UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Mobile Robot Optimum Trajectory Development using a Hybrid Reactive Navigation Model

By

Ngwenya Thabang

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Submitted in partial fulfilment of the requirements for the degree,

Master of Engineering (Industrial and Systems Engineering)

In the Faculty of Engineering Built Environment and Information Technology

November 2021

Supervisor:

Dr MK Ayomoh

ABSTRACT

Path planning for mobile robot navigation in workspaces with varying obstacles complexity levels was addressed in this research. The domain problem is that for a specific class of obstacles referred to as the concave shaped and lengthy obstacles, the likelihood of local minima trap occurring is often significantly high. For instance, a labyrinth premised on concave shaped obstacles often misleads a navigating robot into the concave hollow region in a bid for the robot to reach its desired target point. Apart from the use of reactive algorithms, for an autonomous navigation process which is often premised on continuous path trajectory development, the literature clearly alleges that most non-reactive algorithms get trapped in the concave hollow and along the edges of lengthy obstacles. The purpose of this research is to adapt a reactive mobile robot (MR) navigation algorithm premised on the Hybrid Virtual Force Field (HVFF) concept for the exploration of robot navigation in both developed and literature based obstacle constrained workspaces. The obstacles considered in this research work are mostly premised on concave shaped and lengthy obstacles cul-de-sac. The HVFF approach evolved from the Virtual Force Field (VFF) approach which is premised on the Potential Field Method (PFM). This method of path planning operates by utilizing the resultant of forces emanating from the combination of repulsive and attractive forces acting on a navigating robot. The algorithmic validation was carried out via the conduct of simulation trials using the Python software. The simulations conducted span across newly developed workspaces and literature based workspaces for a comparative study. Furthermore, the behaviour of the robot navigation with and without the HVFF algorithm per workspace was presented. Of a particular interest was the navigation time with and without the HVFF algorithm per workspace. The results obtained in all the simulations showed a much efficient navigation completion time with the use of the HVFF algorithm. Efficiency in arriving at the target point implies that the robot was able to come out of the local minima trap each time it entered the hollow region of a concave shaped obstacle or around the edges of a lengthy stretched out obstacle. The time difference recorded between deploying the HVFF approach and not deploying the HVFF algorithm across the different simulations conducted spanned between 14.27 to 287.44 seconds which corresponds to a percentage gain time of 31.87% and 89.70% including a simulation with an unending target point (TP) arrival time for the without HVFF algorithm. As the concave trap increased in its depth, the tendency of the robot to escape from the trap becomes much more difficult. The outputs of this research justify the effectiveness and efficiency of the HVFF algorithm.

Key Words: Mobile Robot, HVFF Algorithm, Concave Obstacles, Target Point, Hybrid Approach.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3IR - Third Industrial Revolution

LMT – Local Minima Trap

MR – Mobile Robot

AVGs - Automated Guided Vehicles

LiDAR - Light Detection and Ranging

VFH - Vector Field Histogram

VFF- Virtual Force Filed

NN - Neural Network

RF – Repulsive Force

RN – Robot Navigation

FA – Firefly Algorithm

AF – Attractive Force

AP – Attractive Potential

RP – Repulsive Potential

PSO - Particle Swarm Optimisation

RO - Real Obstacle

ACO - Ant Colony Optimisation

GA - Genetic Algorithm

APF - Artificial Potential Field

HVFF - Hybrid Virtual Force Field

VOC - Virtual Obstacle Concept

VO – Virtual Obstacle

PP – Path Planning	
OA - Obstacle Avoidance	
SO – Static Obstacle	
TP – Target Point	
DO – Dynamic Obstacle	
VGC - Virtual Goal Concept	
F-ACHA - Fireworks-Ant Colony Hybrid Algorithm	
DAPF - Discrete Artificial Potential Field	
CNN - Convolutional Neural Network	
IMU - Inertial Estimation Unit	
JPSO - Jumping Mechanism Particle Swarm Optimisation	
SGOA - Safety Gap Obstacle Avoidance	
PGA - Parallel Genetic Algorithm	
UAV - Unmanned Aerial Vehicle	

CHAPTER ONE

INTRODUCTION

1.1 Background to the study

Mobile robot (MR) navigation and obstacle avoidance (OA) has over the years gained the attention of researchers, manufacturers, enthusiasts and operators of robotic vehicles stated by Li and Savkin in 2018 [1]. MRs have widely been deployed in everyday human activities including manufacturing, agriculture, military, medicine, and several other domains. Safe navigation is an essential requirement for MRs in their respective environments. Therefore, research on autonomous robot navigation as they move from one point to another in a given workspace without making contact with obstacles is fundamental for their physical and functional safety in an-environment littered with obstacles. Abiyev et al [2] stated that these obstacles often act as navigation obstructers along the path between the robot and its target point (TP). The robot navigation space can be littered with both static obstacles (SOs) and dynamic obstacles (DOs) of varying convex and concave shapes resulting in cul-de-sac capable of creating a local minima problem. Furthermore, workspace obstacles can be further classified as being virtual, real, concave, convex, static or dynamic with the aid of sensing devices and algorithmic procedures.

The problem of robot navigation can be summarised using the three principal questions viz: where am I?", where am I going?", and how should I get there?" Question one is about localisation, question two is goal or TP centred while the third question is about planning a path that results in achieving the defined goal.

Investigations of the latter two questions often come under the domain of path planning (PP) and OA.

The use of reactive hybrid navigation algorithms is beginning to take the centre stage in the MR path planning problem for enhanced intelligence. The integration of different navigation techniques has the ability to produce more efficient and effective navigation results. Olunloyo and Ayomoh [38] proposed a hybrid approach referred to as the Hybrid Virtual Force Field (HVFF) approach. This technique like a few others, has the benefit of high TP attainment. A MR constrained by this methodology can generally arrive at its TP without colliding with the workspace obstacles [39]. One of the most significant merits of using reactive navigation algorithms is that they are usually online compliant i.e. they have the ability to make the robot self-governing hence, making it capable of coping in unstructured workspace domains in a bid to be fully autonomous while navigating. To achieve its goal, the robot must be able to perceive its environment sufficiently to allow it navigate safely. In recent times, some areas of success have been reported in the literature but nonetheless research in autonomous MR navigation may be well off the infancy stage however, far-fetched from the desired target of attaining the human dexterity. Extensive research is still on going for improvement in this research space to make its widespread use in all facets of the human endeavor possible.

Robot PP, in a wider sense may well be alluded to as the method of distinguishing impediments free configurations inside a given workspace in arrangement to upgrade a collision free path from its start position to its TP. To be beyond any doubt, PP for MRs is an intricate issue that also requires smoothness and clearances besides guaranteeing a collision-free path with minimum traveling distance. Two crucial classification recommended by Fu et al [3] to portray the robotic PP issue are Obstacle Limitation and Path Limitation. Obstacle Limitations show that there are a few focuses in space which are occupied and are not free for the MR to pass through. Path Limitations are more often than not given as points on a path which the MR must take after. As a robotic vehicle translates and orientates her member components at different points with time and in space, while accomplishing an assigned task, it does this in some defined paths ensuring that obstacles are avoided both locally and globally. Local and global path planning are next discussed below:

1.1.1 Local Path Planning (sensor based planning)

Sometimes information may not be available at the inception of tackling an issue in this way we must illuminate the issue in stages as data is continuously made accessible. Sensor-based planning is a vital work when situations alter with time, are obscure, or there are mistakes in the robotic equipment. A postieri information can be utilised to discover the next trajectory in a path (by collecting data about the results of the previous trajectory) or may also be utilised to guide the MR in a random sense when investifating an environment. These techniques correspond to an “execute and evaluate” methodology. The information feedback in such cases is acquired with the help of sensors while the sensors utilised may range from vision frameworks to contact switches.

1.1.2 Global Path Planning (information based planning)

It is much simpler to solve an issue in a case if all the information needed about the workspace is available at the beginning and prior to the onset of movement. In robotics we may plan paths before their execution if we have adequate information of the environment. The PPs before execution facilitates solutions of a shorter path time, more efficient dynamics, and absolute collision evasion. When working in this mode, a priori information (i.e. known before) is utilised. Techniques are available to solve a variety of problems, when given a priori information. A few of the knowledge which we utilise for a priori PP may come from diverse sources such as vision frameworks, designing details, or CAD programs. Such a priori information may moreover be pertinent to moving objects, in the event that they have an unsurprising recurrence or movement. However, a priori information cannot be utilised for eccentric or arbitrary moving objects.

1.2 Problem Statement

The robot navigation problem is that which focuses on an autonomous MR navigating from an initial position to an endpoint without colliding with objects in sight along its trajectory of navigation. The robot navigation problem can be divided into two sub-categories viz: The local minima navigation problem where the MR moves in an environment with unknown obstacle information and the global minima problem, where the MR meander's in a workspace with prior knowledge of the obstacles' information and the environment as a whole. MRs must complete some complicated tasks in a time-efficient manner and avoid virtual obstacles (VOs) and real obstacles (ROs). Hence one of the problems is that of the total travel time minimisation of a MR amidst a workplace with obstacles. Over the years researchers have pioneered several methodologies for MR navigation. An in-depth information of individual algorithms for navigation given by Patle [5] highlight the following gaps:

- Limited research in DOs, multi TPs, multi robots environment: Limited research has been accounted for particularly in an environment consisting of DOs, multi TPs, real-time experiments, and multi robots framework.
- Few papers on hybrid approaches

The complexity in MR navigation has been increasing gradually due to changing environments. As a result, among the specific issues to be addressed in this research is the problem of cul-de-sac traps which are caused by deep concave-shaped obstacles. Imagine a real-life environment where an MR fall into a 1km deep concave-shaped obstacle only to realise while it is inside the obstacle that the target is behind the obstacle. This scenario can be put in perspective in mining industry where MRs are used to do inspection where it is dangerous for humans. Figure 1 and Figure 2 show a local minima trap (LMT) by a deep concave-shaped obstacle and a lengthy obstacle respectively, where R represents the MR and T the target. A real-life scenario can be in mines where MRs are deployed to do a safety inspection.

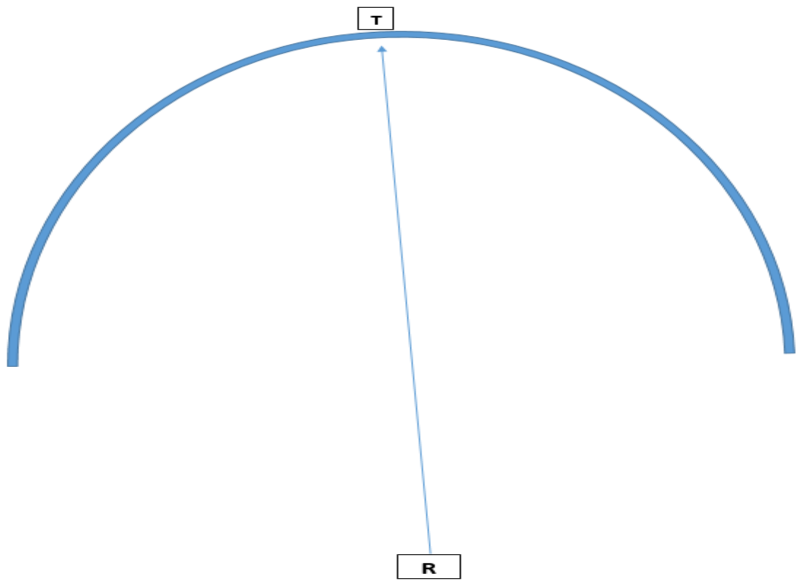


Figure 1: LMT by a concave-shaped obstacle

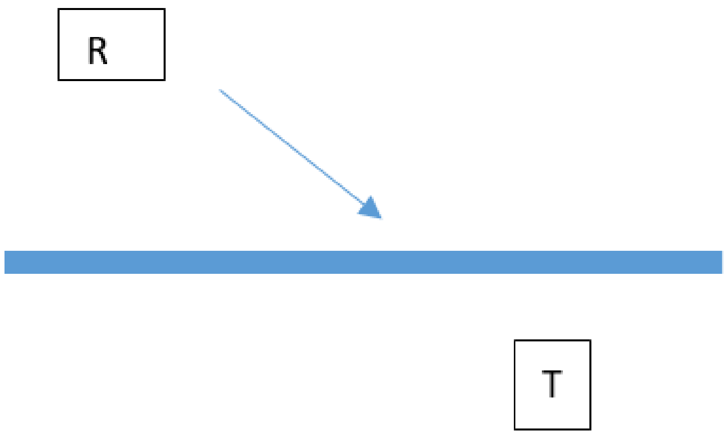


Figure 2: LMT by a Lengthy obstacle

1.3 Research Questions

- How can an optimal navigation algorithm for a robotic vehicle in a domain of obstacles be actualised?
- How can a robotic vehicle avoid the local minima traps (LMTs) amidst VOs and ROs?
- What specific optimisation control metrics are required to effectively deploy a real robotic vehicle to a test environment?

1.4 Research Aim

This study is aimed at adapting a reactive MR navigation algorithm premised on the HVFF concept to deep concave and lengthy obstacles cul-de-sac randomly distributed in developed workspaces as a measure towards addressing the local minima problem posed by these categories of obstacles.

1.5 Research Objectives

The objectives of this research are to:

- Explore and adapt the HVFF concept for effectiveness and efficiency studies of robot navigation in workspaces with concave or lengthy-stretched out obstacles.
- Adapt the HVFF algorithm to a multi-target point navigation problem
- Validate the HVFF algorithmic performance in the different workspaces using simulation trials premised on the Python software.

1.6 Motivation

The introduction of robotised subsystems had grown gradually since the Third Industrial Revolution (3IR) because of their profitability and comfort. A significant part of the time base is determined to the inclinations these structures have and give less thought to their system unpredictability with the change in atmosphere plans. Therefore, the 3IR has left the thin scene with systems unequipped for the adaptable and independent end. An enormous bit of the genuinely steady organisations across metropolitan territories, structures, adventures, and present-day motorisation cannot pick or prepare decisions in an anticipative manner. As needs are, there are pointless deficiencies similarly to obtained costs in the steady movement of these genuinely strong organisations. Inefficient information about the complexity of robots in industrial production may result in a poor assessment of installing measures that could redesign MRs and how profitable/economical they can be. Change in environmental conditions has increased system complexities in industrial automation. Planning and coordination of MRs and automated guided vehicles (AVGs) in workspaces in which these utilities are deployed is very vital. Accordingly, there is a need to introduce smartness into these systems so they can have the option to confer among themselves and their incorporating areas with enlargement to have the alternative to recognise, act, portray, and qualifying conditions. These require

1.7 Scope of the Research

This research will focus on the efficient robot vehicle navigation model amidst workplaces with obstacles. The scope will consist of mathematical algorithms, programming using python, and thorough theoretical research to investigate methodologies needed for navigating a MR to enable it to cope in real-world situations consisting of obstacles.

1.8 Limitations of the Research

- This research focus is on a productive robot vehicle route model in an objective area amid workspace snags. Two major constraints towards achieving the time-efficient objective include the path length and obstacle state identification/avoidance especially when they are along the robot's line of sight.
- The navigation environment considered for the robot is an unknown environment in which the robot acquires basic obstacle information such as position and geometry through sensory inputs.
- The programming algorithms will be validated through simulation trials.

1.9 Delimitations of the Research

- The global navigation path will not be looked at since the research will look at an unknown environment.

- Due to current circumstances linked to the pandemic, the real-world application will not be considered for validation instead simulation environment will be used.

1.10 Notations

The following table gives the notations used in Chapter Three.

Table 1: Mathematical Notations

No	Symbol	Meaning	No	Symbol	Meaning
1	x_r	x-coordinate of the robot	15	$U_{att}^{x}(q_r)$	x-component of attractive potential generated by goal state
2	y_r	y-coordinate of the robot	16	$U_{att}^{y}(q_r)$	The goal state y-component of attractive potential
3	i	Grid-line occupied by robot on x-axis	17	$U^{x}(q_r)$	Resultant potential in the x-axis
4	j	Grid-line occupied by robot on y-axis	18	$U^{y}(q_r)$	Resultant potential in the y-axis
5	$c_{i,j}$	Dynamic window cells	19	$U_{att}(q_r)$	Total attractive potential from the target point
6	x_t	x-coordinate of the robot goal state	20	$U_{rep}(q_r)$	Total repulsive potential from obstacles
7	y_t	y-coordinate of the robot goal state	21	q_r	Configuration space describing the current position of the robot
8	x_o	x-coordinate of the obstacle in robot's active window	22	q_t	Configuration space describing target point
9	y_o	y-coordinate of the obstacle in robot's active window	23	$U_{rep}^{x}(q_r)$	x-component of repulsive potential
10	d_t	Robot distance from the goal state	24	$U_{rep}^{y}(q_r)$	y-component of repulsive potential
11	d_o	Robot distance from an obstacle			
12	x_{grid}	Grid-line similar on the x-axis of the robot compartment			
13	y_{grid}	Grid-line similar on the y-axis of the robot compartment			
14	F_{cr}	Repulsive force variable of sensed obstacle			

1.11 Organisation of the Research

The remaining chapters of this research are structured in the following way: Chapter Two gives a literature review where the review gives an in-depth understanding of the methodologies that already exist to develop optimum MR navigation models. Chapter Three gives the research approach that will be taken to meet the objectives of this research. The Chapter presents the chosen methodology from the thorough literature review in Chapter Two. Chapter Four gives the simulation runs results and discussion of results. Chapter Five it is the conclusion, research findings, and future work.

1.12 Chapter Summary

The introductory chapter gave background study to this research and clear details of the problem that will be addressed and gave the objectives. It further gives the motivation, limitations and scope of this research. And the aim of this research work is to at adapt a reactive MR navigation algorithm premised on the HVFF concept to deep concave and lengthy obstacles cul-de-sac randomly distributed in developed workspaces as a measure towards addressing the local minima problem posed by these categories of obstacles.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter reviews exploration and applications on a scope of subjects of significance for portable robots and AGVs route amid obstacles. The utilisation of MRs, and AGVs specifically, is developing as the scope of robot applications in industrial facilities, clinics, places of business, and so on increments [4]. The objective of this literature review is to analyse techniques/methods that currently exist for the MR navigation in an environment where OA is required. The chosen technique will be used to build up an ideal robot vehicle navigation model in an objective area amid a workspace with obstacles.

2.2 Review

The review gives an in-depth understanding of the methodologies that already exist to develop optimum MR navigation models. This review aims to further find gaps that still exist in research for MR navigation problems and add the latest research work from 2019 to 2021 that is not included in techniques reviewed in the paper [5]. Over the years researchers have applied several methodologies to addressing MR navigation amidst obstacles [6]. The following was extracted from the research papers.

- The problem/s solved in the field of research
- How the problems were solved
- Who solved these problems
- Limitations and outcomes of the research

- Possible future work

2.2.1 Simultaneous Localization and Mapping (SLAM)

SLAM refers to methodologies that solve the issue of constructing the map of the workplace with no prior knowledge and localizing MR into this map without any human involvement [7]. This technique was applied in MR navigation for OA in 2014 by Moreno-Armendariz and Calvo [8]. Iizuka et al [9] in the same year did a study that used the SLAM technique in MR navigation in presence of moving obstacles. The technique was combined with an artificial potential field (APF). The hybrid approach was a success since the MR did not collide with obstacles till it reached the target point (TP). Their future work will focus on deploying the technique in real-time application. Sqalli et al [10] used the technique for OA for an improvement of RN. Moreover, several other studies extended research in using this technique for RN [11-13].

2.2.2 Light Detection and Ranging (LiDAR) Technique

LiDAR is a dynamic eminent distance detecting innovation, arranged as a range estimation sensor that consistently sends a beam of light utilising pivoting radiates at a steady rate and registers the distance between the objective and itself with high precision. Local navigation methods use sensors for the position and orientation of a MR. In the autonomous industry, the LiDAR technique is often utilised for automation. LiDAR works autonomously when contrasted with the GPS framework; along these lines, it has the capacity of planning the climate. LiDAR can be utilised freely yet when combined with different sensors it gives improved outcomes [14].

In 2017, Ghorpade et al [15] proposed an efficient OA model using the 2D LiDAR technique for a MR. The objective was to accomplish acceptable constant execution and improve the precision of OA focused on independent mechanical frameworks intended for military applications. The proposed strategy extricates spatial data from laser point-cloud utilising division and bunching strategies. The convex body calculation was used to recognise the precise geometry of the impediment. Convex-shaped objects were considered for this research as SOs. The proposed OA and obstacle identification technique used a basic numerical model and productively accomplished great ongoing execution. The unwavering quality of the model was simulated in MATLAB. The technique was made economical, portable, and robust because of the inclusion of Raspberry pi 3 in the system. In the future, the work will focus on improving the handling time of the proposed model.

In 2019, Madhavan and Adharsh [16] extended this work using 2D LiDAR. A deliberate methodology was utilised to dodge impediments on the guideline of least expense work. The simulation results classified obstacles to be circular, quadrilateral, and linear-shaped. The future work will consider dynamic OA. Moreover, Baras et al [17] used LiDAR and Raspberry Pi to address navigation problems while the autonomous vehicle avoided impediments. The paper considered an environment with static impediments. The experiments were conducted in real-time in presence of multiple impediments of various sizes and shapes. Future adjustments to their algorithm might be made to anticipate impediments movement and explore more efficiently in a dynamic workspace. Moreover, other real-time experiments were conducted for OA using 2D LiDAR in 2020 by Dong et al [18] and Ren Yee et al [19]. The papers considered both concave and convex-shaped obstacles. Dong et al [18] future work will focus on 3D models for MR navigation. One of the key factors is that the researchers used hardware that is less expensive for the 2D LiDAR technique.

2.2.3 Vector Field Histogram (VFH) Technique

The VFH technique was pioneered by Borenstein and Koren [20]. The technique was found to be very robust, efficient, and not sensitive to readings. Moreover, Ulrich and Borenstein improved the VFH in 1998 [21] and 2000 [22]. The technique is intended to diminish the restriction of potential-field strategies, for example, robot motions while dodging the obstacles [23]. The VFH technique is another method of RN utilised to solve the PP of MRs. In 2014, Yim and Park [24] used VFH in MR navigation. Their simulations considered SOs which were convex shaped. Furthermore, Kumar and Kaleeswari [25] implemented the VFH in a robot where DOs and SOs were considered. The experiments were conducted in real-time and their future work will consider the use of potential field strategy.

In 2019, Alagic et al [26] proposed a modified VFH technique in a MR framework. Their VFH calculation gave both local movement arranging and obstruction evasion dependent on ready sensor estimations. The adequacy of the proposed system was confirmed in both static and dynamic obscure conditions. The acquired results demonstrated the capability of the VFH calculation in exploring the portable MR from the beginning to the objective location evading impact with impediments. The drawback with this technique is that local route planning does not give the best results when it comes to travel time and distance covered. Similarly, Diaz and Marin [27] improved on the algorithm proposed by Ulrich and Borenstein [21]. The experiments used two robots in the presence of convex-shaped obstacles.

2.2.4 Artificial Potential Field (APF)/ Virtual Force Field (VFF) Technique

The APF way of arranging innovation previously proposed by Khatib is based on a basic level and appropriate for constant control [28]. The APF technique is known as the VFF technique. This VFF technique was pioneered by Borenstein and Koren [29]. Moreover, this technique has been broadly used in research for route planning, robot optimisation, and MR navigation. However, the drawback with APF is that it falls into the local minima trap (LMT) and neglects to arrive at the objective. The essential thought of the APF technique is to make the robot move using forces such that obstacles produce a repulsive force (RF) and the objective delivers an attractive force (AF) on a MR. Complete forces of the AF and the RF control the moving of the MR. Along these lines, the MR can effectively maintain a strategic distance from obstructions and arrive at the objective. Furthermore, the following functions below are imperative in understanding the APF technique [30].

Assume the position of a robot is given as $P_r = (x, y)$, and the position of the goal is given as $P_g = (x, y)$, then the attractive potential (AP) function is defined in Equation (2.1).

$$U_a = w_a * (P_r - P_g)^2 \qquad (2.1)$$

The repulsive potential (RP) function is defined in Equation (2.2), the RP expands to prevent the robot from hitting the impediment.

$$U_{rep} = \{0 \ \frac{1}{2} * \eta * (\frac{1}{\rho} - \frac{1}{\rho_0})^2 \qquad (2.2)$$

Such that η is a positive constant, ρ is a minimum length between an obstacle and a robot, ρ_0 is a maximum effective length of one impediment. There is no effect for a robot when the length between a robot and an impediment is bigger than ρ_0. Then the total potential field is defined as follows:

$$U_t = U_a + \sum U_{rep} \tag{2.3}$$

In 2015, a paper by Chiang et al [31] proposed the route-guided artificial potential field-stochastic reachable (APF-SR) strategy for successful routes in complex situations. The problem of guidance for a robot was in both inactive impediments as well as various dynamic and stochastic impediments. The obstacles were concave and convex-shaped. The strategy works by beginning with a sampling-based method to recognise a substantial, collision freeway within the nearness of inactive impediments. Then the APF is used to move the robot through moving impediments. This technique is successful where the APF method alone fails by falling into traps. This work was extended in 2017 by Malone et al [32] for route planning in a highly intricate and dynamic workplace with impediments.

Additionally, Sudhakara et al [33] investigated the OA and guidance pathway of a wheeled portable robot using the amended APF technique. In their work, this technique does not think about the impact of RFs and AFs. The great point of proposing this technique was to beat the issue that the traditional APF could not adjust to the complex direction of arranging and falling prey to LMTs. Simulations were done in recursive U-formed, long divider, unstructured, labyrinth-like, and jumbled situations. The results showed that the proposed enhanced APF may very well be adequately used in the direction arranging of wheeled portable robots and can be applied progressively in real-time situations. The great advantage of the proposed improved APF is that the calculation adjusts well in both straightforward and complex conditions with a short travel time. Similarly, Lu et al [34] proposed an algorithm dependent on the improved APF to tackle the issue of local optimum. Another paper that focused on the improvement of the APF method is by Lin et al [35].

Moreover, a discrete artificial potential field (DAPF) technique for portable robot PP was introduced by Lazarowska [36]. The DAPF calculation utilises the idea of an APF and alters it for use in a discrete setup space. The outcomes of investigations showed that the DAPF calculation is fit for finding a crash-free way for a portable robot in both dynamic and static conditions. The advantage of this technique is the close ongoing activity, which makes it helpful for pragmatic applications. Future works will focus on trials with multiple DOs. In 2021, Shin and Kim [37] pioneered a hybrid approach that combines positioning risk (PR) and the APF technique. The results showed that the PR-APF method generated more than 90% success paths while the APF failed to generate 50% success paths. The simulations were tested using MR and aerially in real-time. The limitation of this technique is a failure if a lot of positioning errors occur. The figure below (figure 3) shows the flow of this proposed algorithm.

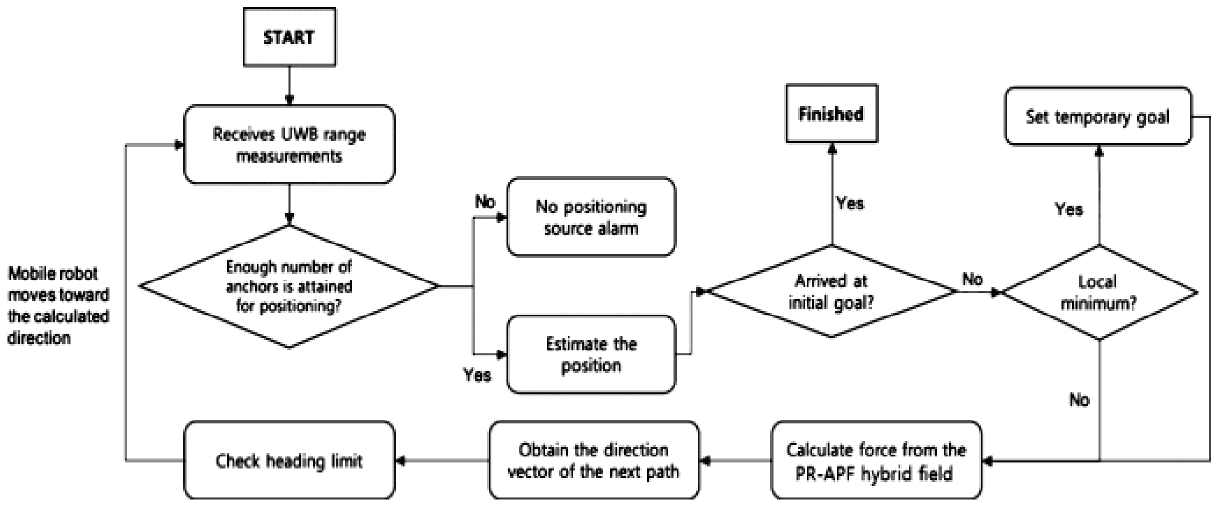
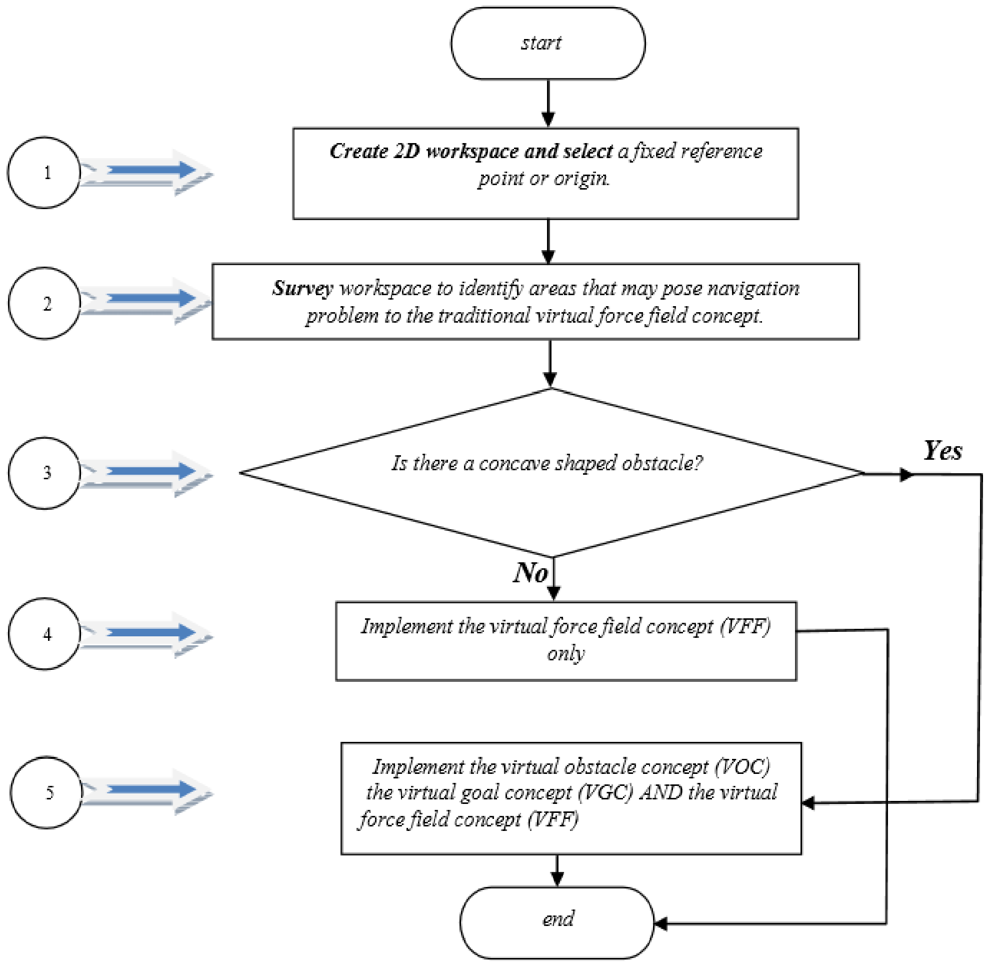

START
Receives UWB range measurements
Enough number of anchors is attained for positioning?
No
No positioning source alarm
Yes
Estimate the position
Finished
Yes
Arrived at initial goal?
No
Set temporary goal
Yes
Local minimum?
No
Mobile robot moves toward the calculated direction
Check heading limit
Obtain the direction vector of the next path
Calculate force from the PR-APF hybrid field

Figure 3: The flowchart of the PR-APF methodology, adapted from [37]

2.2.5 Hybrid Virtual Force Field (HVFF) Technique

A general new pattern in research work of portable robot optimisation, route planning, navigation, and obstacle avoidance is the hybrid approach. The explanation behind that is to accomplish preferable outcomes over-utilising the techniques independently. One of the hybrid approaches is known as the hybrid virtual force field (HVFF). This approach integrates the virtual obstacle concept (VOC), virtual goal concept (VGC), and VFF. The HVFF technique was pioneered by Olunloyo and Ayomoh to take care of the MR route issue for either a totally or halfway known static workplace of impediments [38].

In 2009, one of their research papers was focused on the overall issue of DOs in an obscure environment with extraordinary consideration given to the class of curved molded impediments and extensive impediments for which the VFF method is known to be inclined to LMTs. In their other paper, the HVFF technique was created and adjusted to LMT in SOs of the inward class is a somewhat known environment. Their findings validated that the HVFF technique is robust and versatile. Furthermore, it has the benefits of high productivity and viability as far as objective state achievement. Regardless of how much a workspace is bunched with impediments of various shapes or sizes, MR constrained by this methodology can generally arrive at the objective state without crashing into impediments given a possible way exists [39]. An outline of how the conventional VFF is combined with VGC and VOC is presented in Figure 4 below.



start
1
Create 2D workspace and select a fixed reference point or origin.
2
Survey workspace to identify areas that may pose navigation problem to the traditional virtual force field concept.
3
Is there a concave shaped obstacle?
Yes
No
4
Implement the virtual force field concept (VFF) only
5
Implement the virtual obstacle concept (VOC) the virtual goal concept (VGC) AND the virtual force field concept (VFF)
end

Figure 4: HVFF methodology flowchart, adapted from [38]

In 2010, Olunloyo and Ayomoh [40] further used the HVFF method to address the nearby minima robot route issue brought about by one or the other a curved or protracted impediment from a novel point of view. This was done through techniques that can empower a robot to check and arrange impediments into nearby or non-nearby minima causative impediments, stomach muscle initio, from the robot's underlying situation, preceding route. Such a methodology endeavors to impersonate human knowledge whereby on recognising a nearby minima causative obstacle an individual rather than continue into the trap, promptly suspends further route along its planned line of direction and reclassifies another way of route.

Moreover, in 2011[41] the HVFF technique was extended from the work presented in [40]. The extension incorporates route productivity concentrated between the current route plots ensuring papers on the constructed model and summed-up model conducted in a dynamic obstacle (DO) environment. The two figures (Figures 5 and 6) below show some of the obstacle sketches that were used in the paper ref [40]. The other characteristics of these obstacles' geometry are the lengths, height, and other features that can be extracted using sensor methods to identify the obstacles.

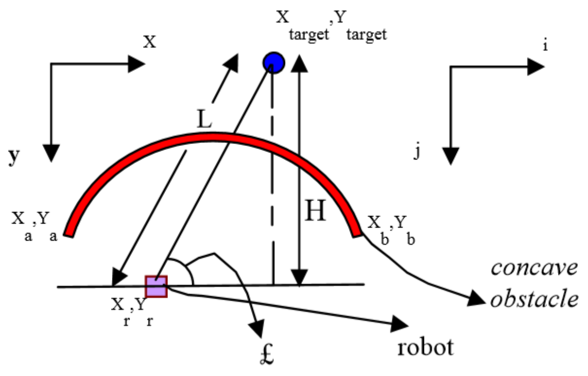


Figure 5: Concave obstacle sketch, adapted from [40]

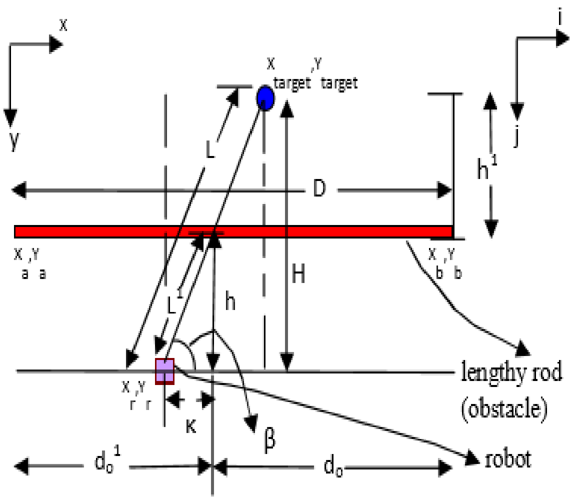


Figure 6: Straight obstacle sketch, adapted from [40]

2.2.6 Fuzzy Logic (FL) Technique

The FL technique was introduced by Zadeh [42] and extensively used in engineering. This technique plays a huge role in the domain of robotics. Many researchers have successfully used this technique to guide the MR in a certain environment. FL control is very appropriate for minimal effort portable robots that do not need extremely complex routes since FL is a mix of numerous types of rational estimations of the sources of info. The FL systems are motivated by human thinking, which is dependent on recognition. The FL technique that was behavior-based was designed by Qing-yong et al [43]. One of the behaviors included obstacle avoidance behavior in a MR. Jaradat et al [44] investigated a MR in a dynamic surrounding that had static and moving objects using a hybrid approach. They integrated FL with APF and the results found were simulated under a dynamic environment. Under the considered approach, it was seen from the simulated scenarios that the proposed approach had the alternative to give the robot a crash-free approach to carefully show up on the moving goal. One disadvantage of this technique is the LMT, where the robot was caught in a position sitting tight for a hindrance or the objective to change their positions which does not happen in static conditions. Moreover, Pandey et al [45] developed an FL technique for taking care of the movement arranging issue of a portable robot in the presence of various states of SOs to discover crash freeway. The outcomes showed that the proposed technique empowers the portable MR to securely arrive at the objective without impacting. In the future, the current technique can be improved by streamlining with the assistance of optimisation algorithms.

In 2016, Almasri et al [46] created crash evasion and line train procedures for portable robot routes in dynamic and static conditions with the joining of FL combination. The crash shirking method utilised vicinity sensors to distinguish SOs and DOs. The proposed technique was effectively tried in the Webots Pro test system and progressively explored. Furthermore, Singh and Thongam [47] investigated the issue of OA in the route of the portable robot by building up an advanced FL deduction framework. In their research energy was diminished by decreasing the trip time to arrive at the objective. The reliability, verification, and novelty of the utilised technique were simulated in the FL tool kit of MATLAB.

Additionally, in 2019 Batti et al [48] extended the use of FL for OA in labyrinth workspace. The future work would consider combining this approach with algorithms like a genetic algorithm (GA) or neural network (NN) to produce better results. Mohanty et al [49] proposed a new model called Takagi-Sugeno (T-S) FL to address the issue of route planning. The model was validated in simulation and real-time experiments. Figure 7 below shows the navigation space consisting of various obstacles considered in their study. Future work will focus on dynamic conditions using multiple robots.

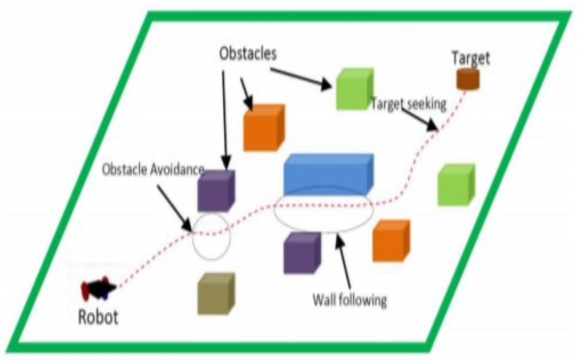


Figure 7: Navigation Space, adapted from [49]

Recently, Oleiwi et al [50] research paper addressed the MR navigation and route planning problem. They addressed the improvement of the FL technique by applying it in a complex environment that involves more than two DOs. The environment is considered unknown and known DOs. The results obtained through simulations outline that newly improved FL overcame all limitations that were presented prior by other researchers. Hence, future work will consider the application in drones.

2.2.7 Neural Network (NN) Technique

The NN is a huge plan of equivalent spread planning segments (neurons) related to graph geology. Learning in the neural association can be controlled or independent. Coordinated learning vocations portrayed plan information, while independent learning uses simply the least information without pre-classification. Solo learning counts offer less computational multifaceted nature and less precision than directed learning estimations. The neural association could convey the data irrefutably in the loads, resulting in learning. The NN had filled quickly in the domain of recognising objects and obstacle discovery in a picture. Recently, the issue of recognising obstacles in the robot navigation system is important [51]. The most popular methodology used to solve this problem in the past years has been convolutional neural networks (CNN) [52, 53].

In 2011, Chi and Lee [54] in their paper various principles were actualised for the control technique to keep away from the obstacle effectively. The proposed framework with the NN control approach has illustrated the adequacy of dodging the obstacles and robots can explore through the direction with dependability and unwavering quality. In the future, to accomplish better reaction from the proposed NN approach, it needs further exploratory examination to have better information preparation. The more substantial prepared information, the better the NN framework to have.

In 2014, Motlagh et al [55] have proposed avoiding obstacles and the objective of looking for practices utilising NN. Janglova [56] addressed the issue of robot guidance using NN amidst objects. The research looked at a way of arranging and canny control of a self-governing robot that should move securely in a mostly organised workplace. This environment included quite a few obstacles of self-assertive shape and size. The results were simulated, and the researcher concluded that the NN technique is usable by and large for the movement of the robot in a self-assertive workplace. In the future, this strategy will be looked at for the safe movement of our trial versatile vehicle in indoor conditions.

Moreover, Yu et al [57] addressed the area of an arrangement control issue with impact and OA for multi-robot frameworks within the sight of model vulnerabilities and outside unsettling influences. Although the NN has numerous benefits, the determination of the number of shrouded hubs in the NN should be controlled by the calculations. The assurance of the middle estimation of the concealed layer hub requires a further report. Further exploration work will principally focus on the ideal control issue of the various automated frameworks dependent on the NN.

Additionally, in 2020 Saleem et al [58] roused by the benefits of the various levels including extraction of profound learning, their work examined the improvement of a CNN calculation to tackle the issue of the portable robot OA in an indoor climate. The eventual outcomes showed that the precision can be improved by remembering the MR direction for the dataset, expanding the size of information, and tuning the network's hyperparameters. The CNN calculation has indicated the incredible potential to get highway order exactness for hindrance evasion for portable robots.

Furthermore, Wei and Ye [59] proposed an OA framework dependent on GA-supported OIF-Elman NN. Figure 8 shows the architecture of the OIF-Elman NN.

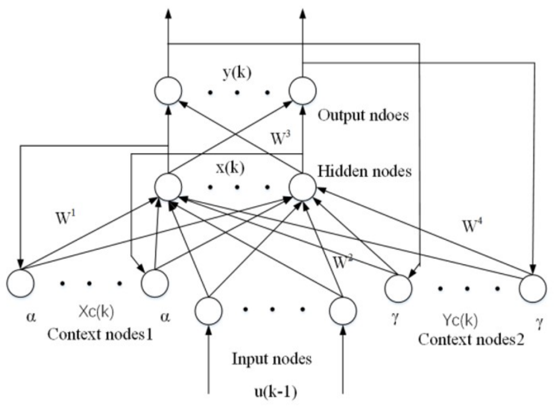


Figure 8: OIF-Elman structure network, adapted from [59]

The framework can manage versatile robots to finish the development and impediment evasion in the workspace with deterrents. Considering the information gathered by the MR's six infrared sensors, the framework changes its heading and changes the MR's movement at the following second. The test results demonstrated that the framework planned by the GA-helped OIF-Elman network is more successful for OA. Another paper by Zhang et al [60] focused on the improvement of NN for RN in complex environments.

2.2.8 Particle Swarm Optimisation (PSO) Technique

The PSO is broadly utilised in the field of versatile robot routes tending to the planning and confinement issues of portable robot routes in the obscure workplace [61]. The utilisation of PSO assists with limiting the count and holds more steady intermingling attributes. In 2015, the examination of different methodologies was introduced and the results showed that the FL matched with PSO provides the ideal outcomes in separation voyaged [62]. The uses of PSO are not restricted to versatile robot routes just in the guard area. Atyabi et al [63] introduced an extension of the PSO technique in robotics to improve the performance of this method. Their research considered the environment with SOs and DOs. The technique showed potential under the conditions that were considered; however, the method cannot be deployed in the whole domain of robotics. The future work would examine the effectiveness of this method under real-world applications with mobile robots.

Furthermore, one of the research papers presented another PSO technique, named the PSO-IAC technique [64]. This algorithm was developed to determine the objective of coming to terms with the hindrance evasion issue for a 6- degrees of freedom (DOF) controller of the home assistance robot. The suggested PSO-IAC calculation coordinates the improved versatile idleness weight and the tightening factor with the standard PSO. Both the free-space and obstruction shirking states were set up for assessments in real-time and simulations examinations. Simulation outcomes demonstrated that the PSO-IAC calculation gives the quickest combination capacity. Lastly, the suggested control plan can cause the controller of the home assistance robot to show up at the objective situation with and without impediments in all continuous trials.

In 2018, Meerza et al [65] built up a PSO-based robot way arranging calculation that has an impact shirking capacity for SOs and DOs. In the future, they will test their proposed calculation in a certifiable workplace. Their point was to fuse a few profound fortifications figuring out how to accomplish more perplexing swarm conduct. Additionally, Alaliyat et al [66] in their paper, proposed a powerful way of arranging calculations dependent on PSO, ready to manage the dynamic complex workplace. The outcomes indicated that, without any earlier information on the workplace, the robot can accomplish its objective evading SOs and DOs. The robot moves easily and does not stall out even in complex dynamics. They intend to stretch out this work to acquire a keen robot that can learn and retain the circumstances during its route through the workplace. Moreover, they plan to improve the proficiency in anticipating the speed and course of DOs. Figure 9 shows one of the navigation setups considered for their simulations.

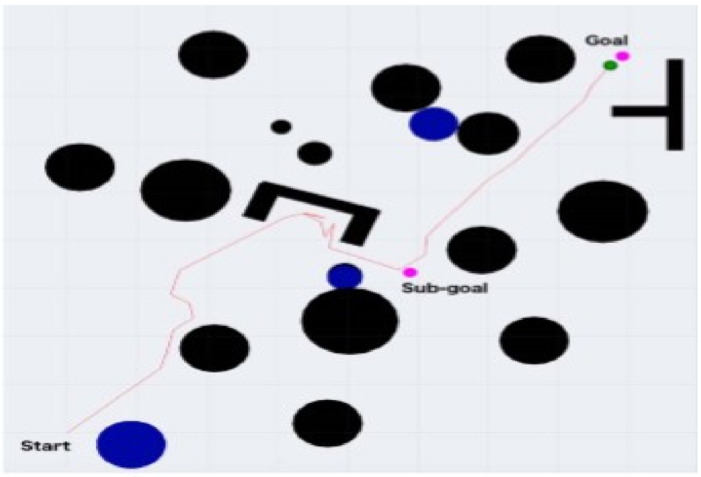


Figure 9: Complex navigation space with obstacles, adapted from [66]

In 2020, Tian et al [67] in their paper embraced remote sensor organisation to find robots and impediments. The technique proposed utilised an improved counterfeit clever calculation to design way. Their simulations used multiple robots. The limitations to the proposed method are the calculation of the union speed to improve the worldwide pursuit execution and failure to manage the circumstance that numerous robots may collide. In the future, hypothetical exploration of PSO calculation and obstruction evasion calculation to manage different testing improvement issues will be looked at.

2.2.9 Genetic Algorithm (GA) Technique

This is a well-known technique-based enhancement instrument that follows the guideline of hereditary qualities and common determination found first in 1958 [68]. The application to the field of software engineering was introduced first in 1975 [69]. The utilisation of GA for the versatile robot route issue has been given to a static workplace. The investigation is introduced by reproduction results as they were within the sight of a polygonal impediment. This strategy is embraced by Xiao et al [70] to accomplish the objective of the route, for example, way length, way perfection, and OA. Many of the scientists have given routes in a static workplace simply by utilising GA yet the route within the sight of a moving impediment in an uncertain workplace [71]. To improve results in robot way arranging, numerous scientists have joined the use of GA along with another shrewd calculation to get a mixture approach [72]. Patle et al [73] state that in the future the work may stretch out to cause the crossbreed regulator and it to be tried for the ongoing open-air workplace. The execution of the suggested regulator for the submerged condition can be checked. It might likewise apply to the improvement of the self-ruling vehicle for the different ecological conditions.

In 2018, Germi et al [74] paper tended to be an alteration to the first potential field calculation to better the exhibition of the calculation in dynamic conditions. The change depended on adding deterrent elements as a term to the shock field. The adaptiveness of the GA stems from changing the proportion of the populace created by each strategy. Additionally, Choueiry et al [75] research paper introduced a survey of the path planning enhancement issue and a calculation for robot way arranging in a static environment using GA as a device. The motivation behind the calculation was to discover the quickest course in each number of steps while staying away from obstacles in the space. The calculation's exhibition was upgraded by consequently overlooking all recommended courses that cross the limits of the workplace. For enhancement and search issues, GA was utilised as a hunting instrument in figuring to discover precise or a surmised arrangement. Figure 10 show the flowchart of the proposed approach.

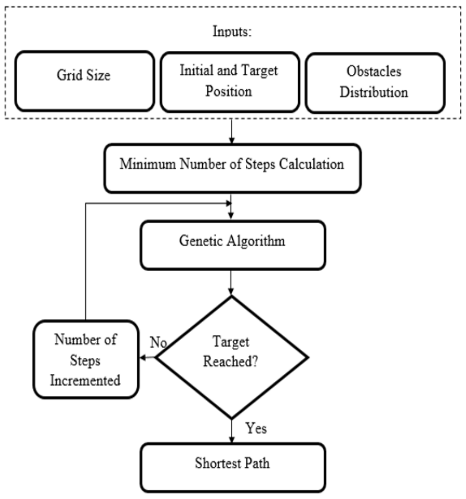

Inputs:
Grid Size
Initial and Target Position
Obstacles Distribution
Minimum Number of Steps Calculation
Genetic Algorithm
Target Reached?
No
Number of Steps Incremented
Yes
Shortest Path

Figure 10: Approach Flowchart, adapted from [75]

Furthermore, Lopez-Gonzalez et al [76] utilised GA to accomplish distance-based development, with the usage of two unique sorts of chromosomes, one for the distance arrangement, and the other for impact evasion. The proposed answer for this was the utilisation of the consolidated preparing force of all robots in a parallel GA that relocates potential arrangements in request to diminish preparation time and accomplish agreement between the robots to an objective.

Additionally, in 2020 Aghda and Mirfakhrae [77] consolidated the GA-FL technique that was utilised to improve directing. The explanation behind applying the fuzzy element was the vulnerability of the information data and the avoidance of experiencing obstructions. To exploit the two strategies together, the joined GA-FL technique was utilised because in FL, the ideal nearby arrangement was found, and the GA was utilised to manage it.

2.2.10 Ant Colony Optimisation (ACO) Technique

The ACO technique is now applied to different domains of science and designing, for example, workshop booking, vehicle steering, quadratic task issue, mobile sales rep issues, diagram shading, and some more. The ACO is a technique that is applied in the field of the robot system, in particular the path planning of mobile robots [78]. To determine the automated flying vehicle course issue for a war zone, the ACO calculation has been introduced to resolve this issue [79]. The ACO technique is used for deterrent avoidance and course in exceptional conditions. The basic ACO for portable robot way arranging exists numerous issues, for example, absence of steadiness calculation, untimely convergence, more difficulty to track down an ideal answer for complex issues, etc. In 2011, Zhangqi et al [80] proposed improvement measures. Their research work applied GA to the advancement and arrangement boundaries of the essential ACO. The simulation outcomes showed that the improved ideal way length is essentially not exactly the fundamental ACO and instability is more modest, steadiness essentially improves.

In 2018, Wang et al [81] in their paper the APF calculation was improved first, and the strategy for piecewise capacity of fascination potential was proposed to take care of the issue that the robot can without much of a stretch slam into the obstruction when the length between a robot and an objective point is huge. They too adjusted the shocking likely capacity to tackle the wavering issue. The improved APF calculation and ACO are joined as seen in the flowchart in Figure 11. On the premise of finding the ideal or imperfect way, the assembly speed of ACO is improved. The limitation of this approach is that the model contains numerous boundaries which make it difficult to tune. They will discover the relations between these boundaries in future work and attempt to tune these boundaries by calculations.

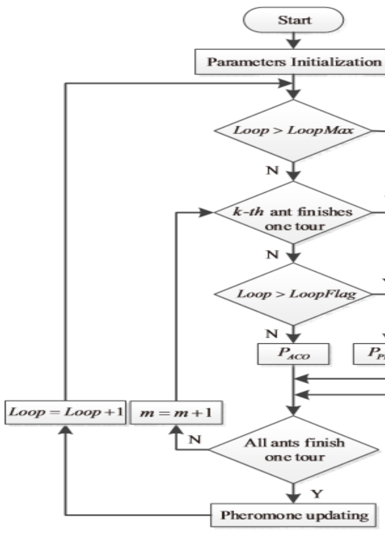


Figure 11: Improved APF combined with ACO flowchart, adapted from [81]

Moreover, in 2019 Yi et al [82] paper produced dynamic change data as indicated by the contrast between the best way of the past age and the best way of the current cycle in ACO. In 2020, Ma et al [83] address the automated submerged vehicle two-dimensional independent way arranging issue in the climate influenced by sea momentum and obstacles. The paper applied a better fireworks-ant colony hybrid algorithm (F-ACHA). Trial outcomes showed that this calculation can rapidly locate the global ideal arrangement, and the more unpredictable the workplace. The calculation proposed in this study gave another approach to the self-sufficient way of arranging submerged vehicles.

Additionally, Zhao [84] in his paper the ideal way of anticipating robots dependent on ACO was proposed by contemplating the connected writing and significant methods of robot way arranging in China and abroad. The paper primarily thinks that under the condition of detecting the current workplace, the robot powerfully keeps away from the hindrances and meets the necessities of the least time. Based on presenting the robot way of arranging the issue, the numerical model was set up for the ACO. The exploratory outcomes showed that the model can wisely pick a way for the robot with DO evasion, having the most limited time and more limited distance, and has a specific commitment to the smart advancement for the robot.

2.2.11 Firefly Algorithm (FA) Technique

The FA technique was introduced by Yang in 2008 [5]. This algorithm can also be called the metaheuristics algorithm and the idea comes from fireflies flashing behavior. In 2015, the problem of robot navigation using FA in the domain of SOs was addressed by Paniagua et al [85]. Three objectives were met in their paper which were route path, route smoothness, and route length. The future work stated that the work will be extended in the domain of a dynamic work environment. This problem was also addressed by Brand and Xiao-Hua [86] for a free collision path using FA in a simulation platform environment. Other researchers considered solving the underwater robot navigation problem [87, 88]. Moreover, many researchers have addressed the robot route planning and navigation problem through various hybrid algorithms that incorporate FA [89-91]. The hybrid algorithms are growing more and are considered since the individual technique does not guarantee the best solutions in some unstructured spaces. For dynamic conditions under the analysis of FA, it was addressed by Patle et al [92]. In 2020, Li et al [93] extended research on FA in a static environment.

2.2.12 Graphical Techniques

Over the years researchers have developed graphical techniques to also address the issue of RN in workplaces. These techniques are limited to static workplaces. Recently, a paper by Chi et al [94] used one of the graphical methods known as the Voronoi diagram. They addressed the issue of PP in complex workplace obstacles that are moving and static. The experiments were successfully run in real-time and the technique was robust. Another known method is cell decomposition, it was best used by Wahyunggoro and Cahyadi [95] in a hybrid approach with the FL method to improve the duration it takes to reach the TP. Similarly, Zhou and Liu [96] combined a roadmap approach with SLAM to address the issue of navigation for MRs. Their future work will focus on optimising this approach to increase accuracy. Some of the other techniques that exist in the domain of graphical methods are transformed space, oct trees vgraph, and configuration space.

2.2.13 Vision-Based, Transient Virtual Obstacles (TVO) and Other Techniques

Aggarwal et al [97] in their paper looked at OA using vision-based techniques by graphing virtual impediments. This technique is based on the intensity of each pixel in an image captured by a camera. The intensity value of each pixel is used to classify it as terrain. The pixels with the highest brightness are classified as terrain, while other pixels are classified as obstacles. Black dots indicated obstacles in simulations. The disadvantage of this technique is that the robot shows abnormal behavior, such as detecting false alarms when it was tested.

In 2019, Ravankar et al [98] presented a TVO technique in presence of VOs to change the methods of the robots for safe navigation. Their current implementation needs manually adding the VOs within the map. In the future, they are going to modify the rule to mechanically decide the addition and removal of VOs exploitation knowledge from totally different sensors. Similarly, they extended this work in 2020 [99] looking at temporary VOs. The main advantage of their proposed method is that there is no need to change the route planner when adding and removing VOs on the map. This allows powerful functions to prevent or guide the robot through specific passages without using real obstacles, changing the route planner, or reprogramming it.

Other additional techniques include the tangent bug method was implemented by Yousuf and Kadri [100] in their research, which included concave and convex-shaped obstacles. A study by M et al [101] used a reinforcement learning technique to address navigation issues in MRs. Another approach called lifelong learning was used for RN by Liu et al [102] and Xie et al [103] used a stochastic approach for RN. Other reactive methodologies like bacterial forging optimisation, artificial bee colony, shuffled frog leaping, cuckoo search, not included in this paper can be seen in the paper ref [5].

The above sections looked at previous research on methodologies that were used to address the issue of MR navigation, PP, and optimisation in robotics. Table 2 below summarises these methodologies and the taxonomy breakdown is as follows: Environment consisting of what type of obstacles, show if single or hybrid approach, results simulated or not, the year, geometry of obstacles, target point and robot if it was one or multiple. Obstacles can be VOs, SOs and DOs.

Table 2: Analysis of various path planning and navigation algorithms amidst obstacles

Cv =Concave, Cx = Convex, SR = Simulation Result, RTR= Real Time Result

Ref No	Techniques	Environment consists of			Technique used as		Result		Year	Obstacle(s) Shape		Target Point (TP)		Robot	
		SOs	DOs	VOs	Stand Alone	Hybrid	SR	RTR		Cv	Cx	Single TP	Multi TP	Single Robot	Multi Robot
Classical Approach															
[8]	SLAM	Yes	No	No	Yes	No	No	Yes	2014	Yes	Yes	Yes	No	Yes	No
[9]		No	Yes	No	No	Yes	Yes	No	2014	No	Yes	Yes	No	Yes	No
[10]		Yes	Yes	No	Yes	No	No	Yes	2016	Yes	Yes	Yes	No	Yes	No
[11]		Yes	Yes	No	No	Yes	No	Yes	2018	Yes	Yes	Yes	No	Yes	No
[12]		Yes	Yes	No	No	Yes	No	Yes	2018	Yes	Yes	Yes	No	Yes	No
[13]		Yes	Yes	No	No	Yes	Yes	No	2021	Yes	Yes	Yes	No	Yes	No
[15]	Light Detection and Ranging (LiDAR)	Yes	No	No	Yes	No	Yes	No	2017	No	Yes	Yes	No	Yes	No
[16]		Yes	No	No	Yes	No	Yes	No	2019	No	Yes	Yes	No	Yes	No
[17]		Yes	No	No	Yes	No	No	Yes	2019	Yes	Yes	Yes	No	Yes	No
[18]		Yes	Yes	No	Yes	No	Yes	Yes	2020	Yes	Yes	Yes	No	Yes	No
[19]		Yes	Yes	No	Yes	Yes	No	Yes	2020	Yes	Yes	Yes	No	Yes	No
[20]	Vector Field Histogram (VFH)	Yes	No	No	Yes	No	No	Yes	1991	No	Yes	Yes	No	Yes	No
[21]		Yes	No	No	Yes	No	No	Yes	1998	No	Yes	Yes	No	Yes	No
[22]		Yes	No	No	Yes	No	Yes	Yes	2000	No	Yes	Yes	No	Yes	No
[23]		No	Yes	No	Yes	No	No	Yes	2012	No	Yes	Yes	No	Yes	No
[24]		Yes	No	No	Yes	No	Yes	No	2014	No	Yes	Yes	No	Yes	No
[25]		Yes	Yes	No	Yes	No	No	Yes	2016	Yes	Yes	Yes	No	Yes	No
[26]		Yes	Yes	No	Yes	No	Yes	No	2019	No	Yes	Yes	No	Yes	No
[27]		Yes	Yes	No	Yes	No	No	Yes	2020	No	Yes	Yes	No	No	Yes
[28]	Artificial Potential Field (APF)/ Virtual Force Field (VFF)	Yes	No	No	Yes	No	Yes	No	2014	No	Yes	Yes	No	Yes	No
[29]		Yes	No	No	Yes	No	No	Yes	1989	Yes	No	Yes	No	Yes	No
[30]		Yes	No	No	Yes	No	No	Yes	1985	Yes	Yes	Yes	No	Yes	No
[31]		Yes	Yes	No	No	Yes	Yes	No	2015	Yes	Yes	Yes	No	Yes	No
[32]		No	Yes	No	No	Yes	Yes	No	2017	Yes	Yes	Yes	No	Yes	No
[33]		Yes	No	No	Yes	No	Yes	No	2018	Yes	No	Yes	No	Yes	No
[34]		Yes	No	No	Yes	No	Yes	No	2020	No	Yes	Yes	No	Yes	No
[35]		Yes	No	No	Yes	No	Yes	No	2020	Yes	Yes	Yes	No	Yes	No
[36]		Yes	Yes	No	Yes	No	Yes	Yes	2019	No	Yes	Yes	No	Yes	No
[37]		Yes	No	No	No	Yes	Yes	Yes	2021	Yes	Yes	Yes	No	Yes	No

Table 2: Continued

Ref No	Techniques	Environment consists of			Technique used as		Result		Year	Obstacle(s) Shape		Target Point (TP)		Robot	
		SOs	DOs	VOs	Stand Alone	Hybrid	SR	RTR		Cv	Cx	Single TP	Multi TP	Single Robot	Multi Robot
Classical Approach															
[38]	Hybrid Virtual Force Field (HVFF)	Yes	No	Yes	No	Yes	Yes	No	2009	Yes	Yes	Yes	No	Yes	No
[39]		No	Yes	Yes	No	Yes	Yes	No	2009	Yes	Yes	Yes	No	Yes	No
[40]		No	Yes	No	No	Yes	Yes	No	2010	Yes	Yes	Yes	No	Yes	No
[41]		Yes	No	No	No	Yes	Yes	No	2011	Yes	Yes	Yes	No	Yes	No
Heuristic Approach															
[43]	Fuzzy Logic (FL)	Yes	No	No	Yes	No	Yes	No	2009	Yes	Yes	Yes	No	Yes	No
[44]		Yes	Yes	No	No	Yes	Yes	No	2012	No	Yes	Yes	No	Yes	No
[45]		Yes	Yes	No	Yes	No	Yes	No	2014	No	Yes	Yes	No	Yes	No
[46]		Yes	Yes	No	Yes	No	Yes	Yes	2016	Yes	Yes	Yes	No	No	Yes
[47]		Yes	No	No	Yes	No	Yes	Yes	2018	No	Yes	Yes	No	Yes	No
[48]		Yes	No	No	Yes	No	Yes	No	2019	Yes	Yes	Yes	No	Yes	No
[49]		Yes	No	No	Yes	No	Yes	Yes	2020	Yes	Yes	Yes	No	Yes	No
[50]		Yes	Yes	No	Yes	No	Yes	No	2021	No	Yes	Yes	Yes	Yes	Yes
[54]	Neural Network (NN)	Yes	No	No	Yes	No	No	Yes	2011	No	Yes	Yes	No	Yes	No
[55]		Yes	No	No	Yes	No	Yes	No	2014	No	Yes	Yes	No	Yes	No
[56]		Yes	No	No	Yes	No	Yes	No	2004	Yes	Yes	Yes	No	Yes	No
[57]		Yes	Yes	No	Yes	No	Yes	No	2019	No	Yes	Yes	No	No	Yes
[58]		Yes	No	No	Yes	No	No	Yes	2020	Yes	Yes	Yes	No	Yes	No
[59]		Yes	No	No	No	Yes	Yes	No	2020	Yes	Yes	Yes	No	Yes	No
[60]		Yes	Yes	No	Yes	No	Yes	No	2020	Yes	Yes	Yes	No	Yes	No
[63]	Particle Swarm Optimisatio n (PSO)	Yes	Yes	No	Yes	No	Yes	No	2010	No	Yes	Yes	No	Yes	No
[64]		Yes	Yes	No	Yes	No	Yes	No	2016	No	Yes	Yes	No	Yes	No
[65]		Yes	Yes	No	Yes	No	Yes	No	2018	Yes	Yes	Yes	No	Yes	No
[66]		Yes	Yes	No	Yes	No	Yes	No	2019	Yes	Yes	Yes	No	Yes	No
[67]		Yes	Yes	No	Yes	No	Yes	No	2021	Yes	Yes	No	Yes	No	Yes
[74]	Genetic Algorithm (GA)	Yes	Yes	No	Yes	No	Yes	Yes	2018	No	Yes	Yes	No	Yes	No
[75]		Yes	No	No	Yes	No	Yes	No	2019	Yes	Yes	Yes	No	Yes	No
[76]		No	Yes	No	Yes	No	Yes	Yes	2020	No	Yes	No	Yes	No	Yes
[77]		Yes	Yes	No	No	Yes	Yes	No	2020	No	Yes	Yes	No	Yes	No

Table 2: Continued

Ref No	Techniques	Environment consists of			Technique used as		Result		Year	Obstacle(s) Shape		Target Point (TP)		Robot	
		SOs	DOs	VOs	Stand Alone	Hybrid	SR	RTR		Cv	Cx	Single TP	Multi TP	Single Robot	Multi Robot
Heuristic Approach															
[80]	Ant Colony Optimisation (ACO)	Yes	No	No	Yes	No	Yes	No	2011	Yes	Yes	Yes	No	Yes	No
[81]		Yes	No	No	No	Yes	Yes	No	2018	Yes	Yes	Yes	No	Yes	No
[82]		Yes	No	No	Yes	No	Yes	No	2019	Yes	Yes	Yes	No	Yes	No
[83]		Yes	Yes	No	No	Yes	Yes	No	2020	No	Yes	Yes	No	Yes	No
[84]		Yes	Yes	No	Yes	No	Yes	No	2020	Yes	Yes	Yes	No	Yes	No
[85]	Firefly Algorithm (FA)	Yes	No	No	Yes	No	Yes	No	2015	Yes	Yes	Yes	No	Yes	No
[86]		Yes	No	No	Yes	No	Yes	No	2013	No	Yes	Yes	No	Yes	No
[87]		Yes	No	No	Yes	No	Yes	Yes	2015	No	Yes	No	Yes	No	Yes
[88]		Yes	No	No	Yes	No	Yes	Yes	2013	No	Yes	No	Yes	No	Yes
[89]		Yes	No	No	No	Yes	Yes	Yes	2015	No	No	Yes	No	Yes	No
[90]		Yes	No	No	No	Yes	Yes	No	2018	No	No	Yes	No	Yes	No
[91]		Yes	Yes	No	No	Yes	Yes	Yes	2019	Yes	Yes	Yes	Yes	Yes	Yes
[92]		Yes	Yes	No	Yes	No	Yes	Yes	2018	Yes	Yes	Yes	No	Yes	Yes
[93]		Yes	No	No	Yes	No	Yes	No	2020	No	Yes	Yes	No	Yes	No
Graphical Approach															
[94]	Graphical	Yes	Yes	No	Yes	No	Yes	No	2021	Yes	Yes	Yes	No	Yes	No
[95]		Yes	Yes	No	No	Yes	Yes	No	2016	No	Yes	Yes	No	Yes	No
[96]		Yes	No	No	No	Yes	No	Yes	2019	No	Yes	Yes	No	Yes	No
Other Approaches															
[97]	Vision Based	Yes	No	Yes	Yes	No	No	Yes	2010	No	Yes	Yes	No	Yes	No
[98]	TVO	No	No	Yes	Yes	No	No	Yes	2019	No	Yes	Yes	No	Yes	No
[99]		Yes	No	Yes	Yes	No	No	Yes	2020	Yes	Yes	Yes	No	Yes	No
[100]	Other Techniques	Yes	No	No	Yes	No	Yes	No	2020	Yes	Yes	Yes	No	Yes	No
[101]		Yes	No	No	Yes	No	Yes	No	2019	No	Yes	Yes	No	Yes	No
[102]		Yes	No	No	Yes	No	Yes	Yes	2021	No	Yes	Yes	No	Yes	No
[103]		Yes	No	No	Yes	No	No	Yes	2021	Yes	Yes	No	Yes	Yes	No

2.3 Alternative Solutions

The methodologies in Table 3 have been used to solve guidance and optimisation issues in mobile robots amidst obstacles. These techniques solved the problems closely related to the current problem of this research. Table 3 below gives the limitations and strengths of some of these methodologies reviewed in the literature.

Table 3: Methodologies strengths and limitations

Methodologies	Strengths	Limitations
NN	• Helps in learning capabilities. • Real time experiments as well as simulations can be conducted	• Slow convergence speed • Complexity increases with layers.
HVFF	• High productivity and viability • Robust and Versatile	
APF/VFF	• Robust • Efficient and Versatile	• Local trap complexities
ACO	• Easy to implement. • Produce good outcomes in simulation. • Can be easily utilised for hybrid methodologies. • Require minimal control parameters	• Convergence is slow
PSO	• Easy to implement. • Produce good outcomes in simulation. • Faster convergence	• Performance analysis is complicated
GA	• Produce good results when integrated with other methodologies. • Great capability in terms optimisation • Produce good outcomes in simulation	• Incompetent in a dynamic environment • Local minima problems cause oscillations in the framework

From the review the researchers have generally utilised delicate processing methods when contrasted with hard registering; that is deterministic, non-deterministic, and evolutionary algorithms for MR route, optimisation, and OA. The disadvantages and advantages of these techniques were also looked at. The methods were looked at from an efficiency perspective and complexity level. For this research, not all methods will be considered because of the drawbacks they each have based on complexity and efficiency.

2.4 Preferred Solution

The hybrid approach is the preferred solution to achieve the objectives of this research. The chosen methodology from the literature review is a hybrid approach that adopts the concept HVFF algorithm [38-41] which integrates VGC, VFF and, VOC techniques. This approach minimises the drawbacks that individual techniques have while addressing the route and OA problem for MRs. This approach can achieve multiple objectives and is less costly. From the review in section 2.2, the key findings by researchers is that the hybrid approach is robust, effective, and tends to give better results.

2.5 Chapter Summary

In this chapter research papers were reviewed. The literature review looked at methodologies that solved problems closely related to that of this research. An outline of PP procedures for self-sufficient MRs, the benefits, and faults of these strategies were introduced and examined momentarily. An exhaustive conversation of each approach in this expansive research field of PP for MRs has been shown. An intriguing perspective is that the course of this research is despite the significant improvement in the area over recent many years, limited research has been accounted for particularly in multi-robotic frameworks. Most results were captured through simulations more than the real-world. Moreover, most research in MR navigation was conducted more in static environments. A large portion of the papers manages the subject of single advanced robotics frameworks while leaving a wide assortment of regions in composed and organised multi-robotic frameworks that are yet open for future works. In conclusion, from the literature review the key findings by researchers is that the hybrid approach is robust, effective, and tends to give better results. This approach minimises the drawbacks that individual deterministic and stochastic techniques have while addressing the PP, navigation, and OA problem for MRs. The approach is capable to produce better results for the problem at hand and can achieve multiple objectives. This study is an extension of the work of Olunloyo and Ayomoh [39], which incorporate new SOs. The basis of introducing the new obstacles will be in relation to the real-life obstacle problems.

CHAPTER THREE

RESEARCH APPROACH

3.1 Chapter Overview

This chapter presents the methodology deployed in this research. The chapter is divided into three sections viz: the conceptual framework which presents a comprehensive research roadmap for this dissertation, the theoretical framework which extends the conceptual framework via a detailed generic illustration of the governing theories associated with each of the identified research concepts and finally, the development of specific models in respect of the problem domain earlier articulated with the aid of modeling procedures as depicted in the theoretical framework. The chosen research methodology is premised on findings from the literature as presented in section 2.4 in chapter two. The methodology deployed in this research as presented in this chapter, is the HVFF concept for efficient robot vehicle navigation and control. The HVFF [38-41] is a hybrid approach that integrates the VGC, VFF and VOC techniques. This approach, minimises the drawbacks that most isolated individually operated robot navigation methodologies often encounter in their bid to navigate along an optimal path towards a desired target point. This algorithmic challenge is brought to the fore while the robot, apart from trying to sustain itself on the optimal trajectory is also trying to simultaneously avoid colliding with the workspace obstacles along its trajectory.

3.2 Conceptual Framework

The research design as contained in the conceptual framework presented in figure 12, depicts a generalised flow of work in this research from the initial to the concluding phase. Figure 12, Point 1 defines the problem and set the objectives of this research. This research focus on the mobile robot (MR) navigation problem with the aim of developing an optimum navigation algorithm. Point 2, gives an exploration and applications on a scope of subjects of significance for the path of MRs amidst obstacles. Moreover, Point 3 focus on formulating mathematical models that make up the methodology chosen for this research. The outlook of this methodology is given in Figure 13. The mathematical models are to aid with developing a solution algorithm for this research. Additionally, Point 4 will focus on validating the developed solution algorithm using the python software. Various cases with different workspaces will be simulated. The simulation results will be analysed by comparing some results with past research results. Point 5 will give research findings and make recommendations about future work of this research. Lastly, the methodological outlook is outlined in Figure 13, it is divided into three sub problem blocks which connect with the research objectives and methods.

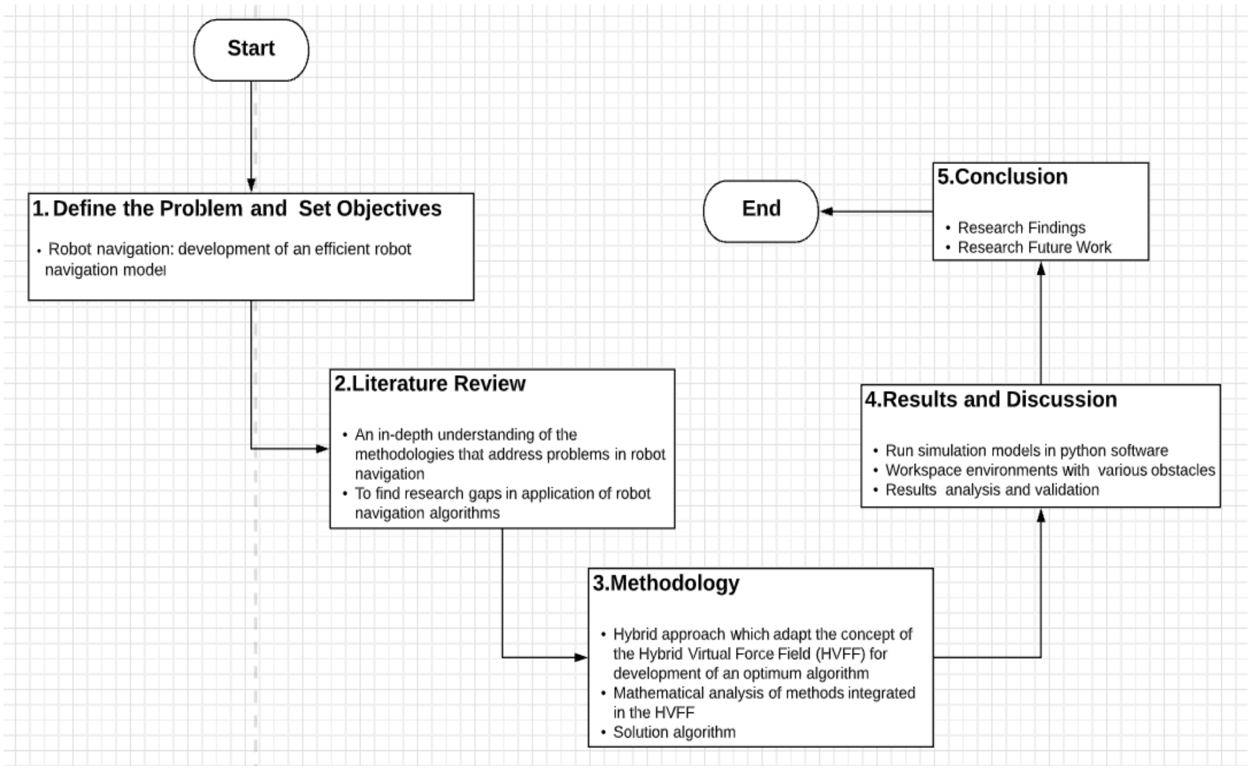

Start
1. Define the Problem and Set Objectives
• Robot navigation: development of an efficient robot navigation model
2.Literature Review
• An in-depth understanding of the methodologies that address problems in robot navigation
• To find research gaps in application of robot navigation algorithms
3.Methodology
• Hybrid approach which adapt the concept of the Hybrid Virtual Force Field (HVFF) for development of an optimum algorithm
• Mathematical analysis of methods integrated in the HVFF
• Solution algorithm
4.Results and Discussion
• Run simulation models in python software
• Workspace environments with various obstacles
• Results analysis and validation
5.Conclusion
• Research Findings
• Research Future Work
End

Figure 12: Conceptual framework: Research Comprehensive Outlook

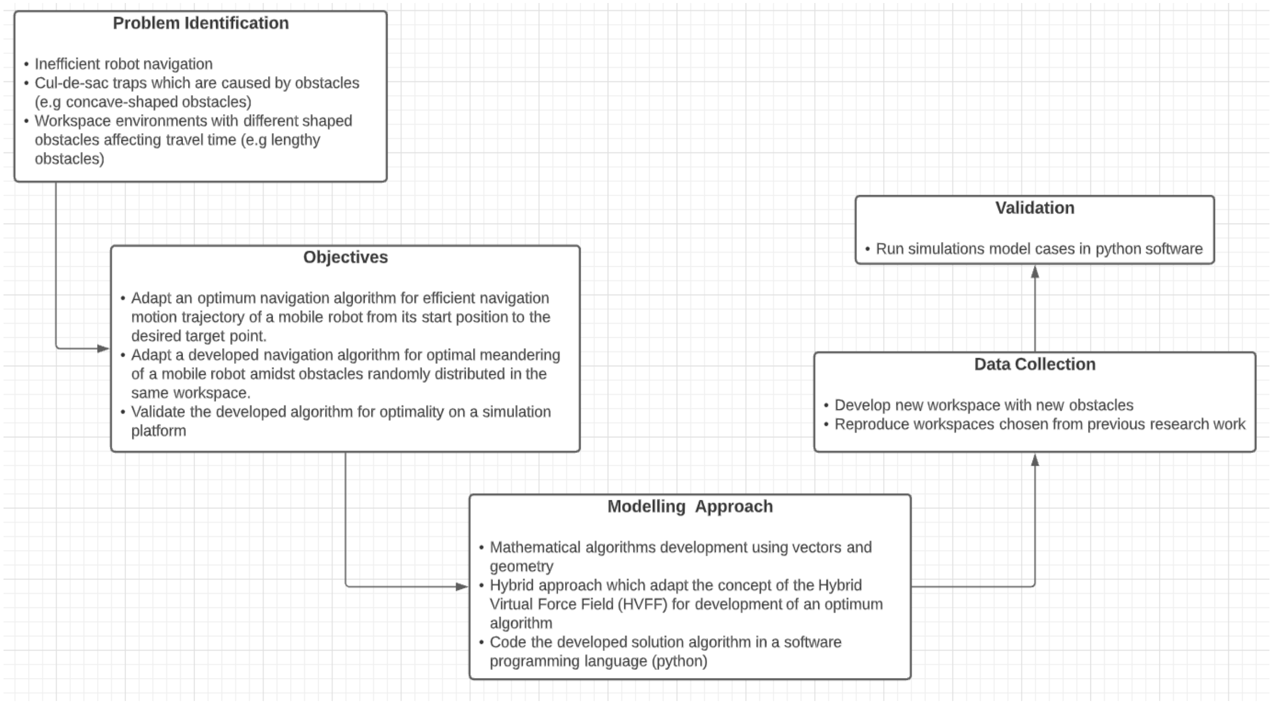

Problem Identification
• Inefficient robot navigation
• Cul-de-sac traps which are caused by obstacles (e.g concave-shaped obstacles)
• Workspace environments with different shaped obstacles affecting travel time (e.g lengthy obstacles)
Objectives
• Adapt an optimum navigation algorithm for efficient navigation motion trajectory of a mobile robot from its start position to the desired target point.
• Adapt a developed navigation algorithm for optimal meandering of a mobile robot amidst obstacles randomly distributed in the same workspace.
• Validate the developed algorithm for optimality on a simulation platform
Modelling Approach
• Mathematical algorithms development using vectors and geometry
• Hybrid approach which adapt the concept of the Hybrid Virtual Force Field (HVFF) for development of an optimum algorithm
• Code the developed solution algorithm in a software programming language (python)
Data Collection
• Develop new workspace with new obstacles
• Reproduce workspaces chosen from previous research work
Validation
• Run simulations model cases in python software

Figure 13: Conceptual framework: Research Methodological Outlook

3.3 Theoretical Framework

This section presents a detailed set of theoretical models in respect of the conceptual framework. Sub-section 3.3.1 contains the theoretical representations of the VFF algorithm, Sub-section 3.3.2 contains the theoretical representation of VGC and VOC algorithms, and lastly subsection 3.3.3 gives the HVFF methodology steps.

3.3.1 VFF algorithm

The method of route and obstacle evasion of a MR is performed by the VFF algorithm. Similar to the APFs, the VFF technique pivots with relation to the concept of attraction and repulsion. It further employs a two-dimensional cartesian lattice framework to communicate to the work environment inside which the MR is restricted to explore. A certainty value (CV) is assigned to each compartment within the work environment, indicating the accuracy of the computation as to whether an obstruction is detected on a certain block inside the work environment. In an endeavor to resolve a few of the issues related to APFs, the VFF technique was pioneered. One of the issues concerned with VFF is that of LMT. Olunloyo and Ayomoh [39] solved this problem utilising the HVFF method.

To itemise VFF technique in [38], let q_r describe the layout space of the robot in 2D range such that $q_r = f(x_r, y_r)$ where $x_r \equiv i \text{ } and \text{ } y_r \equiv j,$ which follows this directional representation $q_r = q_r(i) + q_r(j)$. Additionally, if vector operator ∇ given by $\nabla = i\frac{\partial}{\partial x} + j\frac{\partial}{\partial y}$ is looked at, then:

$$\nabla(q_r) = \left(i\frac{\partial q_r}{\partial x} + j\frac{\partial q_r}{\partial y}\right) \tag{3.1}$$

And if in every motion point the MR *R* is subjected to forces from SOs and target position then the resultant potential will be represented as follows; $U_{att}(q_r)$ – the sum of AP and $U_{rep}(q_r)$ – the sum of RP. Moreover, the gradient of U at q_r is defined below where $\nabla U(q_r)$ is a vector that faces the direction of the quickest change of U at layout q_r.

$$\nabla U(q_r) = \left[\frac{\partial U}{\partial x} \quad \frac{\partial U}{\partial y}\right]^T \tag{3.2}$$

Additionally, force potentials can be split into x and y axes in form of attraction and repulsion such that they are defined as follows and Figure 14 below show the force diagram for the robot:

$$U_{att}(q_r) = U_{att}^{x}(q_r) + U_{att}^{y}(q_r) \tag{3.3}$$

$$U_{rep}(q_r) = U_{rep}^{x}(q_r) + U_{rep}^{y}(q_r) \tag{3.4}$$

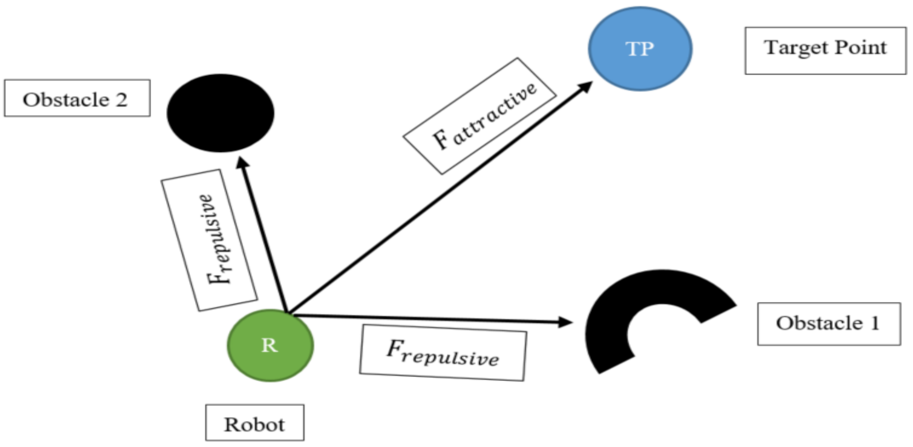


Figure 14: Vector Force Diagram illustration for the robot

Furthermore, assuming that

$$d_t = \sqrt{((x_t - x_r)^2 + (y_t - y_r)^2)} \tag{3.5}$$

Where, d_t = robot distance from the target point and the layout space for robot is given as $q_r(x_r, y_r)$ and for the target point it is given as $q_t(x_t, y_t)$. And $x_r = 1, .., (n-1), n \; and \; y_r = 1, ..., (n-1), n.$

Such that
$$d_t = \|q_t - q_r\| \tag{3.6}$$

If $q_r(x_r) \; and \; q_r(y_r)$ are direction considered the following is depicted respectively:

$$d_t = \|q_t - q_r\| = \frac{(x_t - x_r)}{\sqrt{\sum_{r=1}^{n}(x_t - x_r)^2}} \tag{3.7}$$

$$d_t = \|q_t - q_r\| = \frac{(y_t - y_r)}{\sqrt{\sum_{r=1}^{n}(y_t - y_r)^2}} \tag{3.8}$$

From Equation (3.7) and Equation (3.8) the following is depicted:

$$\nabla d_t(q_r(x_r, y_r)) = \frac{(q_t - q_r)}{\sqrt{\sum_{r=1}^{n}(x_t - x_r)^2} + \sqrt{\sum_{r=1}^{n}(y_t - y_r)^2}} = \frac{q_t - q_r}{\|q_t - q_r\|} \tag{3.9}$$

As a result, AP is as follows:

$$U_{att}^{x}(q_r) = \frac{(x_t - x_r)}{d_t} \tag{3.10}$$

$$U_{att}^{y}(q_r) = \frac{(y_t - y_r)}{d_t} \tag{3.11}$$

Furthermore, in similar context RP is represented following the work of [29]:

$$U_{rep}^{x}(q_r) = U_{rep}^{x}(q_r) + \frac{F_{cr}}{3} * \frac{c_{ij}}{d_o^2} * \frac{x_{grid} - x_o}{d_o} \tag{3.12}$$

$$U_{rep}^{y}(q_r) = U_{rep}^{y}(q_r) + \frac{F_{cr}}{3} * \frac{c_{ij}}{d_o^2} * \frac{y_{grid} - y_o}{d_o} \tag{3.13}$$

Where:

$$\mathrm{F_{cr}} = \frac{3 * \mathrm{rel_{angle}}}{\mathrm{d_o}} \tag{3.14}$$

Equation (3.14) is activated when obstacles are sensed in the dynamic window of a robot (c_{ij}). Without obstacles Equation (3.14) the outcome is zero and the repulsive initial value is also zero. This is following the presumption that the MR preferably starts from a condition of rest where it is uninformed of its current circumstance. Such that the following is considered:

$$c_{ij} = \begin{cases} 0 \; no\ obstacle\ present\ in\ robot's\ active\ window \\ 1 \; obstacle\ present\ in\ robot's\ active\ window \end{cases}$$

If c_{ij} = 1 and

if d_t > 0,

then $d_o = \sqrt{((x_r - x_o)^2 + (y_r - y_o)^2)}$ (3.15)

Where d_o = robot distance from the obstacle? The position of an obstacle is represented by $x_o\ and\ y_o$

As a result, the following Equations (3.14) and (3.15) mean the RP in the x and y direction. The route way of the MR is controlled by these two factors as they together decide the MR's new situation as it explores from one state-space towards the TP.

$$U^x(q_r) = U^x_{att}(q_r) + U^x_{rep}(q_r) \quad (3.16)$$

$$U^y(q_r) = U^y_{att}(q_r) + U^y_{rep}(q_r) \quad (3.17)$$

3.3.2 VGC and VOC algorithm

The drawback of using the traditional VFF algorithm alone is the issue of LMTs. Hence, VGC and VOC are adopted to solve the problems interlinked with the use of the VFF algorithm. Although there has been built up within the writing that either of these concepts may be utilised freely, the double utilisation of both concepts as an indispensable portion of the arrangements prepare is novel.

Additionally, an outline of how the conventional VFF is combined with VGC and VOC will be given as illustrated in Figure 15. VGC and VOC focus on guiding the robot to the objective point using sensors to avoid obstacles. Cases for VOC consider the acceptable distance of a robot from the obstacles while moving towards the objective point. Firstly, the VOC regarding this study is a proactive technique that guarantees that the MR is not pulled in into a corner locale of obstacles that have no outlet. It is pivoted on the concept of totally blocking off such paths that may lead the MR to a LMT. The approach [38] proposed for the representation of VOC is the concept of meeting vertices. This includes the presentation of a modern line that closes the edges of the impediments that outline the LMT. The crossing point of the line of the location of the target from the robot with this modern line represents the most remote area the virtual obstacle (VO) can be put from the robot; something else it is located right following to the robot along the line of the location of the objective from the MR.

To itemise VOC from [38], firstly, there is a need to identify if the VO is needed therefore the VOC function is defined as VOC = $f(I_{LS})$. Where I_{LS} is a file for portraying the statue of the line of locating from the objective position regarding capture attempts by obstacles that might cause a LMT such that the following is observed:

$$I_{LS} = \begin{cases} 0 \ \ line\ of\ locate\ is\ not\ disturbed\ by\ an\ obstacle \\ 1 \ \ line\ of\ locate\ is\ disturbed\ by\ an\ obstacle \end{cases}$$

If I_{LS} $<>0$ then VOC is actioned.

Moreover, positioning of the VO takes the following steps:

Step One: Find the vertices at the corners of the impediments that frame the passage into suspected marginal traps.

Step Two: Extend a line between vertices at the corners of the obstacle.

Step Three: Sketch a straight line from the MR's location to the specified TP. This appears as the normal direction of the MR to the TP accepting that there was no obstacle.

Step Four: On the off chance that line in Step Three converges to a line in Step Two at any point along with Step Two, it infers that the MR's common direction is in course of a trap.

Steps One-Four are confirmation steps as to whether the VOC ought to be started. When confirmed that the MR's direction is within the course of a trap, then Step Five is triggered.

Step Five: This presents a VO at the current position of the MR. It is done by utilising a few shapes of generation rule as a control command to obstruct MR motion. This anticipates the MR from moving within the heading of LMT and consequently encourages the possible minimisation of travel time. Quickly after this step, continuation is to actualise the VGC as itemised underneath by letting $f_{voc} = 1$.

Furthermore, to itemise VGC from [38] accepting a protest at a self-assertive position say $p_{i,j}$ is seen from an interpreted position $p_{i+1,j+1}$ at that point the perceivability of this question at its starting state $p_{i,j}$ from its current state, $p_{i+1,j+1}$ shapes a critical address postured by the VGC. This VGC in [38] pivots on the concept of relative perceivability. It works on the guideline of falling back from a first known location to a recently explored location. The essential objective of the VGC in this setting is to lead the MR from a point where the VO was enacted to the ultimate TP through the help of one or more brief goals referred to as virtual objectives. The number of virtual objectives and their relative location could be a work of the geometry of the impediment's interference the perceivability of MR from the objective state. The degree of perceivability of a question from its beginning location $p_{i,j}$ as prior portrayed, relative to its unused location $p_{i+n,j+n}$ determines whether the modern location ought to be considered a virtual location of the protest. On the off chance that the object's perceivability is darkened at point $p_{i+n,j+n}$ at that point going before point $p_{i+n-1,j+n-1}$ is considered for the virtual location of the question. For this reason, the middle focuses between $p_{i,j}$, and $p_{i+n,j+n}$ are chosen along and near to the vertices of the mediating impediment limit.

The procedure for the VGC is as follows:

Firstly, check the need for a virtual goal such that if $f_{voc}\ or\ T_{nso} = 1$ then virtual goal is needed. Hence, the following is observed:

$$f_{voc} = \begin{cases} 0 & VO\ not\ activated \\ 1 & VO\ \ activated \end{cases}$$

Where T_{nso} has been presented to depict the circumstances that happen when the MR and TP are at inverse closes of the space in between two barely dispersed impediments.

Finding the virtual goal takes the following steps:

Step One: Find the objective position.

Step Two: Find the position of the robot.

Step Three: From the objective location, fall back along an anticipated line of locating towards the MR location skirting the border of the mediating impediment.

Step Four: Find and locate virtual objectives at suitable focus along the line of locating as backtracking is in progress from the genuine objective to the MR location. The precise area of each virtual objectives is a function of two factors: (v_{rg}, d). Such that, v_{rg} either portrays the relative perceivability of the foremost later virtual objective from the immediately preceding virtual objective or the relative perceivability of virtual objective from the objective. And *d* is the removal between the virtual goal and the impediment edge discouraging MR's line of locate from the real objective.

Imperative parameter which is vital to end the method of presenting virtual goal is v_{mr} which speaks to the relative perceivability of the foremost later virtual objective from the robot. Such that the following is observed:

$$v_{mr} = \begin{cases} 0 \; virtual\ goal\ not\ visible \\ 1 \; virtual\ goal\ visible \end{cases}$$

In case $v_{mr} = 0$ at that point no more virtual goal is required; be that as it may if $v_{mr} = 1$ at that point another virtual goal is required. This cycle is rehash until $v_{mr} = 0$.

3.3.3 HVFF Methodological Steps

The itemisation of how the VOC and VGC concepts are combined with the VFF technique to form the pioneered hybrid approach by giving a guide diagram of the working model of the general algorithm is shown in Figure 15.

Step One: The step starts by lining up the robot to face the target point. The module permits the formation of a two-dimensional work environment of measurement X by Y and selects a settled point of reference for the workspace. The accentuation is the accessibility of the work environment. A route environment may be customary or unpredictably formed: what is vital is the accessibility and outline of the boundaries of the work environment. Where this errand is taken after by the recognisable roof of a point inside the work environment which serves as a point of reference for the MR route. Reference point aid the MR to know it claims relative location, and those of the objective state or TP, and the work environment with obstacles.

Step Two: The step looks at analysing the environment. Furthermore, the module performs workspace mapping to identify areas that cause navigation problems to the conventional VFF concept. Such areas are characterised by navigation barriers that cause LMTs. After identifying areas that are prone to the occurrence of local minima, their relative position vectors are obtained from the reference point. This information is encoded in the MR's information base before the navigation exercise begins. The significance of this point is that the proposed algorithm is partially applicable to the known surrounding, where some prior knowledge of the navigation space is required.

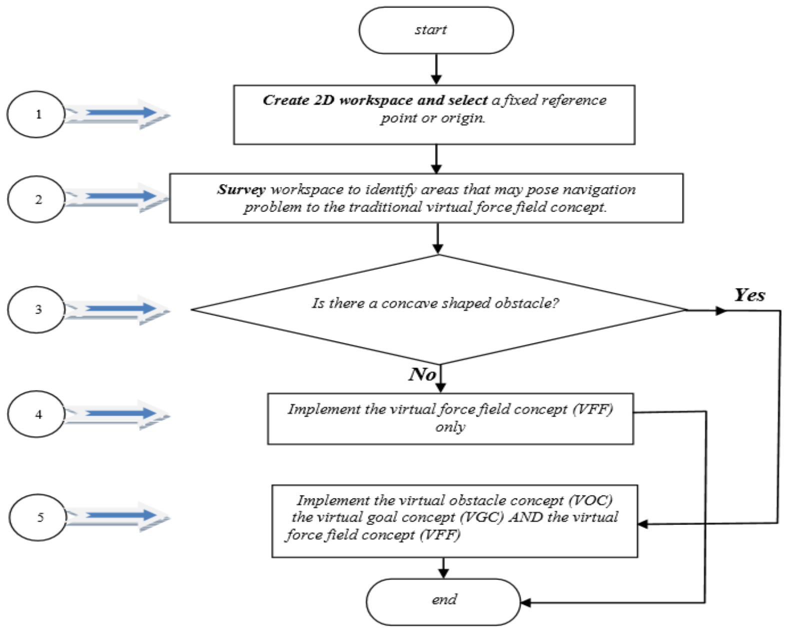

start
1
Create 2D workspace and select a fixed reference point or origin.
2
Survey workspace to identify areas that may pose navigation problem to the traditional virtual force field concept.
3
Is there a concave shaped obstacle?
Yes
No
4
Implement the virtual force field concept (VFF) only
5
Implement the virtual obstacle concept (VOC) the virtual goal concept (VGC) AND the virtual force field concept (VFF)
end

Figure 15: Flowchart for the methodology HVFF, adapted from [38]

Step Three: This step is the decision step for obstacle identification. The step primarily serves as a sequel to the above Step Two, which determines whether to enable or disable certain algorithms in a control sub-unit because of a survey exercise in Step Two. If the work environment is not likely to be a LMT, Step Four is initiated. But if the navigation trap is in doubt Step Four is skipped and goes to Step Five where VOC, VGC, and VFF are implemented. Hence, the aim is to combine Step Two with Step Four or Step Five depending on the event that may be.

Step Four: This step implements the conventional VFF technique at startup. If the result of the study of the work environment in Step Two is such that it is unlikely to capture local minima, go from the third step to the fourth. At this moment, the navigation environment is referred to as a completely unfamiliar environment as the MR's navigation controls are fully reactive and no prior knowledge of obstacles in the work environment is required. If the result of the work environment is such that a LMT is likely to happen, then the algorithm proceeds from the third step to the last one.

Step Five: The step is actioned when the LMT arises from the underlying shape constraints. In this step, VOC, VGC, and VFF are implemented together. At this stage, the navigation environment is categorised as little known surrounding because the robot investigation at the objective location relies on prior knowledge of some of the obstacles in the work environment.

3.4 Model Development and Solving

The problem undressed in this research work will be carried out using the hybrid method. The algorithm will address the concept of objective target and avoiding obstacles for MR path planning. The aim is to add value in developing an optimum algorithm for MR navigation that will optimise the total travel time of a MR from the start point to the TP in the domain of obstacles.

3.4.1 Assumptions and Navigation Modelling

In this study, the navigation environment considered for the MR is an unknown environment in which the MR acquires obstacle information such as position and geometry through sensory inputs. The constraints are obstacles and route length. The additional assumptions considered in this research include:

- All obstacle coordinates and geometry are automatically detected by the robot.
- Neither new targets nor obstacles can be added to the workspace once the robot is in motion.
- A robot has prior knowledge of the target location.

3.4.2 Mathematical Modelling

3.4.2.1 Static Scenario MR Distance from TP

Step I: Robot distance from TP

This step looks at the model that calculates the robot distance from objective point (TP) in the work environment.

$$d_{TP} = \sqrt{((x_{TP} - x_r)^2 + (y_{TP} - y_r)^2)} \tag{3.18}$$

Step II: Identifying static impediments

This looks at the procedure to identify static impediments in a workspace. The procedure is adopted from the research paper by Ayomoh et al [3] investigated an effective robot optimum route modelling in a multi-target space. Close to the current study, the issue was that of developing an ideal model to minimise the complete travel duration of a MR as it visits its desired TPs amidst workspace obstructions. A hybrid modelling approach was used to address the problem in a multi-objective domain.

Consider a 2D work environment defined by $x \; and \; y$ such that they linked to origin point O. Hence, the following is observed$(x_r, y_r) = f(x_r, y_r)$. Furthermore, take set of virtual radiation objects with equal length of $l(m)$ which is radiated by the travelling robot (r) at a certain velocity (v). Then the underlying situation moving robot (r) comparative with static obstruction straightforwardly in the middle of the robot and an ideal objective point is given as l_n. This length proceeds inside an example of decrease after some time to: l_{n-1} , l_{n-2} and hence l_{n-3} units of length. This is depicted in Figure 13. Moreover, consider that $v = \frac{d}{dt}(s)$ for moving robot at the same velocity (v) therefore $s = [l_4 - l_3 = l_3 - l_2 = l_2 - l_1]$ as a result, $[l_{(i+n)} - l_{(i+n-1)}] = [l_{(i+n-1)} - l_{(i+n-2)}]$. It is further observed that after travel time t_i the robot covers $[l_{(i+n)} - l_{(i+n-1)}]$ in the distance and the radiation cone length of the assumes new length l(m) as observed in Figure 16. If these remaining parts hold in ensuing distances covered, it suggests the deterrent is static comparative with the robot. The supposition holds if there should arise an occurrence of non-uniform speed as seen below:

$$a = \frac{d}{dt}(v) = \frac{d}{dt_i}(s_i)\,, for \; i = \{1, .., n-1, n\} \; and \; t_i \neq t_{i+1}$$

In this case, the robot covers a distance(s) between any two stretches at different times notwithstanding, the overall length of the radiation object from the robot's situation to the hindrances is likewise used to distinguish the static conduct of the obstacle. Hence, the conditions of computational steps to detect if an obstacle is static are such that it considers the following:

// Consider $(x_r, y_r) \; at \; t = 0 \; - \; robot \; prepares \; to \; move, also$

$(x_{obs}, y_{obs})\ at\ t = 0\ \ -\ obstacle\ prepares.$

At t > 0, if robot (r) co-ordinate $(x_{r+i}, y_{r+i}) \neq (x_r, y_r)$,

obstacles co-ordinate $(x_{obs+i}, y_{obs+i}) = (x_{obs}, y_{obs})$

and if $[l_{(i+n)} - l_{(i+n-1)}] = [l_{(i+n-1)} - l_{(i+n-2)}]$ then obstacle is static//

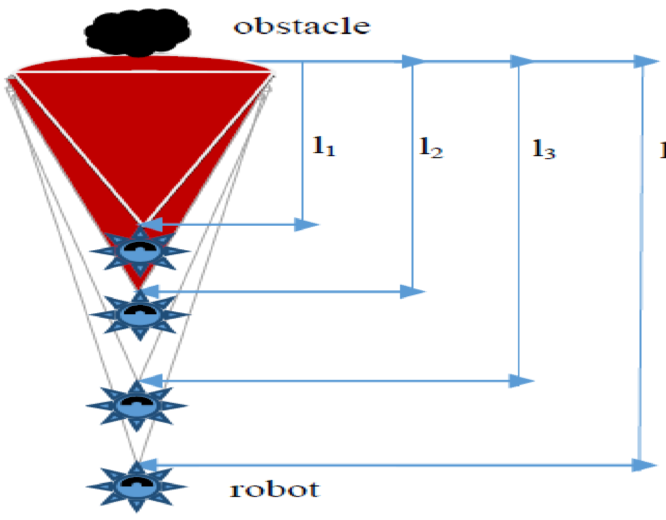


Figure 16: Static Obstacle Detection, adapted from [3]

3.4.2.2 Repulsive OA and Goal Seeking Navigation Algorithm

This algorithm assumes that the MR (r) is standing in a hill on a plane and it has to reach a goal downhill by rolling towards it. If there are obstacles in between the algorithm use the analogy of gravity or electrostatic field in which the MR and the obstacles are both of the same charge. So they repel each other proportionally, which is the concept taken from the VFF technique explained in section 3.3.1. As the MR gets closer to an obstacle the RF keeps increasing until it is great that the MR cannot touch the obstacle but avoids it and move on.

Step I: Global Functions

This step give the two global equations used in repulsive algorithm. The first equation (Equation 3.20) is the normalisation vector which calculate the unit vector and the second equation calculates the RF between the MR and the obstacle. Algorithm 1 and 2 give the pseudocode for their calculation. Algorithm 2 determine the RF between two points and returns the vector. Where d is the distance between two points, v_x *and* v_y represents vectors units for x and y, and G_o gravitational constant value of an obstacle.

- Normalise Vector

$$d = \sqrt{(v_x^2 + v_y^2)} \tag{3.19}$$

$$v = \left\langle \frac{v_x}{d}, \frac{v_y}{d} \right\rangle \tag{3.20}$$

Algorithm 1 *Nomalise Vector*

```
1  procedure NormaliseVector (input vector)
2     use Equation 3.19 to calculate $d$ (distance)
3     compute the vector using Equation 3.20
4     end
5  end
   Take the output as the normalised vector
```

- The Obstacle RF

$$F_{cr} = \frac{G_o}{CV} \tag{3.21}$$

$$Unit\_Vector = \left\langle \frac{x_r - x_o}{d_o}, \frac{y_r - y_o}{d_o} \right\rangle \tag{3.22}$$

Algorithm 2 *Obstacle Force*

```
$P_r = MR\ position$
$P_o = Obstacle\ position$
$CV = Certainty\ Value$
$rSafe$ = safety radius (it is introduced to cater to robot actual dimensions)
**procedure** ObstacleForce ($P_r, P_o, rSafe$)
    use Equation 3.15 to calculate $d_o$
    **if** $d_o \leq rSafe$ then
        Assign $CV$ the value 0
    **else**
        $CV$ is computed by $d_o$ minus $rSafe$ squared
        And gravity force is maximum when $d_o$ is equal to $rSafe$
    use Equation 3.21 to compute repulsive force proportional to $\frac{1}{cv}$
    use Equation 3.22 to compute unit vector
    multiply Equation 3.21 with Equation 3.22 to get force vector
    **end**
**end**
Take the output as force vector and unit vector
```

Step II: Repulsive OA Algorithm

This step will give the pseudocode for the repulsive OA navigation algorithm on how it is implemented.

Algorithm 3 *OA Navigation Algorithm*

```
Class Navigation:
   procedure initial ($P_r$, $P_o$, rSafe)
      set MR position
      set goal position
      set a safety radius to cater to robot actual dimensions
      create an empty list of obstacles
      end
   procedure Navigate
   determine the best route to goal and return this direction vector by default
      set steering vector
      set own position of the MR
      compute direct vector to the goal using Equation 3.18
      if length of obstacles list is > 0 then there are obstacles in sight
          for each obstacle in the list do
              generate the repulsive gravitational force in Algorithm 2
              then generate the steering vector
      use normalised vector return to the recommended steering vector
      based on this the MR turn into whatever is recommended
      end
   procedure SetGoal
    create a new goal to assist the robot when trapped deep in a long
    extreme size  concave partially trying the concept of VGC
      end
end
```

Step III: Goal Seeking Algorithm

This step will give the pseudocode for the TP navigation algorithm.

Algorithm 4 *TP Navigation Algorithm*

```
Class GoalSeeker(input MR):
   procedure initial ($P_r$, $P_o$)
      set MR position
      set goal position
      set a safety radius to cater to robot actual dimensions
      initiate Algorithm 3 by inputting $P_r$, $P_o$ and rSafe
      create a list of currently visible obstacles
      when in goal seek mode check for traps as the MR move towards the TP
      when trapped apply Algorithm 3 while updating the steering vector to the TP
      end
end
```

3.4.3 Solution

The general mathematical model that will be utilised to decide the plausibility of robot colliding with impediments in the workspace is given in section 3.3. The optimal solution that will be used to achieve the objectives of this study is the HVFF approach that applies the VFF technique combined with the VGC technique. Figure 17 below shows the solution algorithm flowchart of the approach that will be used in this study towards the aim of developing an optimum algorithm for MR navigation in complex environments. The computational algorithm that will be validated is MR navigation in the domain of SOs; such that the robot is to identify the sequential order in which the target point is to be visited. The output results will be compared with the published work that used a similar approach. Computation algorithm below is proposed for the solution to the problem of cul-de-sac traps which are caused by concave-shaped obstacles. These traps can be observed in real-life environments such as mines which are normally unstructured and complex. Mining environments have deep concaves of which in some areas humans cannot explore fully hence a need for MRs to do those tasks. In these areas an MR still needs to navigate optimally and detect lengthy or concaved obstacles. The cases presented in the next chapter test for that optimality by starting with easily detectable obstacles to non-detectible ones.

[Pre-Route Phase]

- **Step I**: The robot adjusts its detecting view to the wanted TP.

[Navigation and Obstacle Location Phase]

- **Step II**: The robot explores the TP and constantly check for the presence of impediments along its route.
- **Step III**: If a deterrent is available, the robot figures its Euclidean separation from the obstacle. Where the Euclidean robot distance from impediment is defined by Equation (3.15).
- **Step IV**: Robot checks impediments as moves towards the TP.
- **Step V**: If the impediment is present follow section 3.4.2 model procedure which incorporates the robot using sensory inputs to sense the impediments.

[Obstacle Avoidance Phase]

- **Step VI:** If the impediment cause LMT or blocks the path, implement Algorithm 3 which is taken from the VFF technique explained in section 3.3.1, and integrate the VGC technique explained in section 3.3.2 for effective OA.
- **Step VII:** If there are no obstacles continue navigating to the TP
- **Step VIII:** End

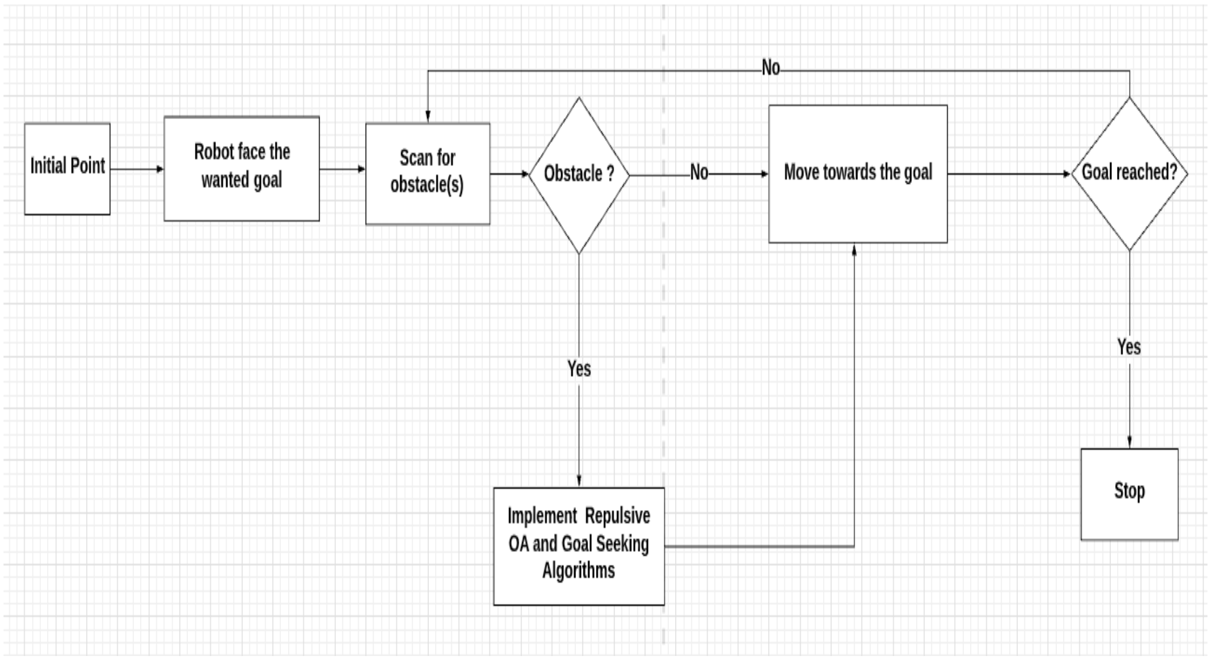


Figure 17: Solution algorithm flowchart

3.5 Chapter Summary

This chapter gave an in-depth of the research approach divided into a conceptual framework, theoretical framework, model development, and solving. The chapter gave an overview of how this research will be approached and addressed in detail to meet the objectives of this paper. The modelling approach in addressing the objectives of this research is as follows: It will start with mathematical algorithms development using vectors and geometry and hybrid approach which adapt the concept of the HVFF for development of an optimum algorithm. Lastly, the developed solution algorithm will be used in coding programs for running simulations and validation purpose in the next chapter.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This chapter gives validation of our solution algorithm, simulations were conducted both on recently created workspaces and few chosen workspaces created by prior researchers in ref [41]. The simulations were implemented using PyCharm which uses python language. The chapter further gives an analysis and discussion of simulated results.

4.2 Model Validation and Results

The first three cases are created workspaces looking at a concave-shaped obstacle with different sizes that create LMT for a MR. The fourth case is the mixed workspace scenario of SOs with two TPs. The last two cases are reproduced workspaces chosen from previous research work. The TP in the workspace is the black and white flag, MR is the small blue robot and the black circles represent the obstacles. Simulation results were shown in three-phase figures: The initial phase where the MR is at rest, the middle phase where the MR is in motion and the MR's trail is black dotted, and the last phase is when the MR reaches the TP where the MR's is red dotted. In all cases time is calculated in seconds from the initial point to the TP and results are shown in Tables 4 to 9.

Figures 18 to 41 are mazes initially created in this work to illustrate the completeness and generalised applicability of our algorithm in complex spaces with SOs. Comparison with the outcomes of other researchers' work ref [41] is as shown in

Figure 42 to Figure 56. The comparison was carried out by duplicating their workspaces in terms of the shape, estimate, and location of the impediments within the particular workspaces. In any case, it was not continuously conceivable to replicate the chosen workspaces precisely to scale in this research work. This is because of the non-accessibility of point-by-point data such as the impediment measurements and in a few cases, the workspace measurements. All things considered, the need of this study is to be able to replicate such workspace to a reasonable level for validation. Furthermore, the chosen workspaces for this work are for the most part those that customarily see complex or where prior researchers recorded challenges in exploring the MR to the TP. The objective in doing this is to be able to validate by and large the adequacy of our approach relative to a few existing models. The autonomous robot deployed in this study is an animated robot picked as an image is presented in the appendix myriads of codes. The speed was set to five with the robot safety radius of 50 and range reduction of four if trapped, to reduce the detection range to avoid getting lost.

4.2.1 Case 1: A simple and easily detectable concave obstacle

The workspace created for this case has a simple and easily detectable concave-shaped static obstacle. Figures 18 to 22 show the simulated results.

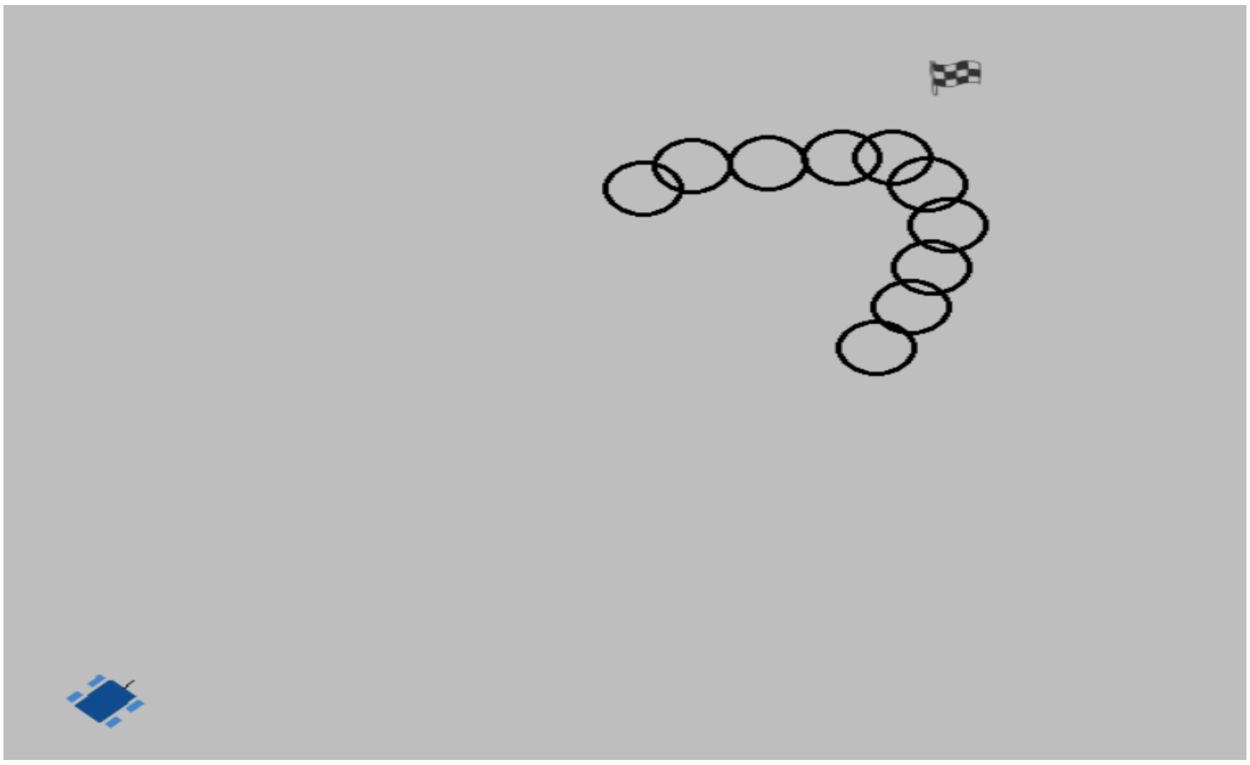


Figure 18: Developed Problem Sample Space01

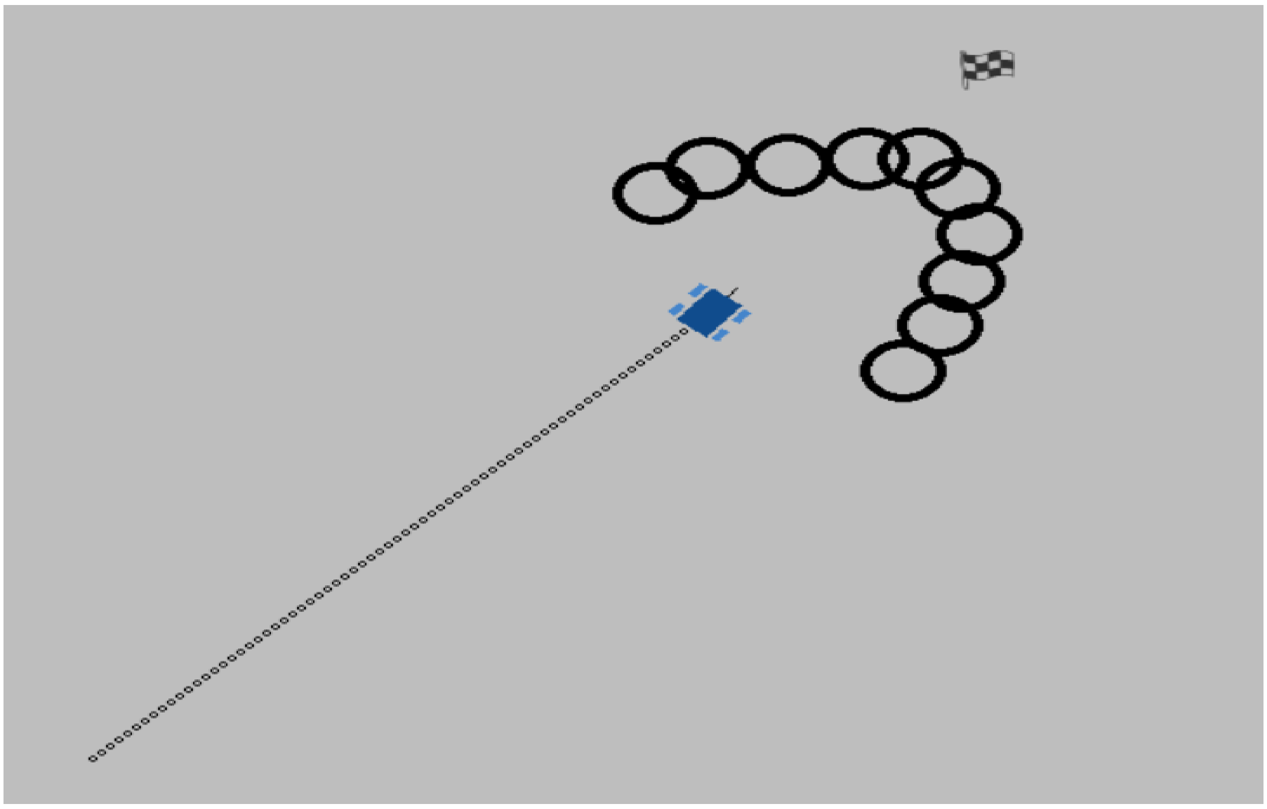


Figure 19: Progressive Solution for developed Sample Space01 (without HVFF)

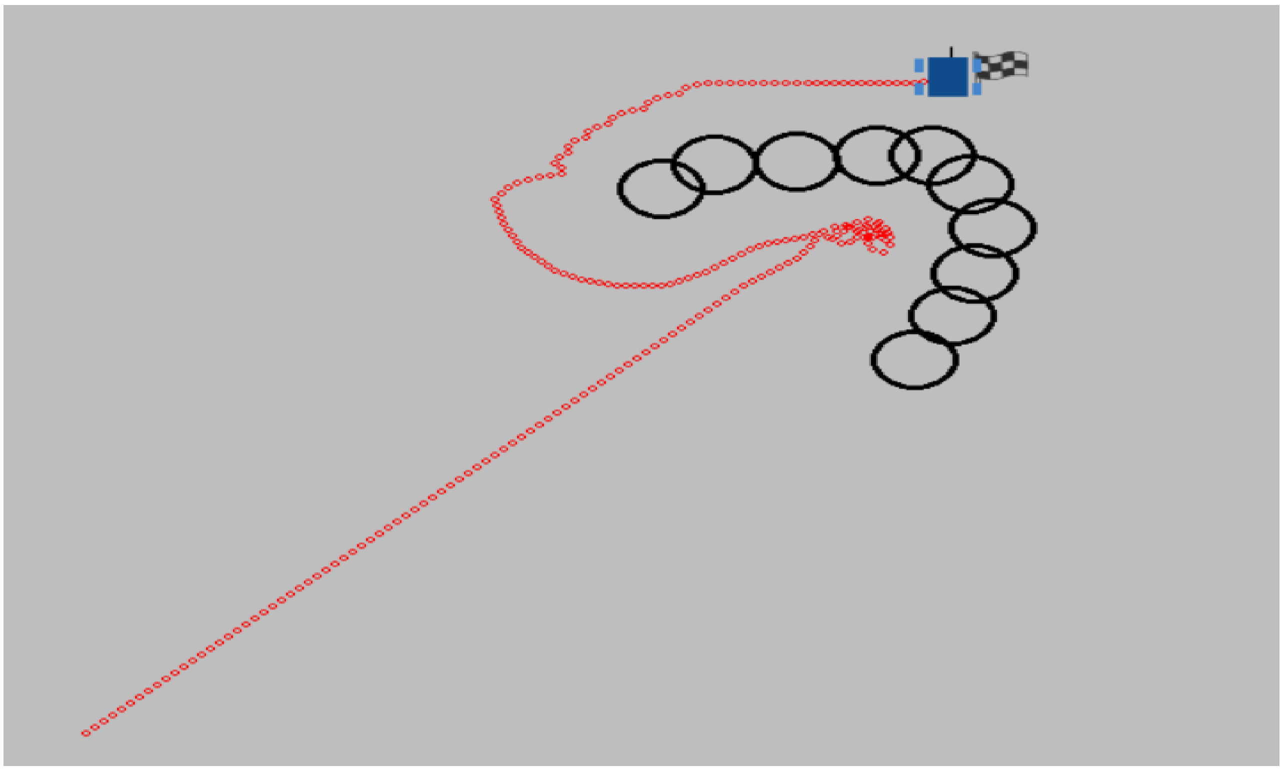


Figure 20: Final Solution for developed Sample Space01 (without HVFF)

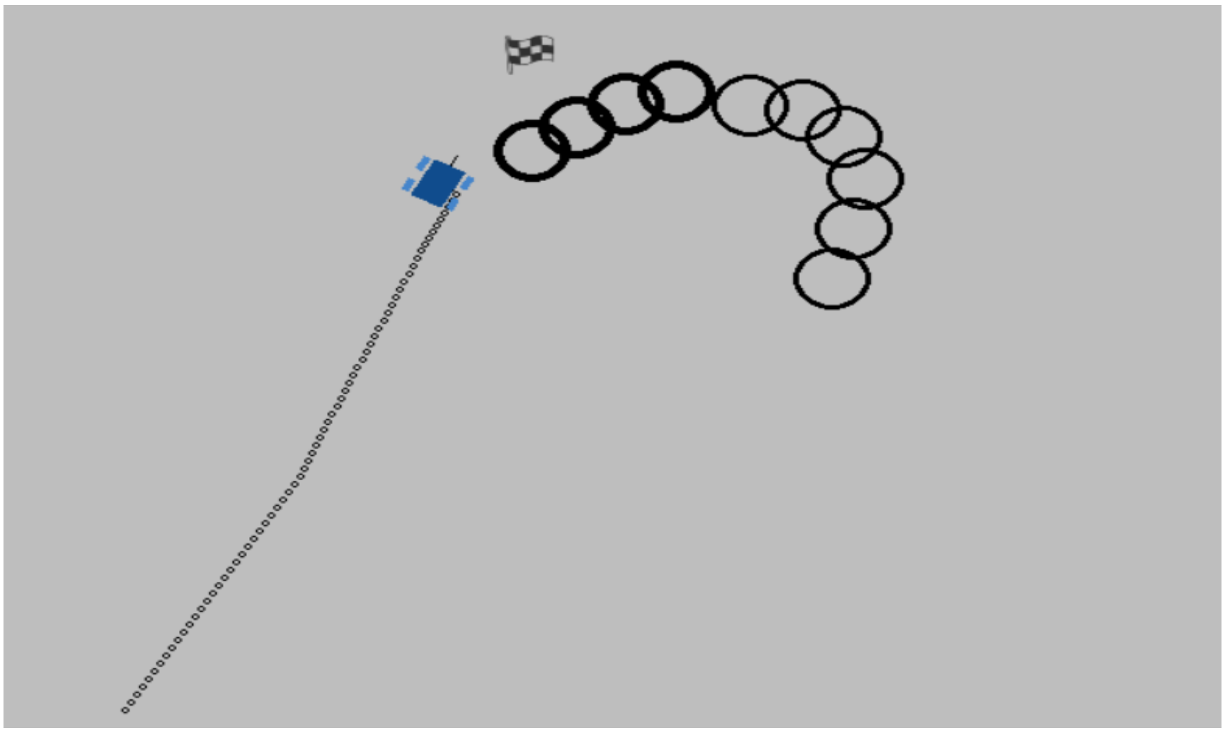


Figure 21 : Progressive Solution for developed Sample Space01 (with HVFF)

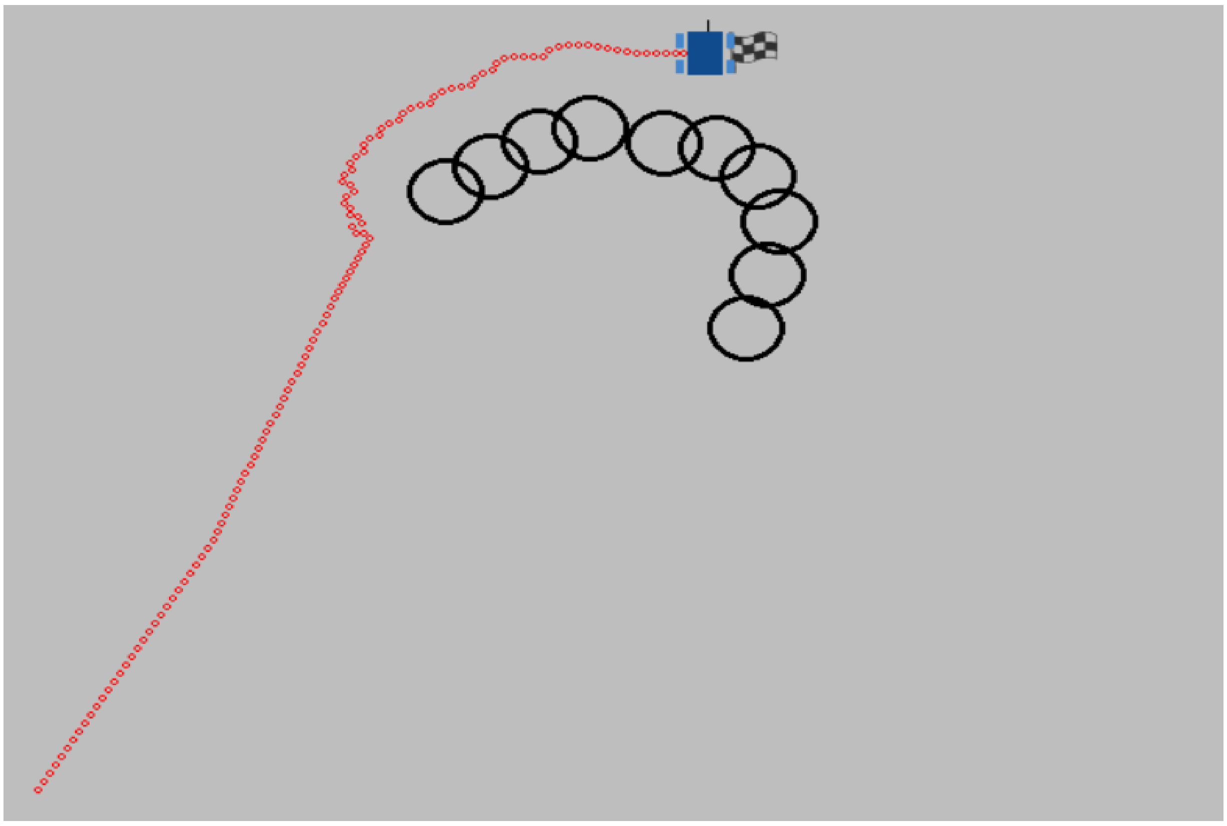


Figure 22: Final Solution for developed Sample Space01 (with HVFF)

Table 4: Results for Case 1

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the target point	46.52	26.90

4.2.2 Case 2: An intermediate size “not easily detectable” concave obstacle

The workspace created for this case has an intermediate size “not easily detectable” concave-shaped static obstacle. Figure 23 to Figure 27 show the simulated results.

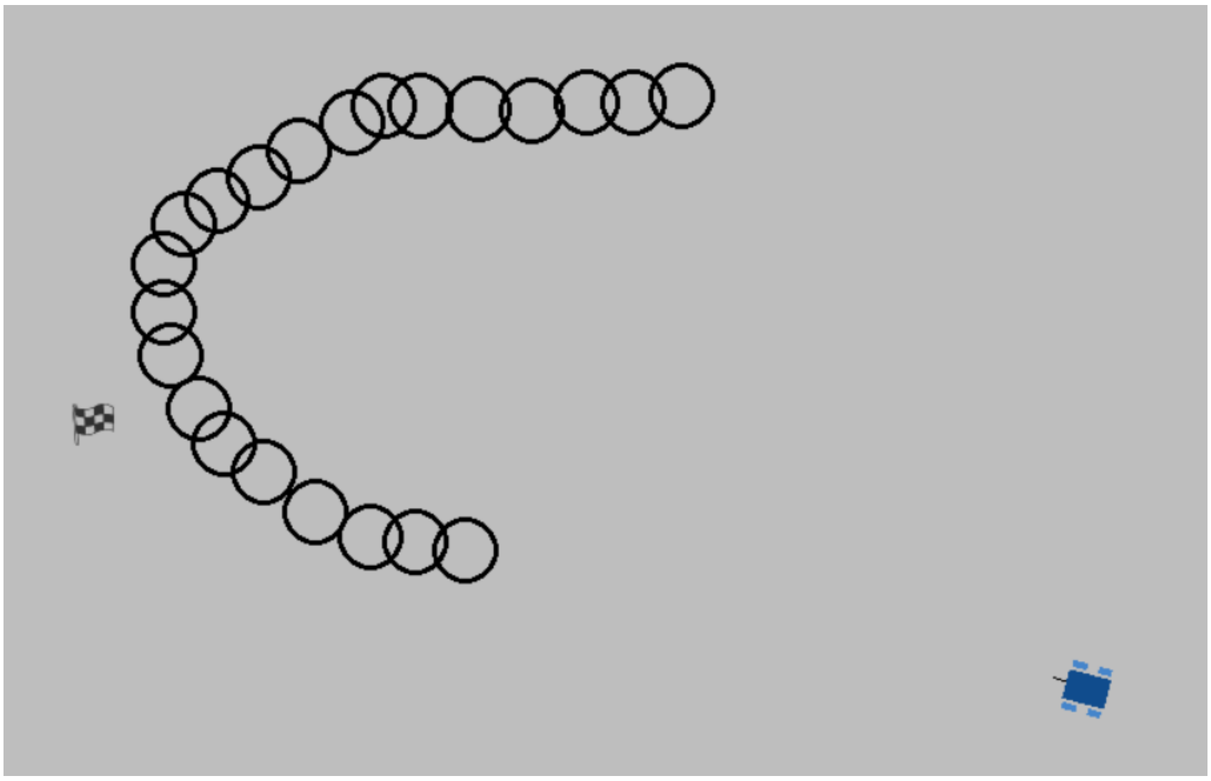


Figure 23: Developed Problem Sample Space02

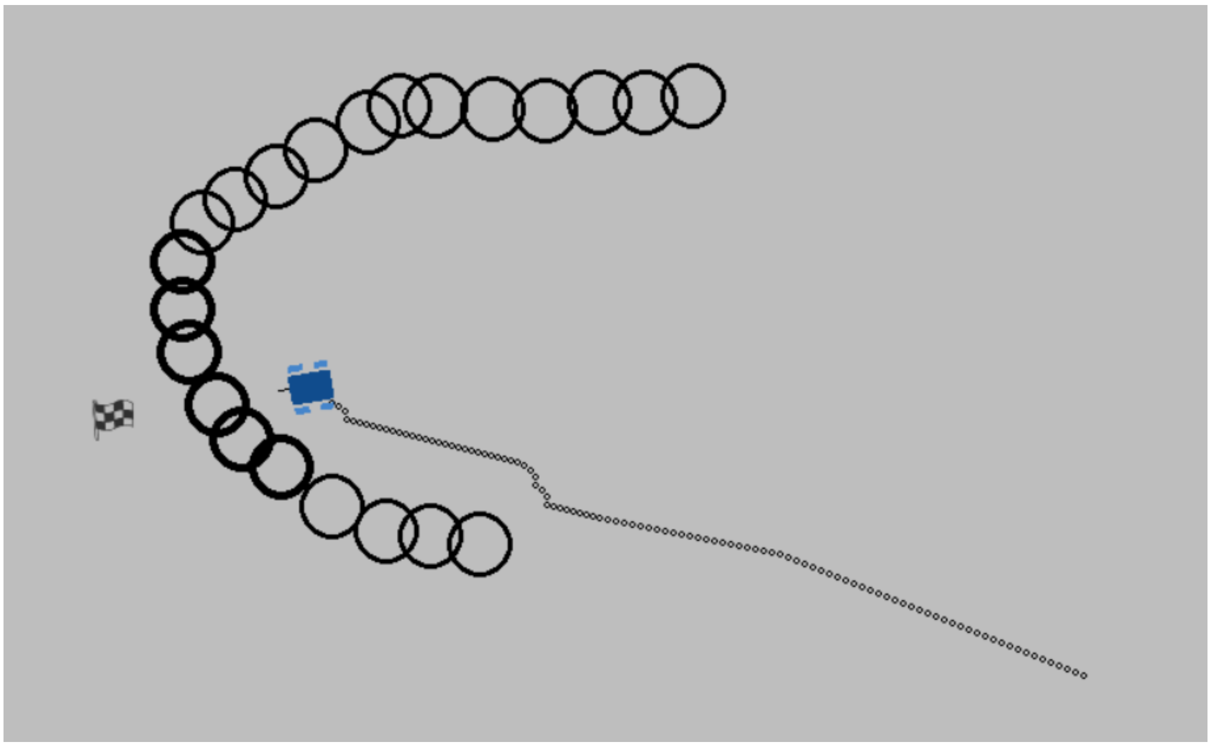


Figure 24: Progressive Solution for developed Sample Space02 (without HVFF)

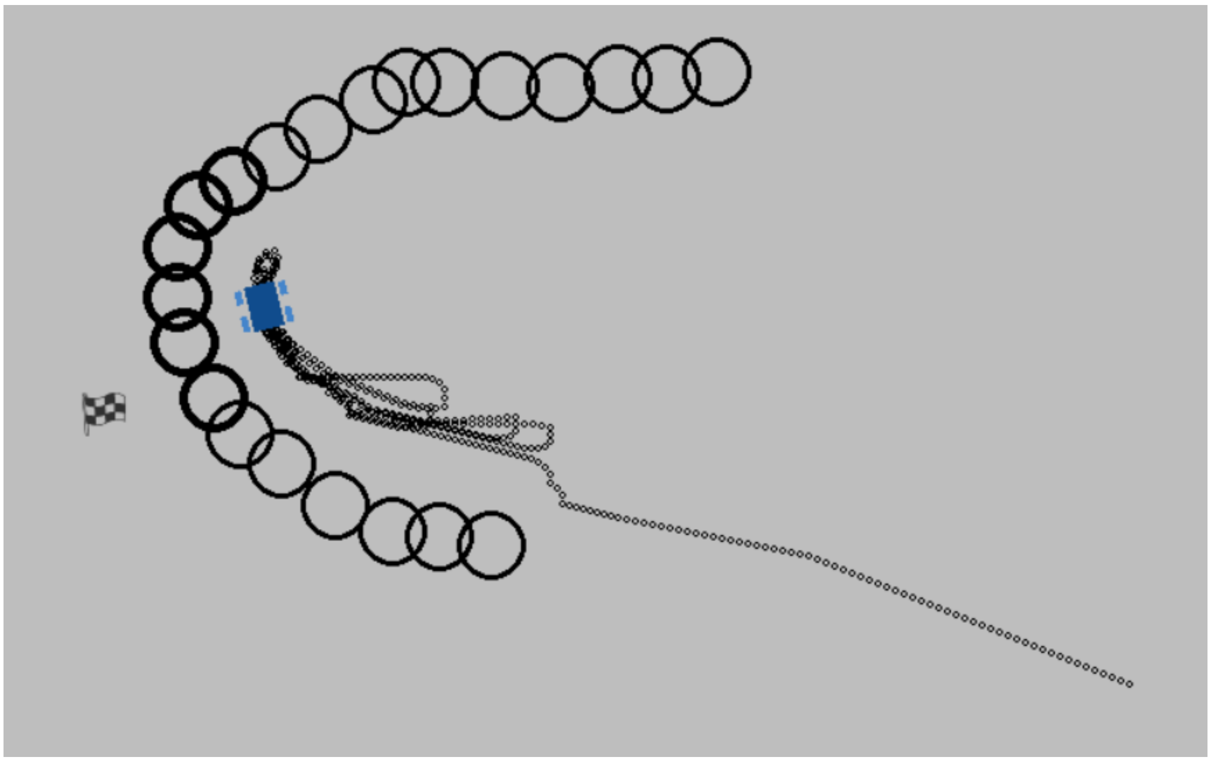


Figure 25: Progressive Solution for developed Sample Space02 (MR got trapped)

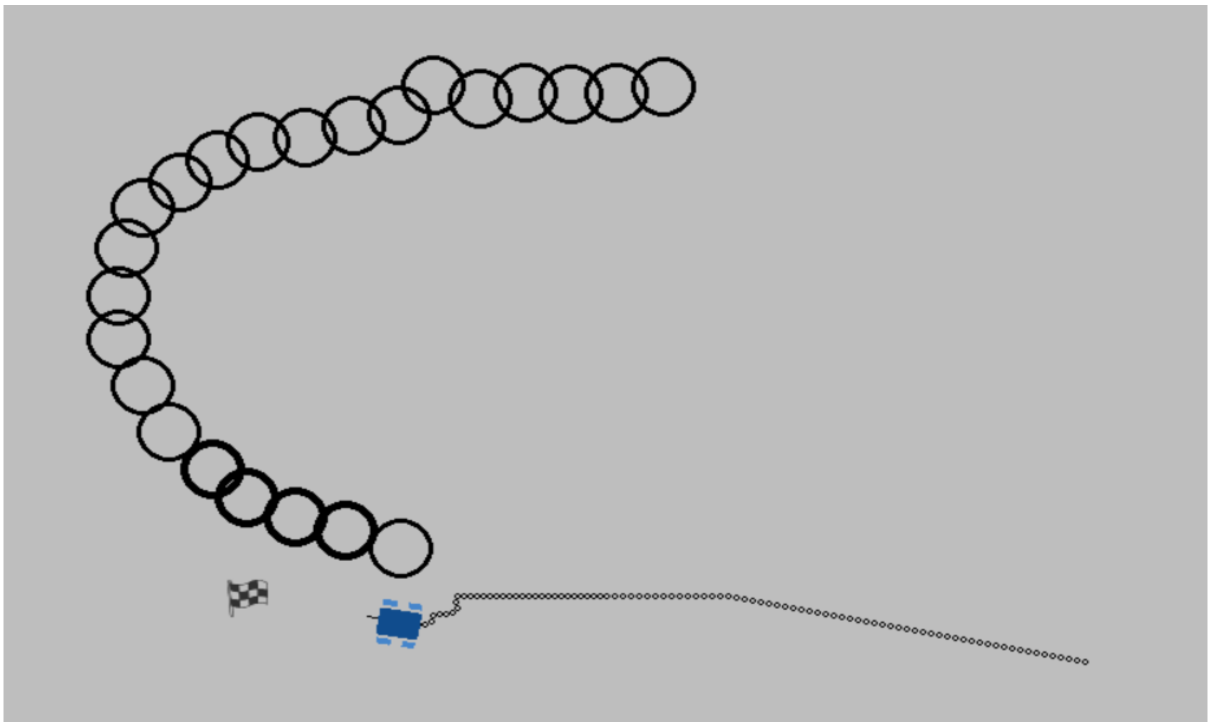


Figure 26: Progressive Solution for developed Sample Space02 (with HVFF)

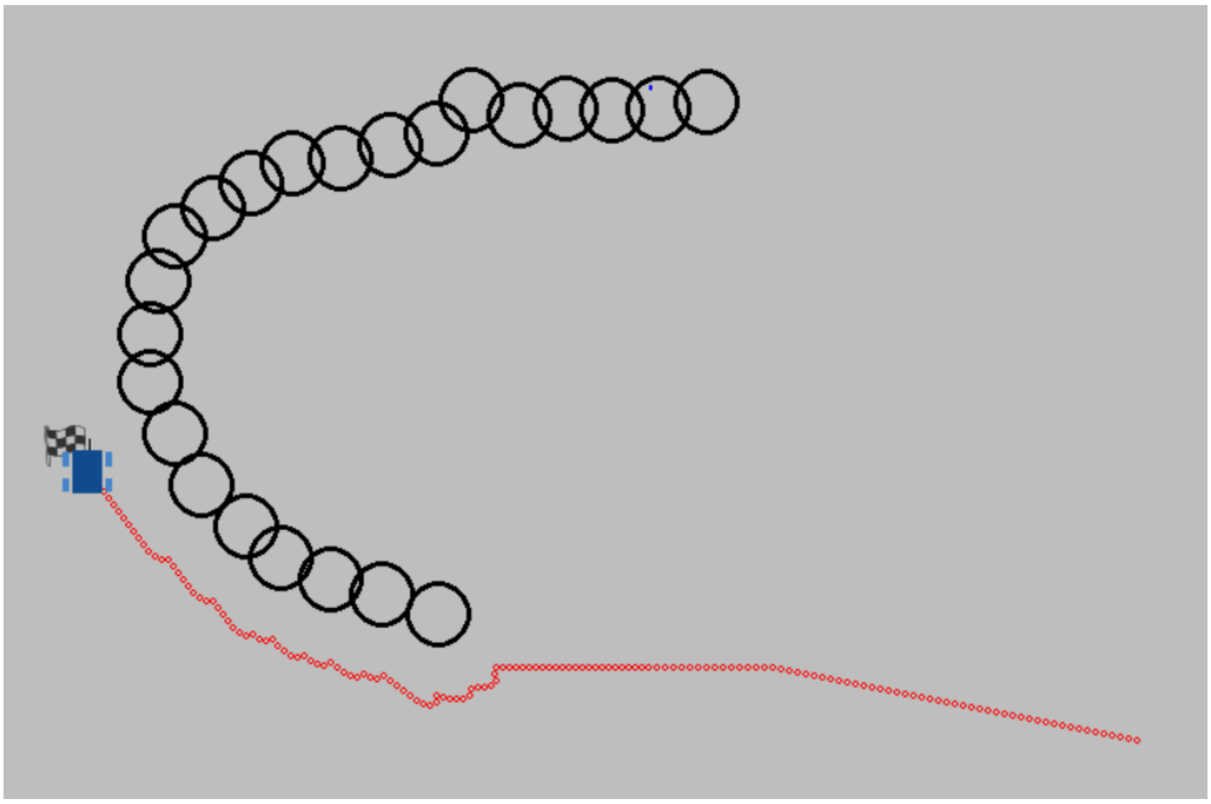


Figure 27: Final Solution for developed Sample Space02 (with HVFF)

Table 5: Results for Case 2

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the TP	MR got trapped until the simulation runs was stopped.	33.02

4.2.3 Case 3: An extreme size undetectable concave obstacle

The workspace created for this case has an extreme size undetectable concave-shaped static obstacle. Figure 28 to Figure 32 show the simulated results.

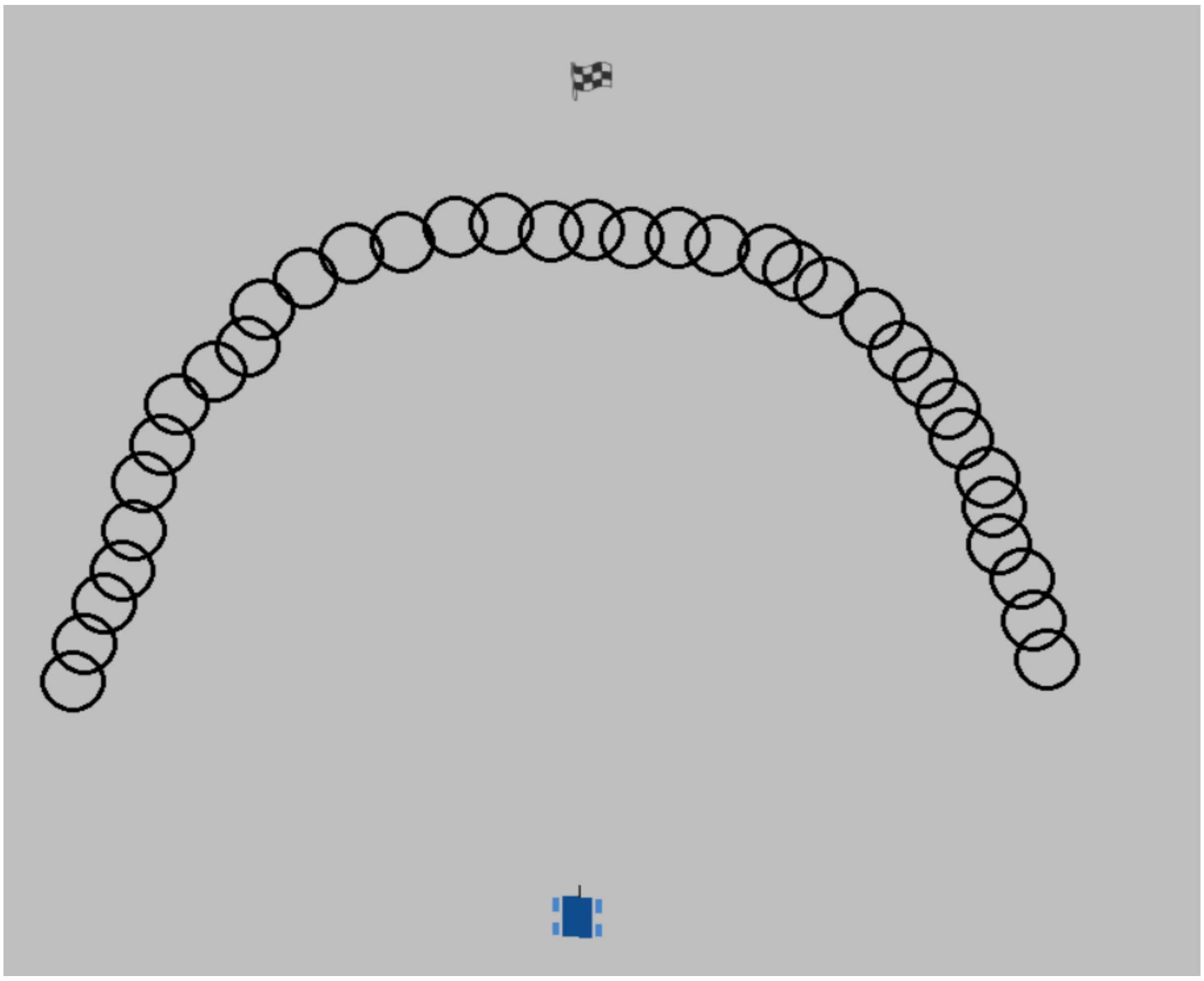


Figure 28: Developed Problem Sample Space03

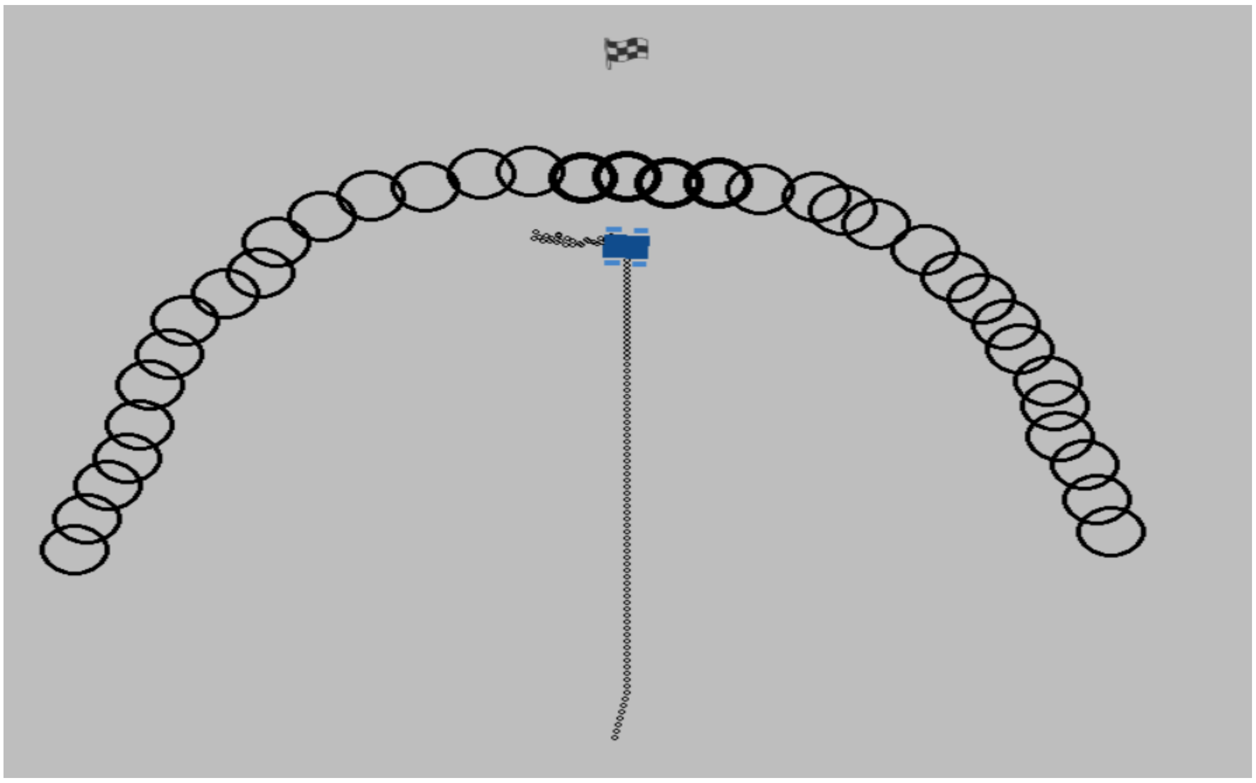


Figure 29: Progressive Solution for developed Sample Space03 (without HVFF)

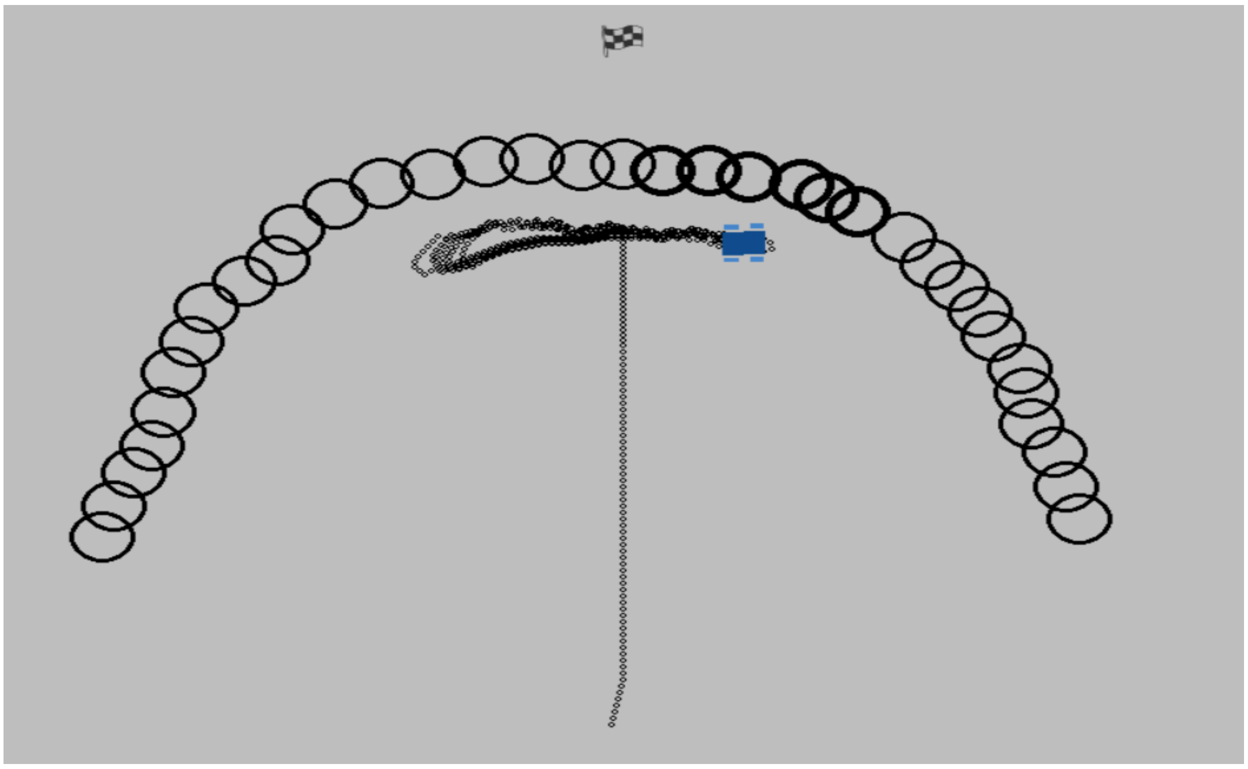


Figure 30: Progressive Solution for developed Sample Space03 (MR got trapped)

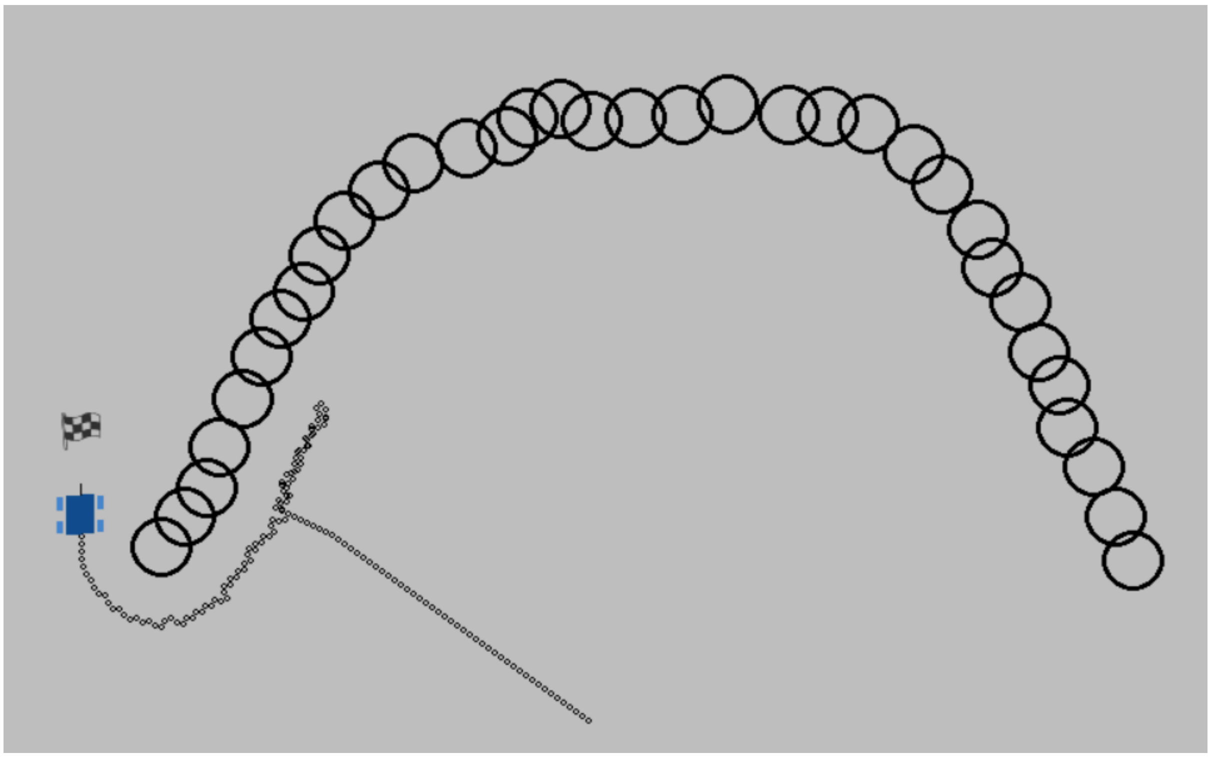


Figure 31: Progressive Solution for developed Sample Space03 (with HVFF)

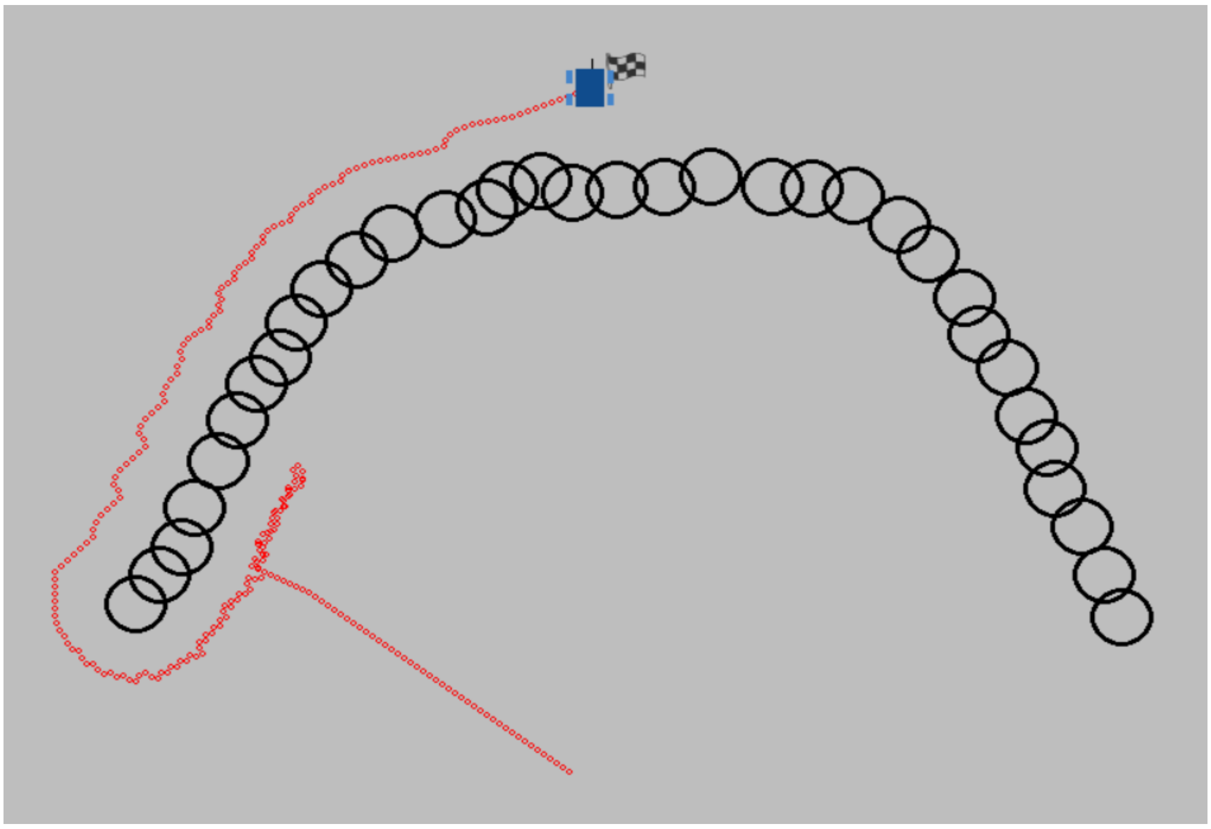


Figure 32: Final Solution for developed Sample Space03 (with HVFF)

Table 6: Results for Case 3

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the TP	MR got trapped (Infinite seconds) until simulation was stopped.	53.20

4.2.4 Case 4: Multiple Target Points Workspace scenario with static obstacles

The workspace created for this case has simple and easily detectable concave-shaped SOs with two TPs. Figure 33 to Figure 41 show the simulated results.

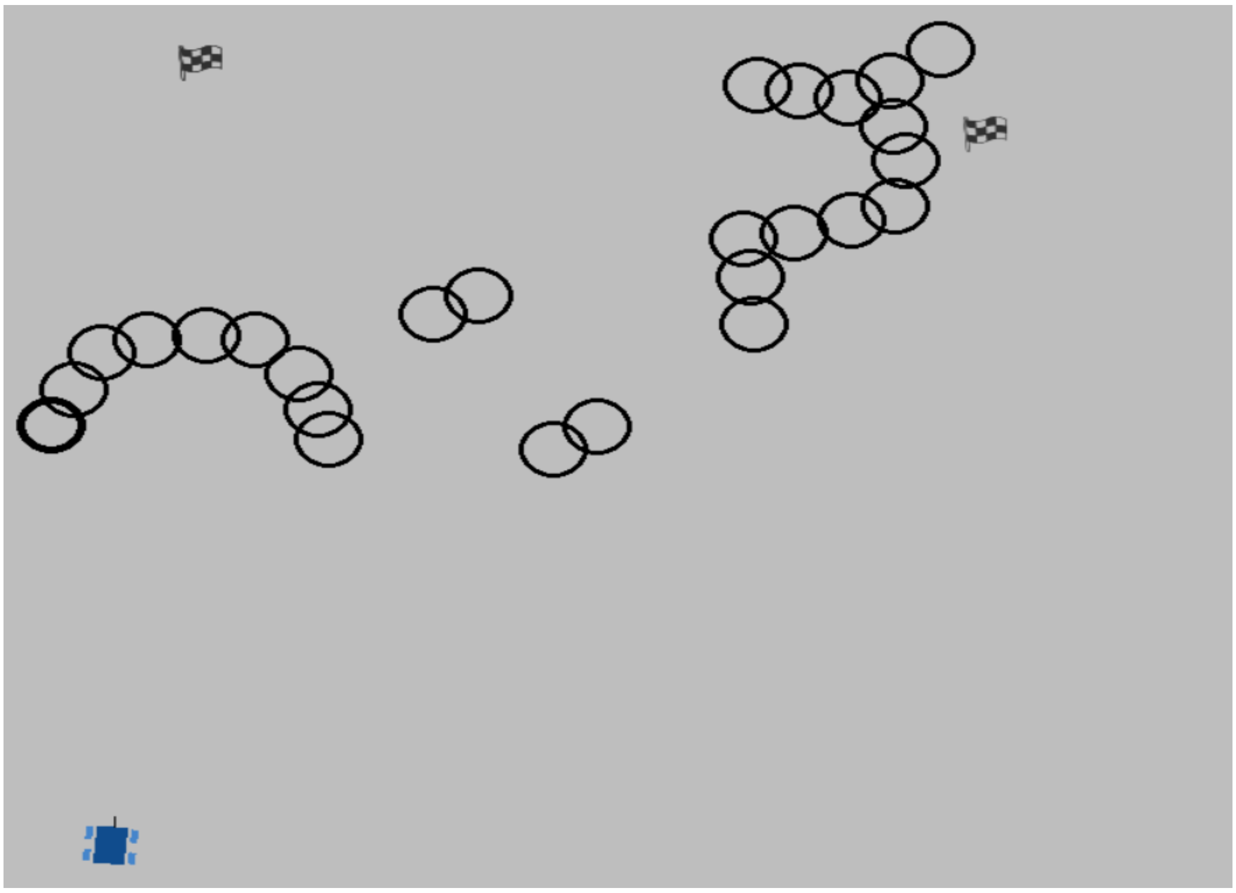


Figure 33: Developed Problem Sample Space04

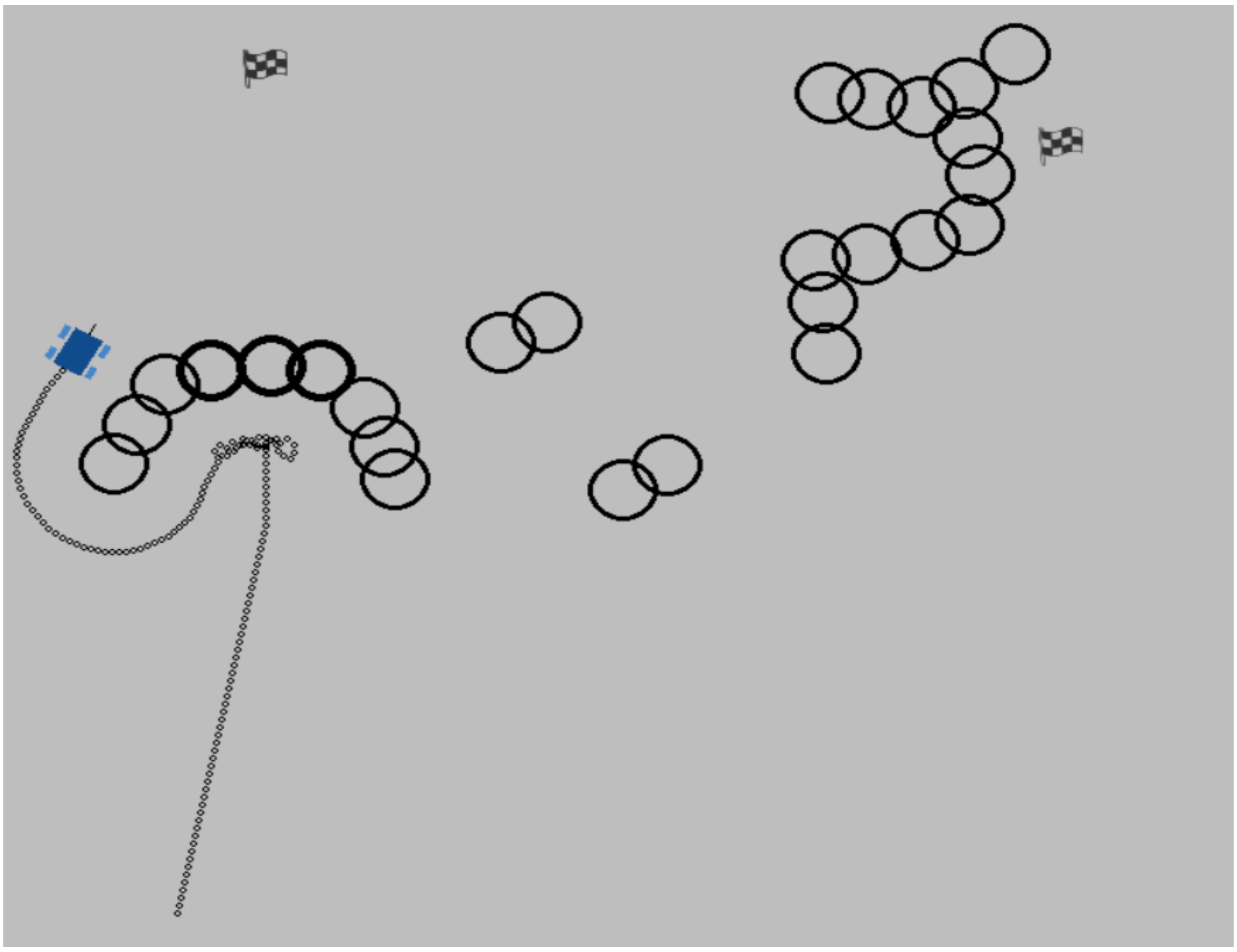


Figure 34: Progressive Solution (First TP) for developed Sample Space04 (without HVFF)

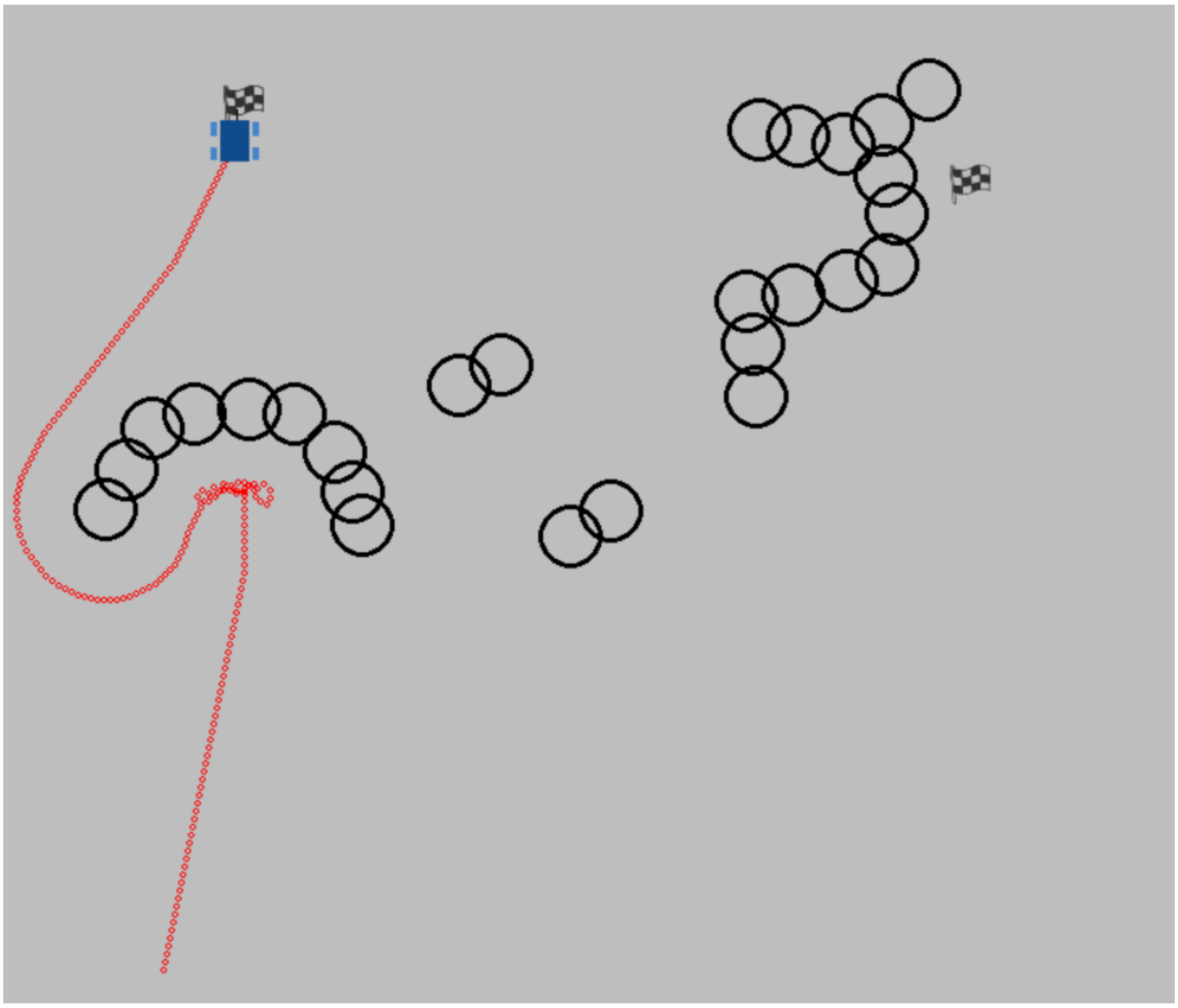


Figure 35: Final Solution (First TP) for developed Sample Space04 (without HVFF)

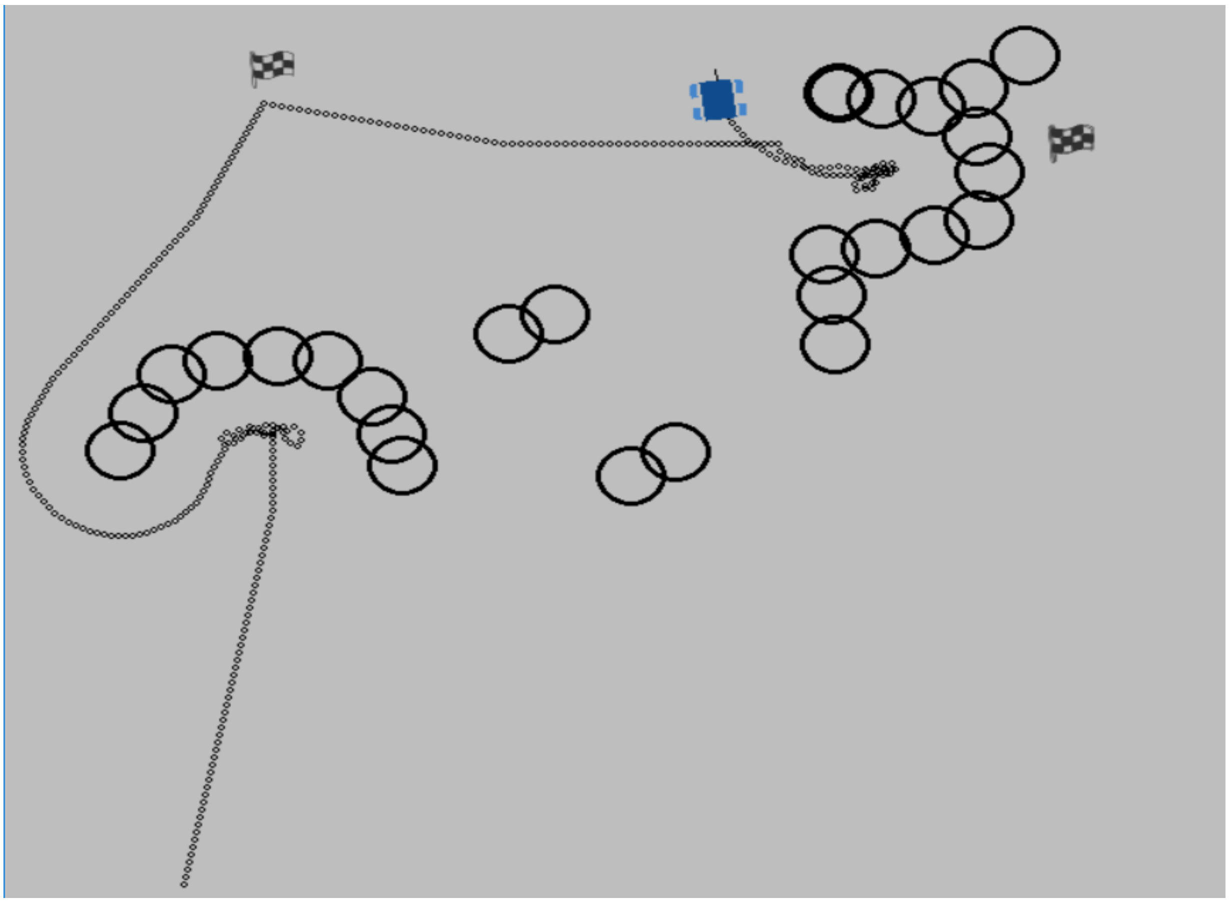


Figure 36: Progressive Solution (Second TP) for developed Sample Space04 (without HVFF)

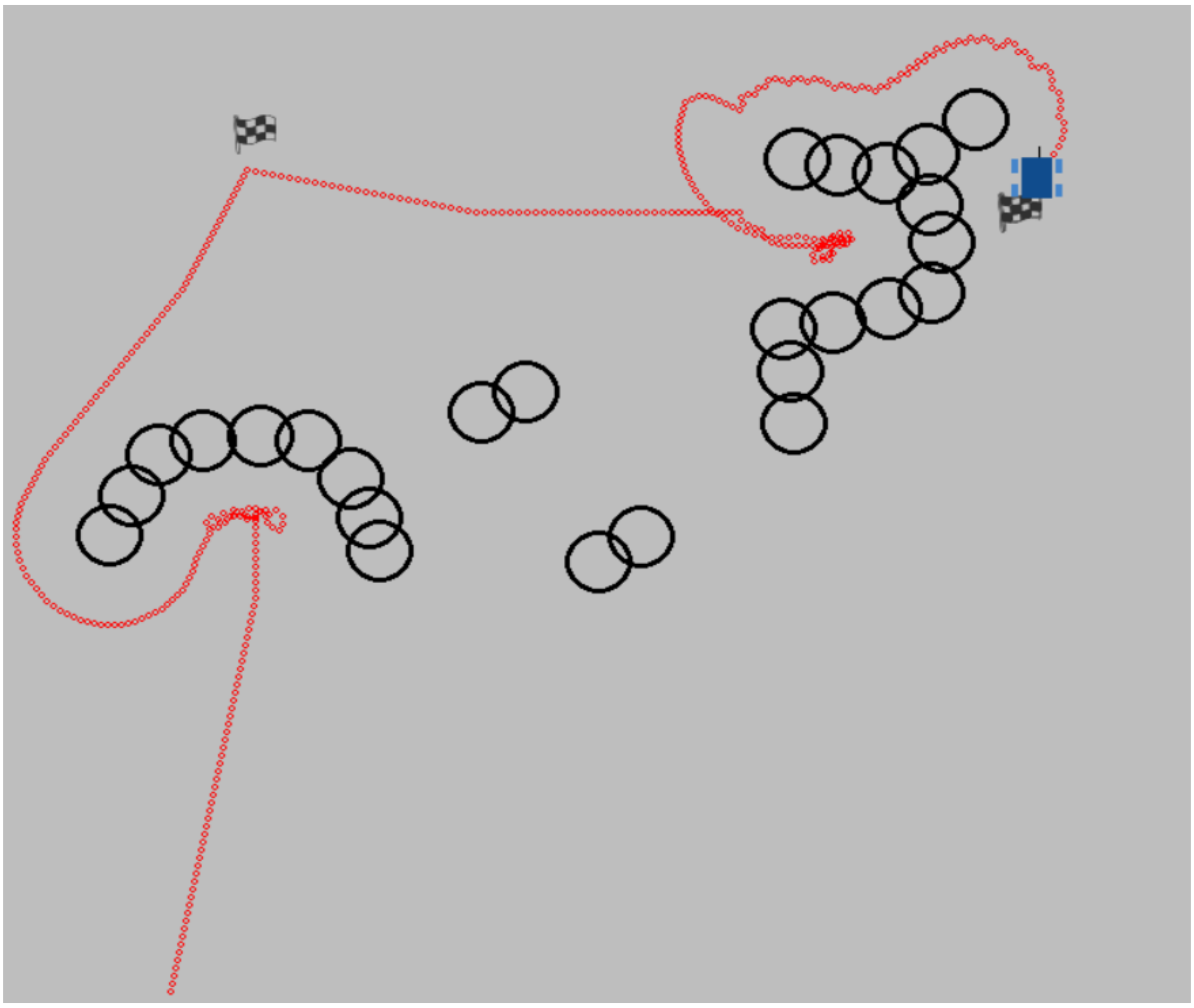


Figure 37: Final Solution (Second TP) for developed Sample Space04 (without HVFF)

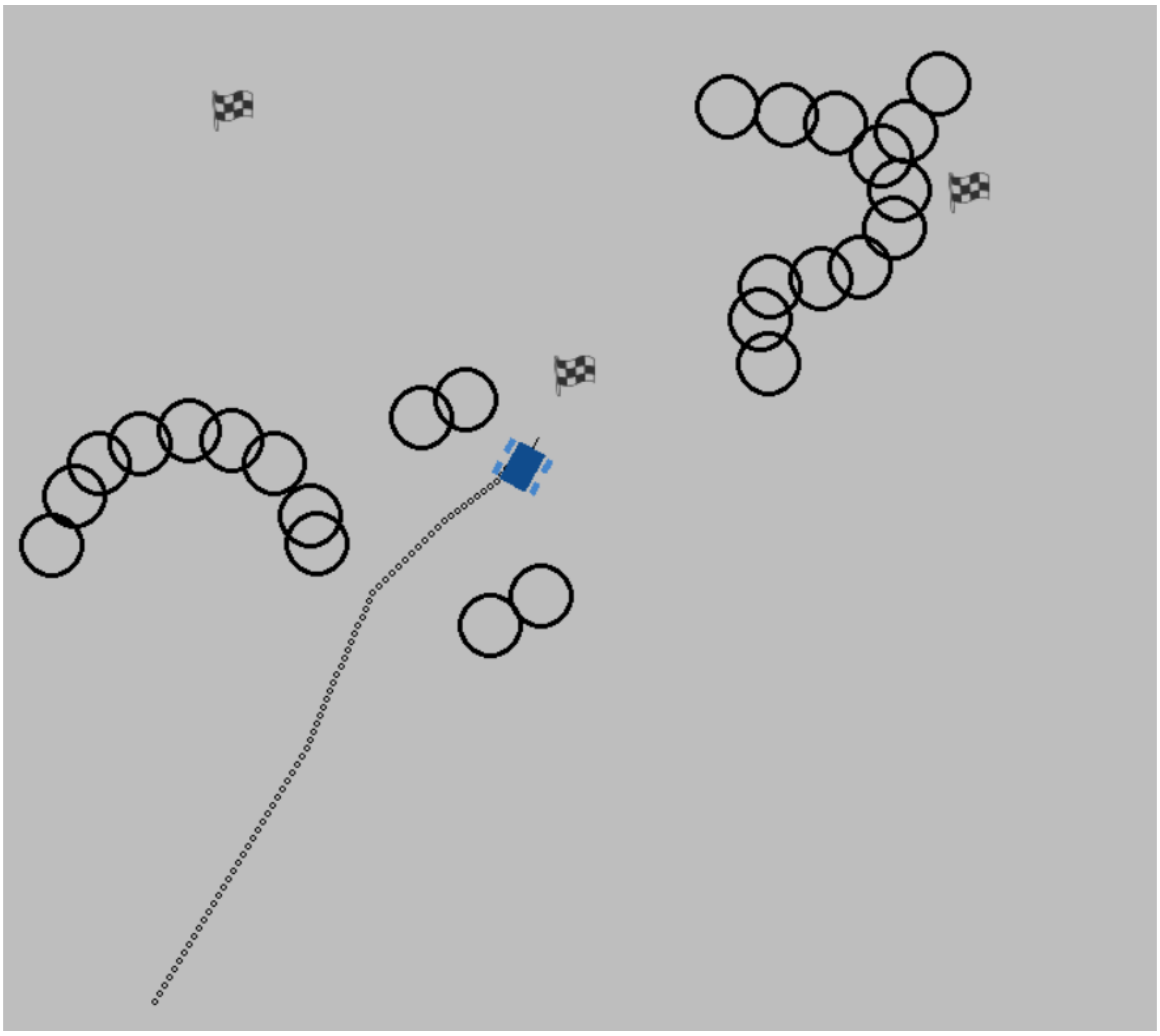


Figure 38: Progressive Solution (First TP) for developed Sample Space04 (With HVFF)

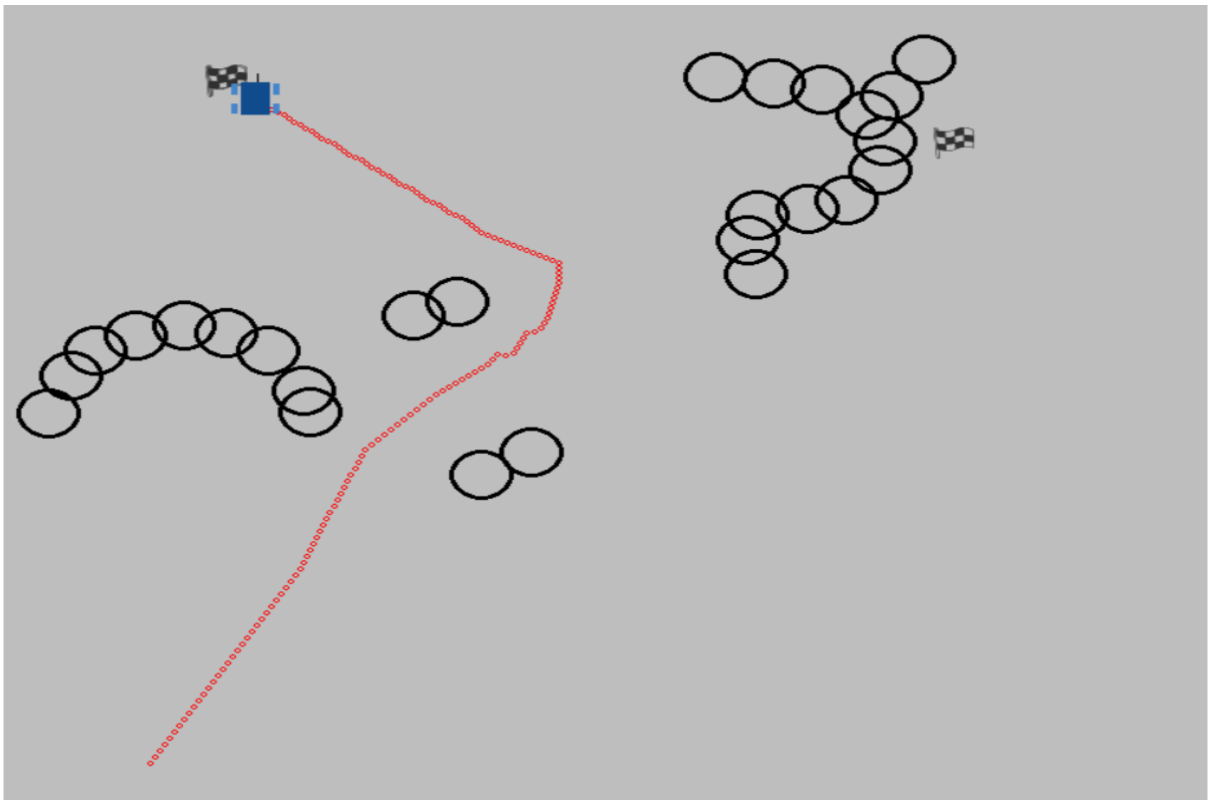


Figure 39: Final Solution (First TP) for developed Sample Space04 (with HVFF)

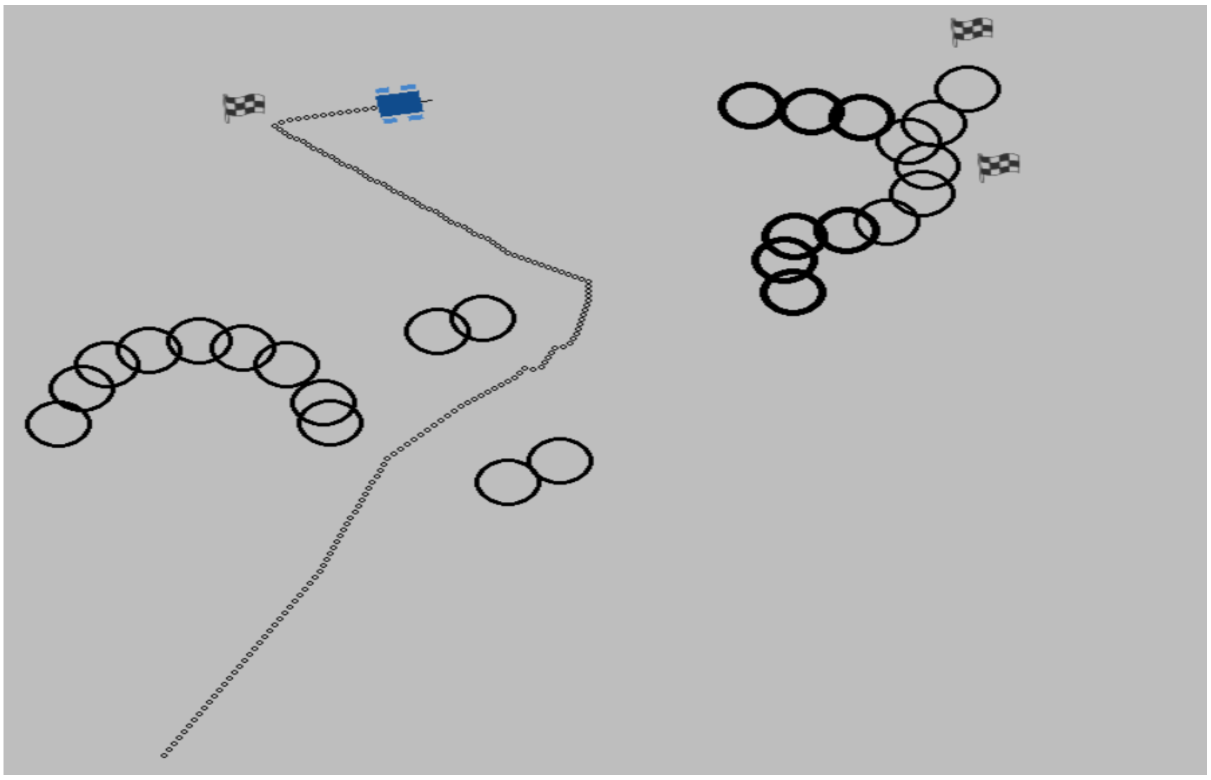


Figure 40: Progressive Solution (Second TP) for developed Sample Space04 (with HVFF)

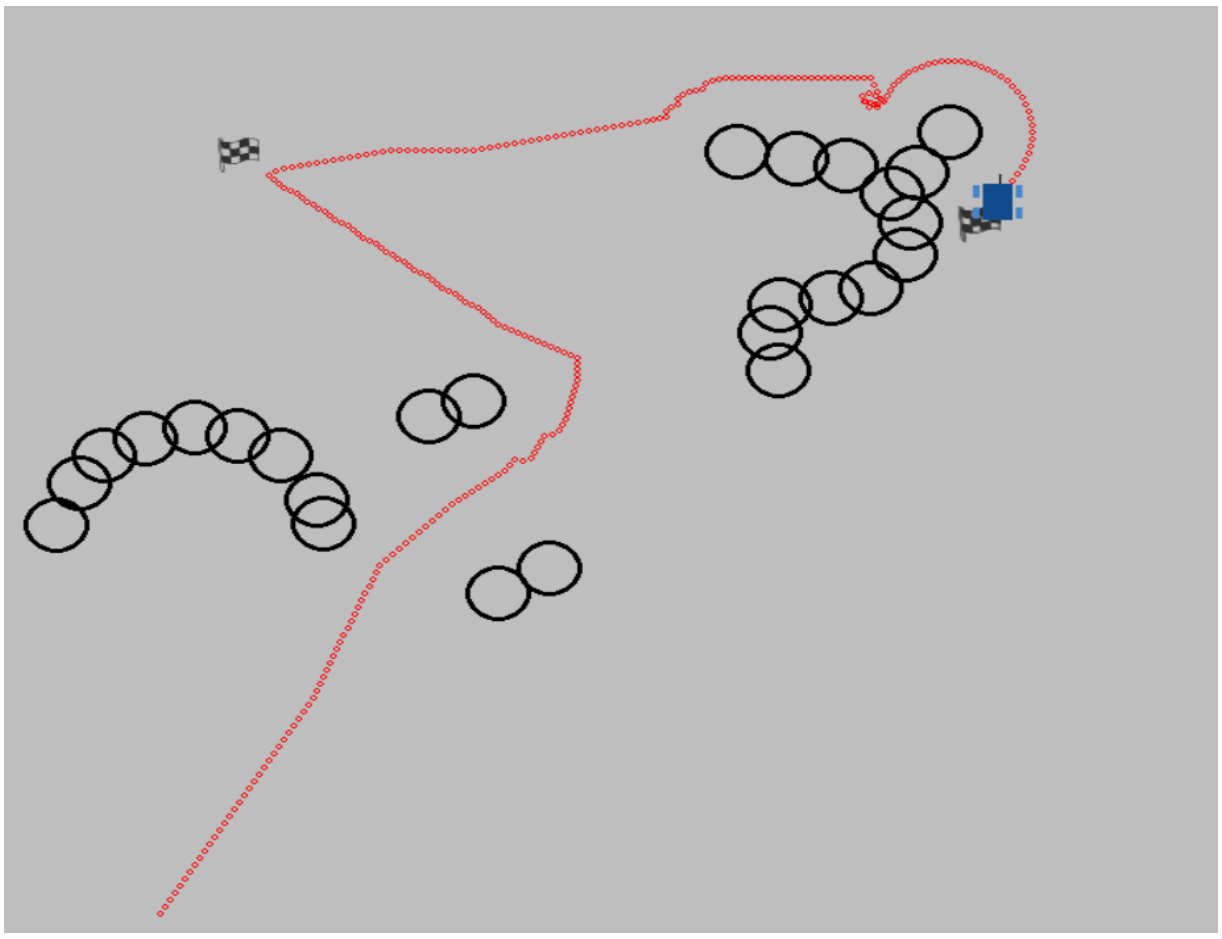


Figure 41: Final Solution (Second TP) for developed Sample Space04 (with HVFF)

Table 7: Results for Case 4

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the TP 1	40.79	31.80
Time (seconds) from TP 1 to TP 2	44.77	30.50

4.2.5 Case 5: Lengthy & Concave Obstacles workspace scenario from paper Ref [41]

The workspace created for this case, it has mixed SOs (not easily detectable) which is a reproduced workspace from paper Ref [41]. Figure 42 to Figure 49 show the simulated results.

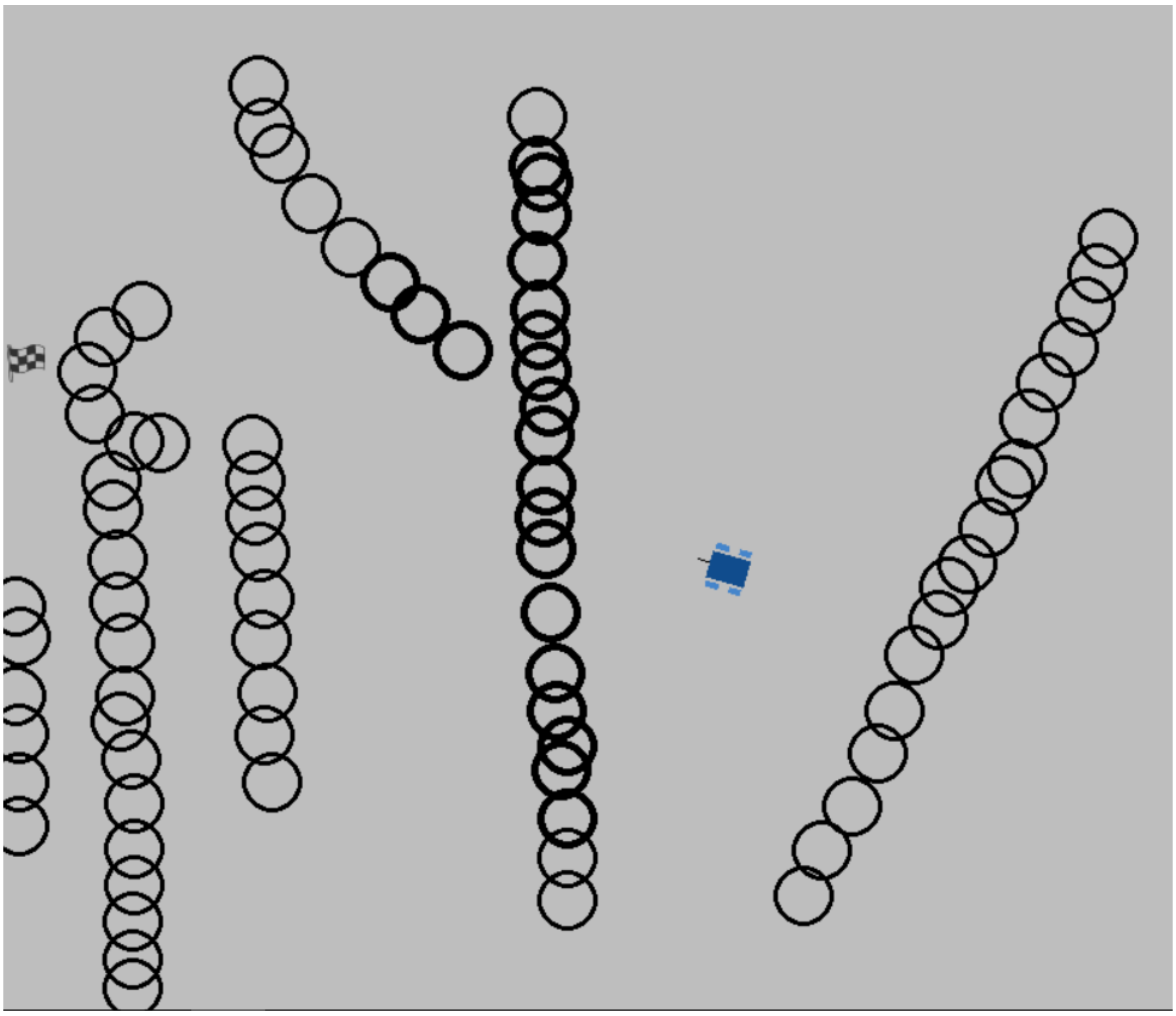


Figure 42: Developed Problem Sample Space05

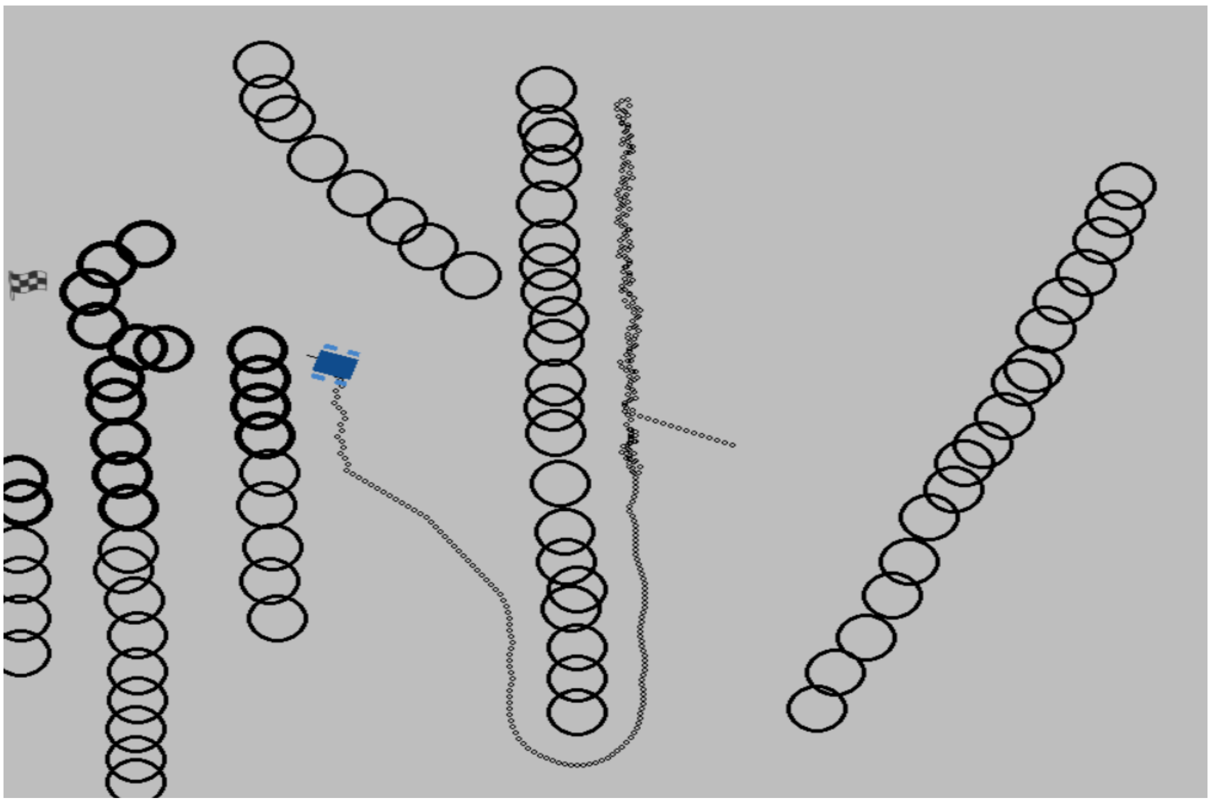


Figure 43: Progressive Solution for developed Sample Space05 (without HVFF)

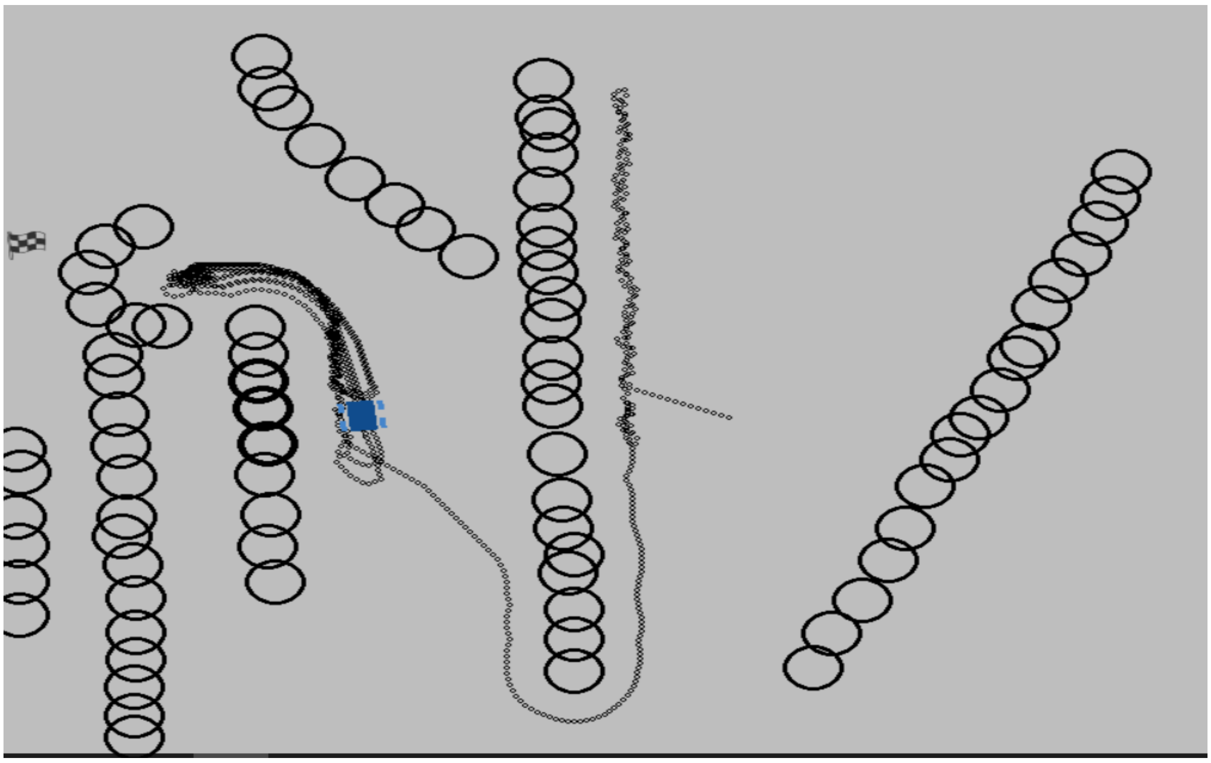


Figure 44: Progressive Solution for developed Sample Space05 (MR got rapped)

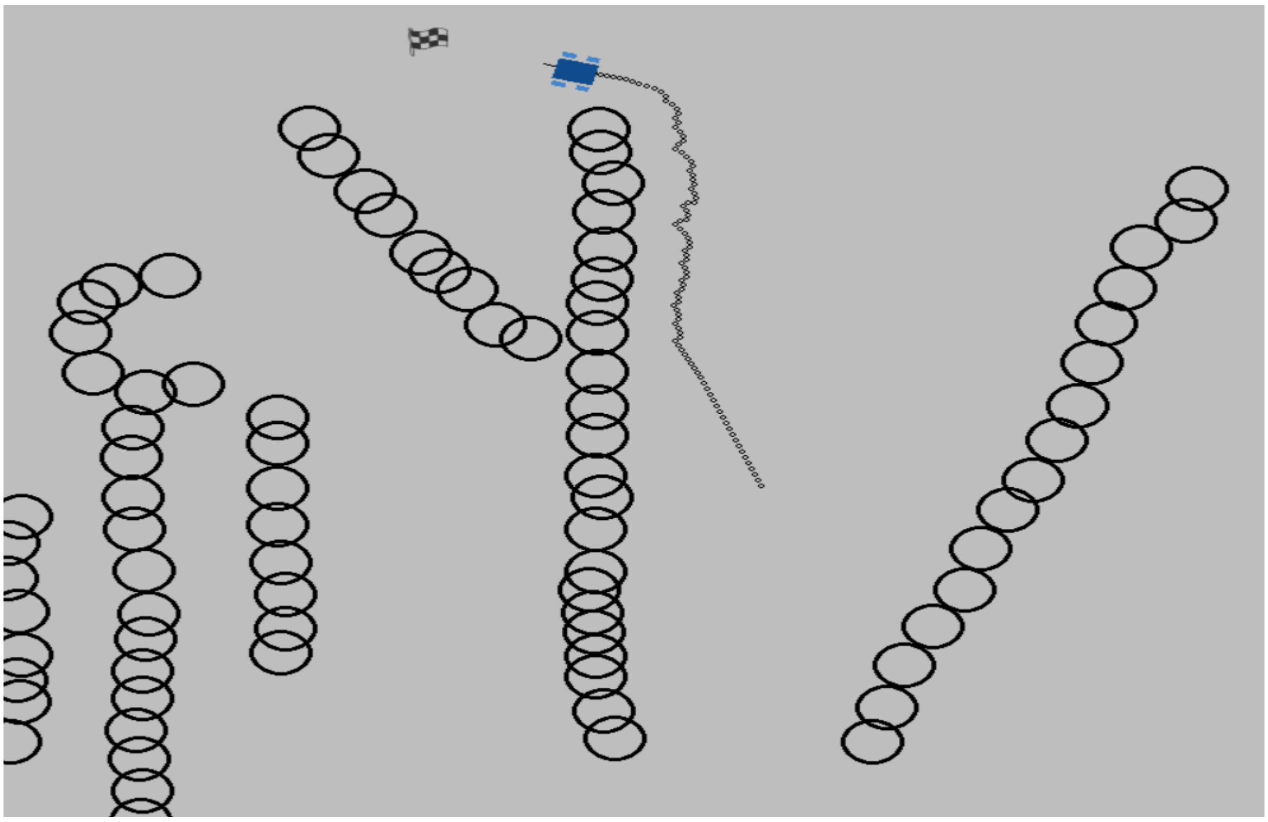


Figure 45: Progressive Solution for developed Sample Space05 (with HVFF)

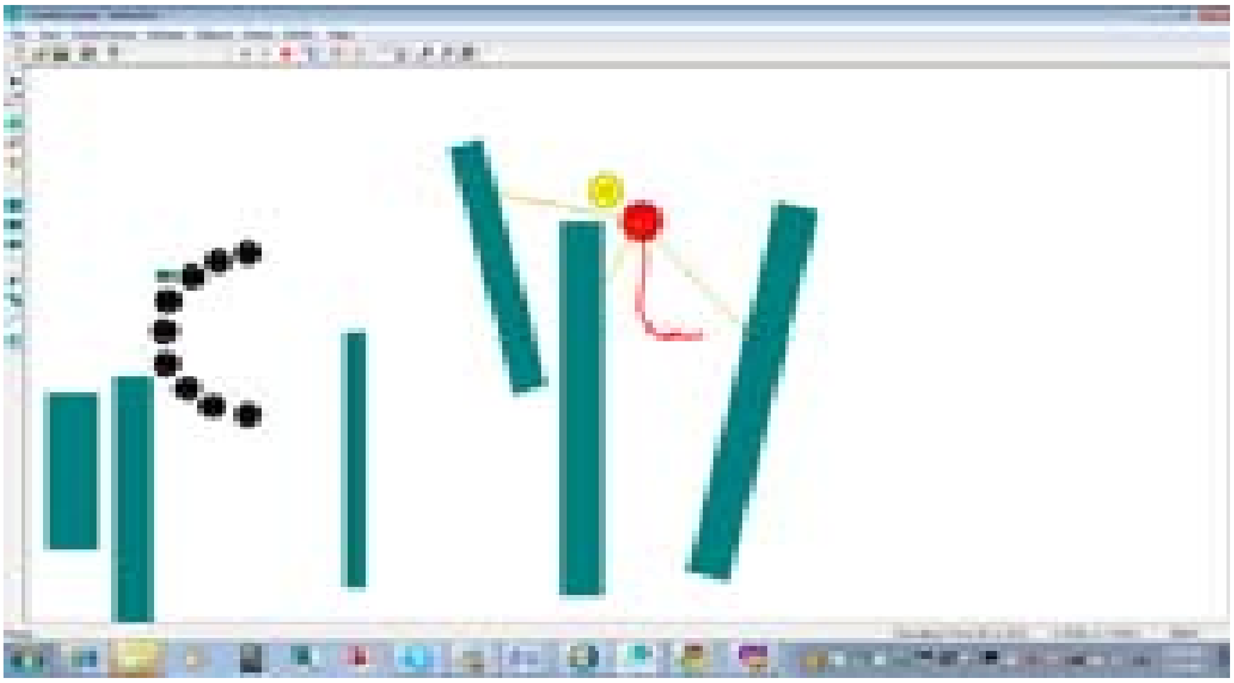


Figure 46: Progressive Solution for developed Sample Space05, adapted from [41]

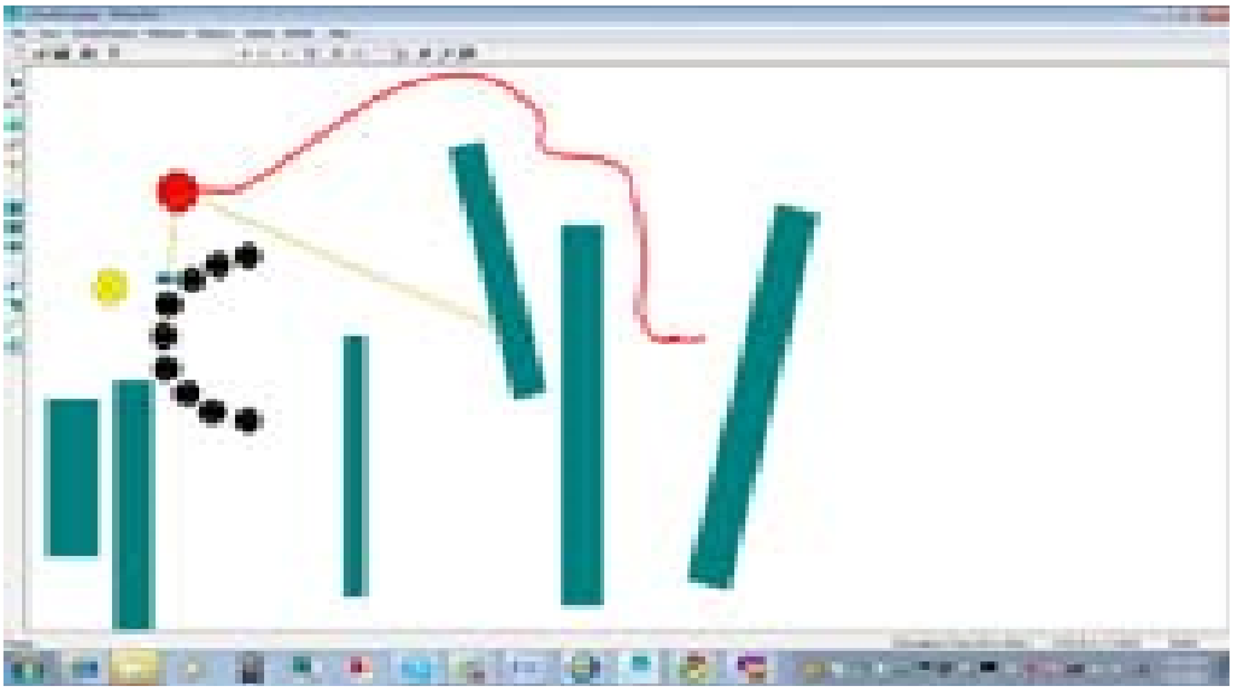


Figure 47: Progressive Solution for developed Sample Space05, adapted from [41]

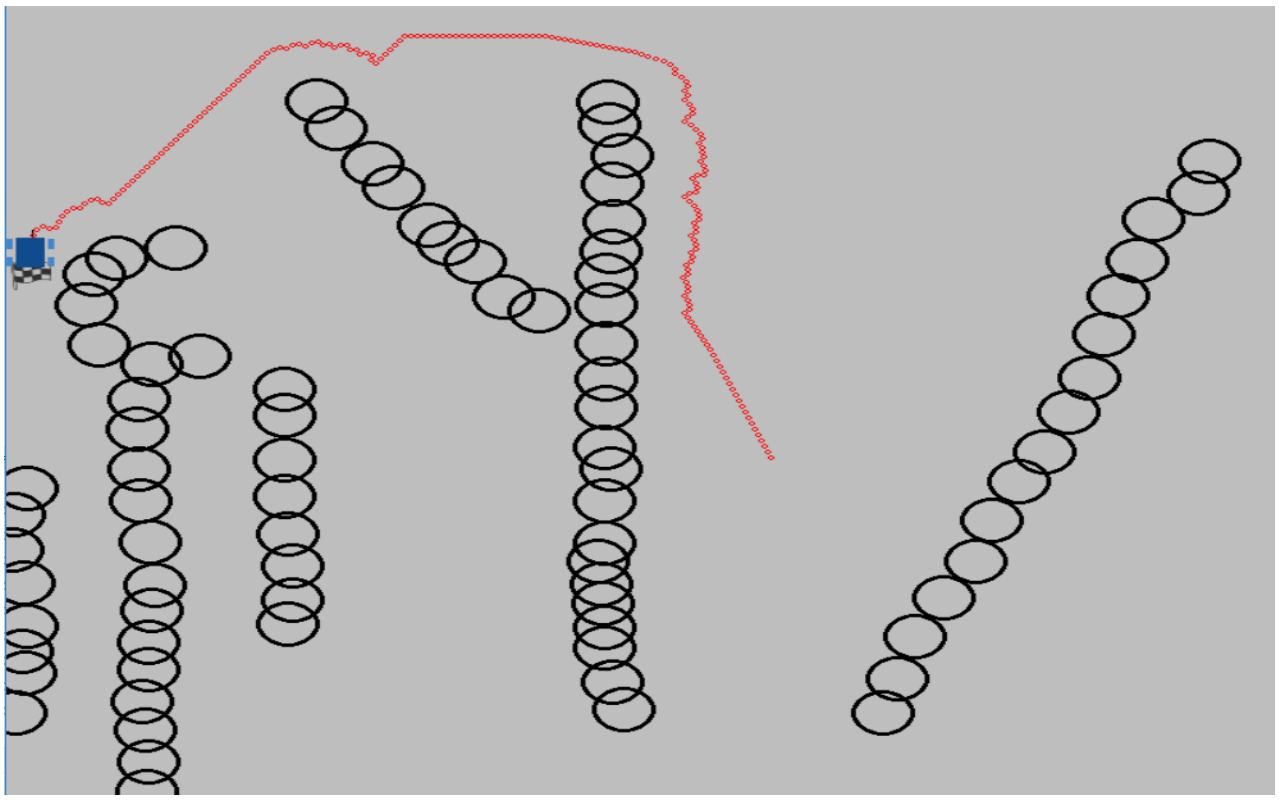


Figure 48: Final Solution for developed Sample Space05 (with HVFF)

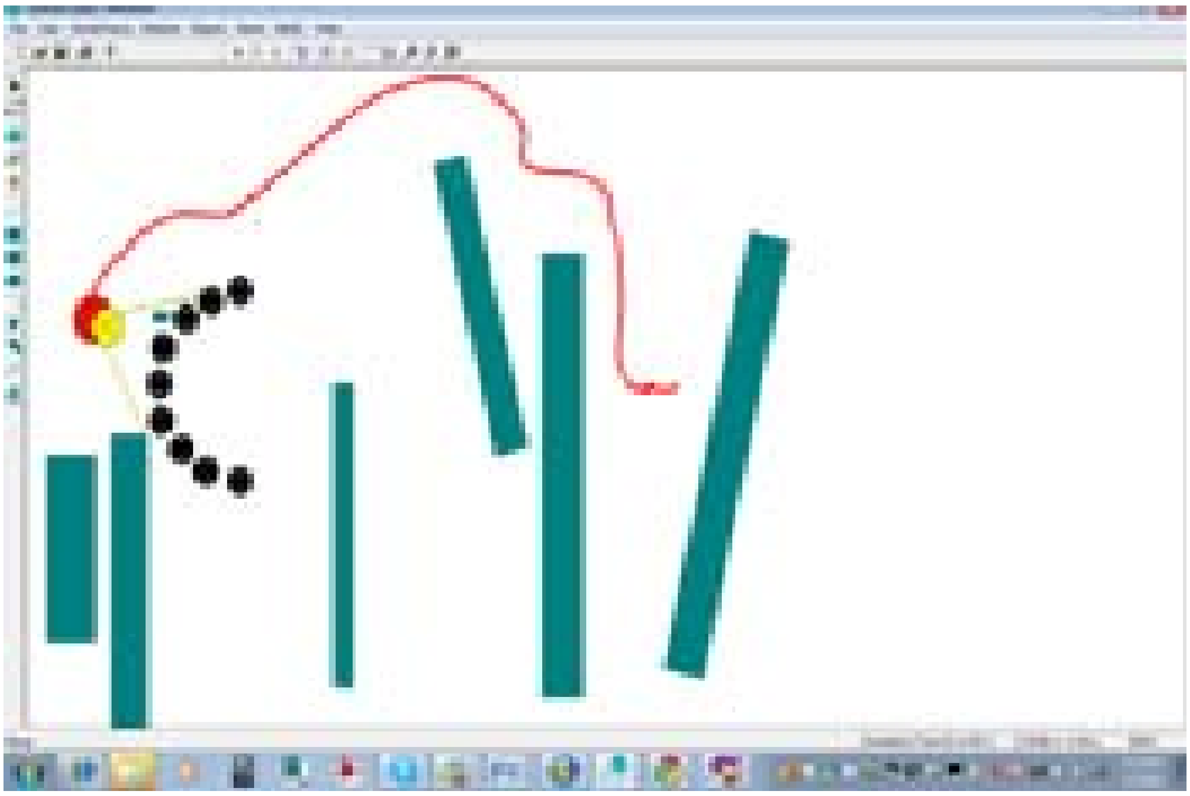


Figure 49: Final Solution for developed Sample Space05, adapted from [41]

Table 8: Results for Case 5

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the TP	MR got trapped for at least 300.35 seconds, the simulation runs was stopped.	43.37

4.2.6 Case 6: Additional workspace scenario with static obstacle from paper Ref [41]

The workspace created for this case, it has a static obstacle (not easily detectable) which is a reproduced workspace from paper Ref [41]. Figure 50 to Figure 56 show the simulated results.

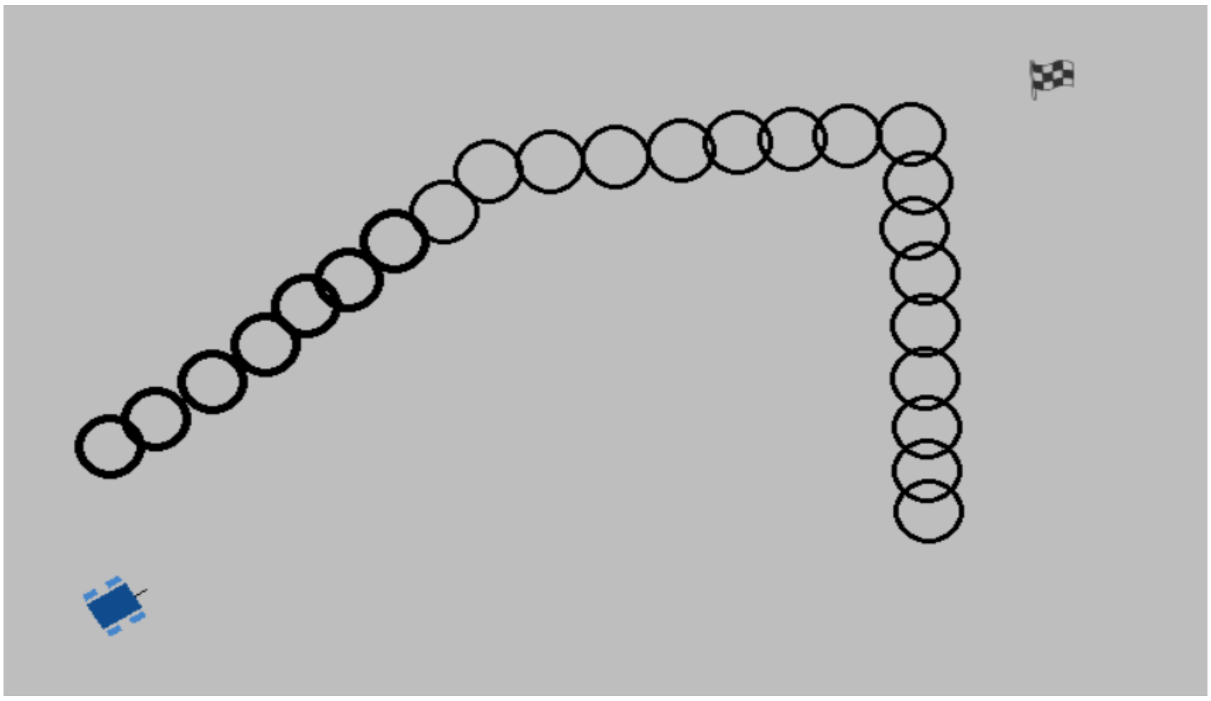


Figure 50: Developed Problem Sample Space06

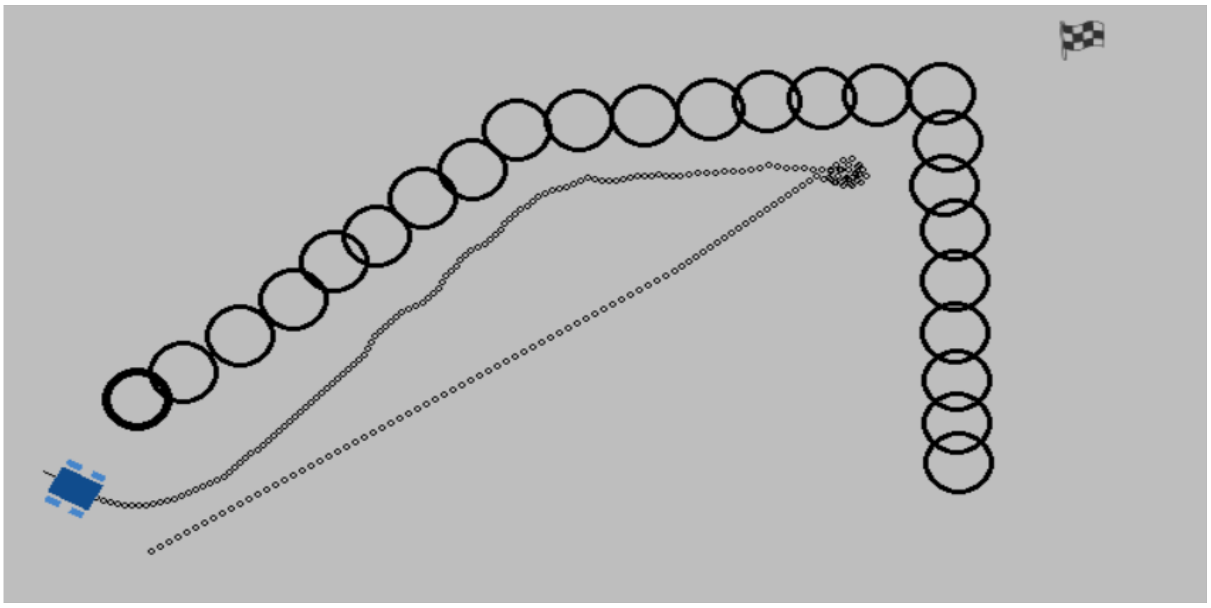


Figure 51: Progressive Solution for developed Sample Space06 (without HVFF)

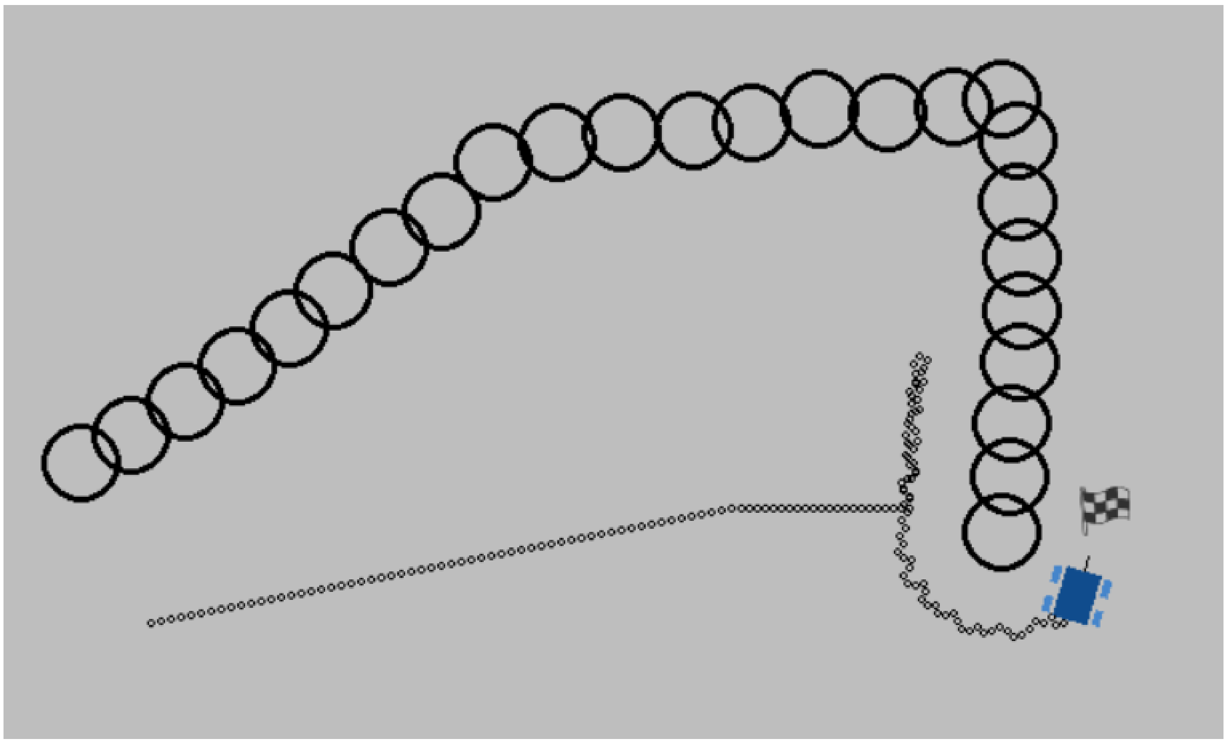


Figure 52: Progressive Solution for developed Sample Space06 (with HVFF)

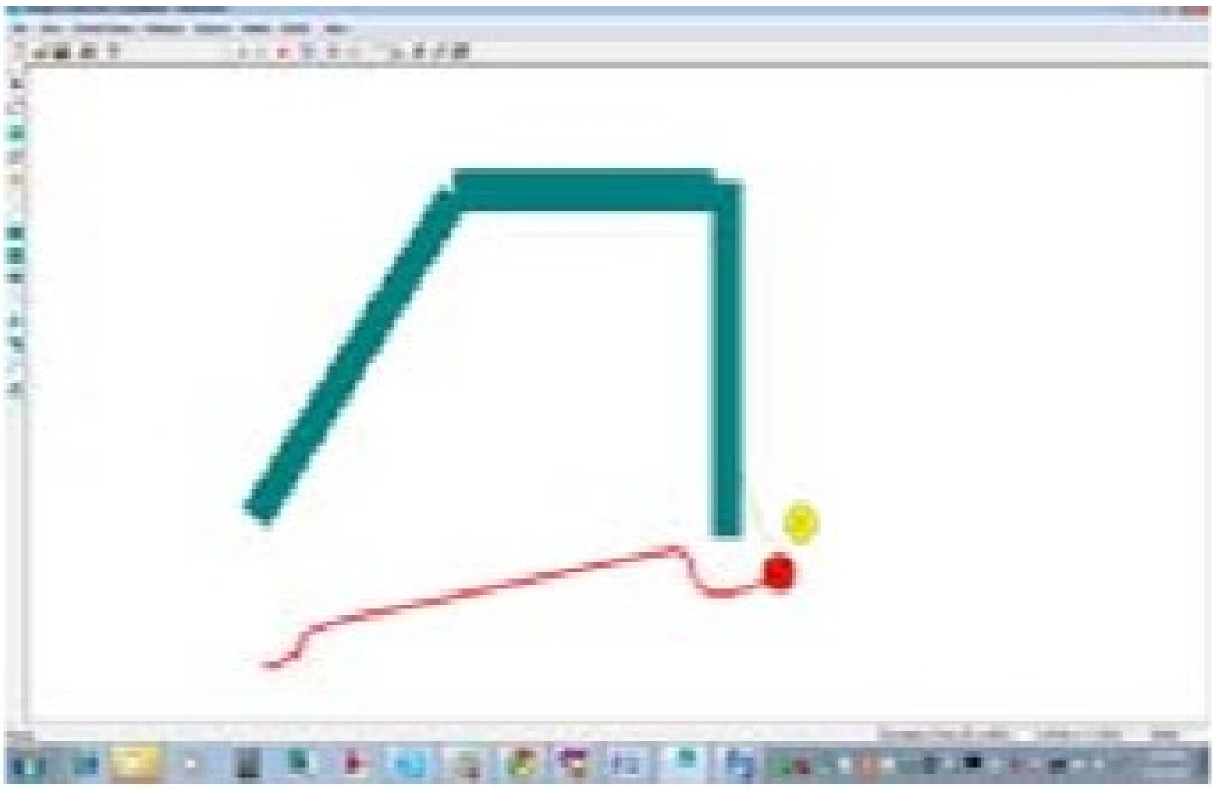


Figure 53: Progressive Solution for developed Sample Space06, adapted from [41]

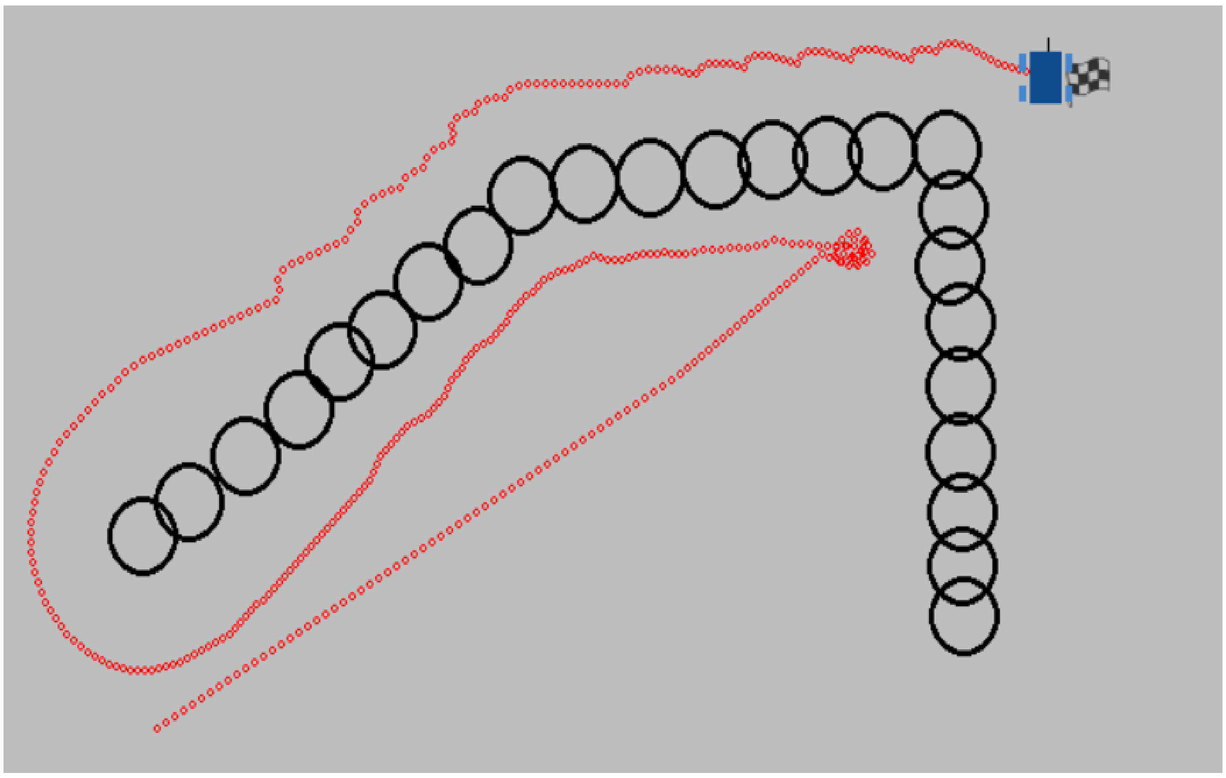


Figure 54: Final Solution for developed Sample Space06 (without HVFF)

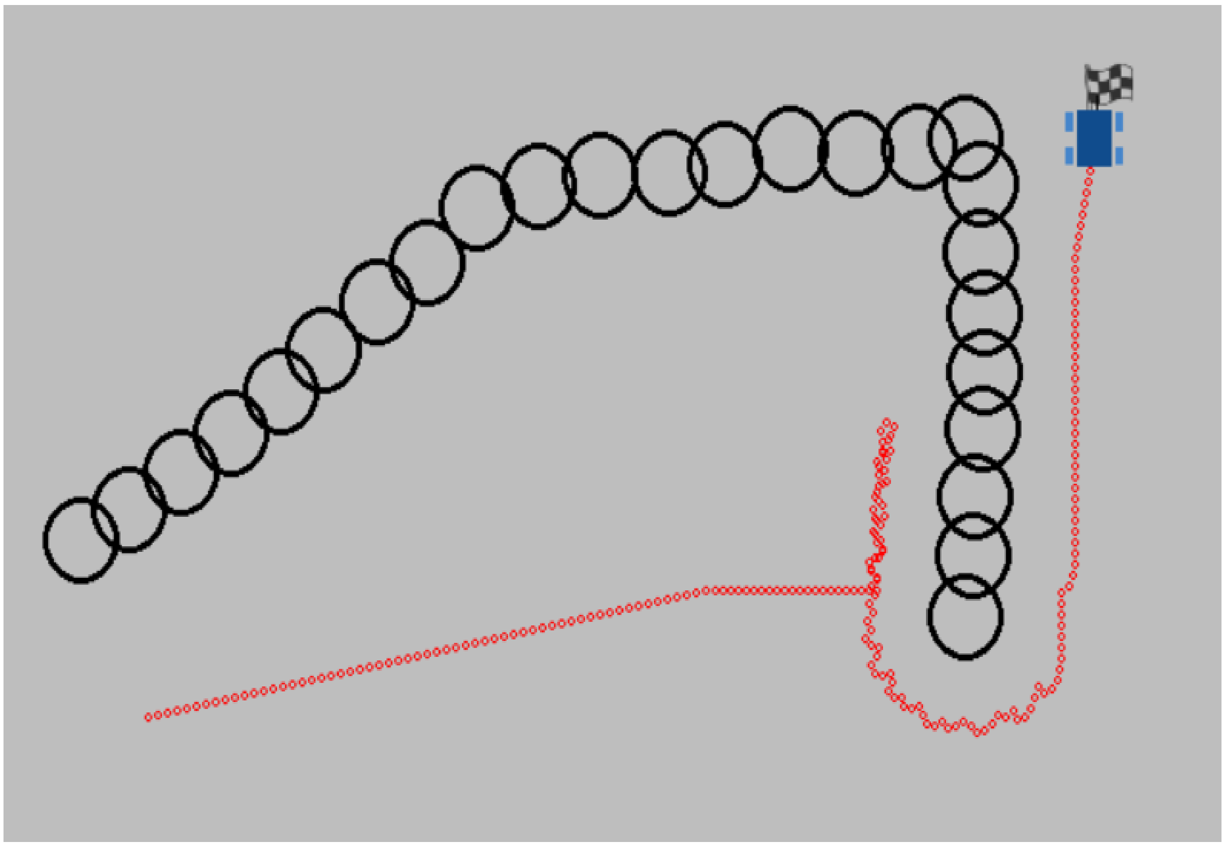


Figure 55: Final Solution for developed Sample Space06 (with HVFF)

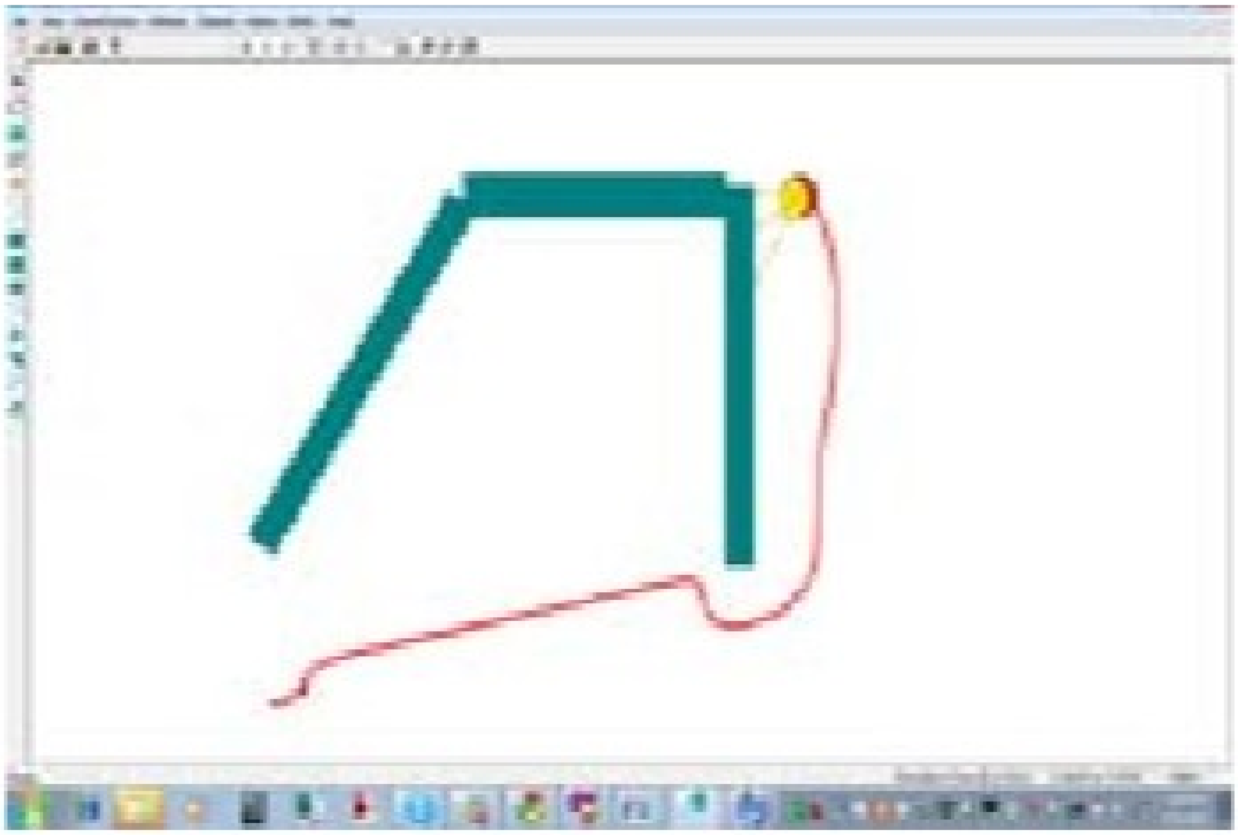


Figure 56: Final Solution for developed Sample Space06, adapted from [41]

Table 9: Results for Case 6

Time	Without HVFF	With HVFF
Time (seconds) from initial point to the TP	79.25	44.91

4.3 Analysis and Discussion of Results

Case 1 in the simulation series depicts navigation towards a simple concave shaped obstacle. Figure 18 presents the initialised workspace which comprises the robot, concave shaped obstacle along the trajectory of robot's navigation and the TP. Figure 19 presents a workspace with the navigating robot displaying a trajectory along the shortest path depicted by a straight line. This navigation is without the HVFF algorithm in place. The sense of direction is towards the hollow section of the concave hollow directly ahead of the TP. Figure 20 presents a navigation clip showing the robot's trajectory in the concave trap, some oscillatory forth back motion, trajectory of the robot off the concave trap and at the TP.

Sequel to the above discussion on case 1 without the use of HVFF algorithm, the current discussion presents MR navigation towards the same simple concave shaped obstacle with the HVFF algorithm in place. Figure 21 presents the navigation trajectory of the moving robot. The robot's trajectory was diverted off the concave trap with the aid of the VGC which is an integral part of the HVFF concept. On reaching the virtual goal (VG), the robot is now redirected to the real goal or the desired TP. The travel time of the robot without the HVFF concept is 46.52seconds while a duration of 26.90 was recorded with the deployment of the HVFF concept. In both scenarios i.e. with and without HVFF, the robot was able to reach it is TP however, at a shorter time with the use of the HVFF concept.

Case 2 in the simulation series depicts navigation towards an intermediate concave shaped obstacle (i.e. a concave shape with an enhanced difficulty). Figure 23 presents the initialised workspace which comprises the robot, concave shaped obstacle along the trajectory of the robot's navigation and the TP. Figure 24 presents a workspace with the navigating robot displaying a trajectory along the shortest path depicted by a straight line. This navigation is without the HVFF algorithm in place. The sense of direction like the previous is towards the hollow end of the concave directly ahead of the TP. Figure 25 presents a navigation frame showing the robot's trajectory in the concave trap with back forth oscillatory motion. The robot was trapped in the concave hollow. This was observed for a period of 320.46 seconds after which the simulation was stopped.

Following the above discussion on case 2 simulation without the use of HVFF algorithm, the discussion herein this subsection presents the MR navigation towards the same intermediate concave shaped obstacle and TP however, with the HVFF algorithm in place. Figure 26 shows the navigation trajectory of the robot towards the virtual goal (VG). The robot's trajectory was diverted off the concave trap with the aid of the VGC upon which the HVFF concept hinges. On reaching the virtual goal (VG), the robot was redirected to the desired TP as shown in figure 27. The travel time of the robot with the HVFF concept was 33.02seconds. Unlike in the simple scenario of concave obstacle, the robot got stuck here in the intermediate concave obstacle without the use of the HVFF algorithm.

Case 3 in the simulation series depicts navigation of the MR towards an undetectable concave shaped obstacle (i.e. a concave shape with an extremely wide apart vertex) as presented in figure 28. The initialised workspace in figure 28 comprises the robot, concave shaped obstacle along the trajectory of the robot's navigation and the TP. Figure 29 presents a workspace with the navigating robot displaying a trajectory along the shortest path depicted by a straight line. This navigation is without the HVFF algorithm in place. The sense of direction like the previous is towards the hollow end of the concave directly ahead of the TP. Figure 30 shows a trapped MR in the concave hollow with a back forth oscillatory motion. This directionless motion of the MR continued endlessly. This was observed for quite a long period without any sign of escape of the MR from the local minima trap. The simulation was eventually truncated.

Presented in the navigation control of figure 31 is the HVFF concept with the VGC actively in place. Figure 26 shows the navigation trajectory of the robot towards the virtual goal (VG). The robot's trajectory was diverted off the concave trap with the aid of the VG. On reaching the VG as shown in figure 31, the robot was redirected to the desired TP figure 32. The travel time of the robot with the HVFF concept in place was 53.20seconds. Based on the outcome of the last two simulation cases, it can be inferred that as the complexity of the concave shaped obstacle increases based on its width and depth, the likelihood of a MR getting stuck in the local minima trap significantly increases.

Case 4 in the simulation series depicts navigation of the MR in an environment of a few cascaded "simple concave shaped obstacles" amidst multiple TPs (2 Target Points). This is as represented in the initialised workspace shown in figure 33. Following this is figure 34 which presents the trajectory of a navigating robot without the HVFF algorithm. The robot got into the first simple concave trap, manoeuvred its way out and proceeded towards the first target point (TP1) as presented in figure 35. From this point, the robot continued to the second target point (TP2). Firstly, the robot got into a second local minima trap presented by the second concave obstacle. However, it got out of this in figure 36 and proceeded to the second target point (TP2) as shown in figure 37. The entire navigation duration from the start point of the robot through TP1 and TP2 was 85.56 seconds as distinctly presented in table 7.

Furthermore, in respect of case 4 simulation series, figure 38 presents the MR navigation from its initial position to TP1 with the aid of the HVFF algorithm. Unlike in the earlier discuss of case 4 without the HVFF concept, the navigation trajectory herein is premised on the prompting of the VG. Firstly, the MR was routed via the VG placement as seen in figure 38. This continued to the first target point (TP1) in figure 39. Following this was the prompting and placement of a second VG as shown in figure 40. This finally led the robot to TP2 as presented in figure 41. The combined navigation time with the HVFF algorithm was 62.30seconds as presented in table 7. It's obvious from the developed and presented workspaces as contained in this dissertation that the HVFF controlled navigation is quite efficient.

Presented in figure 42, case 5 in the simulation series is the initialised workspace combining both a concave shaped and a lengthy obstacle. Unlike the previous workspaces contained in figures 18 to 41, the current workspace was adapted from the literature. Figure 43 presents the navigation trajectory of the MR towards the target point without the use of the HVFF concept. Figure 44 shows a further progressive motion of the robot however with the robot running into a trap and never reaching the TP. On deploying the HVFF concept as seen in figure 45, the MR went in pursuit of the optimally positioned VG. This was in a bid to recover a MR navigation result from an adapted literature based labyrinth (see figures 46 and 47). Figure 48 presents the navigation trajectory of the MR to the TP in an optimised route at a duration of 43.37seconds as presented in table 8.This is in agreement with the output obtained from the literature as shown in figure 49.

Case 6 in the simulation series depicts navigation of the MR in a literature adapted concave shaped obstacle. The goal herein is to recover or improve on the navigation trajectory obtained in the literature. Figure 50 presents the initial workspace showing the robot and TP. Figure 51 shows the navigation of the MR into the concave hollow trap and its outward motion towards the TP not without some forth back oscillatory navigation while in the concave trap. This is a case of navigation without the HVFF concept. Figure 52 presents the trajectory of the robot to the TP. The overall time spent by the robot herein from start to completion is 79.25 seconds as contained in table 9.

Figure 53 on the other hand presents the navigation of the robot via the use of the HVFF algorithm. The robot was directed along the optimal path of the VG as seen in figure 53. This is in conformance with the output from the literature figure 54 and in a bid to show navigation trajectory optimality. The final navigation of the robot to the TP is as presented in figure 55 at a much shorter time of 44.91 seconds (see table 9). Figure 56 is a presentation of the complete navigation trajectory of the robot as obtained from the literature.

4.4 Chapter Summary

This chapter has presented simulation cases and results by validating the solution algorithm of this work. The results showed that the solution algorithm is effective but it is not time efficient without HVFF. The simulation time results of solution algorithm with HVFF are very efficient and effective as discussed in the previous section. Furthermore, it is also noticeable from Cases 2, 3, 5 and 6 that the complexity of concave-shaped and lengthiness of static obstacles increases the likelihood of MR getting struck in the local minimal trap.

CHAPTER FIVE

CONCLUSION AND FUTURE RESEARCH WORK

5.1 Chapter Overview

This chapter has detailed out a summary of the research work as presented in this dissertation. The research has presented an optimum mobile robot navigation model amidst static concave shaped and lengthy stretched out obstacles in a given navigation workspace. Furthermore, algorithmic validation was carried out on a single TP as well as multi TPs. The outcome of the simulated results as presented for each developed workspace as presented in the earlier chapters showed the efficacy of the algorithmic solutions. The use of reactive algorithms such as the HVFF technique has proven to be quite effective in trajectory control of robotic vehicles in obstacle constrained navigation problem domain.

5.2 Conclusion

This research has presented the deployment of the HVFF algorithm for MR navigation in an obstacle constrained workspace dominated with concave shaped or lengthy (stretched-out) obstacles. The degree of effectiveness of the HVFF algorithm in respect of MR optimum trajectory development in a workspace with this class of obstacles is remarkably high. As the level of constraint posed by workspace objects increased, the robot navigation success to the TP generally experienced some significant level of difficulties. However, on comparing the navigation output between the HVFF controlled trajectory and the non-HVFF i.e. conventional navigation trajectory, it was observed that the robot either never reached the TP or spent so much time trying to reach the TP after several back forth oscillations in the case of a conventional navigation approach. However, the scenario with the HVFF algorithm was such that the robot navigation recorded a significantly lesser travel duration to the TP.

Apart from validating the HVFF algorithm on developed workspaces, comparative studies was carried out on some existing literature workspaces comprising concave and lengthy obstacles. In all of these validations, the behaviour of the robot navigation with and without the HVFF algorithm per workspace was presented. Of a special interest is the navigation time of the MR with and without the HVFF algorithm per workspace as presented in table 10. The first simulation gave a completion time difference of 19.62 seconds resulting in a percentage time gain of 42.18% for the HVFF algorithm over the non-HVFF algorithm. The second simulation presented a navigation time difference of 287.44 seconds between the HVFF and non-HVFF algorithms. This resulted in a percentage time gain of 89.7% in favour of the HVFF algorithm over the non-HVFF algorithm. The third case of simulation presented an undefined navigation time difference between the HVFF and non-HVFF algorithm. This was premised on the fact that the non-HVFF algorithm was unable to get the MR out of the concave trap all through the monitoring duration. The outcome herein clearly shows the edge of the HVFF algorithm over the non-HVFF algorithm. The fourth case of simulation presented a cumulative navigation time difference of 23.26 seconds between the HVFF and non-HVFF algorithm. This resulted in an equivalent of 53.91% time gain in respect of the HVFF algorithm. The fifth and sixth cases respectively presented a simulation time difference of 256.98 and 34.34 seconds between the

HVFF and non-HVFF algorithm. This resulted in an equivalent of 85.56% and 43.33% time gain respectively in respect of the HVFF algorithm.

Table 10: Summary of Navigation Travel Duration and Percentage Gain in Efficiency

Simulation Cases	Workspace Type	Completion Time (Non-HVFF Approach)	Completion Time (HVFF Approach)	Time Difference	Percentage (%) Difference in Time
Case 1	Simple concave	46.52	26.90	19.62	42.18
Case 2	Intermediate concave	320.46	33.02	287.44	89.70
Case 3	Advanced concave	∞	53.3	undefined	undefined
Case 4	Multi-Target Points (TP1)	40.79	31.80	8.99	22.04
Case 4	Multi-Target Points (TP2)	44.77	30.50	14.27	31.87
Case 5	Lengthy Obstacle	300.35	43.37	256.98	85.56
Case 6	Concave Shaped	79.25	44.91	34.34	43.33

Research Objectives Addressed

- Explore and adapt the HVFF concept for effectiveness and efficiency studies of robot navigation in workspaces with concave or lengthy-stretched out obstacles – Chapter 4, page: 66-75 (Case 1 to 3) and 84 – 92 (Case 5 to 6)
- Adapt the HVFF algorithm to a multi-target point navigation problem – Chapter 4 (Case 4), page: 76 – 83.
- Validate the HVFF algorithmic performance in the different workspaces using simulation trials premised on the Python software - Chapter 4 (Case 1-6), page: 66 – 92.

5.3 Future Research Work

This subsection presents some future work to be considered in a bid to extend this research. Future work herein would focus on the following underlisted tasks:

i. Local Minima Trap Entry Prevention during MR navigation: The current research did not effectively deploy the virtual obstacle concept (VOC) which is capable of preventing the navigating robot from getting into a local minima trap orchestrated by a concave obstacle in particular. In most instances, the HVFF approach works with both the VGC and VOC. This can greatly impact on the overall navigation efficiency and effectiveness.

ii. Dynamic Obstacles Navigation Analysis: The need to explore a workspace scenario with a cluster of dynamic obstacles or a mixed scenario of static and dynamic obstacles has been reserved for a future research.

iii. Real Vehicle Deployment of Navigation Algorithm: The deployment of the HVFF scheme on real robotic vehicles is already in the pipeline. Three robotic vehicles were recently acquired for deployment of the HVFF algorithm and control of these vehicles autonomously.

REFERENCES

[1] Li H and Savkin AV. 2018. An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. *Robotics and Computer Integrated Manufacturing.* no: 54, pp 65-82

[2] Abiyev R, Ibrahim D and Erin B. 2010. Navigation of mobile robots in the presence of obstacles. *Advances in Engineering Software*. no :41, pp 1179-1186.

[3] Fu KS, Gonzalez RC and Lee CSG. 1987. Robotics: Control, Senisng, Vision and Intelligence. *McGraw Hill*.

[4] Shneier, M. and Bostelman, R. 2015. *Literature Review of Mobile Robots for Manufacturing*. [Online]. Available at https://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.8022.pdf. [Accessed 22 September 2020].

[5] Patle BK, Babu GL, Pandey A, Parhi DRK and Jagedeesh A. 2019. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*. no:15, pp 582-606.

[6] Pandey A, Pandey S and Parhi DR. 2017. Mobile robot navigation and obstacle avoidance techniques: A review. pp 96-105.

[7] Taheri H and Xia ZC. 2021. SLAM; definition and evolution. *Engineering Applications of Artificial Intelligence*.

[8] Moreno-Armendariz MA and Calvo H. 2014. Visual SLAM and Obstacle Avoidance in Real Time for Mobile Robots Navigation. *International Conference on Mechatronics, Electronics and Automotive Engineering*. pp 44-49.

[9] Iizuka S, Nakamura T and Suzuki S. 2014. Robot navigation in dynamic environment using Navigation function APF with SLAM. *10th France-Japan/ 8th Europe-Asia Congress on Mecatronics (MECATRONICS2014- Tokyo)*. pp 89-92.

[10] Sqalli MT *et al*. 2016. Improvement of a tele-presence robot autonomous navigation Using SLAM algorithm. *International Symposium on Micro-Nano Mechatronics and Human Science (MHS)*. pp 1-7.

[11] Song K *et al*. 2018. Navigation Control Design of a Mobile Robot by Integrating Obstacle Avoidance and LiDAR SLAM. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. pp 1833-1838.

[12]Kim P, Chen J and Cho YK. 2018. SLAM-driven robotic mapping and registration of 3D point clouds. *Automation in Construction*. pp 38-48.

[13] Liu Z. 2021 .Implementation of SLAM and path planning for mobile robots under ROS framework. *6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. pp 1096-1100.

[14] Gul F, Rahiman W and Alhady SSN. 2019. A comprehensive study for robot navigation techniques. *Cogent Engineering*.

[15] Ghorpade D, Thakare AD and Doiphode S. 2017. Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR. *International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. pp 1-6.

[16] Madhavan TR and Adharsh M. 2019. Obstacle Detection and Obstacle Avoidance Algorithm based on 2-D RPLiDAR. *International Conference on Computer Communication and Informatics (ICCCI).* pp 1-4.

[17] Baras N, Nantzios G, Ziouzios D and Dasygenis M. 2019. Autonomous Obstacle Avoidance Vehicle Using LIDAR and an Embedded System. *8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. pp 1-4.

[18] Dong H, Weng CY, Guo C, Yu H and Chen IM. 2020. Real-time Avoidance Strategy of Dynamic Obstacles via Half Model-free Detection and Tracking with 2D Lidar for Mobile Robots. *IEEE/ASME Transactions on Mechatronics*.

[19] Ren Yee PD, Pinrath N and Matsuhira N. 2020. Autonomous Mobile Robot Navigation Using 2D LiDAR and Inclined Laser Rangefinder to Avoid a Lower Object. *59th Annual Conference of the Society of Instrument and Control Engineers (SICE)*. pp. 1404-1409

[20] Borenstein J and Koren Y. 1991. The vector field histogram–fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*. pp. 278–288.

[21] Ulrich I and Borenstein J. 1998. VFH+: reliable obstacle avoidance for fast mobile robots. *Proceedings 1998 IEEE International Conference on Robotics and Automation.* pp. 1572-1577.

[22] Ulrich I and Borenstein J. 2000. VFH/sup */: local obstacle avoidance with look-ahead verification. *IEEE International Conference on Robotics and Automation.* pp. 2505-2511.

[23] Babinec A, Dekan M, Duchon F and Vitko A. 2012. Modifications of VFH Navigation Methods for Mobile Robots. *Procedia Engineering.* no:48, pp 10-14.

[24] Yim WJ and Park JB. 2014. Analysis of mobile robot navigation using vector field histogram according to the number of sectors, the robot speed and the width of the path. *14th International Conference on Control, Automation and Systems (ICCAS 2014)*. pp. 1037-1040.

[25] Kumar JS and Kaleeswari R. 2016. Implementation of Vector Field Histogram based obstacle avoidance wheeled robot. *Online International Conference on Green Engineering and Technologies (IC-GET)*. pp. 1-6.

[26] Alagic E, Velagic J and Osmanovic A. 2019. Design of Mobile Robot Motion Framework based on Modified Vector Field Histogram. *International Symposium ELMAR*. pp 135-138.

[27] Diaz D and Marin L. 2020. VFH+D: An Improvement on the VFH+ Algorithm for Dynamic Obstacle Avoidance and Local Planning. *IFAC-PapersOnLine*. pp 9590-9595.

[28] Zhou L and Li W. 2014. Adaptive Artificial Potential Field Approach for Obstacle Avoidance Path Planning. *Seventh International Symposium on Computational Intelligence and Design*. pp 429-432.

[29] Borenstein, J. and Koren, Y. 1989. Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics*. pp 1179-1187.

[30] Khatib O. 1985.Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*. no: 1, pp 90-98.

[31] Chiang H, Malone N, Lesser K, Oishi M and Tapia L. 2015. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. *IEEE*

International Conference on Robotics and Automation (ICRA). pp 2347- 2354.

[32] Malone N, Chiang H, Lesser K, Oishi M and Tapia L. 2017. Hybrid Dynamic Moving Obstacle Avoidance Using a Stochastic Reachable Set-Based Potential Field. *IEEE Transactions on Robotics*. no: 5, pp 1124-1138.

[33] Sudhakara P, Ganaphy V, Priyadharshini B and Sundaran K. 2018. Obstacle Avoidance and Navigation Planning of a Wheeled Mobile Robot using Amended Artificial Potential Field Method. Procedia Computer Science. no:133, pp 998 – 1004.

[34] Lu XS, Li E and Guo R. 2020. An Obstacles Avoidance Algorithm Based on Improved Artificial Potential Field. *IEEE International Conference on Mechatronics and Automation (ICMA)*. pp. 425-430.

[35] Lin X, Wang Z and Chen X. 2020. Path Planning with Improved Artificial Potential Field Method Based on Decision Tree. *27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. pp. 1-5.

[36] Lazarowska A. 2019. Discrete Artificial Potential Field Approach to Mobile Robot Path Planning. *IFAC PapersOnLine*. no:52, pp 277-282.

[37] Shin Y and Kim E. 2021. Hybrid path planning using positioning risk and artificial potential fields. *Aerospace Science and Technology*. pp 106-640.

[38] Olunloyo VOS and Ayomoh MKO. 2009. Autonomous Mobile Robot Navigation Using Hybrid Virtual Force Field Concept. *European Journal of Scientific Research (EJSR).* no:31, pp 204-228.

[39] Olunloyo VOS, Ayomoh MKO and Ibidapo-Obe O. 2009. A path planning model for an autonomous vehicle in an unstructured obstacle domain. *Proceedings of the 14th IASTED International Conference.* pp 180 – 187.

[40] Olunloyo VOS and Ayomoh MKO. 2010. A Hybrid Path Planning Model for Autonomous Mobile Vehicle Navigation. *25th International Conference of CAD/CAM, Robotics & Factories of the Future Conference.* pp 1 – 12.

[41] Olunloyo VOS and Ayomoh MKO. 2011. An Efficient Path Planning Model in an Unstructured Obstacle Domain. *Proceedings of the IASTED International Conference Robotics and Applications*. pp 38-45.

[42] Zadeh LA. 1975. The Concept of a Linguistic Variable and its Application to Approximate Reasoning–I. *Information Science*. no:8, pp 199–249.

[43] Qing–yong BAO, Shun–ming LI, Wei–yan S and Mu-jin AN. 2009.A Fuzzy Behavior–Based Architecture for Mobile Robot Navigation in Unknown Environments. *International Conference on Artificial Intelligence and Computational Intelligence*. pp 257–261.

[44] Jaradat M, Garibeh M and Feilat EA. 2012. Autonomous mobile robot planning using

hybrid fuzzy potential field. *Soft Computing*. no:15, pp 153–164.

[45] Pandey RK, Sonkar KK, and Parhi DR. 2014. Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller. *IEEE 8th International Conference on Intelligent Systems and Control (ISCO)*. pp 39-41.

[46] Almasri MM, Elleithy KM and Alajlan AM. 2016. Development of efficient obstacle avoidance and line following mobile robot with the integration of fuzzy logic system in static and dynamic environments. *IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. pp 1-6.

[47] Singh NH and Thongam K. 2018. Mobile Robot Navigation Using Fuzzy Logic in Static Environments. *Procedia Computer Science*. no:125, pp 11-17.

[48] Batti H, Jabeur CB and Seddik H. 2019. Mobile Robot Obstacle Avoidance in labyrinth Environment Using Fuzzy Logic Approach. *International Conference on Control, Automation and Diagnosis (ICCAD)*. pp 1-5.

[49] Mohanty PK, Kundu S, Srivastava S and Dash RN. 2020. A New T-S Model Based Fuzzy Logic Approach For Mobile Robots Path Planning. *IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*. pp 476-480.

[50] Oleiwi BK, Mahfuz A and Roth H. 2021. Application of Fuzzy Logic for Collision Avoidance of Mobile Robots in Dynamic-Indoor Environments. *2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. pp 131 -136.

[51] Verbitsky NS, Chepin EV and Gridnev AA. 2018. Experimental studies of a convolutional neural network for application in the navigation system of a mobile robot. *Procedia Computer Science.* no:145, pp 611-616.

[52] Ribeiro D, Mateus A, Miraldo P and Nascimento JC. 2017. A real-time deep learning pedestrian detector for robot navigation. *Autonomous Robot Systems and Competitions (ICARSC*). pp 165–171.

[53] Pershina Z, Kazdorf and Abrosimov V. 2018. Application of algorithms for object recognition based on deep convolutional neural networks for visual navigation of a mobile robot. *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. pp. 1–2.

[54] Chi K and Lee MR. 2011. Obstacle avoidance in mobile robot using Neural Network. *International Conference on Consumer Electronics, Communications and Networks (CECNet)*. pp 5082-5085.

[55] Motlagh O, Nakhaeinia D, Tang SH, Karasfi B and Khaksar W. 2014. Automatic navigation of mobile robots in unknown environments. *Neural Computing and Applications*. no:24, pp1569–1581.

[56] Janglova D. 2004. Neural networks in mobile robot motion. *International Journal of Advanced Robotic Systems.* no:1, pp 15-22.

[57] Yu J, Ji J, Miao Z and Zhou J. 2019. Neural network-based region reaching formation control for multi-robot systems in obstacle environment. *Neurocomputing*. no:333, pp 11-21.

[58] Saleem KA, Jabri AA, Maashri WA, Maawali and Mesbah M. 2020. Obstacle-Avoidance Algorithm Using Deep Learning Based on RGBD Images and Robot Orientation. *7th International Conference on Electrical and Electronics Engineering (ICEEE)*. pp 268-272.

[59] Wei Hand Ye Q. 2020. Mobile Robot Obstacle Avoidance System Based on GA-Aided OIF-Elman Network. *4th International Conference on Robotics and Automation Sciences (ICRAS)*. pp 6-10.

[60] Tang X, Li L and Jiang B. 2014. Mobile robot SLAM method based on multi-agent particle swarm optimized particle filter.

[61] Algabri M, Hassan M, Hedjar R and Alsulaiman M. 2015.Comparative study of soft computing technique for mobile robot navigation in an environment.

[62] Atyabi A, Phon-Amnuaisuk S, Ho CK. 2010. Applying area extension PSO in Robotic Swarm. *J Intell Robot Syst*. no:58, pp 253-285.

[63] Lin C, Li TS, Kuo P and Wang Y. 2016. Integrated particle swarm optimization algorithm-based obstacle avoidance control design for home service robot. *Computers and Electrical Engineering.* no:56, pp 748 – 762.

[64] Meerza SIA, M. Islam M and Uzzal MM. 2018. Optimal Path Planning Algorithm for Swarm of Robots Using Particle Swarm Optimization Technique. *3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*. pp 330-334.

[65] Alaliyat S,Oucheikh R and Hameed I. 2019. Path Planning in Dynamic Environment Using Particle Swarm Optimization Algorithm. *8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*. pp. 1-5.

[66] Tian S, Li Y, Kang Y and Xia J. 2021. Multi-robot path planning in wireless sensor networks based on jump mechanism PSO and safety gap obstacle avoidance. *Future Generation Computer Systems*. no:118, pp 37-47.

[67] Bremermann HJ. 1958. The evolution of intelligence. The Nervous system as a model of its environment. Washington, Seattle: Dept. Mathematics, University

[68] Holland JH. 1975. Adaptation in natural and artificial systems. University of Michigan Press.

[69] Xia J, Michalewicz Z, Zhang L and Trojanowski K. 1997. Adaptive evolutionary planner/ navigator for mobile robot. *Transcation on Evolutionary Computation.*

[70] Shi P and Cui Y. Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. *Proceedings of the Chinese Control and decision Conference*.

[71] Patle BK, Babu L G,Pandey A,Parhi DRK and Jagadeesh A. 2019. Review: On path planning strategies for navigation of mobile robot. *Defence Technology.* no:15, pp 582-606.

[72] Patle BK, K Parhi DR, Jagadeesh A and Kashyap SK. 2018. Matrix-binary codes based genetic algorithm for path planning of mobile robot. *Computers and Electrical Engineering*. no: 67, pp 708-728.

[73] Germi SB, Khosravi MA and Fard RF. 2018. Adaptive GA-based Potential Field Algorithm for Collision-free Path Planning of Mobile Robots in Dynamic

Environments. *6th RSI International Conference on Robotics and Mechatronics (IcRoM)*. pp. 28-33.

[74] Choueiry S, Owayjan M, Diab H and Achkar R. 2019. Mobile Robot Path Planning Using Genetic Algorithm in a Static Environment. *Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*. pp 1-6.

[75] Lopez-Gonzalez A, Campana JAM and Martinez EGH. 2020. Multi robot distance based formation using Parallel Genetic Algorithm. *Applied Soft Computing Journal*. no:86, pp 105- 929.

[76] Aghda SAF and Mirfakhraei M. 2020. Improved routing in dynamic environments with moving obstacles using a hybrid Fuzzy-Genetic algorithm. Future Generation Computer System. no:112, pp 250-257.

[77] Guan-Zheng T, Huan HE and Aaron S. 2007. Ant colony system algorithm for real time globally optimal path planning of mobile robots.

[78] Chen Y, Su F and Shen LC. 2009. Improved ant colony algorithm based on PRM for UAV route planning.

[79] Zhangqi W, Xiaoguang Z and Qingyao H. 2011. Mobile Robot Path Planning based on Parameter Optimization Ant Colony Algorithm. Procedia Engineering. no:15, pp 2738 – 2741.

[80] Wang H, Wang ZA, Yu L, Wang X and Liu C. 2018. Ant Colony Optimization with Improved Potential Field Heuristic for Robot Path Planning. *37th Chinese Control Conference (CCC)*. pp. 5317-5321.

[81] Yi Z, Yanan Z and Xiangde L. 2019. Path Planning of Multiple Industrial Mobile Robots Based on Ant Colony Algorithm. *16th International Computer Conference on Wavelet Active Media Technology and Information Processing*. pp. 406-409.

[82] Ma Y *et al.* 2020. Obstacle avoidance path planning of unmanned submarine vehicle in ocean current environment based on improved firework-ant colony algorithm. *Computers and Electrical Engineering.* no:87, pp 106- 773.

[83] Zhao H. 2020. Optimal Path Planning for Robot Based on Ant Colony Algorithm. *International Wireless Communications and Mobile Computing (IWCMC)*. pp 671-675.

[84] Paniagua AH, Rodriguez MAV, Ferruz J and Pavon N. 2015. Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. *Soft Comput*. no:21 , pp 949-964.

[85] Brand M and Yu X. 2013. Autonomous robot path optimisation using firefly algorithm. *International conference on machine learning and cybernetics*. pp 1028-1032.

[86] Sutantyo D and Levi P. 2015. Decentralized underwater multi robot communication using bio-inspired approaches. *Artif Life Robot*. no: 20, pp 152-158.

[87] Sutantyo D, Levi P, Moslinger C and Read M. 2013. Collective-adaptive levy flight for underwater multi-robot exploration. *International conference on mechatronics and automation*. pp 456 – 462.

[88] Mitic M and Miljkovic Z. 2015. Bio-inspired approach to learning robot motion trajectories and visual control commands. *Expert Syst Appl*. No: 42, pp 2624-2637.

[89] Sadhu AK, Konar A, Bhattacharjee T and Das S. 2018. Synergism of firefly algorithm and Q-learning for robot arm path planning. *Swarm and Evolutionary Computation*.

[90] Patle BK, Patel B, Pandey A, Sahu O and Parhi DRK. 2019. Analysis of Firefly-Fuzzy Hybrid Controller for Wheeled Mobile Robot. *3rd International Conference on Computing and Communications Technologies (ICCCT)*. pp 187-194.

[91] Patle BK, Pandey A, Jagadeesh A and Parhi DRK. 2018. Path planning in uncertain environment by using firefly algorithm. *Defence Technology*. no: 14, pp 691-701.

[92] Li F, Fan X and Hou Z. 2020. A Firefly Algorithm with Self-Adaptive Population Size for Global Path Planning of Mobile Robot. *IEEE Access*. pp. 168951-168964.

[93] Chi W, Wang J, Ding Z, Chen G and Sun L. 2021. A Reusable Generalized Voronoi Diagram Based Feature Tree for Fast Robot Motion Planning in Trapped Environments. *IEEE Sensors Journal*.

[94] Wahyunggoro O and Cahyadi AI. 2016. Quadrotor path planning based on modified fuzzy cell decomposition algorithm. *Telkomnika*. no:14, pp 655-664.

[95] Zhou C and Liu J. 2019. Navigation System for Mobile Robot using RGBD Sensor based on Probabilistic Roadmaps. *IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. pp 1258-1261.

[96] Aggarwal A, Kukreja A and Chopra P. 2010. Vision based collision avoidance by plotting a virtual obstacle on depth map. *IEEE International Conference on Information and Automation*. pp 532-536.

[97] Ravankar A, Ravankar A, Hoshino Y and Kobayashi Y. 2019. Virtual Obstacles for Safe Mobile Robot Navigation. *8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. pp 552-555.

[98] Ravankar A, Ravankar A, Hoshino Y, Watanabe M and Rawankar A. 2020. Transient Virtual Obstacles for Safe Robot Navigation in Indoor Environments. *International Institute of Applied Informatics*. pp 59-69.

[99] Zhang Y, Ge R, Lyu L, Zhang J, Lyu C and Yang X. 2020. A Virtual End-to-End Learning System for Robot Navigation Based on Temporal Dependencies. *IEEE Access*. pp 134111-134123, 2020.

[100] Yousuf S and Kadri MB. 2020. Implementation of Modified Tangent Bug Navigation Algorithm for Front Wheel Steered and Differential-Drive Robots. *International Symposium on Recent Advances in Electrical Engineering & Computer Sciences (RAEE & CS)*. pp 1-6.

[101] M MJ, Mathew R and Hiremath SS. 2019. Reinforcement Learning Based Approach for Mobile Robot Navigation. *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. pp 523-526.

[102] Liu B, Xiao X and Stone P. 2021. A Lifelong Learning Approach to Mobile Robot Navigation. *IEEE Robotics and Automation Letters*. no 2, pp 1090-1096.

[103] Xie L *et al*. 2021. Learning with Stochastic Guidance for Robot Navigation. *IEEE Transactions on Neural Networks and Learning Systems*. no 1, pp 166-176.

CODING APPENDIX

Main py-game simulation code

```
''A Python Class to implement a very basic Pygame Plotting
Arena. Plot various objects, animate them, scaling, dealing
with mouse inputs, Updating etc'''

import pygame, sys, math
from pygame.locals import *
import Sprites as spr
import random

pygame.init()
class PyArena:
  def __init__(self, w = 800,h = 800, bgColor = "Grey", axisColor = "Grey", TIMER_DELAY = 30):
    '''Create a Pygame Arena, TIMER_DELAY sets the freq of screen updating in milliseconds'''
    self.w = w
    self.cx = w//2
    self.h = h
    self.cy = h//2
    self.bgColor = bgColor # the background color
    self.axisColor = axisColor # Color of the axes
    self.delay = TIMER_DELAY # The screen refresh  delay im milliseconds
    self.screen = pygame.display.set_mode((self.w, self.h))
    self.clock = pygame.time.Clock()
    pygame.time.set_timer(USEREVENT+1, TIMER_DELAY) # Start the timer, assign it a USEREVENT
    self.counter = 0 # Init a counter to keep track of time passed
    self.obstacles = [] # Empty list to hold all the sprites
    self.robots = []
    self.goals = []
    self.swarming = False # One robot or Swarm?
    self.manualBotID = 0 # Which bot is currently under manual control

  def ReDraw(self):
    '''ReDraw on the Objects on the Screen'''
```

```
        self.screen.fill(pygame.Color(self.bgColor)) # Clear the screen

        #draw the axes
        pygame.draw.lines(self.screen, pygame.Color(self.axisColor), False, ((self.cx,0),(self.cx,self.h))) #
Y Axis
        pygame.draw.lines(self.screen, pygame.Color(self.axisColor), False, ((0,self.cy),(self.w,self.cy))) #
X Axis

        # Update and then Draw all the sprites
        for sprites in self.obstacles, self.goals, self.robots:

            for sprite in sprites:

                if self.counter % 7 == 0: #Sprites update every 00 ms
                    sprite.Update(self.obstacles)
                    ''' Update all the sprites,  Fixed sprites
                    do nothing during Update(). Robots may need to re-position, scan for targets,
                    zombie sprites may just move a little...'''

                sprite.PygameDraw(self.screen)
                ''' Each sprite must be able to draw itself onto the screen
                Rather than sprites passing info to this class, and implementing
                the draw function here, it's simpler if every sprite knows
                how to draw itself. Thus we can have different sub-classes of sprites
                with different drawing attributes, and we can add
                more sub-classes in the future that implement other drawing
                attributes, without having to change any of the code here'''

        #After everything is re-drawn, update the screen
        pygame.display.update()
    def GetManualBot(self):
        '''Returns the currently active robot under
        manual control'''

        return self.robots[self.manualBotID]
```

```
def ObeyRobotCommands(self,event):
    '''Deal with any User genrated Robot Commands'''
    bot = self.GetManualBot() # Find the robot currently under manual control
    if event.key == pygame.K_UP: # UP Arrow pressed, step robot forwards
        bot.StepForward(1)
    if event.key == pygame.K_DOWN: # Down arrow pressed, step robot backwards
        bot.StepBackward(1)
    if event.key == pygame.K_LEFT: #Left arrow pressed turn left
        bot.TurnLeft()
    if event.key == pygame.K_RIGHT: # Right arrow pressed turn right
        bot.TurnRight()
    if event.key == pygame.K_a: # Toggle the robot's auto mode
        bot.ToggleAutoControl()
def AddObstacles(self, event):
    '''Add any user genrated obstacles'''

    if event.key == pygame.K_o: # On pressing 'O' key, an obstacle is added at the mouse curcor
        pos = self.WH2XY(pygame.mouse.get_pos())
        nObs = len(self.obstacles) # New sprite will have ID = len(obstacles) + 1
        self.obstacles.append(spr.Sprite(nObs + 1, pos)) # Add an obstacle at the mouse position

        # Lines below added for debugging pruposes
        '''print("Creating....,")
        for s in self.obstacles:
            print(s.pos)''
def  Add_MoveGoal(self, event):
    '''Move the goal or if in swarming mode, add more goals'''

    if event.key == pygame.K_g: # On pressing 'G' key, the goal is moved to the mouse cursor posn
        pos = self.WH2XY(pygame.mouse.get_pos())
        if self.swarming == True: #multiple robots and goals
            pass
        else:
            self.goals[0].SetPos(pos)
            self.robots[0].SetGoal(pos) # Single robot single goal seeking
            self.robots[0].SetSpeed(5) # Get the robot moving
    def Update(self):
        '''Perform all mandatory repeated tasks'''
       #Deal with user inputs

        for event in pygame.event.get():
            if event.type == USEREVENT+1: #Timer has ticked
                self.counter += 1 # Increment the counter
                # Re-Draw the screen
                self.ReDraw() # ReDraw will also update ALL sprites

            if event.type == pygame.QUIT or (
                    event.type == pygame.KEYDOWN and event.key == pygame.K_q):  # USER CLOSES
PROGRAM
                pygame.quit()
                sys.exit()

            if event.type == pygame.KEYDOWN: # Deal with User Commands

                #Obey User Robot Commands
                self.ObeyRobotCommands(event)

                #Obey User Goal Commands
                self.Add_MoveGoal(event)

                #Add User Generated Obstacles
                self.AddObstacles(event)

    def WH2XY(self,screenWH):
        '''Convert a pygame Screen WH coordinates to Coordinate axis
        XY value tuple'''
        return (screenWH[0] - self.cx, self.cy - screenWH[1])
    def AddDebugObstacles(self, t):
        '''Create a bunch of obstacles for debugging purposes'''
        for pos in t:
```

```
        nObs = len(self.obstacles) # New sprite will have ID = len(obstacles) + 1
        self.obstacles.append(spr.Sprite(nObs + 1, pos)) # Add an obstacle at the mouse position
if __name__ == "__main__":
    arena1 = PyArena() # Create an Arena

    wmax, hmax = arena1.screen.get_width(), arena1.screen.get_height()

    #Create a goal
    pos = arena1.WH2XY((200,150))
    goal1 = spr.Sprite(500, pos)
    goal1.SetImage("images/goal1.png")
    arena1.goals.append(goal1)

    # Create a robot
    pos = arena1.WH2XY((90, 700))
    rob1 = spr.GoalSeeker(1000, pos)
    rob1.SetImage("images/robot11.png")
    rob1.SetHeading(random.randint(0,360))
    rob1.SetSpeed(5)
    rob1.SetSensor(300, 70)
    rob1.SetGoal(goal1.GetPos())
    arena1.robots.append(rob1)




    while True:
        arena1.Update()



        # print(spr.Robot.__init__(spr.Robot, "Test", [100]).trail)
```

Repulsive Navigation Code

```
'''Python class that implements Repulsive Obstacle
Avoidance and Goal Seeking Navigation'''
import math
import pdb
#Define gravitational constants
G_GOAL = 1.0
G_OBS = 1.0
#Define global functions
def NormaliseVector(v):
    '''Return the normalised vector'''
    d = math.sqrt(v[0]*v[0] + v[1]*v[1])
    return [v[0]/d, v[1]/d]
#p1 robot position
#p2 goal position
def ObstacleForce(p1, p2, rSafe):
    '''Determine the gravitational force between two
    points. Returns a vector'''
    d = math.sqrt(float(math.pow((p1[0] - p2[0]),2) + math.pow((p1[1] -
p2[1]),2))) # distance
    if d <= rSafe: # if the obstacle is closer than rSafe, make sure dSqr
is a valid value
        dSqr = 0.000001 # if inside safety radius, dSqr should be very very
small
    else:
        dSqr = math.pow((d - rSafe),2) # Gravity force is maximum when d =
rSafe

    F = G_OBS/dSqr # Repulsive force proportional to 1/dSqr, the force that
the obstacle and mobile robot repel each other.
                   #the greater the force makes the robot not to touch the
obstacle and move in a different direction searching for the goal
    unit_vec = [(p1[0] - p2[0])/d, (p1[1] - p2[1])/d]
    f_vec = (unit_vec[0] * F, unit_vec[1] * F)
    return f_vec, unit_vec

#implementation code
class RepulsorNav:
    def __init__(self, p1, p2, rSafe = 2):
        '''Create the instance of this class'''
        self.ownPos = p1 #robot position
        self.goalPos = p2 #goal position
        self.safetyRadius = rSafe # introduce a safety radius to cater to
robot actual dimensions
        self.obstacles = [] # create an empty list of obstacles

    def __str__(self):
        '''default print function'''
        return "Default Print not defined yet"

    def SetOwnPos(self, p):
        '''Update own position'''
        self.ownPos = p

    def UpdateObstacles(self, obs):
        '''Update the list of obstacles in sight'''
        self.obstacles = obs
    def Navigate(self, edgeholding = False):
        '''Determine the best route to goal and
        return this direction vector
        By default, the navigator will not do edgeholding '''

        steerVec = [0,0] #steering vector
        x1, y1 = self.ownPos
        if edgeholding == False:
            steerVec = [self.goalPos[0] - x1, self.goalPos[1] - y1]  #calc
direct vector to goal

        if len(self.obstacles) > 0: #if there are obstacles in 'sight'
            for obs in self.obstacles: #iterate through all obstacles
                obsForce, unit_vec = ObstacleForce(self.ownPos, obs,
self.safetyRadius)
                #print("OwnPos {op}, Obstacle {ob}, Unit Vec {uv}, ObsForce
{obf}".format(op=self.ownPos,ob=obs,uv = unit_vec, obf=obsForce))

                steerVec = [steerVec[0] + obsForce[0] , steerVec[1] +
obsForce[1]]

        v = NormaliseVector(steerVec) # output the normailised recommended
vector
        c1 = ((math.pi/2) - math.atan2(v[1], v[0]))%(2 * math.pi) # Course
to Steer

        return v,c1 #based on this the robot turn into whatever is
recommended

    def SetGoal(self, newGoal):
        '''Go to a new goal'''
        self.goalPos = newGoal  #use this partially to assist the robot
when trapped in deep concave and trapped. this set new goal as virtaual
goal.

if __name__ == "__main__":

    op = [0,0]
    gp = [10,10]

    r1 = RepulsorNav(op,gp)

    obstacles = [[6,2]]#, [6, 4.2], [6,12], [17,8]]

    r1.UpdateObstacles(obstacles)

    r1.Navigate()
```

Sprites Code

```
'''Sprite class implements basic Sprite functionality
for display in a PyGame Arena'''

import pygame, math
from pygame.locals import *
import RepulsorNav as rpn # Goal Seeking and Obstacle Avoidance Class
import time

# Define constants

HDG_MARKER_LENGTH = 20 # length of heading marker to plot on screen
ROBOT_SAFETY_RADIUS = 50 # Safety circle around the robot
RANGE_REDUCTION = 4 # If trapped, reduce the detection range to avoid
getting lost
#Define global functions

def PlotBearing(pos1, pos2):
        '''Returns the bearing in degrees of pos2 [x2, y2]
        from pos1 [x1, y1] on an equal scale plot'''
        dx = float(pos2[0] - pos1[0])
        dy = float(pos2[1] - pos1[1])
        b = math.atan2(dy,dx)

        if b <=0:
            final = math.pi/2 + abs(b)
        elif b > 0:
            final = (math.pi/2 - b)%(math.pi * 2)

        return final

def FindCentreBearing(hdg, rec):
    '''Given two bearing lines, find the central bearing'''

    h = (math.sin(hdg), math.cos(hdg))
    r = (math.sin(rec), math.cos(rec))

    f = (h[0] + r[0], h[1] + r[1])

    theta = math.atan2(f[1], f[0])

    if theta <=0:
        final = (math.pi/2 + abs(theta))
    elif theta > 0:
        final = (math.pi/2 - theta)%(math.pi * 2)
    return final


def Distance2D(p1, p2):
    '''Returns the 2D distance between two points'''
    x1, y1 = p1[0], p1[1]
    x2, y2 = p2[0], p2[1]
    d = math.sqrt(math.pow(x1 - x2,2) + math.pow(y1 - y2,2))
    return d

class Sprite:
    '''Parent class for all plotting objects'''
    def __init__(self, ID, pos, radius = 20, lineThick = 3, color =
"Black"):
        self.ID = ID
        self.pos = pos
        self.color = color
        self.radius = radius
        self.lineThickness = lineThick
        self.hasImage = False # By default, sprites are not associated with
any image
```

```
    def __str__(self):
        txt = "Sprite, ID={i}, Pos={p}, color =
{c}".format(i=self.ID,p=self.pos, c= self.color)
        return txt

    def SetImage(self, imgFile):
        '''Assign an image to this sprite'''
        try:
            self.image = pygame.image.load(imgFile) # Load an image
            self.hasImage = True
        except:
            print("Error Loading Image for Sprite ID
{i}".format(i=self.ID))
            raise
            self.hasImage = False

    def GetPos(self):
        '''Returns the position tuple of the Sprite'''
        return self.pos

    def XY2WH(self, cx, cy, plotXY):
        '''Convert a normal XY coordinate tuple to Pygame's WH so that
        it fits nicely on the Corrdinate Axis'''

        return (cx + plotXY[0], cy - plotXY[1])

    def SetPos(self, pos):
        '''Set a new position for this sprite'''
        self.pos = pos

    def Update(self, obstacles):
        '''If anything needs to be updated at regular intervals,
        put it here...
        Arguments: counter value, counter increment delay (milliseconds)
        Dead sprites usually dont need to update anything at all.
        Active Sprites can overload this Update() method and do whatever
they
        need to at each update.'''

        pass

    def PygameDraw(self, screen, hdg = 0):
        '''Draw itself onto the screen argument. Child Classes can
implement
        overloaded PygameDraw() methods for more complex drawing operations

        Hdg (degrees) argument is used to rotate the image CCW before
drawing '''
        cx, cy = screen.get_height()//2, screen.get_width()//2

        drawPos = self.XY2WH(cx, cy, self.pos)

        if self.hasImage: # Draw the sprite's image

            img = pygame.transform.rotate(self.image, -hdg)
            imgSize = img.get_size()[0]//2
            screen.blit(img, ((self.XY2WH(cx,cy,self.pos))[0] - imgSize,
                (self.XY2WH(cx, cy, self.pos)[1] - imgSize))) # draw image

        else: # Draw a circle shape
```

```
                pygame.draw.circle(screen,
                        pygame.Color(self.color),
                        drawPos,
                        self.radius,
                        self.lineThickness)

#Define Robot Class


class Robot(Sprite): #
    def __init__(self, ID, pos, goal = (0,0), color =
pygame.Color("Black")):
        '''Create instance of robot class'''
        #Initialise attirbutes of Parent Sprite Classs
        Sprite.__init__(self, ID, pos, color)
        self.SetHeading(math.radians(0)) # Internal to this class, heading
is in radians
        self.stepTaken = False # just a flag to help with keyboard control
see pygame.K_UP

        self.trail = [] # A record of robot's position
        self.TIME_COUNT = 0 # counter to keep track of life time elapsed

    def __str__(self):
        t = "Robot Pos {p1}, Going to {g}, heading {c}, speed
{s}".format(p1 = self.pos, g = self.repulsorNav.goalPos, c =
math.degrees(self.hdg), s = self.spd)

        return t

    def PygameDraw(self, screen):
        '''Overloaded function of the Parent Sprite class
        PygameDraw() Function. The robot needs to draw all its
        extra items like heading marker, visible obstacles etc. and then
        finally call the parent class PygameDraw function to draw its
        body/ image'''

        # Draw the heading marker
        cx, cy = screen.get_height()//2, screen.get_width()//2
        p1 = self.XY2WH(cx, cy, self.marker[0])
        p2 = self.XY2WH(cx, cy, self.marker[1])
        pygame.draw.lines(screen, pygame.Color("Black"), False, (p1, p2)) #
Robot's heading marker

        #TO DO Draw the Robot's Trail
        for pos in self.trail:
            if self.runMode == "Finished":
                pygame.draw.circle(screen, pygame.Color("Red"),
self.XY2WH(cx, cy, pos), 2, 1)
            else:
                pygame.draw.circle(screen, pygame.Color(self.color),
self.XY2WH(cx, cy, pos), 2, 1)


        # Draw the robot's body by calling PygameDraw in the parent class
        super().PygameDraw(screen, math.degrees(self.hdg)) # pygame takes
rotation angle in degrees

    def SetSpeed(self, speed):
        '''Set the robot's speed'''
        self.spd = speed

    def SetHeading(self, hdg):
        '''Set the robot's heading. Heading Agument is in radians!!!'''
        self.hdg = hdg
        self.ResetHeadingMarker()

    def ResetHeadingMarker(self):
        '''Recalculate the heading marker'''

        self.marker = (self.pos, (self.pos[0] + HDG_MARKER_LENGTH *
math.sin(self.hdg), self.pos[1] + HDG_MARKER_LENGTH * math.cos(self.hdg)))

    def ResetStepFlag(self):
        '''Flip the value of stepTaken Flag'''
        self.stepTaken = False

    def GetHeadingMarker(self):
        '''Returns the heading marker tuple'''
        return self.marker

    def SetSensor(self, r1, a1):
        '''Set the parameters of the Robot's Sensor'''
        self.sensor_range = r1
        self.sensor_angle = a1

    def StepForward(self, nSteps = 1):
        '''The robot steps
        forward nSteps at a time'''
        sx = self.pos[0] + nSteps * self.spd * math.sin(self.hdg)
        sy = self.pos[1] + nSteps * self.spd * math.cos(self.hdg)
        self.pos = (int(sx),int(sy))
        #print("Step Taken:{s}".format(s=self.pos))
        self.stepTaken = True
        self.ResetHeadingMarker()
        self.trail.append(self.pos) # add position to trail

    def StepBackward(self, n):
        '''The robot steps backwards'''
        self.spd *= -1 # reverse the speed
        self.StepForward(nSteps = n) # take a step forward (with negative
speed)
        self.spd *= -1 # set the speed to its orginal value
        self.stepTaken = True
        self.ResetHeadingMarker()
        self.trail.append(self.pos) # add position to trail


    def ResetStepFlag(self):
        '''Flip the value of stepTaken Flag'''
        self.stepTaken = False



    def TurnRight(self, angle = 10):
        '''The robot turns clockwise'''
        self.hdg = (self.hdg + 0.01745 * angle) % (math.pi * 2) # default
is turn 5 degrees at a time
        self.ResetHeadingMarker()
    #def TurnLeft(self, angle=10):
        ''' turns anti-clockwise'''
        #self.TurnRight(angle * -1)  # turn right, but with a negative
angle
       # self.ResetHeadingMarker()


    def ToggleAutoControl(self):
        '''Set whether the robot will navigate in Automatic mode or manual
control'''
        self.autoControl = not(self.autoControl)


#Define GoalSeeker Class

class GoalSeeker(Robot):
    def __init__(self, ID, pos, goal = (0,0), color =
pygame.Color("Black")):
        Robot.__init__(self, ID, pos, color)
        self.size = ROBOT_SAFETY_RADIUS
        self.goal = goal
        self.repulsorNav = rpn.RepulsorNav(self.pos, self.goal,
                rSafe = self.size) # create a repulsor algorithm
        self.visible_obstacles = [] # list of currently visible obstacles
        self.runMode = "GoalSeek" # Start off in goal seeking mode
        self.runState = "Clear" # Start off with no obstacles in "sight"
        self.autoControl = False # Run in manual mode
        self.trapHistory = [] # List of flags showing if last ten steps
were 'trapped' or not
        self.maxEdgeHoldTime = 200 # How long to continue edheholding if
goal is not reached

    def __str__(self):
        '''Default print'''
        return "Not coded yet"

    def PygameDraw(self, screen):
        '''Overloaded function of parent class. Draw only the factors
affecting
        the Repulsor Algorithm'''

        cx, cy = screen.get_height()//2, screen.get_width()//2

        # Draw visible obstacles
        for obs in self.visible_obstacles:
            pygame.draw.circle(screen, pygame.Color("Black"),
self.XY2WH(cx, cy, obs.pos), 20, 5)

        #Call parent class and draw robot related everything else
        super().PygameDraw(screen)

    def SetGoal(self, goal):
        '''Set the Goal for the seeking algorithm'''
        self.goal = goal
        self.repulsorNav.SetGoal(self.goal)
        self.runMode = "GoalSeek" # Start Seeking the goal
        self.trapHistory = [] # Clear the trap history

    def ClearOfTrap(self):
        '''Check if the robot is clear of trap'''

        f1 = self.trapTrack > 3 * self.size # Walked enough from trap
        f2 = abs(PlotBearing(self.pos, self.goal) - 
self.goalBrgFromTrap)%360 < math.radians(40) # returned to goal line
        f3 = Distance2D(self.goal, self.pos) < self.goalDistFromTrap # 
Closer to goal than at trap point
        f4 = not(f3) and len(self.visible_obstacles) == 0 # We cleared the 
trap from the opposite side

        print(f1, f2, f3)

        if (f1 * f2 * f3) or f4: # If all three conditions are met.....
            self.trapHistory = [] # we are no longer trapped
            return True
        else:
            return False

    def IsTrapped(self):
        '''Returns True if the robot thinks it is trapped behind an 
obstacle(s)'''

        if len(self.trail) > 30: # Go at least 20 steps before chekcing if 
trapped or not
            d =  Distance2D(self.trail[-1], self.trail[-20])
            if d < self.size: #if robot is not making headway

                self.trapHistory.append(True) # record the 'trap'
                if len(self.trapHistory) > 20: # start checking after 10 
steps
                    # print(self.trapHistory)
                    self.trapHistory.pop(0) # get rid of the earliest flag

                    sum = 0
                    #print(self.trapHistory)
                    for i in range(len(self.trapHistory)):
                        sum += self.trapHistory[i]
                    if sum > 5: # if more than 5 entries are 'true'
                        #we are trapped. Record the necessary trap 
parameters and return True
                        self.trapPoint = self.pos
                        self.goalBrgFromTrap = PlotBearing(self.trapPoint, 
self.goal)
                        self.goalDistFromTrap = Distance2D(self.pos, 
self.goal)
                        return True
                    else:
                        return False
                else:
                    return False
            else:
                return False
        else:
            return False

    def Update(self, obstacles):

        '''Update the robot's status'''

        #Print the current state
        print("ID:{id}, Goal:{g}, Run Mode:{r}, Run State:{s}".format(id =
self.ID, g = self.goal, r= self.runMode, s = self.runState))
        #Refresh the visible obstacles

        self.visible_obstacles = [] # Clear the list of visible obstacles
for this robot

        '''Normally we run in simple "GoalSeek" mode. If the Robot gets
trapped,
        then it records the Goal Vector Lien and changes to "EdgeHold"
mode.
        It does not return to "GoalSeek" until it re-joins the original
Goal Vector Line'''

        if Distance2D(self.pos, self.goal) >= self.size/2: # Robot still
needs to go closer to goal

            if self.runMode == "GoalSeek":
                # print("IsTrapped returned:", self.IsTrapped())
                if self.IsTrapped():# Trapped in Goal Seek Mode
                    self.runState = "Trapped" # Record the change of
Running State
                    v1,v2 = self.pos, self.goal
                    self.savedGoalVector = rpn.NormaliseVector([v1[0] -
v2[0],v1[1] - v2[1]])
                    self.runMode = "EdgeHold" #Change to Edge Holding Mode
of Operation
                    self.trapTrack = 0 # Reset the length of track stepped
record from this trap
                    self.trapTime = time.time() # record the time when the
robot started Edgeholding
                    self.trapHistory = [] # Reset the trap history
                    self.Update(obstacles) # REDO this step after setting
mode to EdgeHold

                else: # In GoalSeek Mode, but not trapped
                    #Update visible obstacles
                    for obs in obstacles:
                        if Distance2D(self.pos, obs.pos) <=
self.sensor_range: # if the robot can 'see' the obstacle
                            dOG = Distance2D(obs.pos, self.goal)
                            dG = Distance2D(self.pos, self.goal)
                            if dOG <= dG: # if the obstacle is closer to
the goal than the robot...
                                self.visible_obstacles.append(obs)


                    if len(self.visible_obstacles) > 0:
                        self.runState = "Evade"
                    else:
                        self.runState = "Clear"


                    #Update Navigator and Calculate Recommended Steering
Vector
                    self.repulsorNav.SetOwnPos(self.pos) # update own
position to navigation algorithm
                    self.repulsorNav.UpdateObstacles([i.pos for i in
self.visible_obstacles])
                    recVector, recHdg = self.repulsorNav.Navigate() #
```

```
Recommended vector to steer
                    self.SetHeading(FindCentreBearing(self.hdg, recHdg))
#Turn halfway to recommended heading

            elif self.runMode == "EdgeHold":

                #Update visble Obstacles. Reduce sensor range to cut out
excess clutter

                if time.time() - self.trapTime > self.maxEdgeHoldTime:
                    self.runMode = "GoalSeek" # After two minutes of
Edgeholding, and we have still not rached the goal, try "GoalSeeking"again

                for obs in obstacles:

                    if Distance2D(self.pos, obs.pos) <=
self.sensor_range/RANGE_REDUCTION:
                        self.visible_obstacles.append(obs)

                #Steer by Repulsion without including the goal direction
                #Determine the current goal vector
                #if CurGoalVec X savedGoalVec is almost = 0, we areback on
the line
                #so, change back to "GoalSeek" Mode

                if self.ClearOfTrap(): # Clear of Trap. Go back to
Goalseeking mode
                    self.runMode = "GoalSeek"
                    self.Update(obstacles) # REDO this step after setting
mode to GoalSeek

                else: # Still in trap, continue edgeholding

                    self.trapTrack += self.spd # increment the track
stepped from this trap
                    self.repulsorNav.SetOwnPos(self.pos) # update own
position to navigation algorithm

                    if len(self.visible_obstacles) > 0:
                        self.repulsorNav.UpdateObstacles([i.pos for i in
self.visible_obstacles])
                        recVector, recHdg =
self.repulsorNav.Navigate(edgeholding = True) # Tell navigator we are
edgeholding
                        self.SetHeading(recHdg + math.pi/2.1) # turn right
and head out of trap

        else: # we are close enough to the goal to stop
            self.runMode = "Finished"
            self.hdg = 0
            self.spd = 0

        #If in AUTO control, Step Forward

        if self.autoControl == True:
            self.StepForward()
```