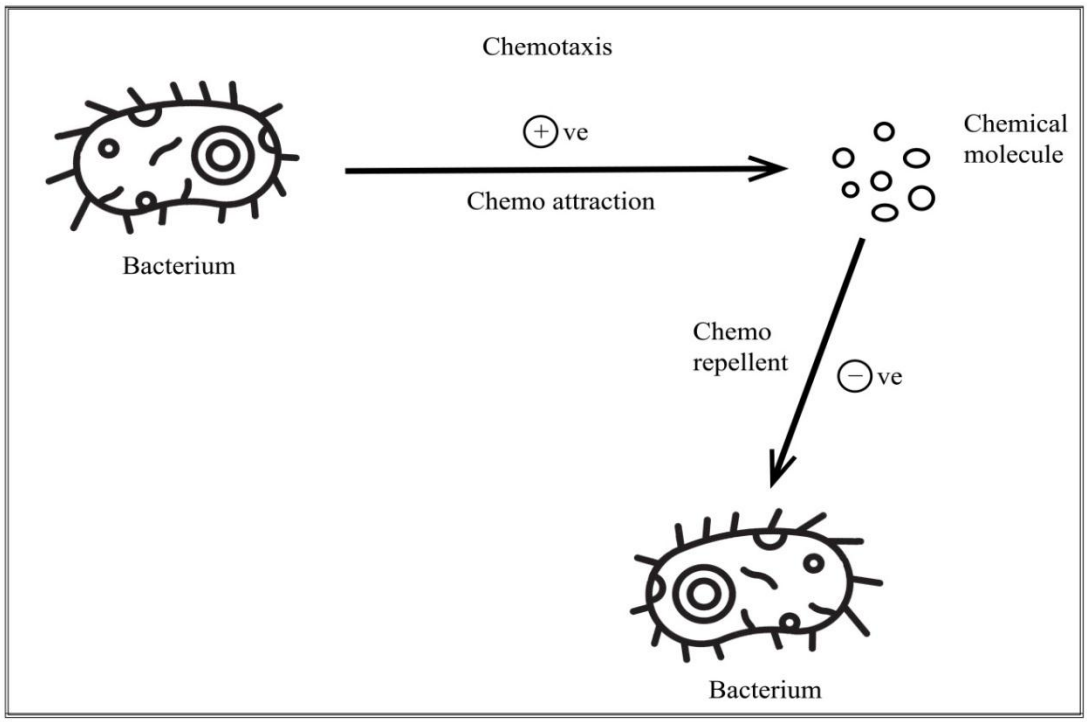

Figure 3.1 Attraction and repellent chemotaxis.

3.2.2. Chemotaxis

The bacterium's swimming and tumbling behavior are simulated during chemotaxis and it is the BFOA's very first stage. So, if the bacterium finds an appropriate area during the foraging, it will travel in the same direction for a while. Otherwise, if the bacterium drops into the noxious area by following a tumble in random directions, it will seek to discover its quest action by exploring the quest for a promising new area.

In chemotaxis step, E-coli movement is observed during swim and tumble as shown in Fig. (3.2). Biologically, in two situations, the motion of the E-coli bacterium is constrained, either it can swim in a certain direction for a period of time as the previous step, or it can tumble throughout its lifespan.

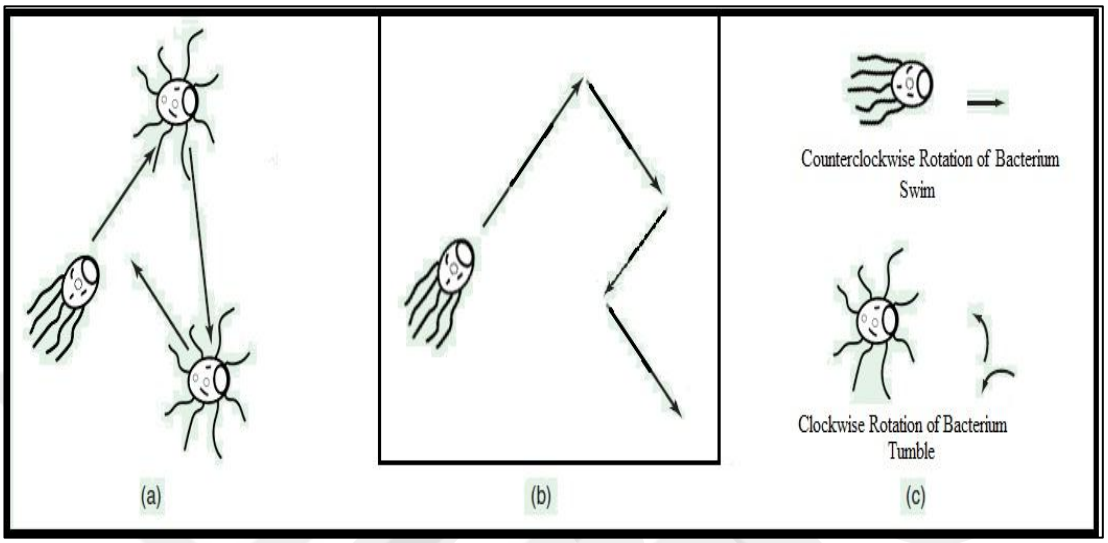

Figure 3.2 Swim and tumble of bacterium.

A unit step in an arbitrary direction is a 'tumble', and a unit step in the same direction means a 'run'. Suppose that at j^{th} chemotactic, k^{th} reproductive, and l^{th} elimination-dispersal stages are combined to make θ^i (j, k, l) that represents the bacterium. $c(i)$ is the step size of the chemotaxis in each tumble or run. It is also possible to design the motion of the i^{th} bacterium as;

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\emptyset(j) \tag{3.22}$$

where, $c(i)$ is the size of the random step, defined by the tumble; $j(i,j,k,l)$ is fitness, which is also the cost of the i^{th} bacterium $\theta^i(j,k,l) \in \mathcal{R}^p$. If the cost $J(i,j,k,l)$ at $\theta^i(j+1,k,l)$ is better than at $\theta^i(j,k,l)$, then another move will be taken in the same direction. Alternatively, the bacteria would tumble spontaneously by taking another step of size $c(i)$ to look for a better nutrient environment. This process contributes to the nutrient-rich areas being collected.

3.2.3. Swarming

In several mobile species of bacteria, such as E.coli and S. typhimurium, an unusual team activity was observed. A system for cell-to-cell contact is developed to simulate

the swarming bacteria's biological behavior. Each bacterium emits chemical attraction during motion to attract other bacteria swarming towards it. By describing the integrated cell-to-cell attraction and repelling effect, BFOA simulates this social behavior. Therefore, it is possible to model the results of combined cell-to-cell attraction and repulsion as;

$$J_{cc}\left(\theta, P(j,k,l)\right) = \sum_{i=1}^{s} J_{cc}^{i}\left(\theta, \theta^{i}(j,k,l)\right)$$

$$= \sum_{i=1}^{i}\left[-d_{attract} exp\left(-w_{attract}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right.$$

$$\left. + \sum_{i=1}^{s}\left[h_{repellant} exp\left(-w_{repellent} exp\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right]\right. \quad (3.23)$$

where, $d_{attractant}$ and $w_{attractant}$ are the depth and width of the attractant released by the cell, respectively. Likewise, $h_{repellant}$ and $w_{repellant}$ are the height and measure of the width of the repellent effect. The values of attractants and repellents coefficients should be chosen properly.

3.2.4. Reproduction

After all chemotactic steps, a reproduction step takes place for bacteria. The bacterium fitness is defined as:

$$J_{fitness}^{i} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \quad (3.24)$$

The population is distributed in ascending order of bacterium fitness for reproduction. Then, the 50 percent least healthy bacteria eventually die in reproduction stage, while the others with good health must split to maintain the stable population.

3.2.5. Elimination- Dispersal

The local environment where bacteria population lives is supposed to change with time (e.g. by nutrient consumption). This occurrence can cause the unexpected death of the bacteria population (e.g. sudden temperature increase) or the distribution of a

bacteria community (e.g. sudden water flow) to a new spot. In BFO, the dispersion event occurs during a variety of cycles of reproduction as well as some bacterias are selected based on the possibility of being destroyed or moved to a different location throughout the area.

The pseudo code for the bacteria foraging optimization algorithm (BFOA) is presented below:

Initialize the parameters viz. p, S,N_c, N_s, N_{re}, N_{ed}, p_{ed}, and

$c(i)(i = 1,2,3, \ldots\ldots, S)$.

1) Elimination-dispersal loop: $l = l + 1$
2) Reproduction loop: $k = k + 1$
3) Chemotaxis loop: $j = j + 1$
   a) For $i = 1, 2, \ldots, S$, take a chemotactic step for bacterium i as follows.
   b) Compute $J(i, j, k, l)$. Let

   $$J(i,j,k,l) = J(i,j,k,l) + J_{cc}\left(\theta^i(j,k,l), P(j,k,l)\right) \quad (3.25)$$

   c) Let $J_{last} = J(i,j,k,l)$ to save this value since we may find a better cost via a run.
   d) Tumble: Generate a random vector $\Delta(i \in \mathcal{R}^p)$ with each element $\Delta_m(i)$, m=1, 2,…, p, random number on [-1, 1].
   e) Move: let

   $$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (3.26)$$

   This results in a step size $c(i)$ in the direction of the tumble for bacterium i.

   f) Compute $J(i, j + 1, k, l)$, and then let

   $$J(i,j+1,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j+1,k,l), P(j+1,k,l)) \quad (3.27)$$

   g) Swim
   i) Let $m = 0$ (counter for swim length).

ii) While $m < N_s$ (if have not climbed down too long)

Let $m = m + 1$

If $J(i, j+1, k, l) < J_{last}$(if doing better), let $J_{last} = J(i, j+1, k, l)$ and let

$$\theta^i(j+1, k, l) = \theta^i(j+1, k, l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (3.28)$$

And use this $\theta^i(j+1, k, l)$ to compute the new $J(i, j+1, k, l)$ as in f.

Else, let $m = N_s$. This is the end of the while statement.

h) Go to next bacterium $(i+1)$ if $i \neq S$ (i.e. go to b) to process the next bacterium).

4) If $j < N_c$, go to step 3. In this case, continue chemotaxis, for the life of the bacteria is not over.

5) Reproduction:

a) For the given k and l and for each i= 1, 2…., S, let

$j^i_{health} = \sum_{j=1}^{N_c+1} J(i, j, k, l)$ be the health of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances).

Sort bacteria and chemotaxis parameters $c(i)$ in order of ascending cost J_{health}(higher cost means lower health).

b) The S_r Bacteria with the highest *J* health values die and the others Sr bacteria with the best values split (and the copies that are made placed at the same location as their parent).

6) If *k* < *Nre*, go to step 2. In this case, reached the number of specified reproduction steps is not reached, so start the next generation in the chemotactic loop.

7) Elimination-dispersal: For i=1, 2…., S, with probability ped, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if you eliminate a bacterium, simply disperse one to a random location on the optimization domain.

8) If *l* < *Ned*, then go to step 1; otherwise end.

According to this algorithm, if the cost function of the next point is smaller (better) than the previous point, a new step in the same direction will be taken. This process will continue for the defined swim length. Then, after

each chemotaxis step, the least healthy bacteria as specified by the cost function are switched by the copies of the fit ones. This process is called reproduction step and it is followed by the elimination-dispersal event. For this event, each bacterium in the group is exposed to elimination-dispersal with a pre-defined probability.

3.3. Kinematic Models of Mobile Robot

3.3.1 Introduction

Sometimes, scope of the problem allows a large range of tools to be implemented: a model of the robot, designs of sensors, coordinate transitions, navigation, state prediction, mathematical problem solving, and others. The aim of this section is to understand the kinematics of the mobile robot in order to achieve favorable results.

A mobile robot can perform various activities, it keeps trace of its position and direction with its supervisor or controller. Throughout the method of steering the mobile robot to a destination, the navigation is very crucial. There is a need for secure and reliable control strategies alongside navigation. The systems of navigation and command must collaborate. Until the mobile robot reaches its destination, the sensors may collect the data required and then just save it for future transfer, or send it to the next stage automatically.

3.3.2 Robot with Differential - Drive Steering

Differential-drive steering shown in Fig. (3.3) is one of the most popular type of steering used for mobile robots. The tyre on each side operates independently. The robot can rotate in a position, drive in a single direction, drive in a circular track, or follow some specified trajectory by combining the two varying speeds. The movement formulas for the robot are now derived from differential tyres speeds. Let R describe the robot's immediate curve radius. The robot width between both tyres is denoted by W. The angle formed by the *y-robot* longitudinal axis w.r.t the y-ground axis is known as ψ, determined in counter clock-wise direction. The instant axis around which the robot moves is the intersection point between the two lines that move through the center of the robot's body.

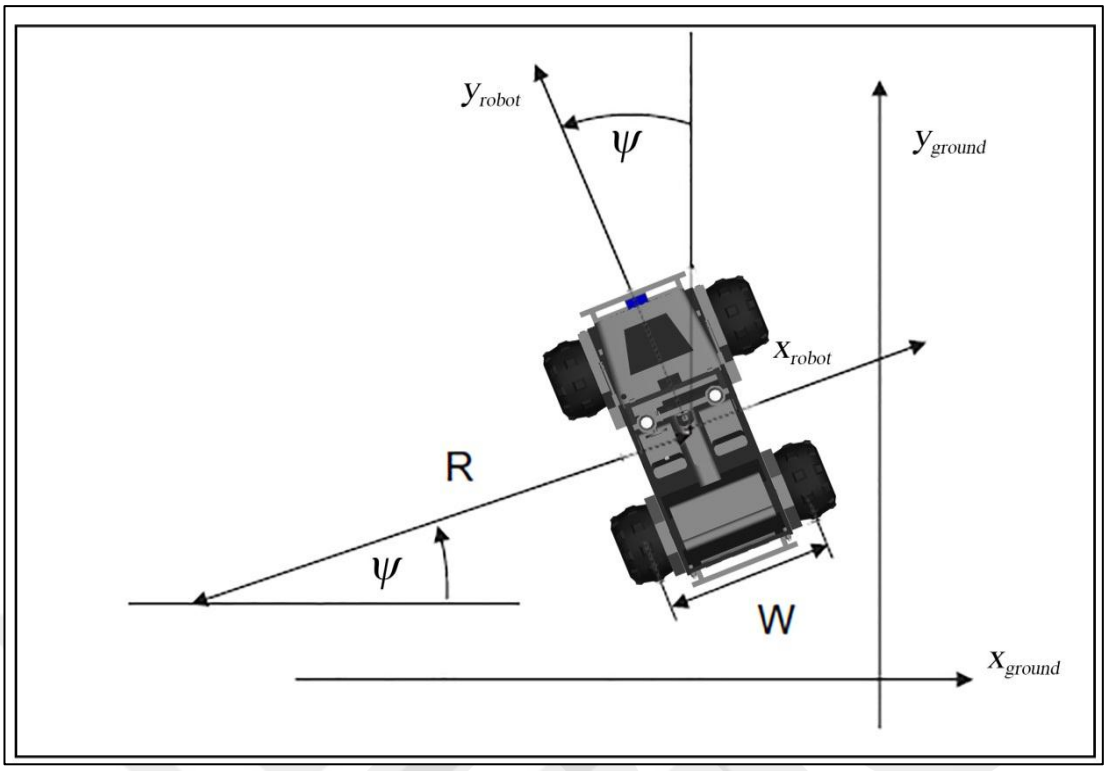

Figure 3.3 Differential Drive Robot.

Now, it should include the following structural factors:

$$v_{left} = \dot{\Psi}\left(\mathrm{R} - \frac{\mathrm{W}}{2}\right) \tag{3.29}$$

and

$$v_{right} = \dot{\Psi}\left(R + \frac{W}{2}\right) \tag{3.30}$$

Now, subtracting the two above equations yields

$$v_{right} - v_{left} = \dot{\Psi}\, W \tag{3.31}$$

So we obtain for the angular rate of the robot

$$\dot{\Psi} = \frac{v_{right}\, v_{left}}{W} \tag{3.32}$$

Solving for the instantaneous radius of curvature, we have:

$$R = \frac{v_{left}}{\dot{\Psi}} + \frac{W}{2} \tag{3.33}$$

or

$$R = \frac{v_{left}}{\frac{v_{right} - v_{left}}{W}} + \frac{W}{2} \tag{3.34}$$

or, finally

$$R = \frac{W}{2} \frac{v_{right} + v_{left}}{v_{right} - v_{left}} \tag{3.35}$$

This results in the expression for velocity along the robot's longitudinal axis:

$$v_y = \dot{\Psi} R = \frac{v_{right} - v_{left}}{W} \frac{W}{2} \frac{v_{right} + v_{left}}{v_{right} - v_{left}} = \frac{v_{right} + v_{left}}{2} \tag{3.36}$$

In summary, the equations of motion in robot coordinates are:

$$v_x = 0 \tag{3.37}$$

$$\mathrm{v_y} = \frac{\mathrm{v_{right}} - \mathrm{v_{left}}}{2} \tag{3.38}$$

and

$$\dot{\Psi} = \frac{v_{right} - v_{left}}{W} \tag{3.39}$$

If we convert to earth coordinates, these become:

$$\dot{x} = -\frac{v_{right} - v_{left}}{2} sin\, \Psi \tag{3.40}$$

$$\dot{\mathrm{y}} = \frac{v_{right} - v_{left}}{2} cos\, \Psi \tag{3.41}$$

and

$$\dot{\Psi} = \frac{v_{right} - v_{left}}{W} \tag{3.42}$$

It may need to consider the fact that speeds cannot vary instantly. It would, therefore, add the speed levels as the regulation parameters:

$$v_{right} = u_1 \tag{3.42}$$

and

$$v_{left} = u_2 \tag{3.43}$$

The kinematic model is now of fifth order. The Euler integration approach could be used to achieve a discrete time model for this nonlinear formula process,

$$x((k+1)T = x(kT) - T\frac{v_{right}(kT) + v_{left}(kT)}{2} sin\Psi(kT) \tag{3.44}$$

$$y((k+1)T = y(kT) - T\frac{v_{right}(kT) + v_{left}(kT)}{2} cos\Psi(kT) \tag{3.45}$$

$$\Psi\big((k+1)T\big) = \Psi(kT) + T\frac{v_{right}(kT) - v_{left}(kT)}{W} \tag{3.46}$$

$$v_{right}\big((k+1)T\big) = v_{right}(kT) + Tu_1(kT) \tag{3.47}$$

and

$$v_{left}\big((k+1)T\big) = v_{left}(kT) + Tu_2(kT) \tag{3.48}$$

There are more advanced and precise approaches for producing discrete-time models; but, if the sampling duration is set relatively short, this Euler formula could be quite effective. Such discrete time scales may be used for process evaluation, controller architecture, evaluator styling and simulation of the process. Pitch, roll and vertical movement might also be included in further complicated systems of mobile robots.

CHAPTER 4

4. IMPLEMENTATION DETAILS

In this research, bacterial foraging optimization algorithm (BFOA) is exploited to seek for an almost efficient solution for mobile robots. The rationale behind the selection of bacterial foraging is its robustness against the area of the work space and non-linearity, and it can handle more numbers of objective functions when compared to other evolutionary algorithms. In several situations, most of the analytical methods struggle at the converging point, whereas, BFOA can provide the most appropriate solution. Robot path planning using bacterial foraging algorithm is implemented by using the equations of BFO sequentially to make the robot move from the starting location to the target while considering the obstacles.

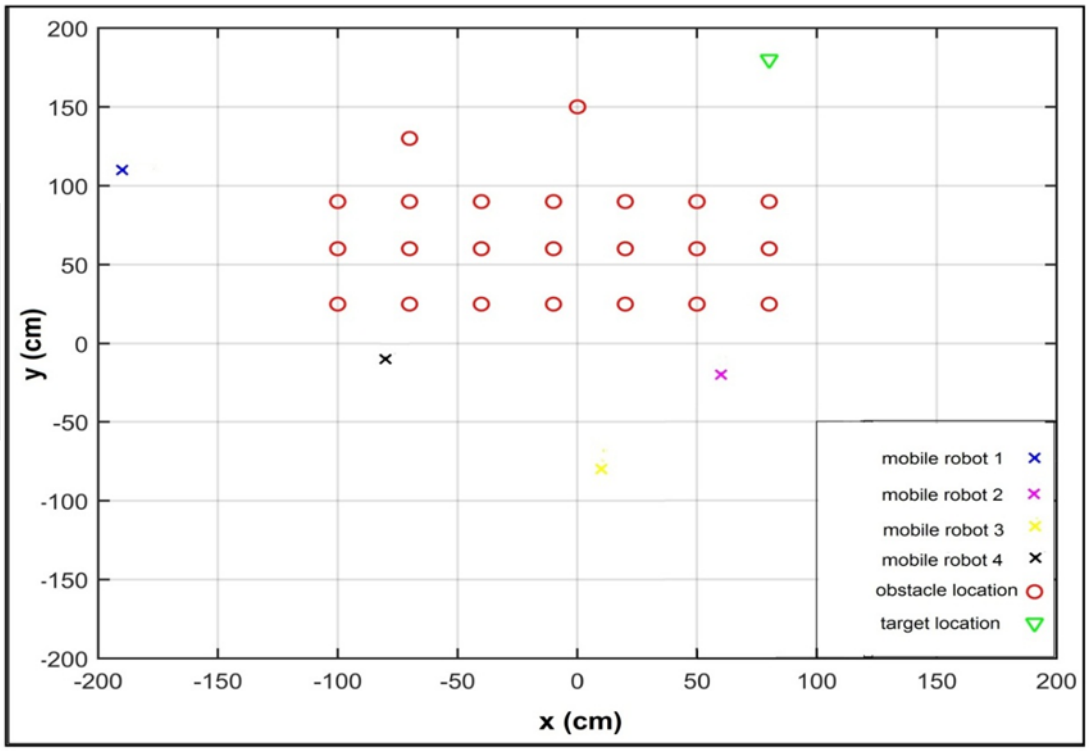

Figure 4.1 Environment of 2-D workspace of robot.

The two-dimensional workspace is depicted in Fig. 4.1, where obstacles, starting points of four robots and target point are specified. Four mobile robots coded with different colors are initiated from different starting points. Each robot's location is represented in the 2D space as (x, y). To obtain the optimized path, shortest possible distance and minimal number of turns are considered. At the same time, the robot should be capable of moving from one position to another by passing the obstacles without any external assistance. In fact, the robot is driven by the virtual forces generated by the target and the obstacles. The resultant forces govern the direction and speed of travel. Therefore, the robot will move through interactive area formed by the attractive and repulsive forces that will cause the robots to change direction based on the strength of these different forces.

The robots can encounter circular and non-circular shaped obstacles. In this work, the regular shaped obstacles like U, L, and circular shaped obstacles are considered. The circular obstacles in the workspace are defined using red color (Fig. (4.1)). In order to avoid the obstacles, a safe region is generated around these obstacles to prevent collision with the robot. The safe zone of radius R is defined around the obstacle according to its size. The location of the goal point is represented using a green triangle with its (x, y) location in the 2D work space.

Navigation enables any mobile robot to achieve the specified target in a static or dynamic environment in a quick and secure manner. Therefore, the robot requires the ability to create an environment map with the help of sensors such as an ultrasonic or proximity sensor. With the use of sensors, the robot gathers environmental data to save it temporarily, then the robot will send this data to the controller for analyzing and decision-making. In global path planning, the robot has all the information of the environment but that information is not known in the local path planning. So, by using sensors, the robot can discover and build its own map.

4.1 The Robot's Trajectory Process

With the bacterial foraging algorithm, the process is initialized by generating 100 particles (bacterium) around the robot with in a circle of radius R. This defines the step size of the movement in the trajectory as:

$$P_S(t + d_t) = P_s(t) + R\left(\frac{\Delta(t)}{\|\Delta(t)\|}\right) \quad (4.1)$$

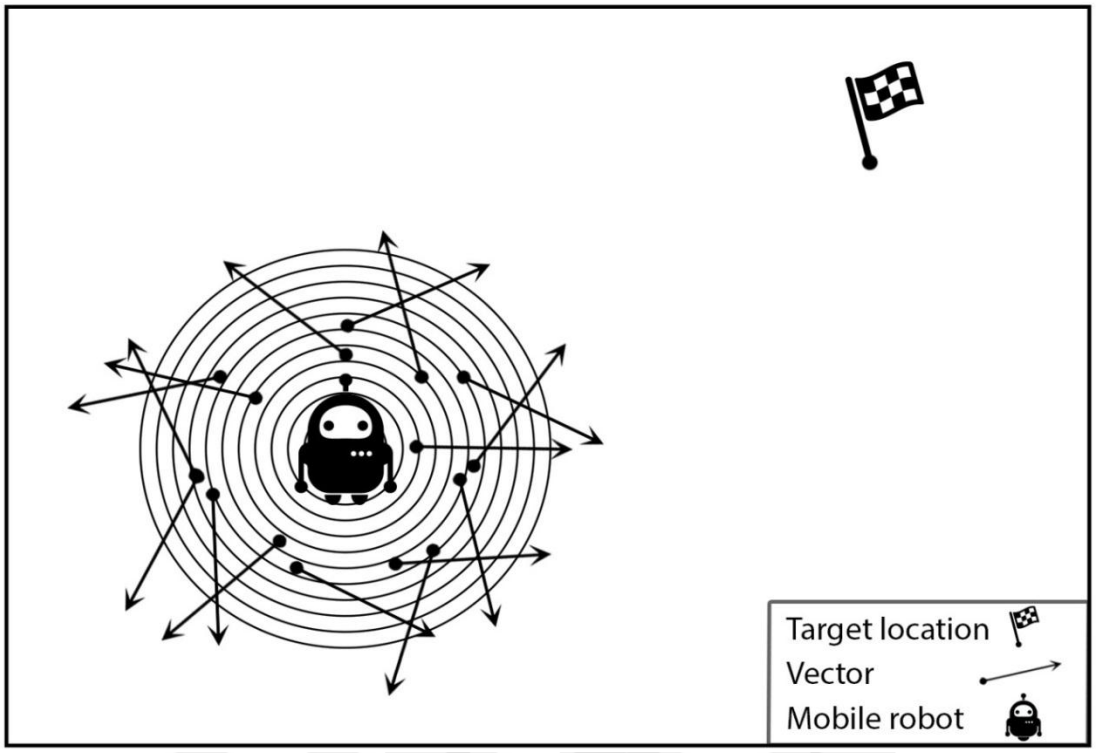

Figure 4.2 Vectors inside robot with start and end point.

Where *Δ(t)* is a random vector in a unit length in order to determine the direction of each of the 100 particles (bacterium), the magnitude of the particle is equal to $\|\Delta(t)\|$. Each one of the bacterium group that are around the robot is looking for the optimal way to the target and it can change direction throughout time. The robot will change its position to one of these 100 bacterium based on the selection of the best one. The process will keep continuing until every obstacle has been avoided and the goal point is achieved, as shown in Fig. (4.2).

The determination of strongest bacterium is based upon two considerations. For each step, the robot should calculate the length between each bacterium, and the length between the current position of the robot to the nearest obstacle. The application of these two variables results in an attractive repellent pattern which guides the robot to the target point which has the global minimum, as shown in Fig. (4.3).

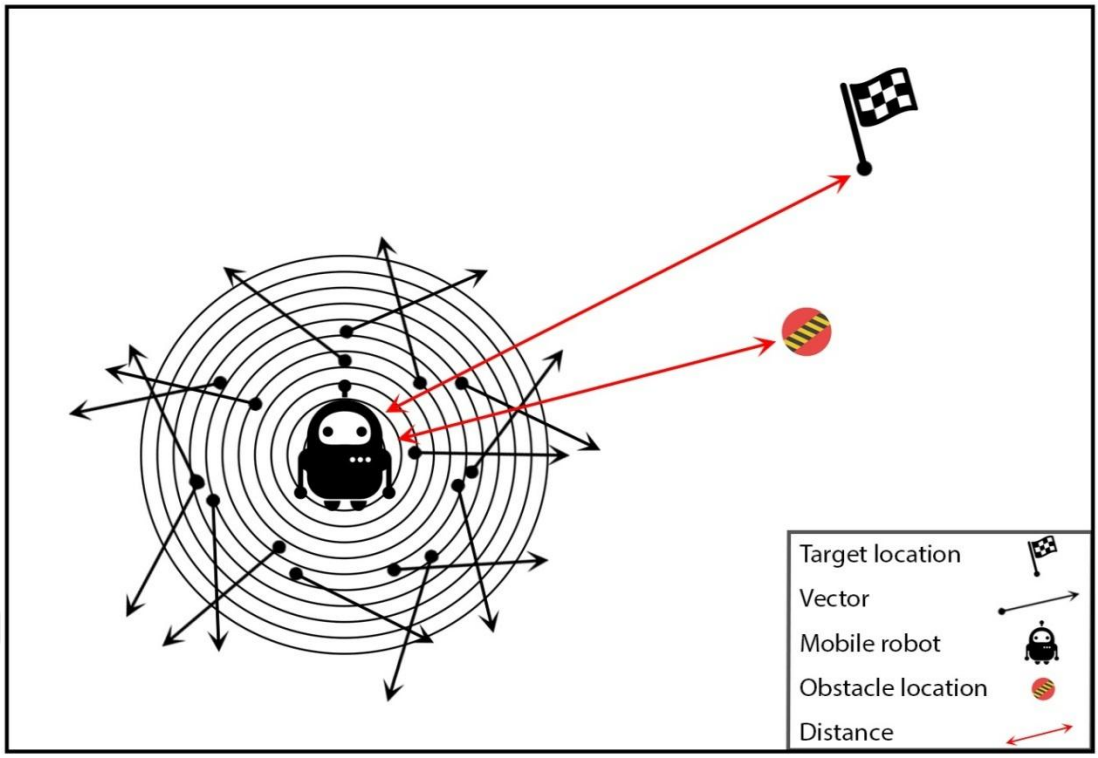

Figure 4.3 Distance between the obstacle location to the robot and the target location to the robot.

During the movement of the robot from one position to another where there is no obstacle, the parameter $J_{obstacle}$ is zero. Otherwise, $J_{obstacle}$ will assume the repulsive Gussian cost function for an obstacle:

$$J_{obstacle} = H_{obstacle} * exp(-W_{obstacle} * (\|\theta_i(t) - P_o(t)\|^2)) \quad (4.2)$$

if $\| P_o(t) - C(t)\|^2 \leq \beta$

$J_{obstacle} = 0$ Otherwise

where, $H_{obstacle}$ and $W_{obstacle}$ are constants that refer to both the repellent's height and width, respectively. With sensor placed at the front of the robot, it can find out any obstacle's location $P_o(t)$. The distance between an obstacle and the robot can be calculated by the Euclidean distance, $\|P_o(t) - C(t)\|^2$, see Fig. (4.3). The value of the $J_{obstacle}$ always remains zero until the robot senses the obstacle in its range, whereas, the range of the sensor is represented by β. The Gaussian cost function of the goal point can be represented as:

$$J_{goal} = -H_{goal} * exp(-W_{goal} * (\|\theta_i(t) - P_o(t)\|^2) \quad (4.3)$$

where, H_{goal} and W_{goal} are constants that refer to both the goal's height and width. The direct distance between the goal and the robot can be calculated by the Euclidean distance $\|\theta_i(t) - P_o(t)\|^2$. Obstacle avoidance and goal search are the two most important concerns during mobile robots' path planning. For the purpose, a combined cost function is defined as:

$$J_{total} = J_{obstacle} + J_{goal} \tag{4.4}$$

Eq. (4.4) is used to find the cost of each bacterium. Based upon their cost values, all particles are sorted from highly desirables to undesirables. In the next iteration, robot will take the position of the highly desirable particle possessing the minimum possible cost. These steps will guide the robot to the target point while preventing the obstacles. The strategy is based on the distance error to the target which can be written as follows:

$$e_s^d = d_s(t + dt) - d_s(t) \tag{4.5}$$

where, $d_s(t)$ is the distance from particle *S* to goal at time *t* which can be computed by $d_s(t) = \|P_s(t) - P_G(t)\|^2$, and again $d_s(t + dt)$ is the distance from the same particle to the goal at time ($t + dt$) which can be calculated by $d_s(t + dt) = \|P_s(t + dt) - P_G(t)\|^2$. Eq. (4.6) is the cost function error to be used for sorting all the the bacterium in an ascending order. The particle with the lowest distance error will be at the top of the list.

$$e_s^J(t) = J\big(P_s(t + dt)\big) - J\big(P_s(t)\big) \tag{4.6}$$

The robot keeps on saving the trajectory in its memory; this information is later used for generating the virtual obstacles and re-routing.

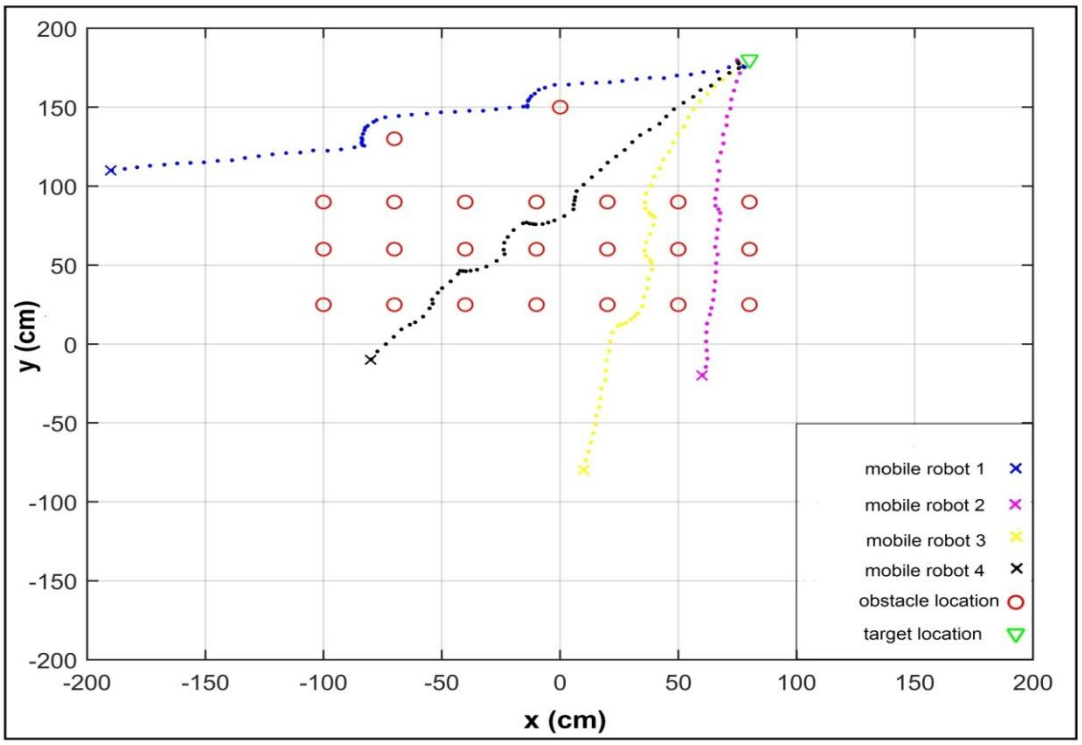

Figure 4.4 Robot movement.

The circular obstacles, shown in Fig (4.4), are scattered over the whole space with enough space left in between for the robot to make safe movements. Therefore, it is possible for the robot to get around the circular obstacle and then proceed towards the target, as shown in Fig. (4.5). The robot's behavior against different type of obstacles is yet to be tested.

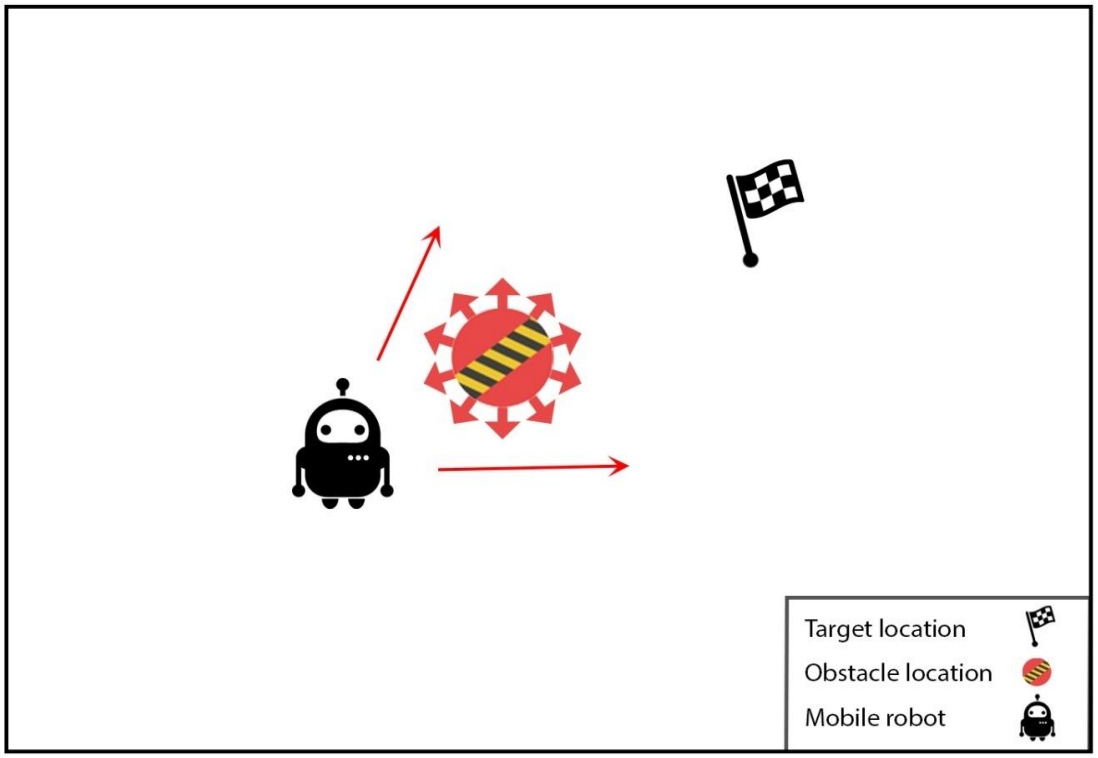

Figure 4.5 Robot rotation around the obstacle

4.2 Different Shapes of Obstacles

The bacterial foraging algorithm does not always guarantee success when different type of obstacles is encountered. Unluckily, with some type of obstacles, the robot can get stuck in the local minima. Thus, local minima problem can cause failure of this algorithm. There are many type of obstacles a robot can face on its path and has to avoid them. Practically, it is difficult to know beforehand whether the robot will stuck in local minima or not. This thesis focuses on developing the improved version of bacterial foraging algorithm to bypass multiple types of obstacles without falling into local minima.

4.3 Unorganized Environment

Based on the theory of bacterial foraging algorithm, the robot will always move step by step. These should be recorded in a temporary memory of that robot. The existing algorithm fails to bypass many type of obstacles as shown in Fig. (4.6). The target location will always attempt to attract the robot where the obstacle will always try to push the robot away. The robot will stuck in the local minima if the robot is

surrounded by the obstacles from two sides because of the two forces (attractive and repulsive forces) acting at the same time. If such an obstacle is encountered, the robot will stop moving, because the obstacle is not of a circular shape. This obstacle surrounds the robot from both sides, hence, the robot will try to find a way to get rid of this local minima.

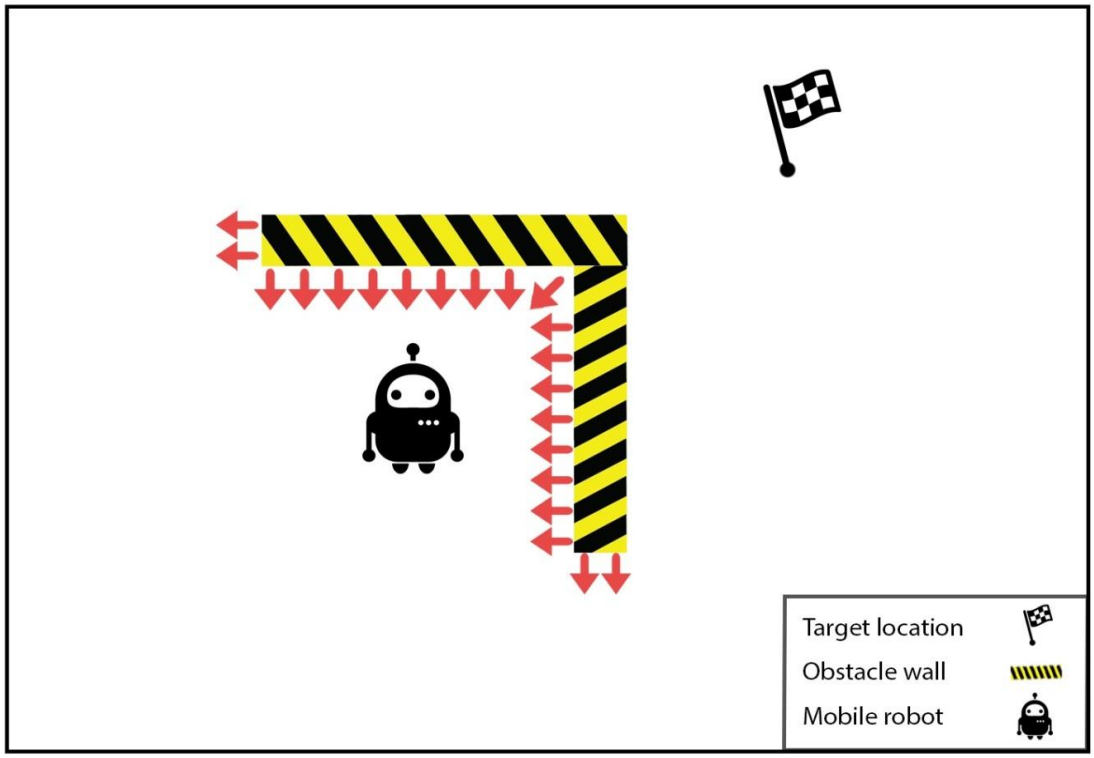

Figure 4.6 Robot get stuck due to shape of the obstacle.

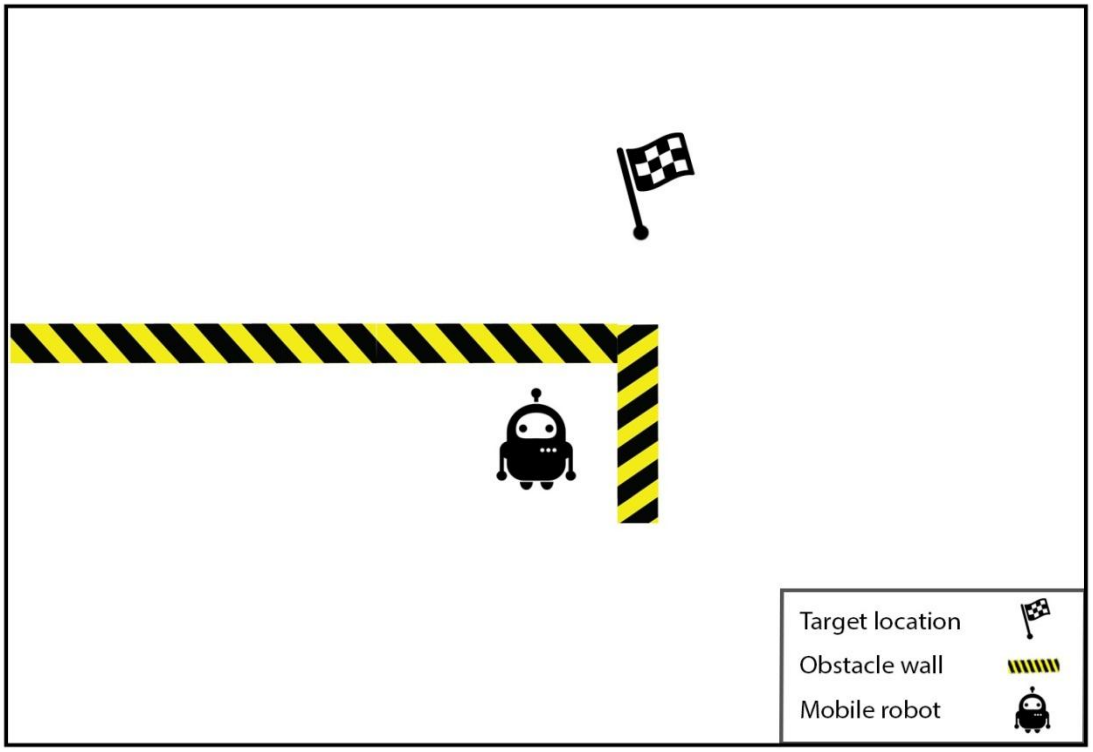

Figure 4.7 Difference between the two lengths of obstacles.

Another scenario is depicted in Fig. (4.7), where it is unknown to the robot that which side of the obstacle's wall is shorten than the other one. Therefore, when the robot tries to escape from the left, it will suffer from long travel with no success towards the target. As a result, the robot will exert considerable distance, energy and time for the length of the existing wall obstacle. Therefore, a solution to this problem must be determined.

In case of a robot stuck into local minima, it will stop for a while and calculate the distance between its current position (in local minima) and each point in its trajectory. Thereafter, the robot will choose one of the points that has specific distance (constant distance) and moves back to it. Once the robot returns to a certain point from its path, the robot will create a virtual obstacle in the local minimum location and share this information with the whole swarm. The robot will continue moving towards the target, it may encounter another obstacle in the surroundings. As a result, the robot will build a number of virtual obstacles with an organized process in the arcs and tight corners. The number of virtual obstacles depend on the size of the enclosed area until the robot can escape from the obstacle and have open area to

move. Using this information, other robots can continue smooth travel without falling into the local minima.

4.4 Proposed Algorithm (SWBFO)

A swarm of robots using bacterial foraging optimization with the reference of path planning finding which is named Swarm Robots using Bacterial Foraging Optimization (SWBFO). With the consideration of ($p_s,\ p_g$) as the start and the target locations for the mobile robot and d_{min} as the radius of the field used for determination of goal location. There is also a circle of radius *r* drawn around the robot. The population of bacterium around the robot in each step is represented by N_s, where l_p represent the iteration that helps the mobile robot to save its trajectory. When the robot fall in local minima, it will choose one of the points in its trajectory that has specific distance dis and moves back to it. The position of the obstacle and the location of the robot are represented by p_o and p respectively. The number of the virtual obstacle that the robot creates in each local minima is denoted by l_o.

With bacterial foraging algorithm, the robot starts by initializing 100 particles (bacterium) with in a circle of radius *r* equals to step size of movement in the trajectory. By using equations 4.1 - 4.6, the error cost function is calculated. The cost function error vector is sorted in an ascending order, and the top one is regarded as a most desirable. When the robot falls in local minima, it will save its current position in the memory p_{vo}, also the number of iterations will be saved in variable *j*. The robot will calculate the distance between its current position and each point in its trajectory. The robot will choose one of the points *p*(*j*) that has specific distance *dis* and move back to it.

Algorithm 4.1: Pseudo-code of SWBFO

Input: $\boldsymbol{p_s}$, $\boldsymbol{p_g}$, $\boldsymbol{d_{min}}$, $\boldsymbol{N_s}, \boldsymbol{l_p}, \boldsymbol{dis}$;
Procedure:

$\boldsymbol{p} \leftarrow \boldsymbol{p_s}$ Initialize robot position,
$l_o \leftarrow 1$; $l_p \leftarrow 0$; $i \leftarrow 0$;
$\boldsymbol{d} \leftarrow \|\boldsymbol{p} - \boldsymbol{p_g}\|$
while $(\boldsymbol{d} \geq \boldsymbol{dmin})$
  $\boldsymbol{l_p} \leftarrow \boldsymbol{l_p} + \boldsymbol{1}$;
  while $(\boldsymbol{i} \leq \boldsymbol{N_s})$
    generate random bacterium a round *p* in radius *r*;
    use Eq. (4.1) to calculate the next point;
    use Eq. (4.2) to calculate $J_{obstacle}$;
    use Eq. (4.3) to calculate J_{goal};
    use Eq. (4.4) to calculate J_{total};
    compute $\boldsymbol{cost_e}$ using the Eqs. (4.5) & (4.6) **;**
    update *d*;
    $i \leftarrow i+1$;
  endwhile
  $i \leftarrow 0$;
  sort $\boldsymbol{cost_e}$
  select best bacterium
  $p \leftarrow best\ point$;
  Save the trajectory; p(l_p) $\leftarrow p$
  If robot fall in local minima
    $p_{vo} \leftarrow$ p;
    $j \leftarrow l_p$;
    while(||p(l_p) - p(j)|| $\leq dis$)
      $j \leftarrow j$-1;
    endwhile
    p ← p(j);
    virobstacle(l_o) $\leftarrow p_{vo}$;
    $l_o \leftarrow l_o + 1$;
  end
  update d;
endwhile

Output:

After repeating the previous operation, the robot can overcome the obstacles (including the non-circular shaped) to get to the goal.

CHAPTER 5

RESULTS AND DISCUSSION

The results are obtained for a mobile robot performing navigation in a cluttered environment with different types of obstacles in two dimensional work space. In order to test the effectiveness of the proposed algorithm, it is evaluated for two different scenarios:

5.1. Scenario 1: Cases without Local Minima Problem

The robot will face many types of obstacle during its movement of which four cases are considered:

5.1.1 Case 1: One Static Obstacle Direct Free-Path

The target is defined at (-20, 90), obstacle is placed at (0, 50), and the robot starts to travel from (50, -90). The mission of the robot is to approach the target location without any impact with the obstacles. In Fig. (5.1) (a) and (b), the robot successfully reaches the target by avoiding a unique spherical obstacle using SWBFO algorithm. The elapsed time required to reach the target is 9.82204 seconds.

The total cost function in each step is the sum of $J_{obstacle}$ and J_{goal} of all the particles around the robot. The next candidate particle is the one having lowest cost function error. For repellant Gaussian cost function, it is maximal only if the robot is near the obstacle and zero if it cannot sense the influence of obstacle. By using mathematical equations of WMR, the robot can go on the path created by the SWBFO until it reaches the goal. The heading angle of the WMR is plotted in Fig. (5.2) against time for the given situation. It can be observed that the mobile robot continuously keeps on changing its heading angle in order to remain on the course or to avoid obstacles. The major spike is observed when WMR encounters the obstacle and tried to make a major shift in its orientation.

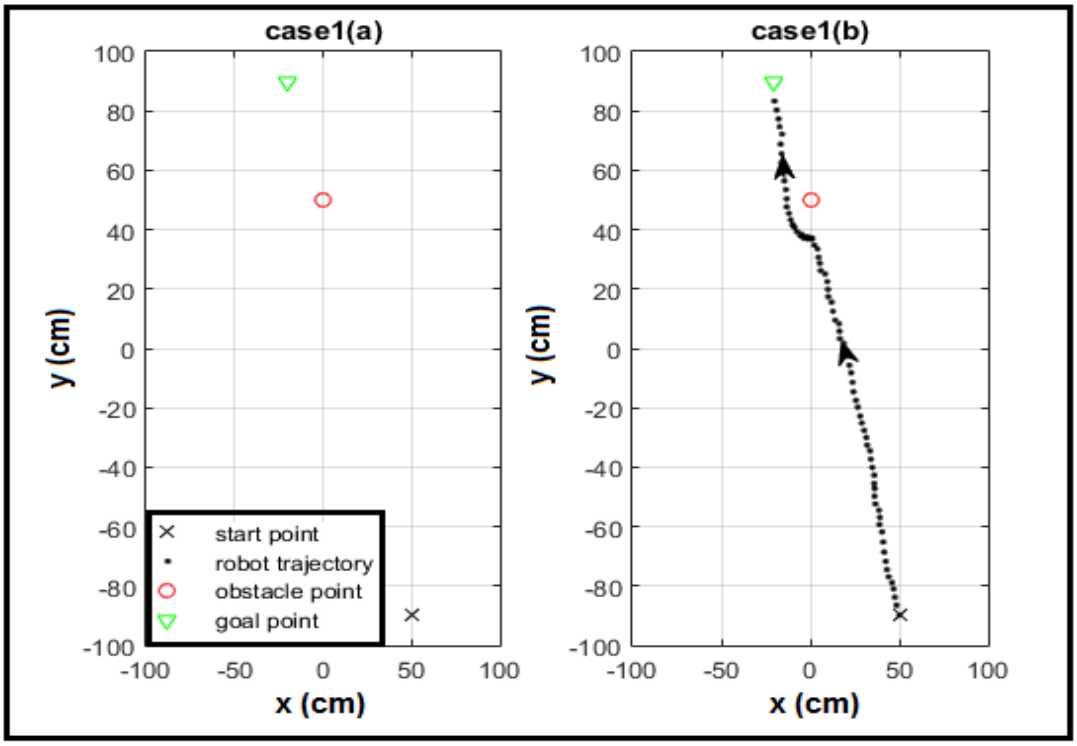

Figure 5.1 case 1 (a) and (b) movement of the robot to the target location.

.

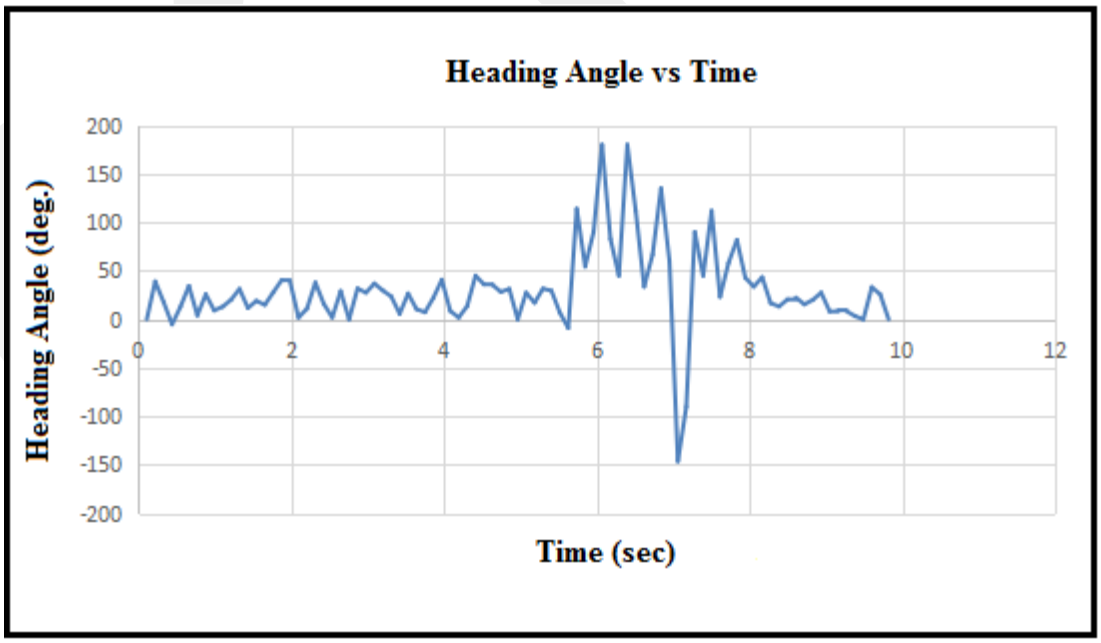

Figure 5.2 Heading Angle vs Time

5.1.2 Case 2: One Moving Obstacle Direct Free-Path

In this case, the target is defined at (-20, 90), the robot starts from (50, -90), and the moving obstacle start at (-25, 50) towards the positive *x*-axis by taking a step of 0.045. The robot moves with on a regular trajectory till it sense the presence of obstacle on its way. The situation is depicted in figs. (5.3) (a) and (b) where the re-route by the robot can be easily observed. The convergence time required for this case is 11.908 sec.

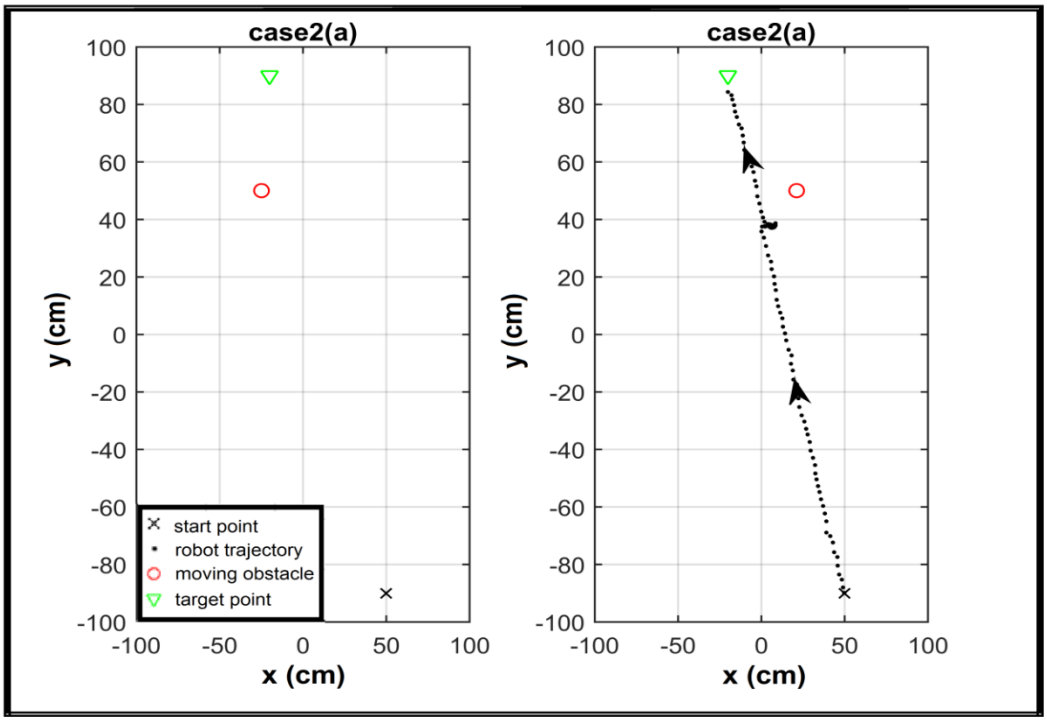

Figure 5.3 Trajectory of the robot when it faces the moving obstacle.

5.1.3 Case 3: Three Obstacles Direct-Free Path

For this case, the target is defined at (-20, 100), the robot starts from (50, -100), and the obstacles are places at (0, -10), (0, 30) and (20, 10). As shown in Figs. (5.4) and (5.5), the robot successfully escapes all obstacles and reach the goal. The process is run for two times; every time different route has been adapted. The convergence time to reach the target in these case is 9.46 and 8.98 seconds. The repellant Gaussian cost function is maximal at the boundary position and near the three obstacles. If the distance between the robot and the obstacles is less than the defined distance, the repulsive field effect on the robot motion is considered zero.

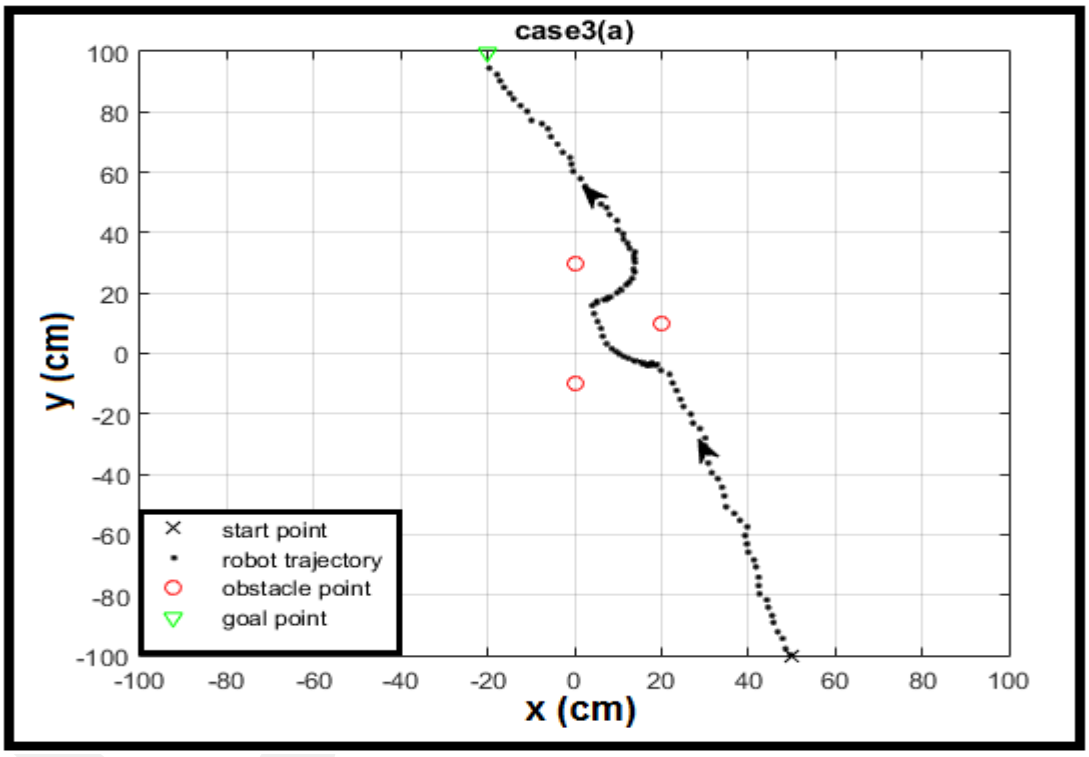

Figure 5.4 Movement of the robot through three obstacles

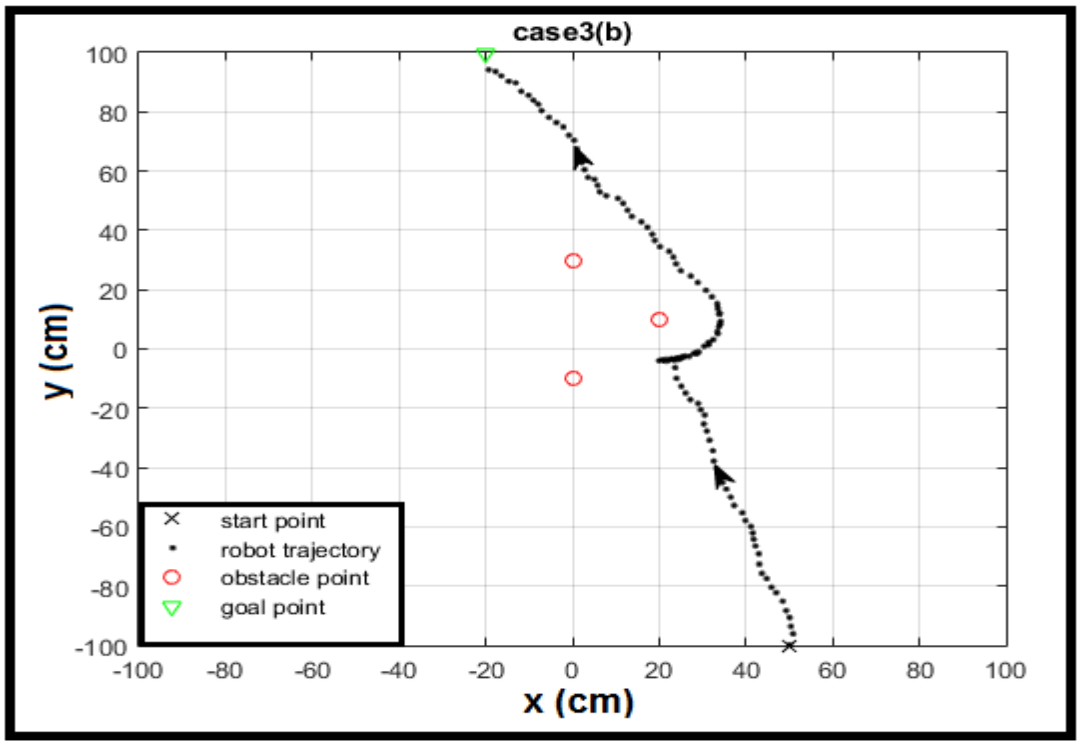

Figure 5.5 Robot escaping from three annoying obstacles.

5.1.4 Case 4: Defeat Local Minima Case with One Obstacle

In this case, the target is located at (0, 100), and the obstacle is defined at (0, -10). The robot starts its travel from (0, -100). It is an interesting case because the straight line trajectory of the robot passes through the obstacle center. The robot can avoid the obstacle and reach the goal point as shown in Fig. (5.6).

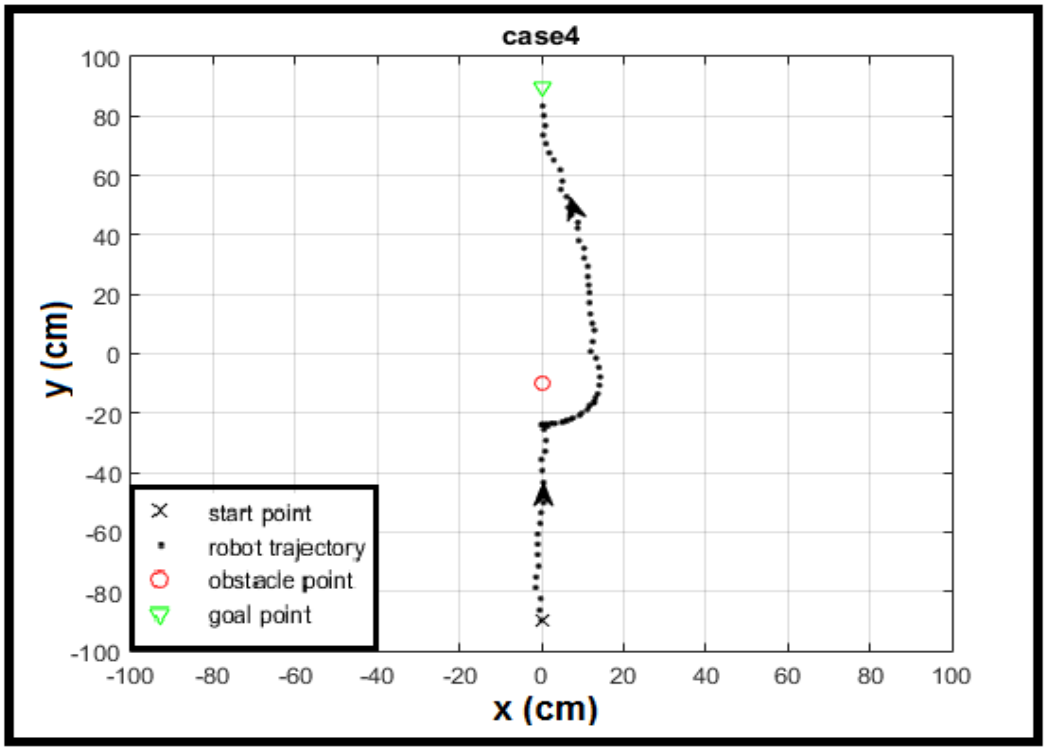

Figure 5.6 Robot defeats local minima.

In other methods like potential field, the robot will trap to local minimum because attractive and repulsive function acts in the opposite directions but existing on the same line cancels each other out. This causes the robot to stay at the same position. But, SWBFO algorithm successfully reaches the target and avoids this problem.

5.2 Scenario 2: Cases with Local Minima Problem

This section explains the behavior of the robot when it falls into a local minimum by using SWBFO.

5.2.1 Case 1: Two Circle Obstacles Close to Each Other

The target is defined at (-20, 100), the obstacles are located at (10, 50) and (-10, 50), and the robot start its travel from (50, -100). The robot can't avoid the obstacles and trapped in local minimum as shown in Fig. (5.7) (a). The resultant field of the two obstacles prevented the robot from moving towards the target.

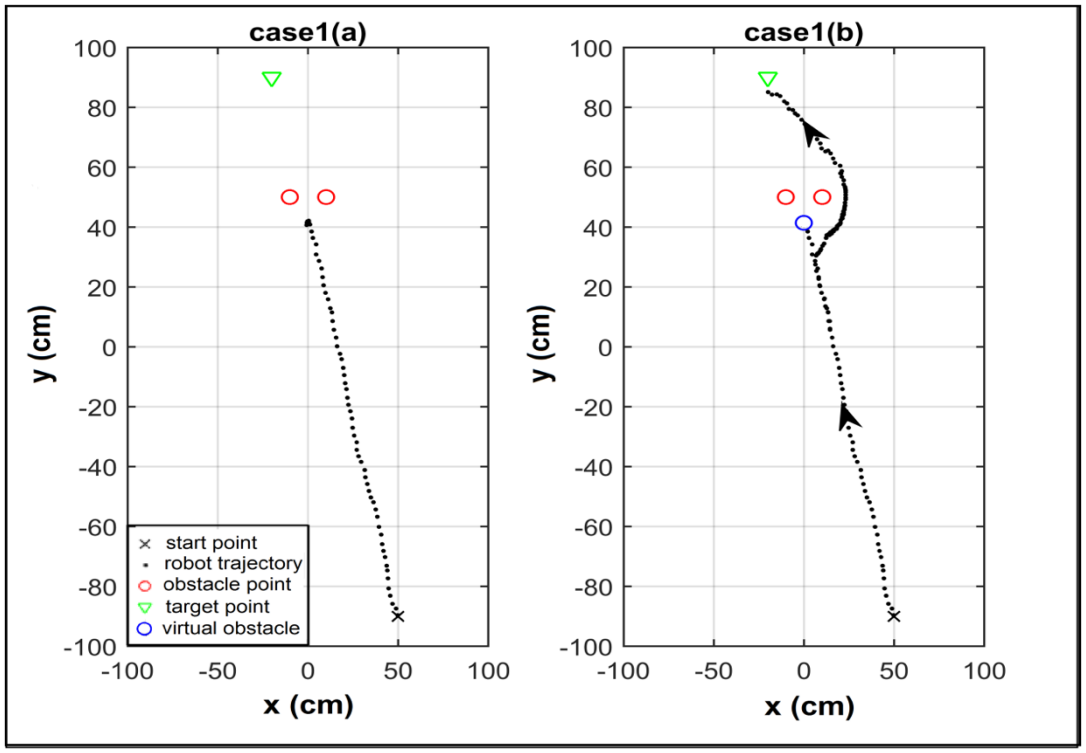

Figure 5.7 Virtual obstacle when the robot falls in to local minima.

When the robot got stuck in local minima, it calculates the distance between its current position and each point in its trajectory. It will choose one of the points that has specific distance from its current position and it will move back to that point. Then, the robot creates virtual obstacle in the local minimum position. The robot avoids to the local minima again by changing its direction and then approach the goal (Fig. (5.7) (b)).

5.2.2 Case 2: L Shape Obstacle

The target is defined at (-20, 90), the robot starts from (50, -90), the coordinates L shape obstacle start from (25, 50) to (-10, 50) and from (-10, 50) to (-10, 30). The robot falls in local minima, then it moves back and creates first virtual obstacle (blue circle), as shown in Fig. (5.8).

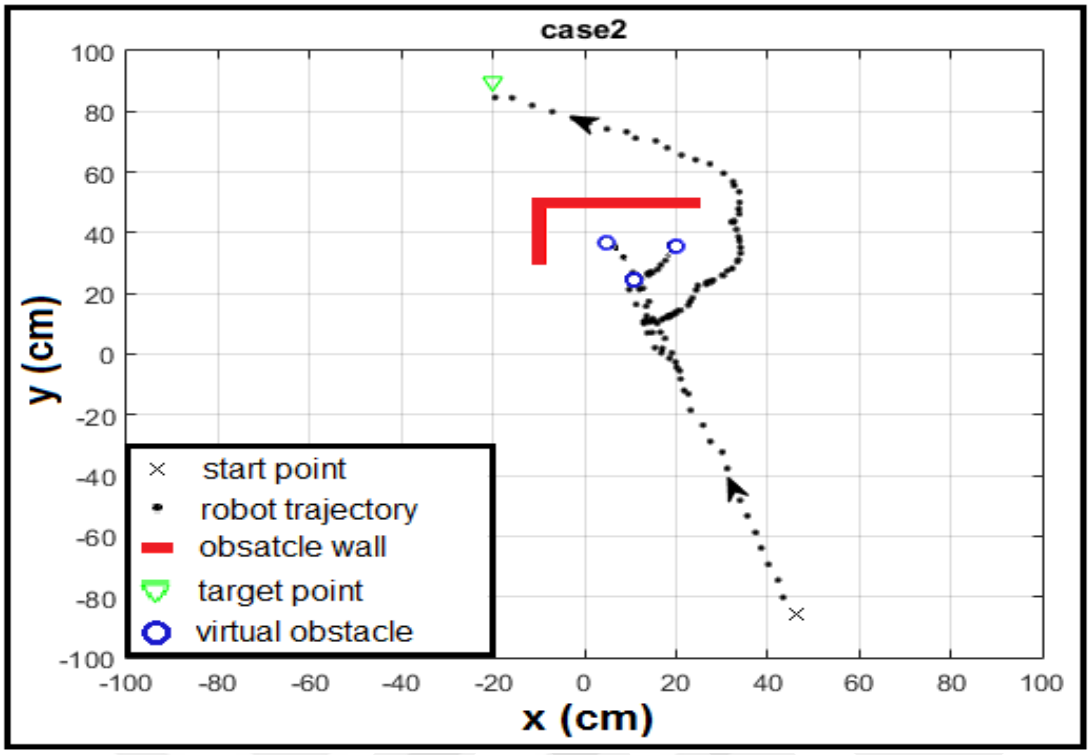

Figure 5.8 Robot behavior towards wall obstacle.

After the creation of three virtual obstacles and recovering from three local minima, it resumes its travel towards the target. It is noticed that the robot keeps changing its direction every time it faces a virtual obstacle. The distance norm (Euclidean distance) between the target and the current position of the robot is plotted against time and shown in Fig. (5.9).

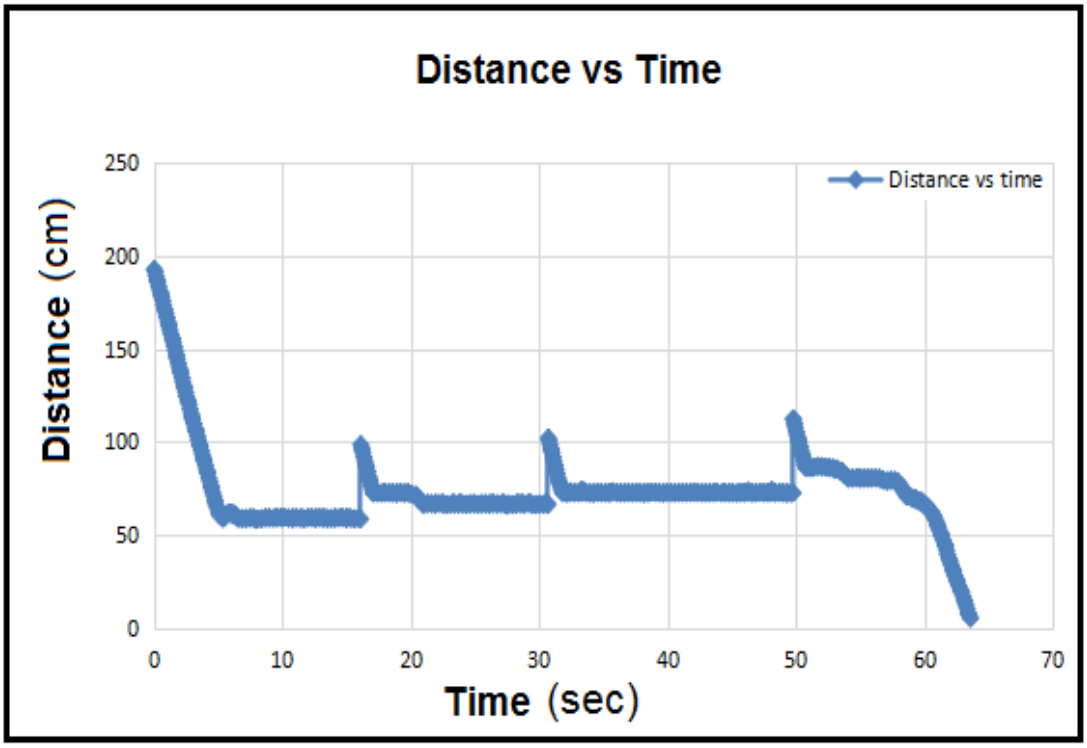

Figure 5.9 Distance norm plot with time.

5.2.3 Case 3: U Shape Obstacle

The target is located at (-20, 90), and the robot starts from (50, -90). The coordinates of the U shape start from (0, 20) to (0, 50), from (0, 50) to (50, 50), and from (50, 50) to (50, 20). The robot falls in local minima for 9 times, as shown in Fig. (5.10). It is noticed that the virtual obstacles created by the robot are arranged in a regular shape, and the distances among each virtual obstacle are uniformly divided. In other words, the robot overcomes the non-circular obstacle by creating virtual obstacles so that the robot can go far enough from the original obstacle and go back to its path in order to reach the goal.

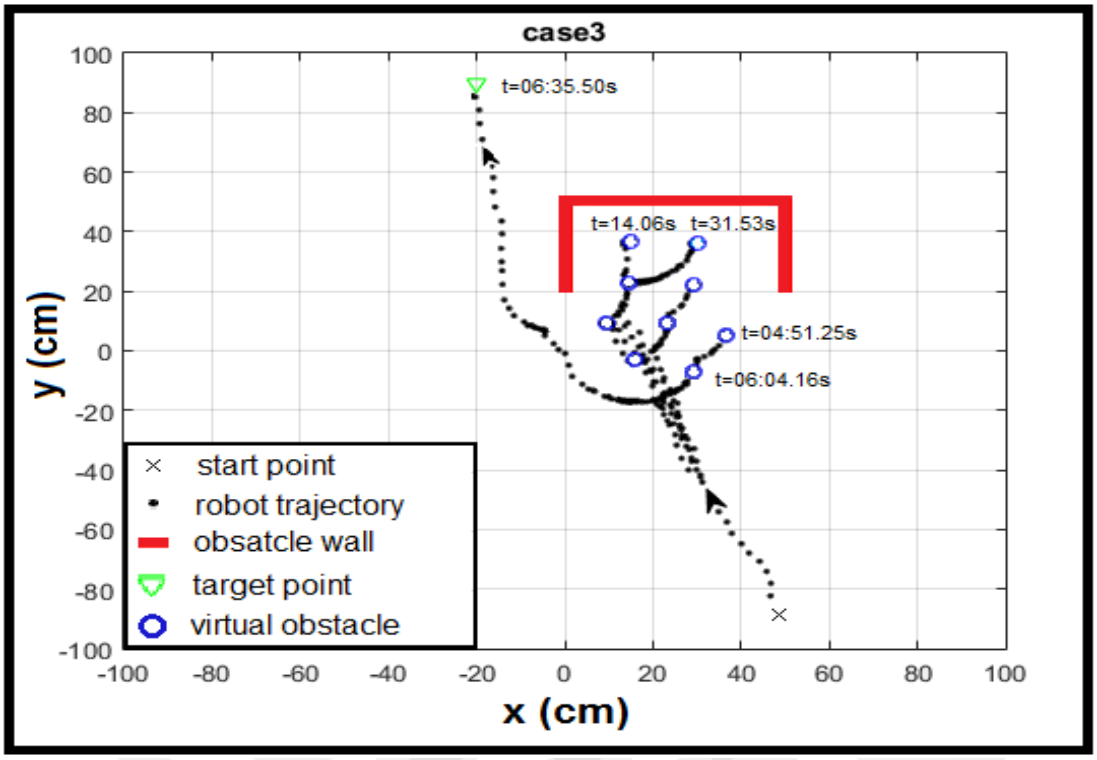

Figure 5.10 Robot behavior with u-shape obstacle.

5.2.4 Case 4: Share the Virtual Obstacles with Other Robots

This case involves swarm of robots with different shapes of obstacles. When one of the robots falls in local minima, it will move back and creates virtual obstacle (blue circle), then it will share this information with other robots to prevent them from falling in local minima. The yellow robot successfully avoids the first obstacle by using the obstacle's information shared by the black robot as shown in Fig. (5.11).

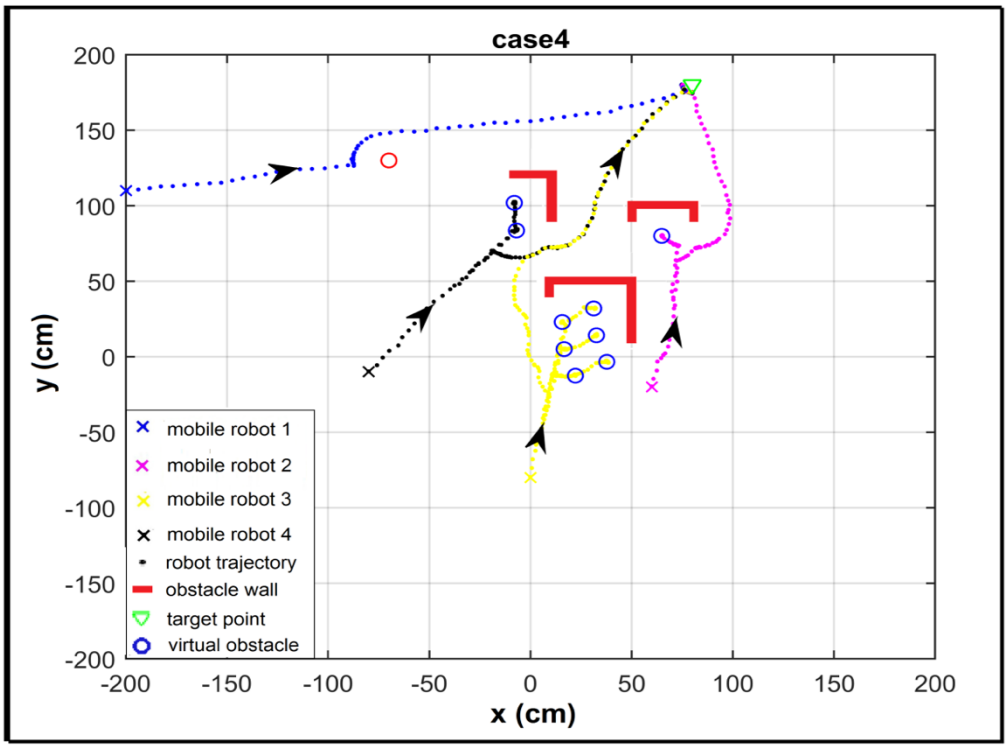

Figure 5.11 Robots behavior and helping each other

The performance of the proposed algorithm is compared against the work of [36]. It is illustrated in Fig. (5.12) that robot fails to reach its goal and falls into local minima following the later approach. The robots (2, 3 and 4) stuck in the local minima of the wall shaped obstacles while robot number (1) is the only one that overcame the obstacle and reach the target because of the circular shaped obstacle that is easy to be avoided.

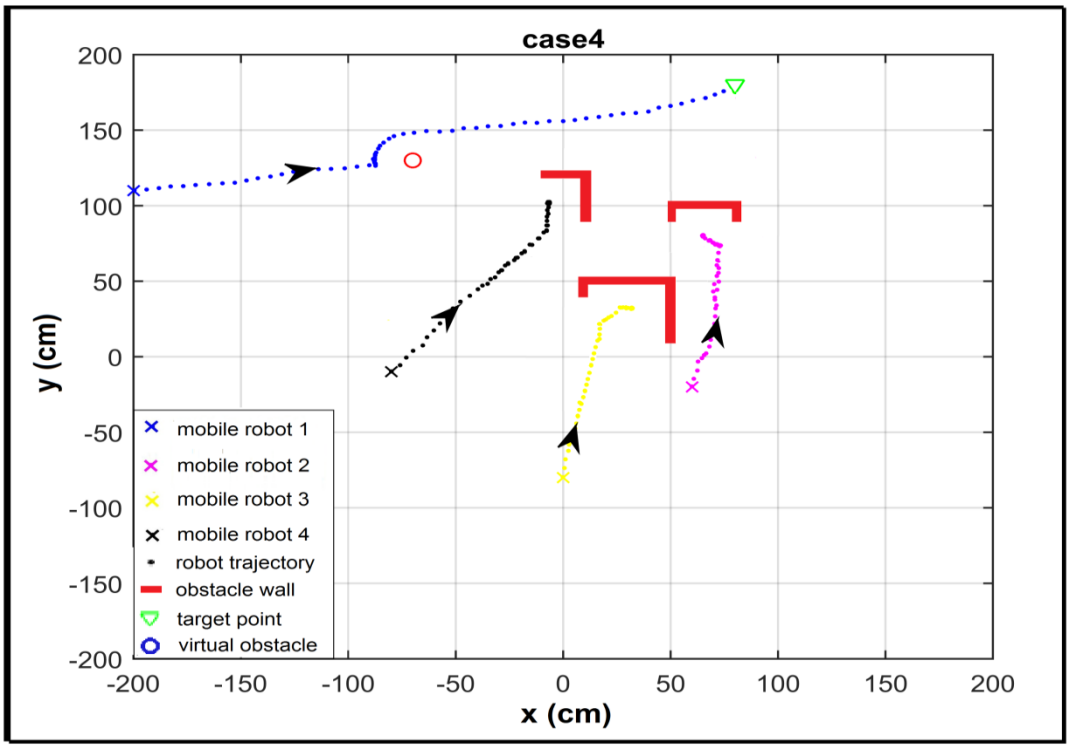

Figure 5.12 Performance of robots and how they get stuck

CHAPTER 6

CONCLUSION AND FUTURE WORK

In the preceding chapters, problem formulation and proposed approach has been detailed for path planning of mobile robots in a cluttered environment. This chapter summarizes the main contribution and conclusions of the SWBFO. Furthermore, the scope of future work is proposed at the end of the chapter.

6.1 The Proposed Work

The prime objective of this work is to reach the target in an effective manner by avoiding maximum type of obstacles. The proposed algorithm is developed to be more efficient and reliable as compared to the existing work. The previous works deals with the Bacterial Foraging Optimization Algorithm (BFOA) to solve path planning problems. It can only deal with circular shape obstacles; hence, it was needed to make some improvements in order to extend its capability. Therefore, proposed work is mainly inspired by the existing BFOA.

The controller uses the input data from different sensors affixed on the robot to identify the environment, locations of the obstacles, and bearing of the goal position. Based on that information, the controller directs the robot to the next position via steering angle as an output according to the SWBFO. The decision scheme permits the robot to choose next position from a number of bacterium around it, where the robot reaches the goal position without colliding with the obstacles that exists along the path.

Sometimes, the robot can encounter obstacles of different shapes (non-circular obstacles), which makes BFO algorithm incapable of overcoming that obstacle. Therefore, a new technique is proposed in order to bypass non-circular obstacles as

well and avoid local minima as well. The information about the explored obstacles is distributed to the whole swarm that can use this information to pre-plan their

trajectory that will reduce the time and error. Through results, it can be claimed that the proposed algorithm proved its success and high performance in many scenarios with unknown environments.

6.2 Future Work

Though the proposed algorithm has been able to perform well in many scenarios, still two scenarios are identified where it stopped working:

1. When the starting point is near the local minima or over the local minima, the robot becomes unable to return to a specific distance placed in its memory, hence, it will stop moving.
2. For the case when robot encounters a closed-form obstacle, it will keep on generating the virtual obstacles in order to avoid actual obstacle. Doing so will fill the space with virtual obstacles, hence, it will not be able to move forward.

REFERENCES

[1] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” Proc. 14th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH 1987, vol. 21, no. 4, pp. 25–34, 1987.

[2] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Roborts,” pp. 500–505, 1985.

[3] R. Bachmayer and N. E. Leonard, “Vehicle networks for gradient descent in a sampled environment,” Proc. IEEE Conf. Decis. Control, vol. 1, no. December, pp. 112–117, 2002.

[4] V. Gazi and K. M. Passino, “Stability Analysis of Social Foraging Swarms,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 34, no. 1, pp. 539–557, 2004.

[5] A. Kamphuis and M. H. Overmars, “Motion planning for coherent groups of entities,” Proc. - IEEE Int. Conf. Robot. Autom., vol. 2004, no. 4, pp. 3815–3822, 2004.

[6] C. Belta, V. Kumar, and S. Member, “IEEE Trans. Rob., vol. 20, no. 5, october 2004 865,” October, vol. 20, no. 5, pp. 865–875, 2004.

[7] M. Kloetzer and C. Belta, “Hierarchical abstractions for robotic swarms,” Proc. - IEEE Int. Conf. Robot. Autom., vol. 2006, no. May, pp. 952–957, 2006.

[8] T. Y. Li and H. C. Chou, “Motion planning for a crowd of robots,” Proc. - IEEE Int. Conf. Robot. Autom., vol. 3, pp. 4215–4221, 2003.

[9] N. Correll, S. Rutishauser, and A. Martinoli, “Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures,” Springer Tracts Adv. Robot., vol. 39, pp. 471–480, 2008.

[10] A. Pandey, “Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review,” Int. Robot. Autom. J., vol. 2, no. 3, pp. 1–12, 2017.

[11] N. HadiAbbas and F. Mahdi Ali, “Path Planning of an Autonomous Mobile Robot using Directed Artificial Bee Colony Algorithm,” Int. J. Comput. Appl., vol. 96, no. 11, pp. 11–16, 2014.

[12] J. Berger, K. Jabeur, A. Boukhtouta, A. Guitouni, and A. Ghanmi, “A hybrid genetic algorithm for rescue path planning in uncertain adversarial environment,” 2010 IEEE World Congr. Comput. Intell. WCCI 2010 - 2010 IEEE Congr. Evol. Comput. CEC 2010, no. August, 2010.

[13] C. Hocaoğlu and A. C. Sanderson, “Planning multiple paths with evolutionary speciation,” IEEE Trans. Evol. Comput., vol. 5, no. 3, pp. 169–191, 2001.

[14] S. Li, M. Ding, C. Cai, and L. Jiang, “Efficient path planning method based on Genetic Algorithm combining path network,” Proc. - 4th Int. Conf. Genet. Evol. Comput. ICGEC 2010, pp. 194–197, 2010.

[15] K. Zhang, E. G. Collins, and A. Barbu, “An efficient stochastic clustering auction for heterogeneous robotic collaborative teams,” J. Intell. Robot. Syst. Theory Appl., vol. 72, no. 3–4, pp. 541–558, 2013.

[16] O. Montiel-Ross, R. Sepúlveda, O. Castillo, and P. Melin, “Ant colony test center for planning autonomous mobile robot navigation,” Comput. Appl. Eng. Educ., vol. 21, no. 2, pp. 214–229, 2013.

[17] D. G. Thelen, J. A. Martin, and J. D. Roth, “Patent Application Publication:,” vol. 1, 2019.

[18] E. Masehian and D. Sedighizadeh, “Classic and Heuristic Approaches in Robot Motion Planning – A Chronological Review,” no. 5, pp. 101–106, 2007.

[19] T. Tsuji, Y. Tanaka, P. G. Morasso, V. Sanguineti, and M. Kaneko, “Bio-mimetic trajectory generation of robots via artificial potential field with time base generator,” IEEE Trans. Syst. Man Cybern. Part C Appl. Rev., vol. 32, no. 4, pp. 426–439, 2002.

[20] L. E. Barnes, M. A. Fields, and K. P. Valavanis, “Swarm formation control utilizing elliptical surfaces and limiting functions,” IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 39, no. 6, pp. 1434–1445, 2009.

[21] M. G. Park, “Artificial Potential Field Based Robots Using a Virtual,” no. Aim, pp. 735–740, 2003.

[22] W. R. Esposito and C. A. Floudas, “Deterministic Global Optimization in Nonlinear Optimal Control Problems,” J. Glob. Optim., vol. 17, no. 1–4, pp. 97–126, 2000.

[23] X. Yang and L. Press, Nature-Inspired Metaheuristic Algorithms, vol. 4, no. C. 2010.

[24] K. M. Passino, “Biomimicry of Bacterial Foraging for Distributed Optimization and Control,” IEEE Control Syst., vol. 22, no. 3, pp. 52–67, 2002.

[25] S. Coelho, “BACTERIA COLONY APPROACHES WITH VARIABLE VELOCITY,” vol. 2, no. 2002, pp. 297–304, 2006.

[26] C. Lucas, “Comments on ‘Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment,’” IEEE Trans. Automat. Contr., vol. 33, no. 5, pp. 511–512, 1988.

[27] E. Rivlin, “TangentBug : A Navigation Algorithm,” vol. 17, pp. 934–953, 1998.

[28] Ishay Kamon and Ehud Rivlin, “Sensory based motion planning with global proofs,” Time Eternity, pp. 1–5, 1995.

[29] J. P. S. Lee, “Cellular Robotic Collision-Free Path Planning,” 1991.

[30] F. Janabi Sharifi and D. Vinke, “Integration of the artificial potential field approach with simulated annealing for robot path planning,” 1993, pp. 536–541.

[31] P. K. R. Volper, “Manipulator Control with Superquadric Artificial Potential Fuction: Theory and Experiments,” vol. 20, 1990.

[32] L. McFetridge and M. Yousef Ibrahim, “New technique of mobile robot navigation using a hybrid adaptive fuzzy-potential field approach,” Comput. Ind. Eng., vol. 35, no. 3–4, pp. 471–474, 1998.

[33] J. O. Kim and P. K. Khosla, “Real-Time Obstacle Avoidance Using Harmonic Potential Functions,” IEEE Trans. Robot. Autom., vol. 8, no. 3, pp. 338–349, 1992.

[34] P. Sudhakara, V. Ganapathy, B. Priyadharshini, and K. Sundaran, “Obstacle Avoidance and Navigation Planning of a Wheeled Mobile Robot using Amended Artificial Potential Field Method,” Procedia Comput. Sci., vol. 133, pp. 998–1004, 2018.

[35] M. G. Park and M. C. Lee, “A new technique to escape local minimum + artificial potential field based path planning,” KSME Int. J., vol. 17, no. 12, pp. 1876–1885, 2003.

[36] M. A. Hossain and I. Ferdous, “Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique,” Rob. Auton. Syst., vol. 64, pp. 137–141, 2015.