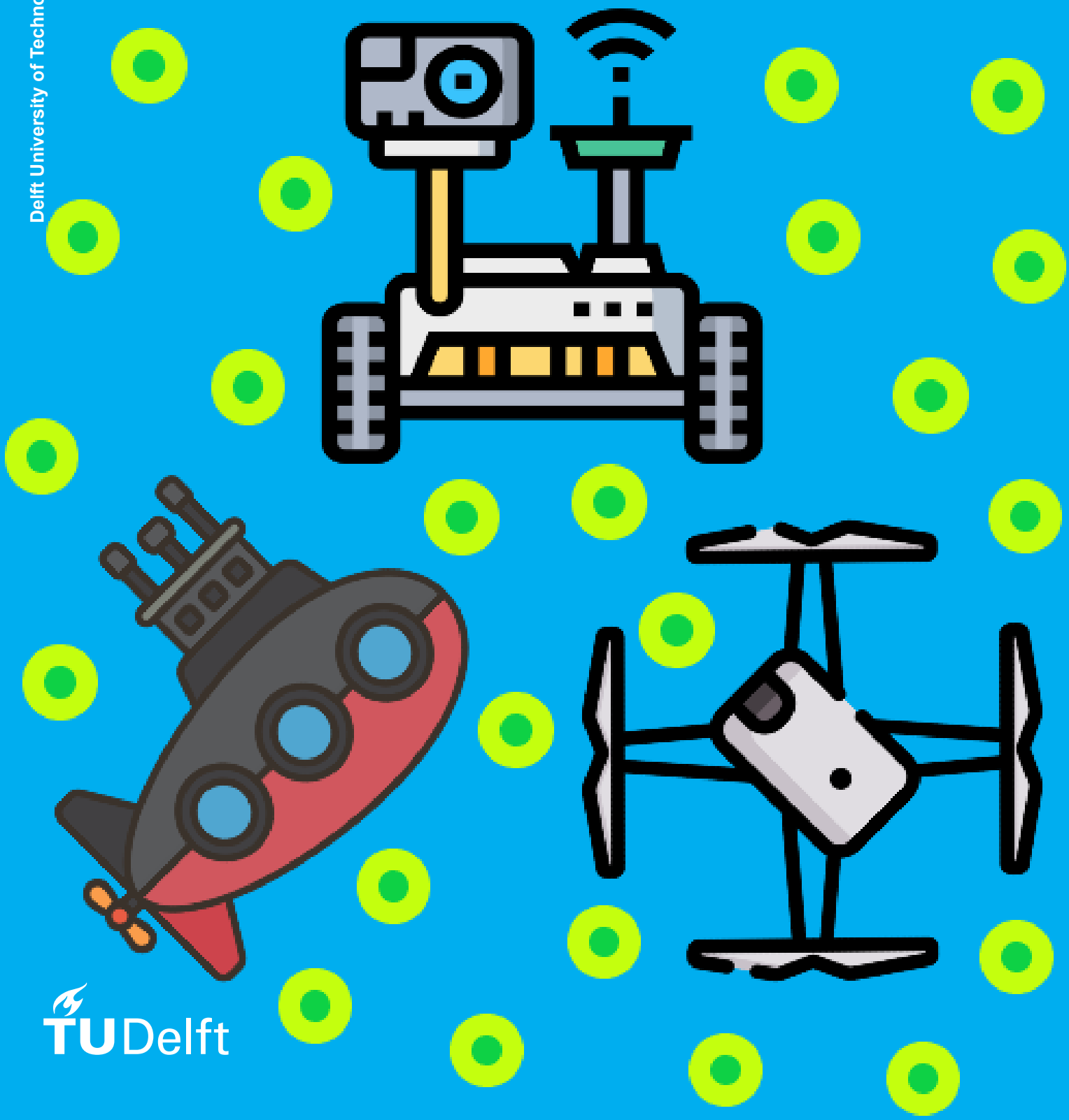
Detect and Avoid for Autonomous Agents in Cluttered Environments

M. Diab

Delft University of Technology



TU Delft

Detect and Avoid for Autonomous Agents in Cluttered Environments

by

M. D'ab

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 24, 2022 at 12:00 PM.

Student number:	5221331	
Project duration:	December 15, 2021 – August 24, 2022	
Thesis committee:	Dr. R. T. Rajan,	TU Delft, supervisor
	Dr. M. Mohammadkarimi,	TU Delft, daily supervisor
	Dr. Ir. C. J. M. Verhoeven,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis is part of completing the master program Electrical Engineering at the Delft University of Technology. It is done under the European Union project Airborne Data Collection on Resilient System Architectures. The mission of the project is to increase the acceptance and regulation of autonomous drones by advancing the technology used. I have focused on the detect and avoid technology that an autonomous agent must implement in order to navigate an unknown cluttered environment successfully.

I always liked automation and robotics and I believe that these fields can develop a lot of other industries and make the whole world work more efficiently. That is the reason why I decided to work on this project concerning autonomous agents that can be used in various applications such as delivery, surveillance, and search and rescue missions.

Furthermore, I enjoyed working on this topic as it was a challenge that made me discover the world of autonomous navigation algorithms and their applicable environments.

I am very grateful to all the supportive people that surrounded me throughout this journey, and who gave me guidance and advice. I want to give special thanks to my main supervisor, Dr. Raj Rajan, for his endless support and availability throughout our meetings and his instrumental feedback. Moreover, I want to thank my daily supervisor, Dr. Mostafa Mohammadkarimi, for his caring approach and for guiding me through the scientific methodology and how to contribute. We can now finally organize some bowling games!

Lastly, I would like to dedicate this to my dear mother and father for always being there for me. To my siblings for their constant motivation and support. To my friends for making this a delightful journey. To the TU Delft student community for being a wonderful and vibrant incubator.

M. Diab
Delft, August 2022

Abstract

Autonomous agents are the future of many services and industries such as delivery systems, surveillance and monitoring, and search and rescue missions. An important aspect in an autonomous agent is the navigation system it uses to traverse the environment. Not much emphasis has been paid in the past on autonomous agent navigation in cluttered environments. Cluttered and unknown environments such as forests and subaquatic environments need to have autonomous navigation systems developed just for them due to their uncertain and changing nature.

Path planning algorithms are used for the navigation of an autonomous agent in an environment. The agent needs to reach a target location while avoiding the obstacles it detects along the path. Such a system is called a Detect and Avoid (DAA) system and there are different implementations for it of which some are explored in this thesis.

The Artificial Potential Fields method or APF for short is a method for mobile agent navigation which is based on generating an attractive force on the agent from the target and a repulsive force from the obstacles. This leads to the agent reaching the target while avoiding the obstacles along the way. The Classical APF (CAPF) method works for structured environments well but not for cluttered environments. The CAPF method can be replaced with a modified version where the agent is surrounded by a set of points (called bacteria points) around its current location and the agent moves by selecting a bacteria point as a future location. This method is named the Bacteria APF (BAPF) method. This selection happens through combinatorial optimization based on the potential value of each bacteria point.

In this thesis, we propose two distinct contributions to the BAPF method. The first one being the use of an adaptive parameter in the repulsive cost function which is determined through a brute-force search. The second addition is a branching cost function that changes the value of the repulsive potential based on predefined perimeters around each obstacle. We show through simulations on densely and lightly cluttered environments that this Improved BAPF (IBAPF) method significantly improves the performance of the system in terms of the convergence to the target by almost 200% and reduced the time it takes to converge by around 25% as well as maintain the safety of the navigation route by keeping the average distance from obstacles around the same value.

Contents

List of Tables

List of Figures

Acronyms

2D Two Dimensional.

3D Three Dimensional.

A-BAPF Adaptive Bacteria Artificial Potential Fields.

APF Artificial Potential Fields.

AUV Autonomous Underwater Vehicle.

BAPF Bacteria Artificial Potential Fields.

CAPF Classical Artificial Potential Fields.

CAS Circuits and Systems.

CPU Central Processing Unit.

CR-BAPF Changing Radii Bacteria Artificial Potential Fields.

CURV Cable-Controlled Underwater Recovery Vehicle.

DAA Detect and Avoid.

DTT Distance to Target.

FAO Food and Agriculture Organization.

IBAPF Improved Bacteria Artificial Potential Fields.

IoT Internet of Things.

IR Infrared.

LiDAR Light Detection and Ranging.

NOAA National Oceanic and Atmospheric Administration.

RADAR Radio Detection and Ranging.

RAM Random Access Memory.

ROV Remotely Operated Vehicle.

SANS Semi-Autonomous Navigation System.

SLAM Simultaneous Localization and Mapping.

SONAR Sound Navigation and Ranging.

UAV Unattended Aerial Vehicle.

1 Introduction

1.1. Motivation

According to the Food and Agriculture Organization (FAO), jungles and forests cover about 4.06 billion hectares which is about 31% of the global land area of Earth [1]. These areas generate around 86 million jobs and for the regions where there is extreme poverty, more than 90% of the people there depend on forests for part of their livelihoods such as food and shelter [1]. Moreover, there are about 391,000 species of plants, 5000 amphibian species, 7500 bird species and over 3700 different species of mammals in forests. The forest market generates more than $580 billion per year considering employment [2]. These numbers represent the richness of forests in fauna and flora and also their great economic capabilities.

The accelerated urbanization and movement of people into larger cities creates a lack of monitoring and preservation of forest regions. Between 2019 and 2020, about 5.8 million hectares of temperate forests in Australia were burned due to wildfires [3]. To tackle these challenges and improve the execution of tasks related to forests and jungles, the concept of Forest 4.0 emerges, as can be seen in Figure 1.1.

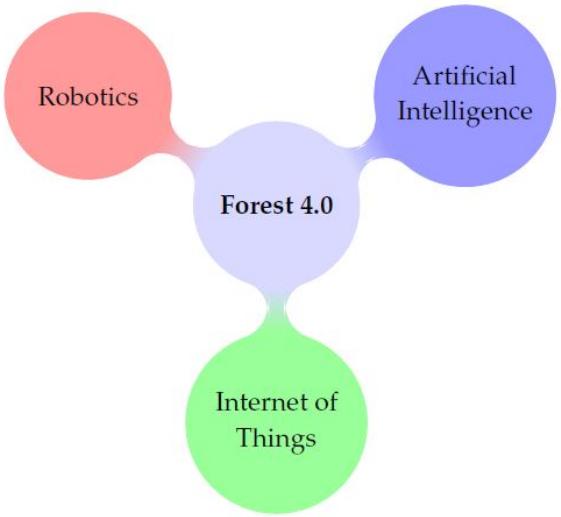


Figure 1.1: Forest 4.0 model [4]

Forest 4.0 became prominent with the use of robotic systems in forestry tasks such as environmental monitoring, fire prevention, planting, and harvesting [5].

Agricultural robots have hundreds of applications around the world, there are not as many applications for robots in forests and jungles yet. The reason for this disparity can be tributed to the fact that

forest environments are generally more complex than agricultural environments. The strongly unstructured and steep slope terrains in addition to the variations in temperature and humidity make it difficult to develop robust robotic systems. There is also no communication infrastructure in remote forests and communication links are obstructed by the dense vegetation which adds to the challenge. Forest robots need special hardware and locomotion systems to operate in these environments where there is no protection against eventual storms [4].

Forests are just one example of a cluttered environment that a robot needs to navigate, another example is subaquatic environments. The National Oceanic and Atmospheric Administration (NOAA) estimates that about 95% of the world's oceans and 99% of the ocean floor are yet to be explored [6]. The dense vegetation that lies underneath the surface of the ocean such as coral reefs and seaweed as well as the dwelling creatures add to the challenge of exploration. Autonomous agents are essential to underwater exploration and collection of samples due to their ability to survive longer than human agents on short missions.

The use of robots in various types of environments is rapidly increasing and the technologies used on these systems need improvement as well to cope with the increase in complexity these environments pose. We will introduce some robotic systems that are used in complex environments and go through how we will develop a navigation algorithm focused towards complex cluttered environments.

1.2. Robot Applications in Forests

Robot applications in jungles and forests include but are not limited to: monitoring and surveillance for environmental preservation and security, forest planting and harvesting, and wildfire firefighting.

The Environmental Hybrid Robot Chico Mendes is an amphibious wheel-legged robot that is designed to operate in the Amazon rain forest to carry out monitoring missions for the Brazilian Oil Company Petrobras S.A.. It is tasked with monitoring the gas pipeline along the forest. The robot is equipped with a robotic arm which has sensors for water quality and gas and an RGB camera to monitor gas leaks and gather data [7], [8]. Figure 1.2 shows an image of Chico Mendes in operation.

Figure 1.2: Chico Mendes monitoring robot in operation [9]

Another surveillance application is the use of Unattended Aerial Vehicles (UAVs) equipped with hyperspectral sensors in mapping forests affected by pathogens [10]. The UAVs were used to detect myrtle rust on paperbark tea trees using various vegetation indices along with an AI classification algorithm. This method achieved a 95% detection rate for healthy trees and a 97% detection rate for infected trees [10]. Figure 1.3 shows the detection and mapping process.

Another forest monitoring technique can be by distributing sensors such as for temperature and humidity and utilizing Internet of Things (IoT) for communication. The distribution of these sensors can occur by a UAV which uses impulsive launching to shoot the sensors into the trees or ground in the

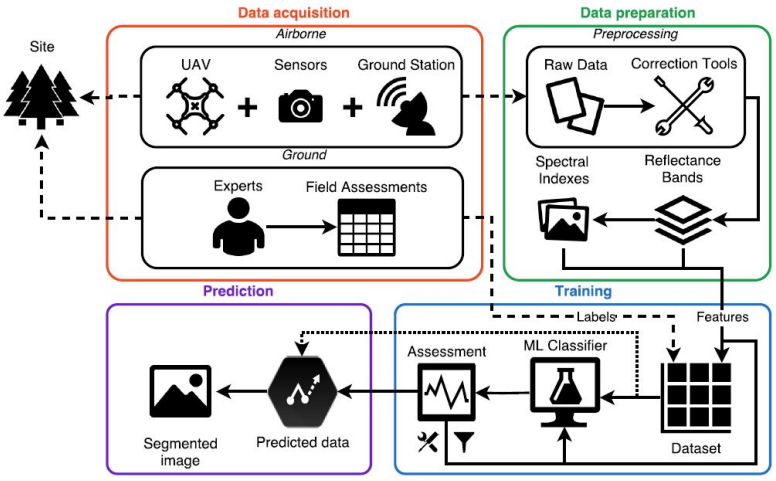


Figure 1.3: Detection and mapping of alterations in forests by diseases pipeline [10]

forest [11]. Figure 1.4 shows the different sensor placement methods and Figure 1.5 shows an implementation of the system.

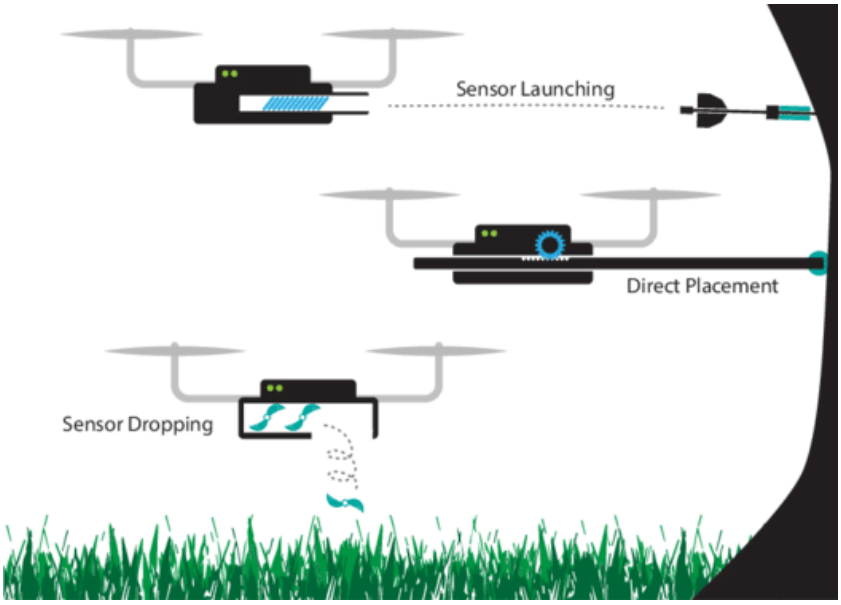


Figure 1.4: Different aerial sensor installation methods [11]

Throughout the year 2020 alone, forest wild fires have caused losses of extraordinary dimensions. A way to prevent wild fires from occurring is by preventing the accumulation of combustible organic material in forests. The SEMFIRE environmental preservation and forest fire prevention project was developed in order to achieve this task. The project is divided into two parts, a swarm of UAVs for scouting for the combustible material and a ranger robot for the removal of the material from the forest. The UAVs use Collective Simultaneous Localization and Mapping (SLAM) for navigation [12].

In [13] they develop a swarm of autonomous sheet pile driving robots for soil stabilization. The task for these robots is to plant steel piles into the ground at designated locations to reduce hydraulic erosion. This is not in a forestry environment but the non-structured terrain with the dunes provide a similar navigational challenge. Figure 1.6 shows the system in place.

Figure 1.5: Drone placing a sensor in a forest environment [11]

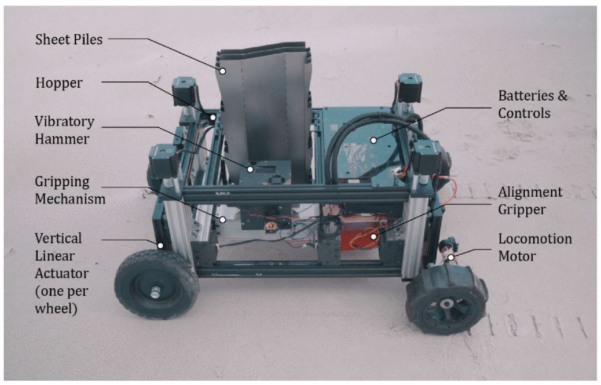


Figure 1.6: Sheet pile driving robot system [13]

1.3. Underwater Robots

Around 37% of the global population lives within 100 km of the ocean [14]. For centuries humans have given the ocean less importance when it comes to exploration and focused more on land. The ocean possesses vast living and non-living resources that if utilized properly can solve some of the main challenges the world faces today [15]. The recent advances in robotics gives an opportunity for developing robotic technologies for underwater exploration. Along with the increase in demand for underwater robotic technologies, this will lead to the development of autonomous, specialized, and reliable underwater robotic vehicles. Unattended underwater vehicles are divided into two main categories which are Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs), the former are tethered to a shore or vessel while the latter are untethered. An important factor for underwater robots is the navigation and sensing system installed due to the unknown and dynamic environments they are to be deployed in [16].

The applications of underwater robots span a wide range from monitoring marine life to removal of underwater sea mines. The US Navy developed a CURV for underwater search and rescue missions. Figure 1.7 shows the first CURV robot developed.

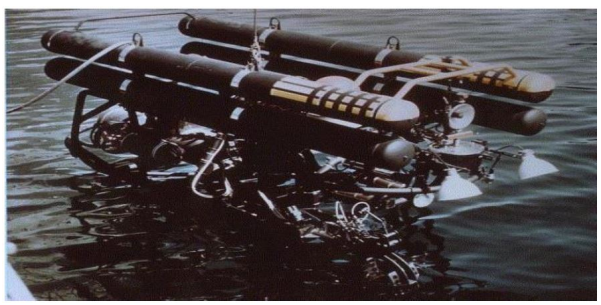


Figure 1.7: CURV robot in operation [17]

1.4. Lunar Zebro

The Lunar Zebro rover is a TU Delft project concerned with developing a lunar rover for the purpose of exploring and gathering data about the lunar surface and earth. The rover shall have a Semi-Autonomous Navigation System (SANS) which is responsible for having the rover avoid craters and boulders on the surface of the moon.

The rover is to go to the Moon while attached on a lander. The rover needs to survive the harsh conditions it will face on the moon for one lunar day (14 Earth days) and it has to maintain communication about its status back to Earth [18].

The DAA algorithm is one of the most important parts in the Lunar Zebro project due to the fact that if the rover hits an obstacle and gets damaged or tipped over then that could possibly jeopardize the whole mission by rendering the rover immobile or damaging some of the internal parts.

Figure 1.8 shows an image of the design of the rover. The legs of the rover are asymmetrically aligned in order to maximize stability and retain a compact format.

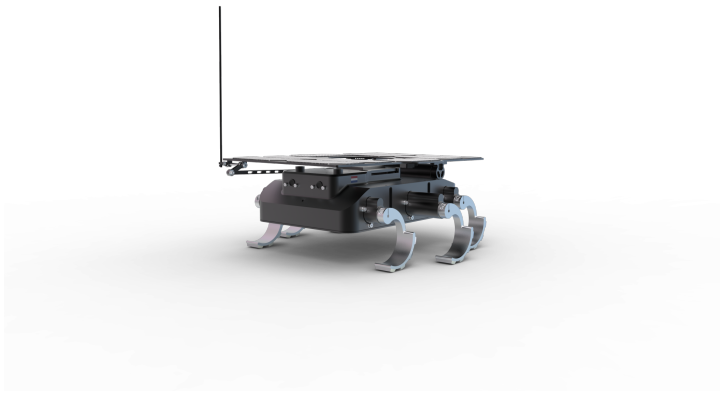


Figure 1.8: Lunar Zebro Rover [19]

1.5. Problem Statement

Autonomous agents such as drones and rovers are the future of delivery systems, search and rescue missions, and surveillance systems. The navigation systems for drones in unknown and cluttered environments like jungles are in their early development stages and still don't have the sufficient performance levels to be deployed into the field.

1.6. Objectives

The objective of this thesis is the development of a DAA algorithm for the navigation of autonomous agents such as aerial drones, terrestrial and extraterrestrial rovers, and underwater vehicles. The objectives can be summarised as follows:

- Development of an autonomous navigation algorithm.
- The emphasis in the development is on unknown cluttered environments.
- The algorithm is designed to outperform the existing DAA algorithms in terms of:
  - The number of convergences to the target location.
  - The average time it takes the agent to converge.
  - The average safety distance the agent keeps from the obstacles.
  - The algorithm complexity in terms of the computation time it takes to converge to the target.

1.7. Thesis Layout

Chapter two contains a review of the available detect and avoid methods in literature and provides a comparison between them with the motivation of choosing the artificial potential field method. Chapter three discusses the classical artificial potential field method and its limitations. Chapter four presents the new developed algorithm proposed in this thesis which is based on artificial potential fields. Chapter five concerns an extra use case for the algorithm which is a Moon rover designed by TU Delft for lunar exploration and data collection. The thesis concludes with chapter six which has the results and conclusions along with the future work which can be carried on based on this thesis.

2

Literature Review

2.1. Introduction

There are several methods for DAA systems for autonomous applications [20]. The focus on this review will be on DAA systems for autonomous agents.

We will present the existing technologies when it comes to the sensory systems used. We will also review the main types of navigation algorithms and contrast between them when it comes to different types on environments and scenarios.

2.2. Overview

There are multiple methods of accomplishing a DAA system from multiple types of sensory systems to multiple collision avoidance approaches. In general, two types of sensors are mainly employed for DAA: active sensors and passive sensors. Active sensors emit an energy signal and measure the reflected backscattered signal. Passive sensors only read the energy discharged by another source [20]. One type of active sensors is a Radio Detection and Ranging (RADAR) sensor which operates by transmitting a radio signal which bounces off an encountered object and returns back to the RADAR. By calculating the time it took for the signal to bounce back the distance to the object can be determined. RADAR systems are accurate but expensive and heavy for battery operated drones [21], [22]. Another type of sensor is the Light Detection and Ranging system more commonly known as LiDAR. LiDAR operates in a similar way to RADAR but it uses laser pulses instead of radio waves. LiDAR is extremely accurate but it cannot detect transparent objects such as clear glass. The last type of active sensor is the Sound Navigation and Ranging (SONAR) system which operates using ultrasonic signals similar to how light signals are used in a LiDAR system. Ultrasonic sensors are much cheaper than the other types and are readily available. Moreover, ultrasonic sensors can detect transparent objects but can't be relied on for detecting materials that have special sound absorbance or reflection characteristics. Passive sensors include visual and Infrared (IR) cameras, These types of sensors have small sizes and low power consumption. However, the visual sensor is highly dependent on weather and lighting conditions where combining it with the IR sensor can come in handy [20]. An overview of DAA methods is in Figure 2.1. As can be seen, DAA systems are composed of perception and action steps.

Collision avoidance can be classified into two main principles: reactive control or deliberative planning. Reactive control is when the robot gathers information using on-board sensors and react based on that data. Reactive control allows for rapid response but can lead to a local minimum and might get the robot stuck so it may need another technique combined with it. Figure 2.2 shows the block diagram for reactive collision avoidance. Deliberative planning on the other hand is when there is an environmental map that is constantly updated by the agent and an optimal collision free path is calculated then executed. The latter method needs an accurate map of the environment and that is computationally extensive specially for a dynamic environment. Figure 2.3 shows the block diagram for deliberative collision avoidance. A hybrid approach between the two is considered more suitable

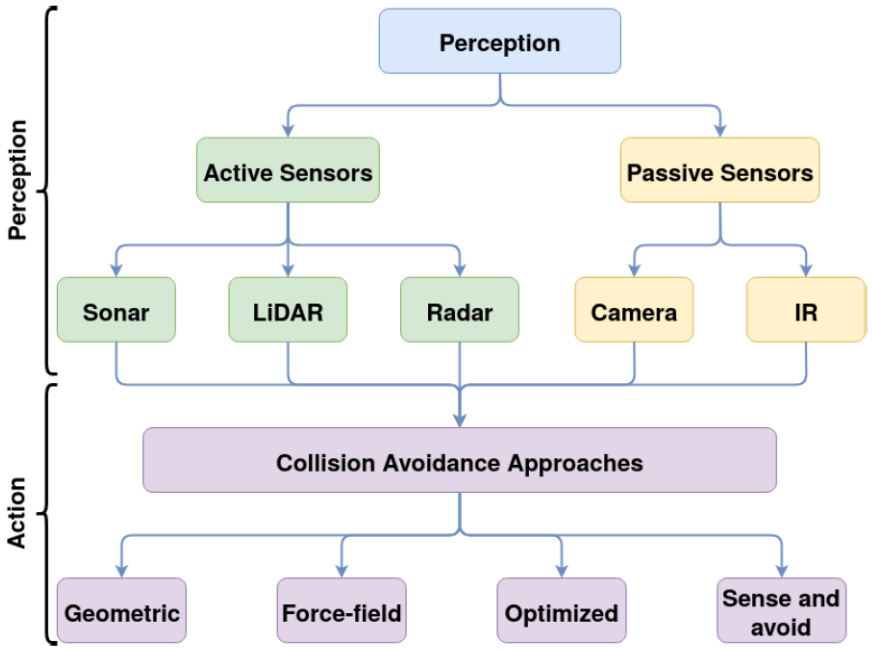


Figure 2.1: Detect and avoid systems overview [20]

for dynamic environments [20]. There are four main methods used for collision avoidance and those are:

- Geometric Methods
- Force-Field Methods
- Optimisation-Based Methods
- Sense-and-Avoid Methods

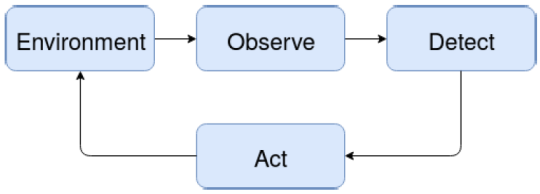


Figure 2.2: Reactive collision avoidance block diagram [20]

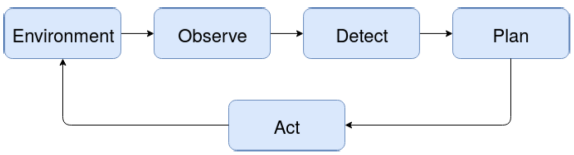


Figure 2.3: Deliberative collision avoidance block diagram [20]

Geometric Methods

Geometric approaches use geometric analysis to make sure that minimum distances to obstacles are not breached. This occurs by computing the time to collision knowing the distance and velocity of the

agent. To predict collision, the algorithms assume that the movement of the drone is at a constant speed without changing direction. Furthermore, each obstacle is surrounded by a spherical safety area. If the distance between the drone and the obstacle is less than the safety radius of the sphere then the collision is detected [23]. This method has been used in static environments before but its effectiveness in dynamic conditions is yet to be investigated [20].

Force-Field Methods

Force-Field methods, also known as potential field methods, rely on the concept of a repulsive or attractive force either to repel a robot from an obstacle or attract it towards the target respectively [24],[25],[26],[27]. However, we need to know the positions of the obstacles and their shape. In dynamic environments, this knowledge is not available. Another idea in literature is to place the potential field around the robot rather than the obstacles [28]. Based on the aggregate force from the attractive and repulsive forces as seen in (2.1), the robot determines the characteristics of it's motion. This method is however sensitive to a local minima and the robot may get stuck in a symmetric environment. This approach is yet to be validated in dynamic environments [20].

$$\mathrm{F}_{total} = \mathrm{F}_{attract} + \mathrm{F}_{repulse} \tag{2.1}$$

In an unknown environment, the agent needs to deal with uncertainty and incomplete information while still navigating safely. Force-Field methods are commonly used for this because of their elegance and simplicity [29].

Optimisation Based Methods

Optimisation based methods calculate the avoidance trajectory based on geographical information. Such algorithms have a high computational complexity so several optimisation methods have been developed. Those include ant-inspired algorithms, genetic algorithms, Bayesian optimisation, gradient descent based methods, particle swarm optimisation, greedy methods, and local approximations [20]. In [30] they use a minimum time search algorithm with ant colony optimisation to ensure the successful calculation of optimised collision-free search paths for UAVs under communication constraints. The authors in [31] present a prediction algorithm for the coordinates of the UAV based on the set of possible commands it is going to execute in the next time period. The algorithm formulates the cost function of the optimal trajectory by considering the position of the UAV and the target coordinates. Through this method, the best set of future commands is selected. If a potential collision is detected, the next set of commands is chosen and executed.

Sense and Avoid Methods

Sense and avoid methods reduce the computation power needed and have short response times while still achieving acceptable performance. These methods simplify the process to an individual detection and avoidance of obstacles. The fast response time makes this algorithm suitable for dynamic environments. In these methods, the robot is equipped with different types of sensors such as LiDAR, SONAR, and RADAR to gather information about the environmental surroundings.

In [32], a 2D LiDAR based approach is proposed. The methodology presented classifies the objects into static or dynamic. It is also capable of approximating the velocities of dynamic obstacles. A computer vision technique for detecting and avoiding animals is discussed in [33]. More than 2200 images have been used to train the algorithm and reached an 82.5% accuracy in successful detection of animals. The solution is however speed limited, in that it cannot prevent collisions at speeds exceeding 35km/h. At higher speeds it may not be able to detect objects at all. The system is also highly sensitive to weather conditions. In [34], the authors use five ultrasonic sensors along with a neural network in MATLAB to triangulate and detect the shape of objects. Locations in 2D settings can be found using only two ultrasonic sensors. Furthermore, adding a third ultrasonic sensor provides a third dimension (depth). The results found are satisfactory for regular shaped objects but the neural network cannot correctly identify irregular shapes [20]. The authors in [35] use sensor fusion of ultrasonic and IR sensors to develop a simple obstacle detection system. Inertial and optical flow sensors are used as a distance derivative for reference to get better data fusion. The end solution has a low computational

cost and saves memory. Furthermore, it enables the UAV to detect and avoid without any need for simultaneous localisation and mapping.

2.3. Comparison Between Methods

Multiple parameters can be used for performance comparison of DAA methods. In this section, we will review some of these methods and try to indicate the discrepancy.

The real time performance of the sense and avoid and geometric methods is better than the force-field and optimisation methods [20, P. 11]. Sense and avoid does not increase computational complexity if a change happens in the environment such as an obstacle moving [35]. Geometric methods are also computationally light but are highly dependent on the algorithm implementation in terms of computation time [36]. Optimisation methods are of medium complexity while the force-field methods are considered complex [20, P. 11]. The velocity constraint is a metric that takes the velocity of the obstacles in consideration. Sense and avoid and geometric approaches are capable to handle this constraint well [32],[37]. However, force-field and optimisation methods are better for predefined planning which does not take changing dynamics into consideration [20, P. 11]. The third metric is static and dynamic environment suitability. For dynamic environments, sense and avoid is the easiest and lightest due to the local computations being performed on board the agent through the changes observed by the sensory system on the agent itself [38]. Geometric methods give an acceptable performance but less optimal than sense and avoid [39]. Force-field methods do not perform well in narrow passages and can lead to a local minima in dynamic environments [40]. Optimisation methods are more suitable for static environments as they require pre-planning and have to optimise the whole route again in case a change is detected [20, PP. 11-12].

When it comes to deadlock, the geometrical and optimisation methods don't have this issue due to the fact that they are designed for known structured environments. Force-field methods can get stuck in a local minima. Sense and avoid methods can reach a deadlock and require another methodology for handling this issue as it can't be solved locally [20, P. 12]. When it comes to swarm compatibility all the approaches can be applied to a swarm of drones but might need an additional algorithm for handling communication between the agents. When it comes to dimensionality, all the mentioned methods need some work done when it comes to scaling the algorithms developed for 2D to handle 3D environments. However, there is current research on the feasibility of force-field methods for 3D dynamic environments [20],[41]. For pre-mission path planning, sense and avoid and force field methods do not require that because the plan is made when the obstacle is detected in-flight. Optimisation and geometric methods on the other hand, require path planning beforehand as the obstacle locations need to be known to the UAV before encountering the obstacles.

The takeaway point here is that for unknown cluttered environments, the force-field method is the best one to use even though it is more complex than the other methods. The reason for that is that the geometric and optimisation methods are designed for known structured environments and do not perform well in unknown cluttered environments while the sense and avoid methods do not take multiple obstacles into account which is necessary for cluttered environments.

2.4. Introduction to Artificial Potential Fields

Artificial Potential Fields or APF for short is a method used for navigation in environments with multiple obstacles that need to be taken into account at once hence in one model. The method works on the basis that a target is attractive while an obstacle is repulsive. The agent is to be attracted by the goal point and hence moves towards it while changing direction whenever an obstacle is encountered. An example for this is a mobile electrical charge moving in a field of static similar charges while being attracted by a static opposite charge. Figure 2.4 shows a simplified example of such a field from a bird's-eye view. An advantage of this method is its reduction of computational complexity which makes it suitable for small agents. Another advantage is that it takes into account the effect of multiple obstacles at once which makes it perfect for cluttered environments which are the scope of this research.

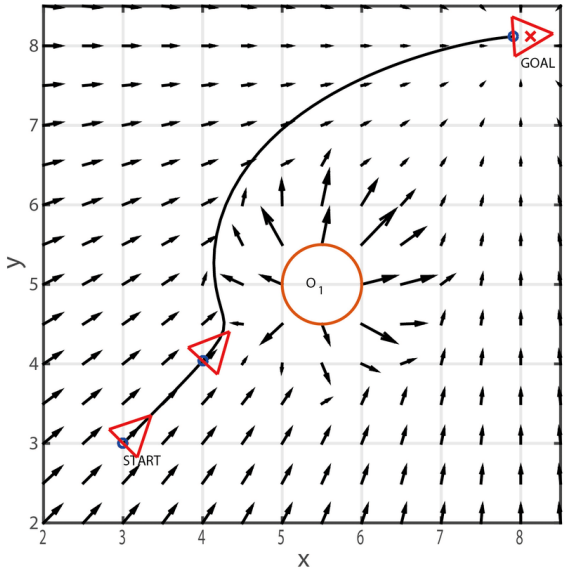


Figure 2.4: Two dimensional artificial potential field quiver plot with one obstacle and one goal [42]

2.5. Simulation Environment Setup

The simulation environment is setup in MATLAB with a fixed starting position for the agent and target location with a set of uniformly distributed obstacles in between. The map size is 30 by 30 meters where it is appropriate to be filled with densely cluttered point sized obstacles whilst giving the agent freedom to move. This simulation setup emulates the navigation through a set of randomly uniformly distributed obstacles such as trees or craters. The same setup will be used throughout the thesis. Figure 2.5 below shows the setup.

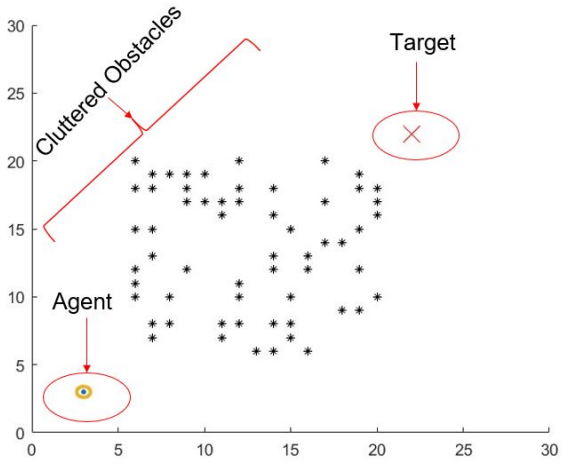


Figure 2.5: Simulation environment setup showing the agent, target, and the obstacles

2.6. Classical APF Method (CAPF)

The original APF method proposed by Khatib in 1985 [25] uses geometric calculations at each iteration to determine the direction and speed the agent should move on in order to avoid the obstacles and reach the target safely. It was originally proposed for robotic arm manipulators but has been extended to mobile agents afterwards [29].

The method operates as follows:

The attractive potential and force for the agent from the target are calculated using the following formulae:

$$J_t(\mathbf{r}) = \frac{1}{2} k_t \cdot DTT^2(\mathbf{r}) \tag{2.2}$$

$$DTT(\mathbf{r}) = \|\mathbf{r} - \mathbf{r}_t\|_2 \tag{2.3}$$

$$\mathbf{f}_t(\mathbf{r}) = -\nabla J_t(\mathbf{r}) \tag{2.4}$$

where k_t is a scaling factor and DTT is the distance to target. $\|\|_2$ is the L2 norm which calculates the Euclidean distance between the two points on the map. $\mathbf{r}$ is the vector locating the agent in the environment from the origin and $\mathbf{r}_t$ is the vector locating the target. J is the potential and $\mathbf{f}$ is the force which is the negative gradient of the potential.

The detection of the obstacles within the detection range (ρ_{rn}) occurs by the agent. The detection range is set as 8 m to emulate the range of an ultrasonic sensor [43].

$$\rho_{obst_j}(\mathbf{r}) = \|\mathbf{r} - \mathbf{r}_{obst_j}\|_2 \tag{2.5}$$

$\rho_{obst_j}(\mathbf{r})$ is the distance between the agent and the jth obstacle and if that distance is less than or equal to ρ_{rn} ($\rho_{obst_j} \leqslant \rho_{rn}$) then the obstacle is detected. The total number of detected obstacles is n.

After the detection of the obstacles, the potentials and forces of each detected obstacle are calculated.

$$J_{obst_i}(\mathbf{r}) = \begin{cases} \frac{1}{2} k_o \left(\frac{1}{\rho_{obst_i}(\mathbf{r})} - \frac{1}{\rho_o}\right)^2 & \text{if } \rho_{obst_i}(\mathbf{r}) \leqslant \rho_o \\ 0 & \text{if } \rho_{obst_i}(\mathbf{r}) > \rho_o \end{cases} \tag{2.6}$$

$$\mathbf{f}_{obst_i}(\mathbf{r}) = -\nabla J_{obst_i}(\mathbf{r}) \tag{2.7}$$

Where k_o is another scaling factor for the obstacle potential and ρ_o is a fixed distance from each obstacle.

The total potential and force from the obstacles is calculated by summing up the individual potentials and forces respectively.

$$J_{obst_T}(\mathbf{r}) = \sum_{i=1}^{n} J_{obst_i}(\mathbf{r}) \tag{2.8}$$

$$\mathbf{f}_{obst_T}(\mathbf{r}) = \sum_{i=1}^{n} \mathbf{f}_{obst_i}(\mathbf{r}) \tag{2.9}$$

The total potential and force from the environment on the agent is calculated by the adding the total potentials and forces from the obstacles to the potential and force from the target.

$$J(\mathbf{r}) = J_{obst_T}(\mathbf{r}) + J_t(\mathbf{r}) \tag{2.10}$$

$$\mathbf{f}(\mathbf{r}) = \mathbf{f}_{obst_T}(\mathbf{r}) + \mathbf{f}_t(\mathbf{r}) \tag{2.11}$$

The pseudo code for this algorithm can be found in Appendix A, Algorithm 1.

Figure 2.6 shows a successful navigation using the classical algorithm on a map containing 33 cluttered obstacles. I took the agent 74 steps with a 0.4 m step size to go from the starting point (3,3) to the target point (22,22). The blue dots mark the path taken by the agent.

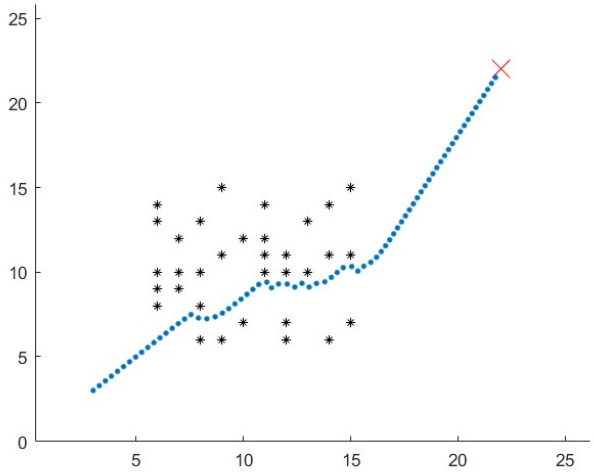


Figure 2.6: A successful run using the classical APF algorithm

The values for the parameters in the equations are determined empirically after testing a spectrum of values and deciding upon the ones that yield the most convergences to the target location. The best value for k_t was found to be 15 and for k_o was found to be 80. The value for ρ_o was found to be 5. However, these values are not terminal in the sense that changing them will change the performance but the algorithm still functions properly.

The CAPF method was originally designed for robotic arm manipulators in structured environments. It can still work in cluttered environments but is far from optimal and is often to get stuck at local minimas and collide with obstacles. Hence, the need arises for a new method that is dedicated for cluttered environments and that is the purpose of this thesis.

2.7. Comparison Criteria Between APF DAA Methods

The following terms and definitions will be used in order to compare different DAA methods with each other in the APF method.

2.7.1. Success Rate

This factor concerns the number of times the agent converges to the target without getting stuck in a local minima or colliding with an obstacles and hence achieves a successful run. It is considered as the most important factor and is calculated using Equation 2.12.

$$Success\ Rate = \frac{Number\ of\ Successful\ Runs}{Total\ Number\ of\ Runs} \tag{2.12}$$

2.7.2. Number of Steps for Convergence

This factor concerns the number of steps it takes the agent to reach the target. Given the size of the step and speed of the agent, the time per step and convergence time are determined as follows.

$$Time\ per\ Step = \frac{Step\ Size}{Agent\ Speed} \tag{2.13}$$

$$Convergence\ Time = Number\ of\ Steps \times Time\ per\ Step \tag{2.14}$$

2.7.3. Safety Parameter

Safety is one of the most important factors when it comes to autonomous agent navigation and especially in cluttered environments where a collision is considered as the worst thing that can happen. A

collision can permanently damage the agent. For this case, a safety parameter has been defined which gives an indication of the likeability of the agent to collide based on the closest distance reached to an obstacle during the run. The parameter is defined as the average distance that the agent maintained from the obstacles throughout the run.

$$Average\ Distance\ from\ Obstacles = \frac{\sum_{i=1}^{n} \rho_{min_{obst_i}}(\mathbf{r})}{n} \tag{2.15}$$

2.7.4. Algorithm Complexity

The computational complexity of an algorithm is determined by the amount of time it takes a computational platform to run the algorithm. It is determined by the maximum amount of time it takes for a certain number of inputs and is denoted by the big O notation ($O(n)$). In the case of navigation algorithms this is determined by the maximum amount of time it takes to navigate the map successfully.

3

Bacteria APF Method

3.1. Overview

The Bacteria APF method or BAPF for short is based on having a point agent surrounded by a circle of possible position points called bacteria points and there are potential functions that are calculated for those points based on the distance from the target and from the detected obstacles. Based on those potential functions a certain bacteria point is selected to be the next position of the agent and the agent moves to it then the whole process is repeated again [44]. This method shall overcome the limitations that the CAPF method faced when it comes to cluttered environments as seen before.

Figure 3.1 shows the layout of the bacteria points around the agent and Figure 3.2 shows the movement mechanism by selecting a certain bacteria point at each step.

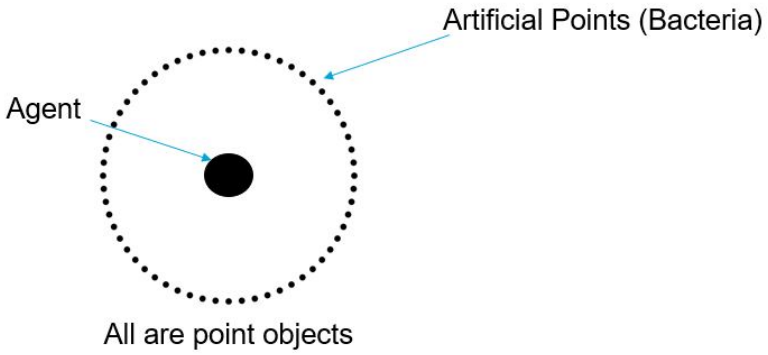


Figure 3.1: Agent and bacteria points around it

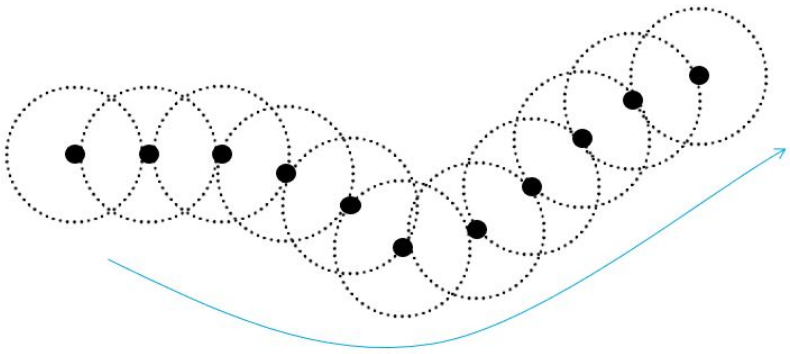


Figure 3.2: Agent movement by selecting different bacteria points

The functions shown in (3.1) to (3.6) calculate the potential values of the agent and the bacteria points around it. A total potential value is calculated for the agent and for each bacteria point. The attractive potential is universal. As in, it is present everywhere on the map while the repulsive potential from an obstacle is only effective once the obstacle is detected by the agent.

The agent only moves to a point with a negative potential and that point needs to have a potential that is more negative than the potential of the agent as shown in (3.7). The latter condition is only in most cases as a random walk and backtrack method can be applied in order to get out of a local minima which will be discussed later on.

Figure 3.3 shows the change in obstacle potential as the robot navigates closer to or away from a single obstacle. This figure represents (3.4) with only one obstacle on the map.

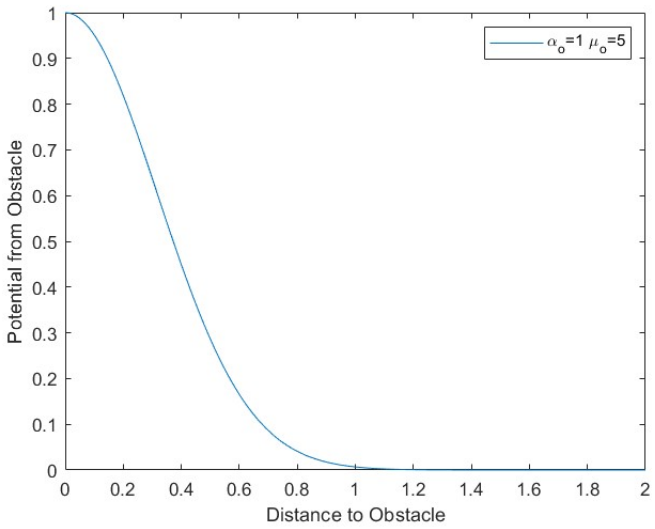


Figure 3.3: Obstacle potential change with distance

The following equations show the calculations done by the agent on the environment:

the Distance to Target (DTT) is first calculated.

$$DTT(\mathbf{r}) = \|\mathbf{r} - \mathbf{r}_t\|_2 \tag{3.1}$$

Then the potential from the target is calculated based on that distance.

$$J_t(\mathbf{r}) = -\alpha_t exp(-\mu_t DTT^2(\mathbf{r})) \tag{3.2}$$

Where α_t and μ_t are constants for determining the magnitude of the potential.

The detection of the obstacles within the detection range (ρ_{rn}) occurs by the agent. The detection range is set as 8 m to emulate the range of an ultrasonic sensor [43].

$$\rho_{obst_j}(\mathbf{r}) = \|\mathbf{r} - \mathbf{r}_{obst_j}\|_2 \tag{3.3}$$

$\rho_{obst_j}(\mathbf{r})$ is the distance between the agent and the jth obstacle and if that distance is less than or equal to ρ_{rn} ($\rho_{obst_j} \leqslant \rho_{rn}$) then the obstacle is detected. The total number of detected obstacles is n. The potential from each of the detected obstacles is then calculated.

$$J_{obst_i}(\mathbf{r}) = \alpha_o exp(-\mu_o \|\mathbf{r} - \mathbf{r}_{obst_i}\|_2^2) \tag{3.4}$$

Where α_o and μ_o are constants for determining the magnitude of the potential.

The total potential from all the detected obstacles is then aggregated. There are n detected obstacles in total.

$$J_{obst_T}(\mathbf{r}) = \sum_{i=1}^{n} J_{obst_i}(\mathbf{r}) \tag{3.5}$$

The total potential on the agent is calculated from the obstacles and target potentials.

$$J(\mathbf{r}) = J_{obst_T}(\mathbf{r}) + J_t(\mathbf{r}) \tag{3.6}$$

The same calculations occur again for each of the bacteria points around the agent. The subscripts "a" and "b" denote the agent and bacteria points respectively. $J_a(\mathbf{r})$ is the total potential on the agent while $J_{b_i}(\mathbf{r})$ is the total potential on the ith bacteria point.

$$J_{b_i}(\mathbf{r}) - J_a(\mathbf{r}) < 0 \tag{3.7}$$

Figure 3.4 shows the flow graph of the algorithm. A linear search on the bacteria points which is described in Figure 3.5 is done to select the best bacteria point to move to. The search starts with testing the bacteria point closest to the target first. If that point meets the potential criteria shown at 3.7 then the agent moves to it with no need to test the other points. If that point does not meet the potential criteria then test the second closest bacteria point to the target and so on and so forth.

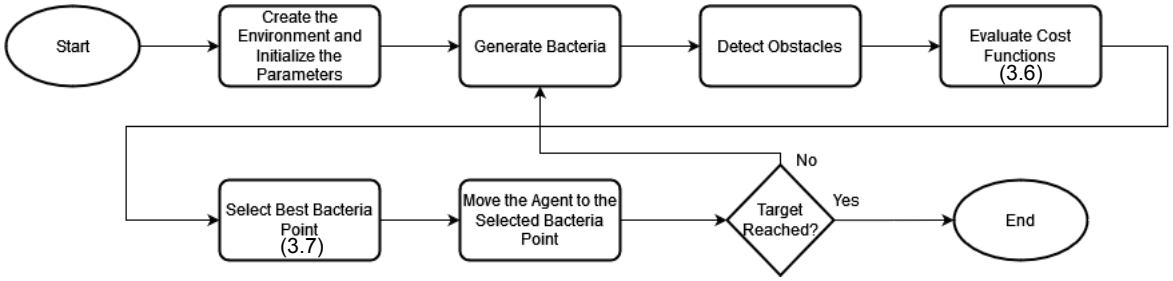


Figure 3.4: Algorithm flow graph for agent navigation

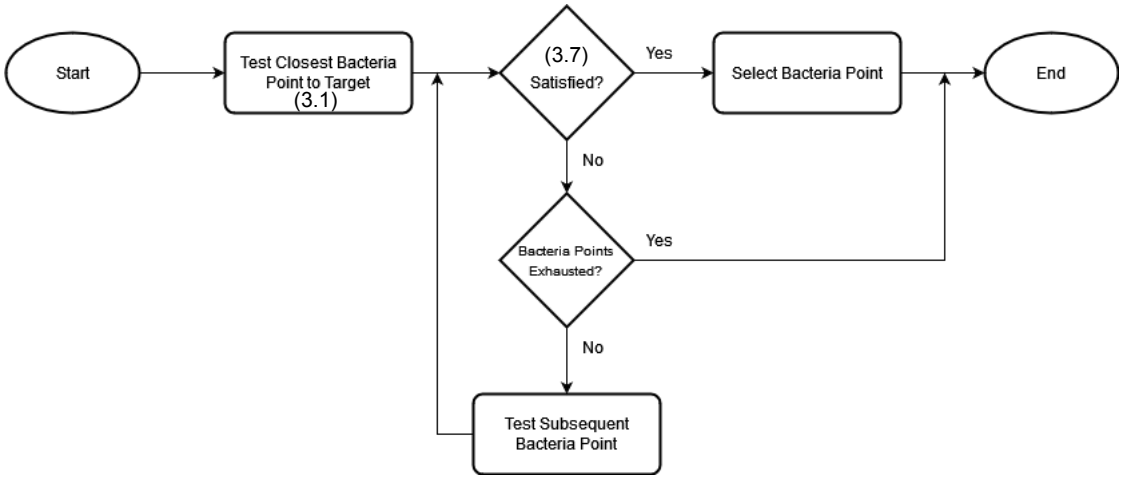


Figure 3.5: Algorithm flow graph for best bacteria point selection

When a bacteria point is selected it becomes the next point that the agent should move to. The distance between the agent and the bacteria points is constant and is determined empirically given the

type of environment in question and the range of the sensor to be used on the agent. The number of bacteria points is fixed at 60 points to give a rotation accuracy of $6°$. The pseudo code for this algorithm can be found in Appendix A, Algorithm 2.

The main difference between the CAPF and the BAPF algorithms at this point is that the CAPF algorithm calculates the potential vectors from the obstacles and target at each step in the navigation while the BAPF algorithm reduces the navigation problem to a linear search problem on the bacteria points at each step.

3.2. Agent Movement

3.2.1. Agent Motion Error

In the real world implementation of the agent, the positioning of the agent is not optimum. There is an error included in the positioning of the agent in the environment. This error stems from measurement errors in the sensors the drone uses for navigation and obstacle detection. This error needs to be simulated as well in order to have the simulation as close as possible to reality.

The error is simulated at the point the agent selects a bacteria point and then starts moving towards it. The agent shall not be positioned exactly at the bacteria point it has selected. It will move with a scaled error that follows a standard normal distribution (zero mean and unity standard deviation)

3.2.2. Successful and Unsuccessful Runs

A successful run is defined as a run where the agent reaches within 0.7 m distance from the target without colliding with any obstacles nor getting stuck in a local minima region.

Figure 3.6 shows a successful run on a 30 by 30 meters simulation environment. The blue dots mark the path taken by the agent while the yellow circle is formed by the bacteria points surrounding the agent at each step.

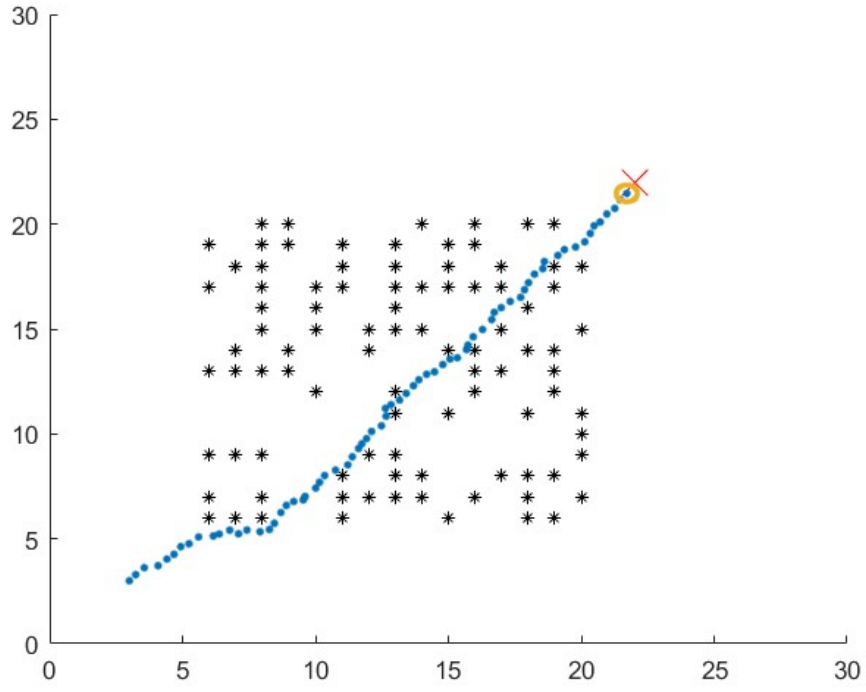


Figure 3.6: Successful agent navigation from start point to target region

An unsuccessful run is defined as a run where the agent gets stuck in a local minima before reaching

the target. It can also be the case of an unsuccessful run if the agent collides with an obstacle in-run. However, this is an extremely rare case in the simulation environment due to the lower radii defined in subsection 3.3.2.

Figure 3.7 shows an example of an agent getting stuck at a local minima in the run.

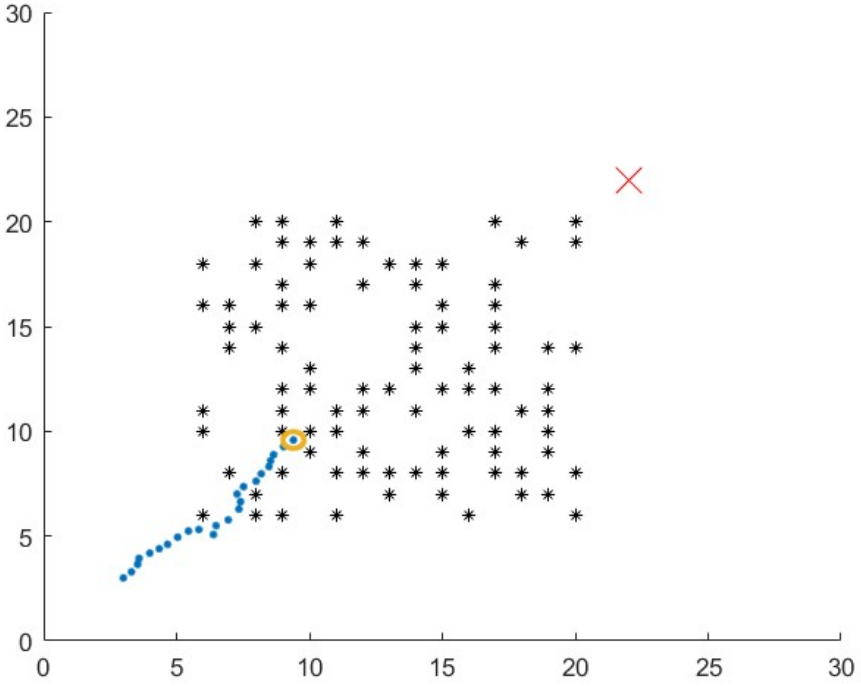


Figure 3.7: Unsuccessful agent navigation to target region

3.3. Improved Bacteria APF Methods (IBAPF)

In this section, we propose the Adaptive BAPF (A-BAPF) and the Changing Radii BAPF (CR-BAPF) algorithms to improve the performance of the classical BAPF algorithm.

3.3.1. A-BAPF

The first one is having an adaptive μ_o value in the repulsive potential equation (3.4) which is applied through a brute force search on a finite set of values ($\mu_o \in [1 : 1000]$) in order to minimize the value of the potential cost function in (3.4). When the μ_o value approaches 1000 the repulsive potential approaches zero and the obstacles' influence on the navigation is disregarded. The minimization of the potential cost function value for the bacteria points can almost guarantee a solution for (3.7) at each step and prevent the agent from ending up in a local minima. This search is performed for every bacteria point and then the selection of the best bacteria point takes place as follows.

$$\begin{aligned} \min_{\mu_o} \quad & J_{b_i}(\mathbf{r}) \\ \text{subject to} \quad & \mu_o \in [1 : 1000] \end{aligned} \tag{3.8}$$

The result of this method defines the upper bound on how many times can a successful run be achieved in various cluttered environments with a uniform random distribution of obstacles.

Figure 3.8 shows the change in the μ_o value that occurs in every step in order to solve (3.8) and achieve convergence to the target on a 30 by 30 meters map.

The value for μ_o changes sparsely and that is due to the fact that the environment the agent is moving in is a cluttered environment where the following factors affect the optimization problem:

- Number of detected obstacles
- Distance from the detected obstacles (affected by obstacle distribution)
- Distance from the target

Those are the only environmental variables that change as the agent is traversing.

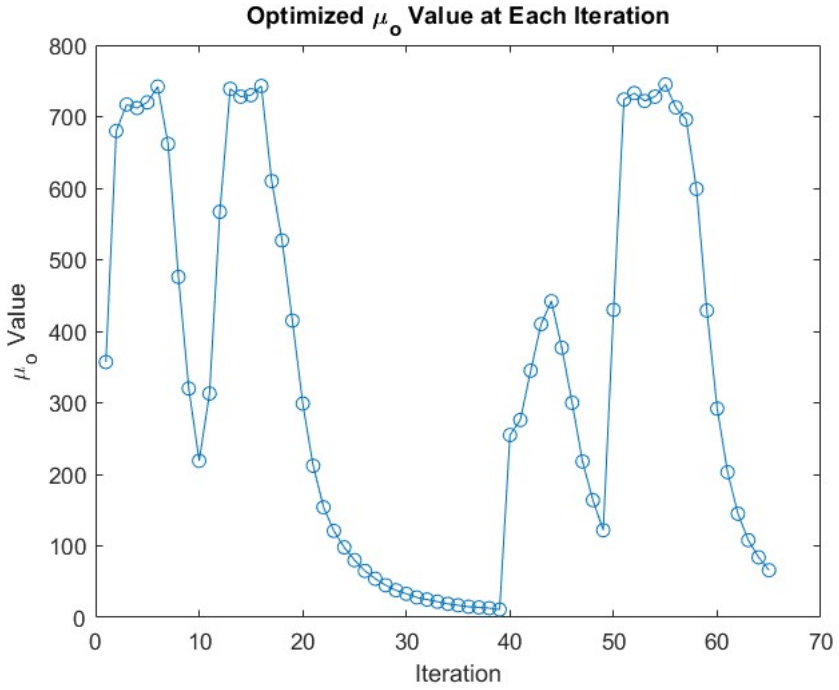


Figure 3.8: μ_o optimized value at every step using brute force optimization

When compared to the constant μ_o value across the whole run, the adaptive μ_o performs better and achieves a higher success rate. This was observed by running the two algorithms on the same environments. The drawback of using the adaptive μ_o is that the brute-force search takes a large amount of time to complete and hence makes the agent take more time to converge to the target. The adaptive μ_o in this case represents the upper bound that the agent can reach in terms of the number of successes (convergences) over various environments.

Figure 3.9 shows the fixed μ_o value versus the success rate for various uniformly distributed cluttered environments. Moreover, in this case it is for 1500 different environments with the number of obstacles ranging from 20 to 45 obstacles per environment under a uniform random distribution. The size of all the environments is 30 by 30 meters.

3.3.2. CR-BAPF

Another contribution is the addition of two radii around each obstacle where the repulsive potential changes to a fixed value if the distance between the agent and the obstacle is beyond that distance. The radii are an upper radius and a lower radius as shown in Figure 3.10. If the agent's distance to the detected obstacle is greater than the upper radius then the repulsive potential from that obstacle on the agent is zero. Moreover, if the agent's distance to the obstacle is lower than the lower radius then the repulsive potential from the obstacle to the agent is infinity. This means that the agent will never select a bacteria point which has a distance to the obstacle that is lower than the lower radius as that point will never satisfy (3.7). The lower radius is meant as a safety perimeter around the obstacles to

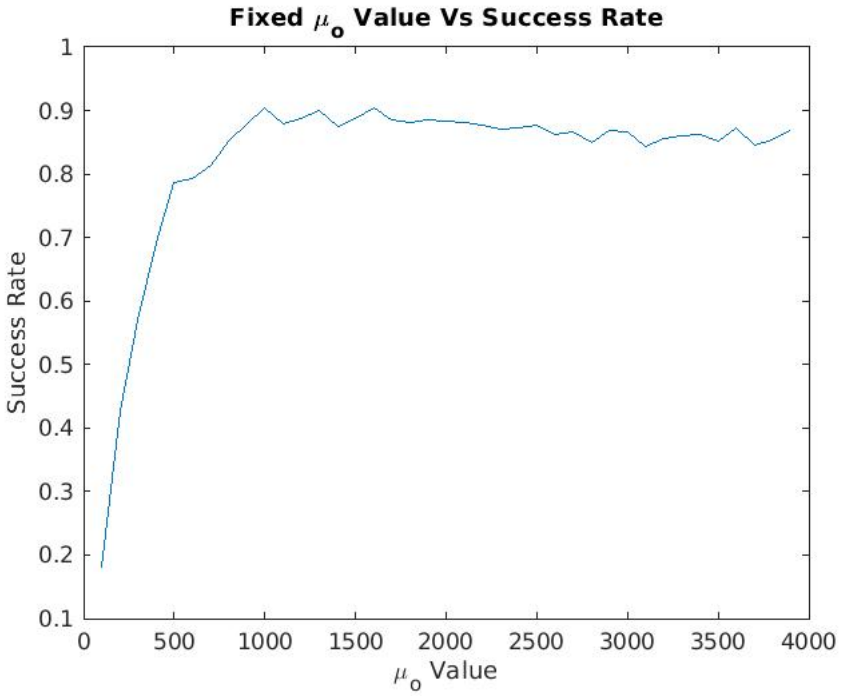


Figure 3.9: Fixed μ_o values versus the success rate over 1500 different environments

prevent collisions. The upper and lower radii are the same for all the obstacles in the map.

The definition of these radii changes the repulsive potential (3.4) into a branching potential equation as can be seen in (3.9) where ρ_u indicates the upper radius and ρ_l indicates the lower radius.

$$J_{obst_i}(\mathbf{r}) = \begin{cases} 0 & \text{if } \rho_{obst_i}(\mathbf{r}) > \rho_u \\ \alpha_o exp(-\mu_o \|\mathbf{r} - \mathbf{r}_{obst_i}\|_2^2) & \text{if } \rho_l \geqslant \rho_{obst_i}(\mathbf{r}) \leqslant \rho_u \\ \infty & \text{if } \rho_{obst_i}(\mathbf{r}) < \rho_l \end{cases} \tag{3.9}$$

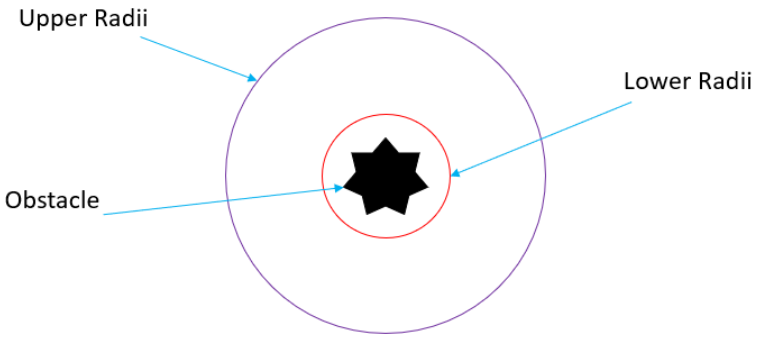


Figure 3.10: The upper and lower radii around a point obstacle

The determination of the upper and lower radii values is relevant to the map. The lower radius is a safety perimeter around each obstacle while the upper radius relates to the density of the obstacles relative to the size of the map. In the 30 by 30 m map that the simulation environment is on, the lower radius is determined to be 0.4 m and the upper radius as 4.5 m.

The upper and lower radii definitions significantly improve the performance of the algorithm in terms of the success rate it achieves over when every detected obstacle has a calculated potential at every step no matter the distance between the obstacle and agent.

Figure 3.11 shows the result of the APF bacteria method without the implementation of the upper and lower radii and after the implementation of the upper and lower radii over 100 different environments. It can be seen from the figures the trend in the improvement in the success rate that the potential changing radii make with the increase in the value of μ_o. The CR-BAPF is what we refer to when we discuss the IBAPF from this point forward. The pseudo code for this algorithm can be found in Appendix A, Algorithm 3.

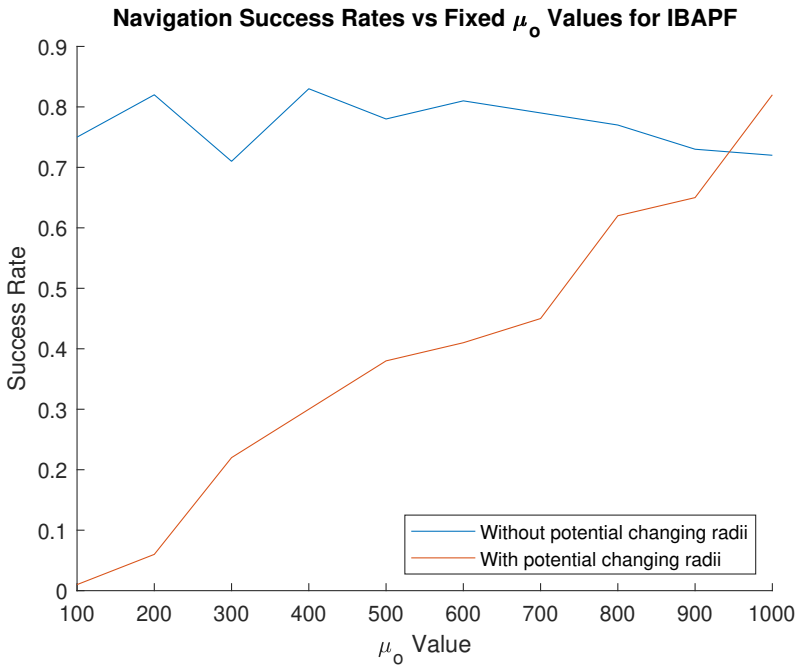


Figure 3.11: Navigation success rates without and with potential changing radii using the improved bacteria based algorithm

3.4. Local Minima Trap

A major drawback for the APF method is that it can get stuck in a local minima where the agent cannot find a solution to the problem on a certain region on the map. The local minima trap is sometimes inevitable in the case of unknown environments and especially unknown cluttered environments.

For the bacteria based APF method, the local minima trap means that no solution was found for (3.7) using any of the bacteria points around the agent at that iteration. This means that the agent cannot make any movement decision at that step and that either the agent cannot converge to the target at that run or another technique needs to be adopted in order to get the agent out of the local minima region.

Figure 3.12 shows two quiver plots, one that has clear paths to the target region without any local minima regions and one with the addition of a single obstacle in a specific location on the same map which creates a local minima region.

From the potential point of view, a successful run means that there was always a bacteria point that satisfies (3.7) until convergence to the target has happened. Figure 3.13 shows the values of the potentials of the agent and the eligible bacteria point selected by the algorithm at every step. On the contrary, Figure 3.14 shows those values in the case that the agent enters a local minima trap region. There is a sudden change in the potentials of the agent and the eligible bacteria point in that case

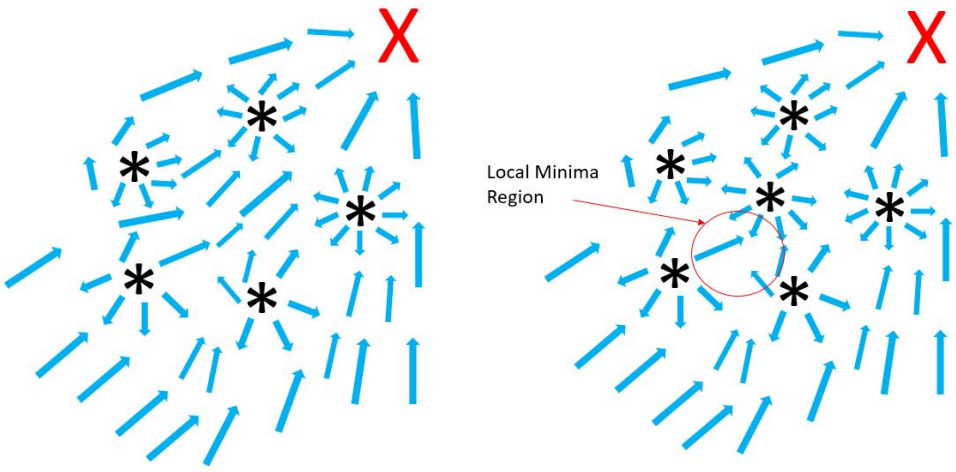


Figure 3.12: Quiver plot of clear navigation paths to the target from any point on the map (left) Quiver plot with the addition of a single obstacle which creates a local minima at that location where the agent will get stuck if it came across it (right)

and it remains static afterwards due the fact that the agent is stuck and is not moving anymore and hence there is no change in the potential values. This sudden change makes the value of the bacteria potential higher than the value of the agent potential and so (3.7) cannot be satisfied anymore.

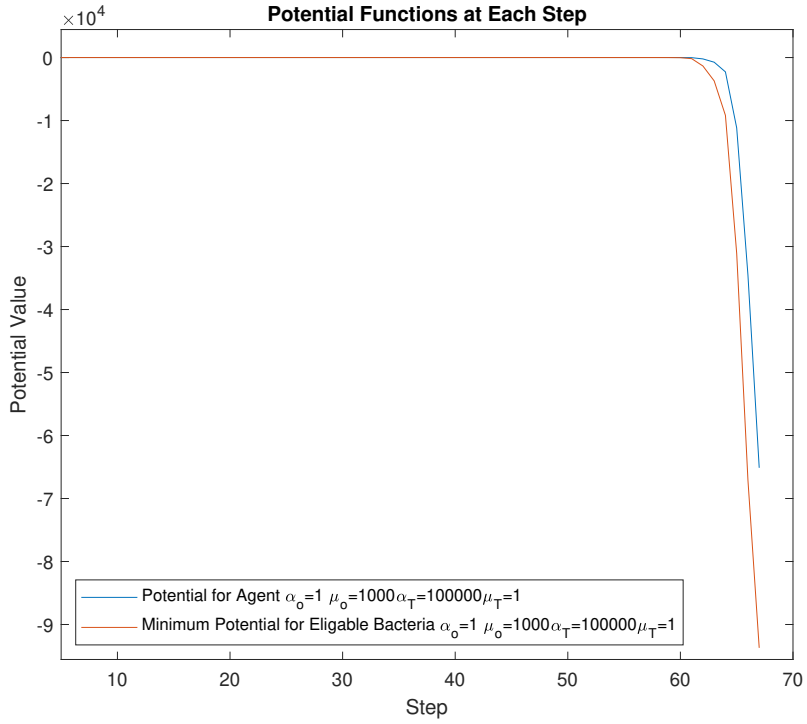


Figure 3.13: Potential values of the agent and the most eligible bacteria point at every step in the case of convergence to the target

To escape the local minima trap, a random walk can be employed where a bacteria point is selected randomly for the agent to move to. A strict condition for this randomly selected bacteria point is that it should not lie in a region where the distance to a detected obstacle is less than the lower potential

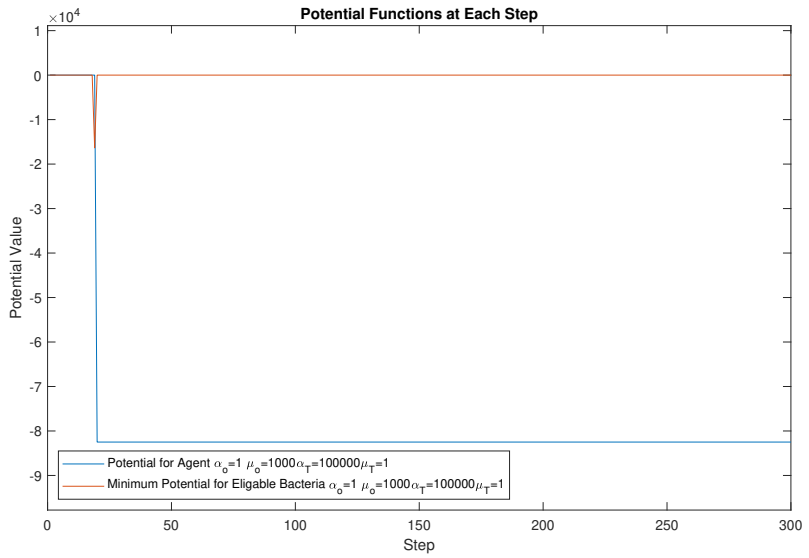


Figure 3.14: Scaled potential values of the agent and the most eligible bacteria point at every step in the case of getting stuck in a local minima. Notice the rapid change in potential and the static values afterwards due to the local minima trap

radius.

Figure 3.15 shows the agent escaping a local minima trap via random walk.

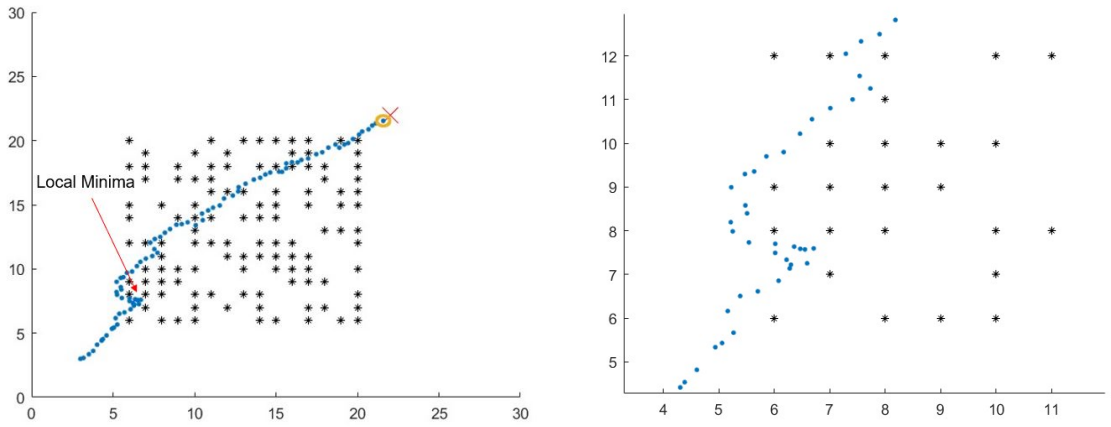


Figure 3.15: Local minima escape through random walk (left) random walk zoomed into (right)

3.5. Comparison Between Classical and Proposed Implementation

The comparison between the different implementation algorithms of the APF method is based on the average rate of success for the algorithm in cluttered environments (tested over 3000 different environments) and the average number of steps it takes to converge as well as the safety parameter and algorithm complexity as explained in 2.7. The step size is fixed (0.4 m) for both algorithms. The tests are run on a 30 by 30 meters map with the number of obstacles ranging from 20 to 45 per map. Table 3.1 shows a comparison between the CAPF, BAPF, and IBAPF algorithms. The simulations were run on the Circuits and Systems (CAS) research group server which has the following specifications: 2 Intel(R) Xeon(R) CPUs E5-2690 v4 @ 2.60 GHz and 384 GB of RAM.

Algorithm	Success Rate	Average Number of Steps	Average Distance from Obstacles (m)	Average Algorithm Execution Time (ms)
CAPF	0.316	91.105	2.717	55.3
BAPF	0.74	68.183	2.3763	178.3
IBAPF	0.929	70.761	2.357	230.8

Table 3.1: Showing the improvement in performance achieved by the bacteria based algorithm

3.6. Lunar Zebro Use Case

The Lunar Zebro rover provides a good physical platform to design and adapt the DAA algorithm for as it is a low-computation autonomous agent [45]. The surface of the moon is a lightly cluttered environment where the obstacles are formed by impact craters that create a coarse and rigid structure [46].

The number of obstacles was decreased to 10 to 30 obstacles in contrary to 20 to 45 obstacles in the densely cluttered environments case. This yielded a better success rate for the IBAPF algorithm as can be seen in Figure 3.16.

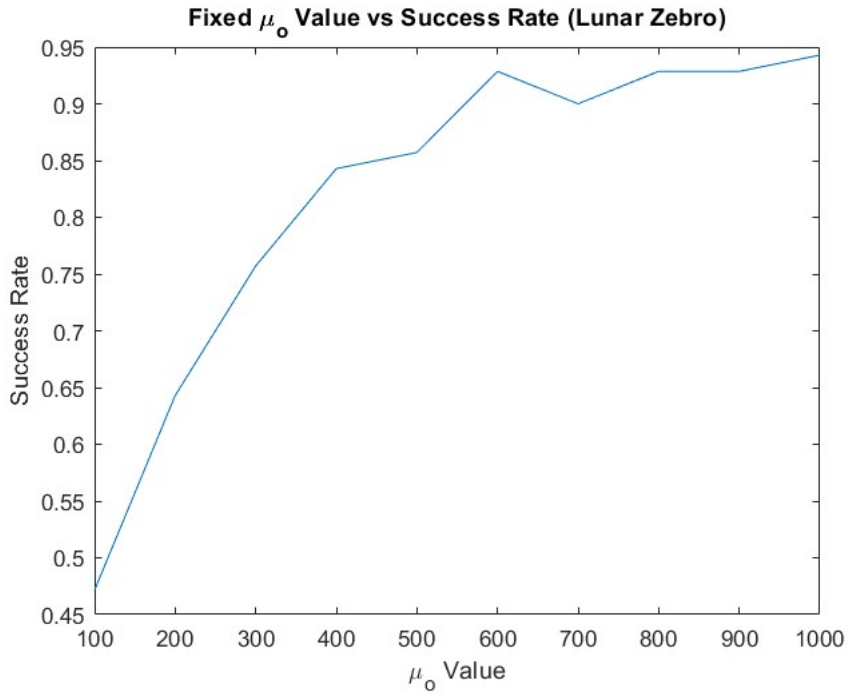


Figure 3.16: The success rate of various fixed μ_o values with a reduced number of obstacles (10 to 30)

The results for the Lunar Zebro case using the IBAPF algorithm can be found in Table 3.2. The

results are from simulations run over 3000 different environments. The results show a higher success rate as expected and a lower average number of steps. Moreover, the results also show a lower average algorithm execution time. All of this can attributed to the fact that there are less objects classified as obstacles for the rover in the lunar environment.

Algorithm	Success Rate	Average Number of Steps	Average Distance from Obstacles (m)	Average Algorithm Execution Time (ms)
CAPF	0.54	82.755	2.544	32.7
BAPF	0.851	67.809	2.368	120.2
IBAPF	0.952	68.52	2.356	149.3

Table 3.2: Showing the improvement in performance achieved by the bacteria based algorithm

3.7. Conclusion

The IBAPF algorithm proves to be the best when it comes to the success rates it achieves in both the densly and lightly cluttered scenarios while the CAPF algorithm under performs in that metric. The bacteria based algorithms (BAPF and IBAPF) are similar when it comes to the average number of steps it takes to converge to the target while the CAPF algorithm takes a longer route than both. All the three algorithms perform well when it comes to the safety criteria in keeping distance from obstacles in that they all keep more than 2 meters on average. The safest algorithm is the IBAPF. The CAPF algorithm has the shortest execution time and the BAPF and IBAPF have longer times respectively due to the linear search done on the bacteria points.

4

Conclusions and Future Work

4.1. Conclusions

The investigation of both CAPF and BAPF algorithms leads to the following conclusions:

- Artificial potential fields are the best suited navigation method for cluttered environments due to their elegance and simplicity.
- The IBAPF algorithm proved to be superior to the CAPF algorithm in cluttered environments.
- The addition of potential changing radii in the BAPF algorithm is a critical contribution that significantly improved the performance.
- The proposed A-BAPF algorithm gives the best performance in terms of success rates but has the drawback of having a lower convergence speed and hence can't be deployed on an actual agent in the field.
- The local minima trap is the major drawback of the APF method regardless of the algorithm used.
- The local minima trap can be escaped through a random walk and the agent can continue in its intended path afterwards.

4.2. Future Work

- Deploy the algorithm in a real-world implementation and test the performance.
- Develop a backtracking method where if the agent ends up in a local minima without being able to escape then it goes back to a saved waypoint and selects another approach.
- Reduce the algorithm complexity by optimizing the algorithm and code further in order to achieve a lower computation time.
- Develop an analytical method for the selection of hyperparamters in the potential functions.

Bibliography

[1] Food, A. Organization, and the United Nations Environment Program, "The state of the world's forests 2020. forests biodiversity and people," 2020. DOI: 10.4060/ca8642en.

[2] Food and A. Organization, "The state of the world's forests 2018. forest pathways to sustainable development," 2018.

[3] M. Boer, V. Resco de Dios, and R. Bradstock, "Unprecedented burn area of australian mega forest fires," *Nature Climate Change*, vol. 10, pp. 1–2, Feb. 2020. DOI: 10.1038/s41558-020-0716-1.

[4] L. F. Oliveira, A. Moreira, and M. Silva, "Advances in forest robotics: A state-of-the-art survey," *Robotics*, vol. 10, p. 53, Mar. 2021. DOI: 10.3390/robotics10020053.

[5] Q. L., P. Nevalainen, J. Peña Queralta, J. Heikkonen, and T. Westerlund, "Localization in unstructured environments: Towards autonomous robots in forests with delaunay triangulation," May 2020.

[6] A. Mustain. "Mysteries of the oceans remain vast and deep." (Jun. 2011), [Online]. Available: https://www.livescience.com/14493-ocean-exploration-deep-sea-diving.html.

[7] G. Freitas, G. Gleizer, F. Lizarralde, L. Hsu, and N. Reis, "Kinematic reconfigurability control for an environmental mobile robot operating in the amazon rain forest," *J. Field Robotics*, vol. 27, pp. 197–216, Mar. 2010. DOI: 10.1002/rob.20334.

[8] N. R. S. d. Reis, "Desenvolvimento de tecnologias como conjunto de ferramentas e suporte às atividades e pesquisas socioambientais na amazônia brasileira: Mobilidade e acessibilidade em áreas de várzea," *Master's Thesis, Universidade Federal do Amazonas, Manaus, Amazonas, Brazil*, Jun. 2010.

[9] Institute of Electrical and Electronics Engineers. "Chico." (Aug. 2022), [Online]. Available: https://robots.ieee.org/robots/chico/?gallery=photo3 (visited on 02/08/2022).

[10] J. Sandino, G. Pegg, L. Gonzalez, and G. Smith, "Aerial mapping of forests affected by pathogens using uavs, hyperspectral sensors, and artificial intelligence," *Sensors*, vol. 18, p. 944, Mar. 2018. DOI: 10.3390/s18040944.

[11] A. Farinha, R. Zufferey, P. Zheng, S. Armanini, and M. Kovac, "Unmanned aerial sensor placement for cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6623–6630, Oct. 2020. DOI: 10.1109/LRA.2020.3015459.

[12] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha, "Semfire: Towards a new generation of forestry maintenance multi-robot systems," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019, pp. 270–276. DOI: 10.1109/SII.2019.8700403.

[13] N. Melenbrink and J. Werfel, "Autonomous sheet pile driving robots for soil stabilization," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 339–345. DOI: 10.1109/ICRA.2019.8793546.

[14] J. Cohen, C. Small, A. Mellinger, *et al.*, "Estimates of coastal populations," *Science*, vol. 278, pp. 1209–1213, Nov. 1997. DOI: 10.1126/science.278.5341.1209c.

[15] S. Chutia, N. Kakoty, and D. Deka, "A review of underwater robotics, navigation, sensing techniques and applications," Jun. 2017, pp. 1–6. DOI: 10.1145/3132446.3134872.

[16] J. Bellingham and K. Rajan, "Robotics in remote and hostile environments," *Science (New York, N.Y.)*, vol. 318, pp. 1098–102, Dec. 2007. DOI: 10.1126/science.1146230.

[17] C. Paschoa. "Pioneer work class rovs (curv-i) - part 1." (Jul. 2014), [Online]. Available: https://www.marinetechnologynews.com/blogs/pioneer-work-class-rovs-(curv-i-iii)-e28093-part-1-700495 (visited on 02/08/2022).

[18] Lunar Zebro and Inholland University of Applied Sciences, “Locomotion simulations of the lunar zebro moon rover drive train. detailed simulations report,” Jan. 2022.

[19] S. Prachi. “Volunteer at lunar zebro.” (Mar. 2020), [Online]. Available: https://www.alten.nl/en/2020/09/03/volunteer-at-lunar-zebro-2/.

[20] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, “Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches,” *IEEEAccess*, vol. 8, 2020, pp. 105 139–105 155, Jun. 2020. DOI: 10.1109/ACCESS.2020.3000064.

[21] B. Korn and C. Edinger, “Uas in civil airspace: Demonstrating, ‘sense and avoid’ capabilities in fight trials,” *IEEE/AIAA 27th Digit. Avionics Syst. Conf.*, pp. 4.D.1-1-4.D.1–7, Oct. 2008.

[22] M. P. Owen, S. M. Duffy, and M. W. M. Edwards, “Unmanned aircraft sense and avoid radar: Surrogate flight testing performance evaluation,” *IEEE Radar Conf.*, pp. 548–551, May 2014.

[23] M. Skowron, W. Chmielowiec, K. Glowacka, M. Krupa, and A. Srebro, “Sense and avoid for small unmanned aircraft systems: Research on methods and best practices,” *Proc IMechE Part G: J Aerospace Engineering*, vol. 233, no. 16, pp. 6044–6062, Dec. 2019. DOI: 10.1177/0954410019867802.

[24] A. Mujumdar and R. Padhi, “Nonlinear geometric and differential geometric guidance of uavs for reactive collision avoidance,” *J. Guid., Control, Dyn.*, vol. 34, p. 69, Jul. 2009.

[25] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Int. Conf. Robot. Automat.*, vol. 2, pp. 500–505, Mar. 1985.

[26] K. L. D. Choi and D. Kim, “Enhanced potential field-based collision avoidance for unmanned aerial vehicles in a dynamic environment,” Mar. 2020. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2020-0487.

[27] M. Radmanesh, M. Kumar, P. H. Guentert, and M. Sarim, “Overview of path-planning and obstacle avoidance algorithms for uavs: A comparative study,” vol. 6, no. 2, pp. 95–118, Apr. 2018. DOI: 10.1142/S2301385018400022.

[28] A. A. Holenstein and E. Badreddin, “Collision avoidance in a behavior based mobile robot design,” *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 898–903, Apr. 1991.

[29] J. C. Mohanta, D. R. Parhi, S. K. Patel, and S. K. Pradhan, “Real-time motion planning of multiple mobile robots using artificial potential field method,” *Journal of Advance Computational Research*, vol. 01, Jan. 2016.

[30] S. Pèrez-Carabaza, J. Scherer, B. Rinner, J. A. Lòpez-Orozco, and E. Besada-Portas, “Uav trajectory optimization for minimum time search with communication constraints and collision avoidance,” *Eng. Appl. Artif. Intell.*, vol. 85, pp. 357–371, Oct. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197619301411.

[31] E. Boivin, A. Desbiens, and E. Gagnon, “Uav collision avoidance using cooperative predictive control,” *Proc. 16th Medit. Conf. Control Automat.*, pp. 682–688, Jun. 2008.

[32] M. Wang, H. Voos, and D. Su, “Robust online obstacle detection and tracking for collision-free navigation of multirotor uavs in complex environments,” *Proc. 15th Int. Conf. Control, Automat., Robot. Vis. (ICARCV)*, pp. 1228–1234, Nov. 2018.

[33] S. U. Sharma and D. J. Shah, “A practical animal detection and collision avoidance system using computer vision technique,” *IEEE Access*, vol. 5, pp. 347–358, 2017.

[34] M. D. Simone, Z. Rivera, and D. Guida, “Obstacle avoidance system for unmanned ground vehicles by using ultrasonic sensors,” *Machines*, vol. 6, no. 2, Apr. 2018. [Online]. Available: http://www.mdpi.com/2075-1702/6/2/18.

[35] N. Gageik, P. Benz, and S. Montenegro, “Obstacle detection and collision avoidance for a uav with complementary low-cost sensors,” *IEEE Access*, vol. 3, pp. 599–609, 2015.

[36] J. Seo, Y. Kim, S. Kim, and A. Tsourdos, “Collision avoidance strategies for unmanned aerial vehicles in formation flight,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 6, pp. 2718–2734, 2017. DOI: 10.1109/TAES.2017.2714898.

[37] D. Bareiss and J. van den Berg, “Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3847–3853. DOI: 10.1109/ICRA.2013.6631118.

[38] S. Hrabar, “Reactive obstacle avoidance for rotorcraft uavs,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4967–4974. DOI: 10.1109/IROS.2011.6094629.

[39] P. Conroy, D. Bareiss, M. Beall, and J. Berg, “3-d reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning,” Nov. 2014.

[40] S. Roelofsen, D. Gillet, and A. Martinoli, “Reciprocal collision avoidance for quadrotors using on-board visual detection,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4810–4817. DOI: 10.1109/IROS.2015.7354053.

[41] J. Sun, J. Tang, and S. Lao, “Collision avoidance for cooperative uavs with optimized artificial potential field algorithm,” *IEEE Access*, vol. 5, pp. 18 382–18 390, 2017. DOI: 10.1109/ACCESS.2017.2746752.

[42] G. Fedele, L. D’Alfonso, F. Chiaravalloti, and G. D’Aquila, “Obstacles avoidance based on switching potential functions,” *Journal of Intelligent & Robotic Systems*, vol. 90, Jun. 2018. DOI: 10.1007/s10846-017-0687-2.

[43] B. Roderick. “Understanding how ultrasonic sensors work.” (Mar. 2020), [Online]. Available: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm.

[44] Y. Ahmad. “Matlab implementation of artificial potential field.” (May 2020), [Online]. Available: https://github.com/Yaaximus/artificial-potential-field-matlab.

[45] Hyperion Technologies B.V., “Ht-cp400.85 computing platform,” p. 5, Jul. 2019.

[46] National Aeronautics and Space Administration, “Mapping the surface of the moon,” 2014.

[47] Flaticon. The cover has been designed using resources from Flaticon.com. (Aug. 2022).

A

Pseudo-Codes

Algorithm 1: CAPF

Input : Environmental constants and obstacles locations
Output: Convergence or not, number of steps, average minimum distance from obstacles, execution time

$Start\ timer$;
while $\|\boldsymbol{r} - \boldsymbol{r}_t\|_2 > 0.7$ **do**
  Save obstacles within detection range;
  Check minimum distance of each obstacle;
  Check safety parameter;
  if $safety == false$ **then**
    break;
  end
  Calculate angles with obstacles and target;
  Calculate attractive potential to target and then force using $J_t(\mathbf{r}) = \frac{1}{2} k_t \cdot DTT^2(\mathbf{r}))$ and $\mathbf{f}_t(\mathbf{r}) = -\nabla J_t(\mathbf{r})$ Calculate repulsive potential from obstacles and then force using

$$J_{obst_i}(\mathbf{r}) = \begin{cases} \frac{1}{2} k_o \left(\frac{1}{\rho_{obst_i}(\mathbf{r})} - \frac{1}{\rho_o}\right)^2 & \text{if } \rho_{obst_i}(\mathbf{r}) \leqslant \rho_o \\ 0 & \text{if } \rho_{obst_i}(\mathbf{r}) > \rho_o \end{cases} \text{ and } \mathbf{f}_{obst_i}(\mathbf{r}) = -\nabla J_{obst_i}(\mathbf{r});$$

  Sum up attractive and repulsive potential vectors;
  Move to the newly calculated position;
  if $timer > limit$ **then**
    break;
  end
end
Record decision on convergence, number of steps, average minimum distance from obstacles, and algorithm execution time;

Algorithm 2: BAPF

Input : Environmental constants and obstacles locations
Output: Convergence or not, number of steps, average minimum distance from obstacles, execution time

$Start\ timer$;
while $\|\boldsymbol{r} - \boldsymbol{r}_t\|_2 > 0.7$ **do**
  Check safety parameter;
  if $safety == false$ **then**
    break;
  end
  Save obstacles within detection range;
  Check minimum distance of each obstacle;
  Calculate potentials on agent using $J_t(\mathbf{r}) = -\alpha_t exp(-\mu_t DTT^2(\mathbf{r}))$ and $J_{obst_i}(\mathbf{r}) = \alpha_o exp(-\mu_o \|\mathbf{r} - \mathbf{r}_{obst_i}\|_2^2)$;
  Sum up potentials from all detected obstacles to get $J_{obst_T}(\mathbf{r})$;
  Calculate total potential on agent $J(\mathbf{r}) = J_{obst_T}(\mathbf{r}) + J_t(\mathbf{r})$;
  for $i = 1 : number\ of\ bacteria\ points$ **do**
    Calculate distance to target for bacteria point i;
    Calculate total potential on bacteria point i;
  end
  for $i = 1 : number\ of\ bacteria\ points$ **do**
    Select closest bacteria point to target;
    if $bacteria\ potential < agent\ potential$ **then**
      Move agent to bacteria point;
      $selection = true$;
      break;
    else
      Disregard bacteria point;
    end
  end
  if $selection == false$ **then**
    Select random bacteria point;
  end
  if $timer > limit$ **then**
    break;
  end
end
Record decision on convergence, number of steps, average minimum distance from obstacles, and algorithm execution time;

Algorithm 3: IBAPF

Input : Environmental constants and obstacles locations
Output: Convergence or not, number of steps, average minimum distance from obstacles, execution time

$Start\ timer$;
while $\|\boldsymbol{r} - \boldsymbol{r}_t\|_2 > 0.7$ **do**
  Check safety parameter;
  if $safety == false$ **then**
    break;
  end
  Save obstacles within detection range;
  Check minimum distance of each obstacle;
  Calculate potentials on agent using $J_t(\mathbf{r}) = -\alpha_t exp(-\mu_t DTT^2(\mathbf{r}))$ and

$$J_{obst_i}(\mathbf{r}) = \begin{cases} 0 & \text{if } \rho_{obst_i}(\mathbf{r}) > \rho_u \\ \alpha_o exp(-\mu_o \|\mathbf{r} - \mathbf{r}_{obst_i}\|_2^2) & \text{if } \rho_l \geqslant \rho_{obst_i}(\mathbf{r}) \leqslant \rho_u \\ \infty & \text{if } \rho_{obst_i}(\mathbf{r}) < \rho_l \end{cases};$$

  Sum up potentials from all detected obstacles to get $J_{obst_T}(\mathbf{r})$;
  Calculate total potential on agent $J(\mathbf{r}) = J_{obst_T}(\mathbf{r}) + J_t(\mathbf{r})$;
  for $i = 1 : number\ of\ bacteria\ points$ **do**
    Calculate distance to target for bacteria point i;
    Calculate total potential on bacteria point i;
  end
  for $i = 1 : number\ of\ bacteria\ points$ **do**
    Select closest bacteria point to target;
    if $bacteria\ potential < agent\ potential$ **then**
      Move agent to bacteria point;
      $selection = true$;
      break;
    else
      Disregard bacteria point;
    end
  end
  if $selection == false$ **then**
    Select random bacteria point;
  end
  if $timer > limit$ **then**
    break;
  end
end
Record decision on convergence, number of steps, average minimum distance from obstacles, and algorithm execution time;