Distributed Intelligent Systems for a Swarm of Robots

Hazem Mohamed Fawzy Zakaria Eissa

A thesis submitted in partial fulfilment of the Requirements of the University of Greenwich for the Degree of Doctor of Philosophy

DECEMBER 2021

DECLARATION

I certify that the work contained in this thesis, or any part of it, has not been accepted in substance for any previous degree awarded to me or any other person, and is not concurrently being submitted for any other degree other than that of Doctor of Philosophy which has been studied at the University of Greenwich, London, UK.

I also declare that the work contained in this thesis is the result of my own investigations, except where otherwise identified and acknowledged by references. I further declare that no aspects of the contents of this thesis are the outcome of any form of research misconduct.

I declare any personal, sensitive or confidential information/data has been removed or participants have been anonymised. I further declare that where any questionnaires, survey answers or other qualitative responses of participants are recorded/included in the appendices, all personal information has been removed or anonymised. Where University forms (such as those from the Research Ethics Committee) have been included in appendices, all handwritten/scanned signatures have been removed.

……Hazem Mohamed Fawzy Zakaria Eissa……. Date ……02/12/2021……

(Student)

……...…… Dr. Wim J. C. Melis …… Date ……02/12/2021……

(First Supervisor)

Abstract

Area exploration is a task where a robot tries to gain information about an unknown environment. Exploring an unknown area is a challenging task for a group of robots as no pre-made map exists, leading to setting a suitable swarm formation compatible with the area to be explored. Having a suitable swarm formation allows the swarm to preserve the overall exploration time, by distributing sub-tasks for each robot, and collecting relevant data. Current swarm formations such as biologically inspired formations or Probabilistic RoadMap (PRM) tend to have a fixed shape, where robots are positioned in a fixed location point within the swarm, preventing the swarm from adjusting its formation to adapt to the unknown area, thus, are not suitable to explore unknown areas. One needs a more flexible formation, where each robot can change its position within the swarm. Consequently, this research aims to build a distributed robotic swarm formation using fractals.

Fractals have the properties of self-similarity, allowing for an equal distribution of the robots, and recursiveness, allowing for a gradual expansion of a swarm formation. Utilising the properties of fractals allow for a robotic swarm to develop a fractal as a swarm formation. Additionally, changing the parameters of each fractal formation, such as a number of branches, will provide the swarm with the flexibility to adjust the fractal formation and to continue exploring an unknown area. In order to determine both advantages and disadvantages of using fractals as a swarm formation, the first step is to classify each selected fractal into either a line or curve-based formation class to distinguish the similarities and differences in each fractal's behaviour. The second step is to implement the growth rule of each fractal formation using robots to explore an unknown area. The last step is to study the effect of changing the parameters of the of implemented fractal formations toward exploring unknown areas.

The research's outcome shows that using fractals as a swarm formation achieved near the amount of area covered by a traditional exploration method, such as PRM, with 88% less use of robots. Furthermore, fractal formations balances between the number of robots used, and the amount of area covered as each fractal uses only the robots needed to develop specific iterations. The effect of changing the parameters of a fractal formation increases the chance of covering more areas.

Acknowledgement

I would like to express my deep appreciation to my ultimate supervisors Dr Wim J. C. Melis, Mrs Radi Dontcheva, and Prof. David Wray for their patience and their guidance throughout the research in tough times, to all my lecturers, family, and friends for their endless support that makes me reach to the point where I can present my work to the audience.

Nevertheless, I would like to express my gratitude to the University of Greenwich for providing me with the necessary facilities and support to present this research work to the world. Also, I would like to express my gratitude to MSA University for their financial support in making this project happen.

Table of Contents

List of Acronyms

ACO	Ant Colony Optimisation
AI	Artificial Intelligence
d	Branch Distance
FBA	Frontier-Based Approach
GNT	Gap-Navigation Tree
IE	Integrated Exploration
LRTA	Learning Real-Time A
MAS	Multi-Agent Systems
MRS	Multi-Robot Systems
N	Number of Branches
PRM	Probabilistic RoadMap
RRT	Rapidly-Random Tree
PSO	Particle Swarm Optimisation
SI	Swarm Intelligence
SR	Swarm Robotics
UAV	Unmanned Aerial Vehicles
V-REP	Virtual Robotic Experimentation Platform
θ	Initial formation Direction
α	Separation Angle

List of Figures

Chapter 1: Introduction

1.1 Swarm Robotics for Exploring Unknown Areas

Swarm Robotics (SR) is the study of cooperative behaviour within a group of robots. The research effort on SR began in the late-20th century, where studies showed that a group of cooperating robots can perform complex tasks more efficiently than a single robot (Cabrera-Mora and Xiao, 2012). Mainly, SR helps human beings handle heavy tasks in industries, and safeguards humans from taking unnecessary risks in rescuing operations. This research aims to prevent people from getting into hazardous situations where human life is at risk, such as being trapped inside an unknown cave.

To improve the functionality of a swarm when performing complex tasks, SR is currently being integrated with Artificial Intelligence (AI). AI gives SR the ability to use a decision-making process as part of Swarm Intelligence (SI). The decision-making process is mainly based on centralised SR because they are easier to build (Schranz *et al.*, 2020). However, centralised SR provides limited swarm flexibility as each robot is positioned in a fixed location point within the swarm formation, and burdens the communication between a master and slaves, resulting in communication overheads. Therefore, this research uses distributed SR to allow the swarm to efficiently distribute sub-tasks for each robot to explore an unknown area.

Changing SR formation from centralised control to distributed control is the key to studying different SR formations and identifying which swarm formation is suitable to explore unknown areas. At the beginning of an exploration task, it is difficult for a swarm to determine a suitable formation because no information is available about the area. The determination issue leads the swarm to set a fixed swarm formation, perform the exploration task, and broadcast the gathered data to each robot. This approach suffers from limited swarm flexibility as each robot perverse its location within the formation. Therefore, current research reviewed different swarm formations to understand the concept of structuring each swarm formation and their behaviour while exploring areas (Dorigo, Theraulaz and Trianni, 2021). Until swarm formations are sufficiently developed for exploring an unknown area, the better option is to use a human operator (Koch, Manuylov and Smolka, 2021).

While there has been some research work related to swarm transformation models, this research proposes using fractals as a framework for swarm formations, changing the current swarm formation during the exploration task to adapt to an unknown area's structure. The benefit of using fractals lies in their properties, such as self-similarity and recursiveness, which gives the swarm the ability to repeatedly expand a fractal formation until the formation fits the structure of its surrounding area. By using these properties to explore unknown areas, a robotic swarm would restructure the swarm formation, thus, allowing the swarm to quickly and efficiently achieve the task of exploring and covering an unknown area. Therefore, the following research question is raised: What are the advantages/disadvantages of using different fractal swarm formations to explore and cover unknown areas? It is expected that using fractal properties will help the robotic swarm improve its functionality by having a flexible swarm formation.

1.2 Research Motivation

Area exploration has been investigated in different research areas, including geological analysis (Roy, Maitra and Bhattacharya, 2021), searching for a treasure (Pang *et al.*, 2021), and rescue missions (Cardona and Calderon, 2019). The latter one is critically important as it relates to human lives and having an effective exploring strategy is essential to reach and save the lives of human beings. An example of a rescue mission is seen in a recent accident in 2018, where twelve associate football members, alongside their team manager, were trapped inside the Tham Luang Cave in the Philippines due to a heavy rainfall flooding the cave entrance. This flood resulted in a significant change in the cave's structure, in which the current map of the cave was not helpful, and an alternative option was to depend on the geophysical exploration technique (Vichalai, 2019; Ahmed *et al.*, 2021). Although all the members were rescued, it came with the cost of losing a rescue officer. Therefore, using a flexible formation by a robotic swarm to safeguard humans from taking unnecessary risks is one factor that drives the motivation to develop fractal formations to adapt to the change of the unknown environment.

While exploring an unknown area is a challenging task for a robotic swarm due to the difficulty to select a suitable formation, the same challenge applies to finding an object on a borderless area. Exploring a borderless area requires a suitable formation that

optimises resources, such as the number of robots, and time as the area has no boundary. An example of exploring a borderless area is seen in another recent accident where a scheduled flight air Malaysia 370 (MH370) was missing from the air traffic controller reader and disappeared in the Indian Ocean, leaving an area size of more than 70 million kilometres square to be explored (Ashton *et al.*, 2015). The only searching strategy applied at the time was to divide the area size into small grids, each to be extensively covered by a group of scanned planes, ships, and submarines (le Hardy and Moore, 2014). The search strategy exhausted much of the resources needed, and at the time of writing this chapter, there was no success in finding the location of the missing aeroplane, and the search stopped. Having a fractal formation that utilises the needed resources, and efficiently distributes the robotic swarm to cover a large area is another factor that drives the motivation to investigate the properties of fractals and uses these properties for building a flexible formation.

The last and the most important reason for the research undertaken lies in understanding how to use fractals as a swarm formation (Eissa *et al.*, 2018). Fractals have been applied in a minimal number of engineering applications, such as antennas (Anguera *et al.*, 2020) and constructions (Wang and Tang, 2021). However, fractals typically have not been used in robotic's exploration applications, which leaves a gap in the swarm robotics field, raising the question of the benefit of using fractals in robotic exploration tasks. Therefore, the research motivation is to fill this gap by answering the research question.

1.3 Original Contribution to the Knowledge

Because fractals have not been used as a robotic swarm, the contribution in this research lies in understanding the behaviour of fractals to be developed as a swarm formation. The contribution can be achieved by analysing the structure and development process of certain fractals, developing mathematical formulas describing the growth rule of a certain fractal, and discussing the advantages and disadvantages of using a particular fractal formation to explore unknown areas. The main contribution of this research is to add fractal formations as a new swarm formation method. With the potential to implement fractals as a swarm formation, it is possible to enhance the robotics field with fractals as a new research theme.

1.4 Research Overview

This research aims to build a distributed robotic swarm to explore an unknown area using fractals as a swarm formation. Achieving the main aim of this research requires completing the following tasks:

1- Understand the concept of growing a fractal by classifying different types of fractals into different classes according to their similar features. The classification will allow for the creation of similar mathematical formulas for each fractal within the same class.
2- Derive a suitable mathematical formula that allows a robotic swarm to form a particular fractal formation and explore an unknown area. This objective can be achieved by creating formulas for each fractal class. For a particular fractal class, named line-based fractals, one formula sets the number of robots needed to create a fractal formation, and the other formula directs each robot's movement while developing a fractal formation. For another fractal class, named curve-based fractals, one formula that describes the robot's location is needed.
3- Analyse the effect of changing each fractal parameter, individually and in combination, while exploring an unknown area. The purpose of changing a fractal parameter is to adjust the current formation to avoid obstructions and, subsequently, continue the exploration process of the unknown area. This analysis allows the robotic swarm to decide which fractal parameter will be the best candidate for value change when facing an obstruction.

Each objective is examined using a robotic simulation to resemble fractal formations, explore an unknown area, and evaluate the area explored for each fractal formation used.

This thesis comprises the following chapters. Chapter 2 is a literature review that discusses various conventional exploration methods in the robotics field, current swarm formations, and fractals for SR. Chapter 3 focuses on implementing different fractal formation types on a robotic swarm using a developed growth rule formula to explore an unknown area. Chapter 4 investigates the effect of changing different fractal parameters and the effect the changes have on the distribution of the robotic swarm

while exploring an unknown area, and presents an optimisation process for a particular fractal model to obtain the maximum area coverage. Based on the analysis made in Chapters 3 and 4, Chapter 5 answers the research question and sub-questions, which shows both the advantages and disadvantages of using fractals as a swarm formation for exploring an unknown area. Additionally, a future work section is included describing the potential of analysing some more fractals as a swarm formation, and suggesting possible approaches to use a decision-making process to address more formation issues, such as overlapping between fractals' branches and facing obstructions.

Chapter 2: Review of Literature

Multi-Robot Systems (MRS) is a broad research area that focuses on the interactions within a group of robots concerning communication, control, and organisation. Swarm Robotics (SR) forms a sub-field of MRS that studies the coordination of a group of robots while performing numerous tasks such as moving specific objects. One exciting application for SR is area exploration, which includes tasks such as: searching for objects/treasures (Ismail and Hamami, 2021), navigation for rescue missions (Faria Dias *et al.*, 2021), and mapping (Roy, Maitra and Bhattacharya, 2021). SR needs a designated exploration method for each task, which requires certain information about an explored area, such as the area's size and boundaries. Without this information, exploration methods may not function, which is a challenge for SR.

An alternative solution for a swarm of robots to explore an area is to rely on other factors such as a number of robots and a swarm formation type to explore unknown areas. However, using a particular swarm formation may not be adequate to explore an unknown area due to the inability to adjust selected swarm formations, leading to a fixed formation design. Therefore, it is essential to study different robotic swarm approaches used to explore unknown areas.

Understanding different types of exploration methods and different types of swarm formations required classifying them into research themes. Therefore, this chapter reviews unknown areas' exploration techniques used by a swarm of robots as a first research theme. Various swarm formation approaches used in area exploration tasks are investigated as a second research theme. Finally, the possibility of using fractals as a swarm formation towards exploring an unknown area is discussed as a third research theme.

2.1 Exploring Unknown Areas

Exploring an unknown area is challenging as the swarm seeks to gather as much information as possible about the area to facilitate the robot's mission. As swarm's challenge is to explore the known environment using suitable traditional exploration methods, such as sweeping and scanning, researchers have attempted to improve these traditional exploration methods to function in an unknown environment.

Exploration applications can be grouped into two categories: area covering and object searching. Depending on the size and shape of an area, alongside the swarm formation type, covering an area would facilitate the robots' task towards targeting and finding a specific object inside the unknown area. Therefore, the area covering techniques will be the focus of this research.

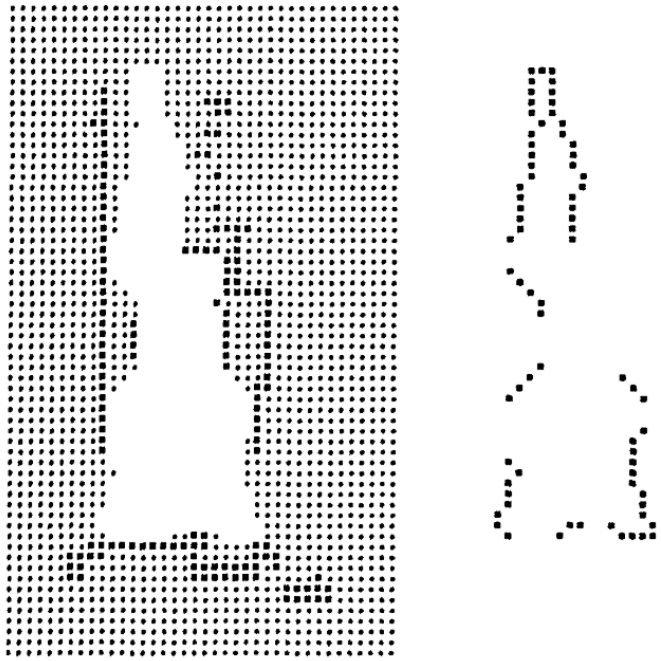
The following sections review conventional exploration methods, including grid patterns, path planning, and a modern exploration method using multi-agent systems.

2.1.1 Grid Patterns for Covering Unknown Areas

The idea behind the grid pattern approach is to divide an area into smaller sections called cells. Each cell is explored using either a single or multiple robots depending on the task requirements, e.g. finding a treasure or surveillance. For example, using this principle, each robot would be required to update a map of a specific area by occupying and covering assigned cells using proximity sensors (Stachniss and Burgard, 2003a). An important application using this mechanism is where a robot scan selected cells to guard a specific area against intruders (Ahmadi and Stone, 2006).

However, for exploring unknown areas, the grid method might be inadequate to use by a robot because the area's information, such as size and boundary, is unavailable. This information is vitally important for the grid method to determine the number of cells and the size of each cell. Hence, the grid pattern method was improved by adding supporting methods that gain as much information as possible from an unknown area. One supportive approach can determine the size of an unknown area by exploring the boundary between open space and undiscovered territory called the Frontier-Based Approach (FBA) shown in Figure 2.1 (a) (Yamauchi, 1998). FBA helps grid methods to determine the number of cells needed to cover an unknown area. However, FBA does not take into account existing obstacles inside an unknown area, which can have a size bigger than the cell's dimension leading to an inaccurate determination of the number of cells needed; therefore, as the area is unknown, determining a suitable size of each cell is impossible. An improved grid method, called the evidence grid, shown in Figure 2.1 (b), uses a spatial representation to determine the possibility of occupying and covering grid cells (Moravec and Elfes, 1986). The evidence grid combines the information about area occupancy coming from different sensors, which helps a robot

build an accurate area map and increases the chance of recognising obstacles. However, the evidence grid is not concerned with determining the size of an unknown area, making it difficult for a robot to determine the number of grid cells needed (Schultz and Adams, 1998; Yamauchi, Schultz and Adams, 1998). In addition, the information gained by the evidence grid method might be incorrect due to the odometry error resulting from a robot's movement (Schultz and Adams, 1998).



(a) (b)

Figure 2.1 The coverage of the area's frontier using (a) FBA, where the black dots represent the edge segments of the area's boundary (b) Evidence grid, where frontier edge segment is detected. It is noticed in both methods that a robot could not ultimately discover the frontier of an area due to facing obstacles (Yamauchi, 1997) - Used with permission.

Using FBA and evidence grid approaches can help the grid method gain more accurate information about an explored area. However, the up-to-date location of each robot, where the information is obtained, is not accurate. The inaccuracy of a robot's location is called the location error, where the asynchronous movement of robots causes an error that is typically accumulated, resulting in an incorrect calculation of a robot's coordinates. Researchers tried to solve the location issue by combining grid methods with location-supporting methods, such as localisation methods, that help a robot detect its position within the area. One particular example is the localisation method

introduced by Yamauchi (1999), where FBA, simultaneous localisation, and map building are integrated into Integrated Exploration (IE) system, enabling a robot to determine its current position. However, the system is heavy and requires a level of processing that a low-cost development board cannot handle, resulting in a robot's difficulty in processing the IE system (Makarenko *et al.*, 2002).

Using the grid method to explore an unknown area benefits the swarm by covering part of the area and learning its structure. However, the grid method reaches a point where it becomes limited in discovering unknown areas, and grid-supporting tools are needed. Combining various grid-supporting methods cannot guarantee to gain information about an unknown area as a single robot may not be able to combine more than a grid-supporting tool due to its limited processing capabilities. Moreover, the overall area coverage completion time is severely affected because a single robot must simultaneously handle multiple tasks, such as collecting the area's information and determining its location. One needs to consider developing a new exploration algorithm that specifies directions for a robot to follow. Therefore, path planning was introduced.

2.1.2 Path Planning for Object Searching

Path planning, also known as motion planning, is a method that identifies an optimal path to safely guide a robot from a start point A to an endpoint B (Lavalle, 2006). One of the most common path-planning techniques is the randomisation technique which includes: Probabilistic Roadmap Algorithm (PRM) (Kavraki *et al.*, 1996) and Rapidly Random Tree (RRT) (LaValle, 1998). The idea of the PRM algorithm is to place random waypoints in free space and connect nearby points to create an optimal path between the start and the finish point using a local planner. Figure 2.2 shows the PRM algorithm setting possible road lines/curves on a simple and a complex area to determine a suitable path for the robot to follow.

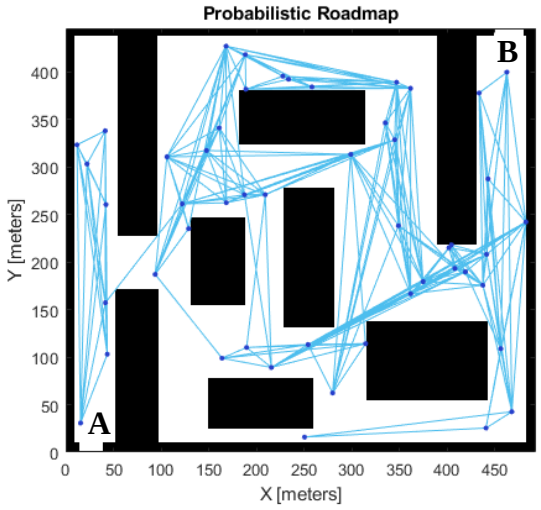


Figure 2.2 A simulation example of a path planning algorithm showing the possible path for a robot from the start point A to endpoint B.

Unlike PRM, RRT uses the concept of a space-filling tree, where a tree is incrementally constructed according to random points placed in a selected area. The structure of RRT is more like a stochastic fractal as the fractal branches are repeatedly generated but then with a random length[1].

Using the randomisation technique in RRT results in a number of problems. The first problem is that RRT does not always find a path between two points, especially in complex areas. The second problem is that RRT requires a pre-made map of an area, and consequently, this technique cannot be used to explore unknown areas. In order to address these problems, alternative path planning techniques, called path-finding, were developed, such as the A-Star algorithm (Hart, Nilsson and Raphael, 1968) and the D-Star algorithm (Stentz, 1994), also known as Dynamic A-Star.

The A-Star algorithm uses an incremental search method to find the nearest path between multiple points, called nodes, while the D-Star algorithm uses a heuristic search method repeatedly to restructure paths when changes occur. Both algorithms are computationally slow when used in a largely unknown area, and their search function may terminate when a robot faces obstacles. Both algorithms need either supporting tools or learning functions to avoid obstacles (Foead *et al.*, 2021).

[1] For fractal definition and more information about fractals, see section 2.3

While path planning algorithms can accurately guide a robot to explore known or unknown areas, the overall exploration task takes as much time to complete compared to the grid method, especially for unknown areas. In addition, guiding a single robot to explore an unknown area is a challenging task because a robot will have to do multiple tasks simultaneously, including scanning, collecting information, updating its location, etc. Therefore, the total exploration time will dramatically increase, and the robot might not complete exploring an unknown area due to the fact that the estimated exploration time cannot be determined. An alternative solution is to consider improving the time and the robot's capability to cover an unknown area. From a robotics perspective, distributing exploration tasks to partner robots is an efficient solution to increase each robot's exploration performance and decrease the total exploration time.

2.1.3 Multi-Agent Systems

Multi-Agent Systems (MAS) is the most recent solution for exploration applications because it can integrate with traditional exploration methods, such as frontier scan and probabilistic roadmap, by distributing sub-tasks between the robots (Kwa, Leong Kit and Bouffanais, 2022). MAS study the interaction between agents that can solve problems based on environmental perception. As MAS contain various applications in different research fields, one particular application of the MAS called Multi-Robot Systems (MRS) is used to focus on the behavioural response of a robotic swarm when exploring areas using different swarm formations.

MRS consider the interactions between a group of robots while they are performing numerous tasks. MRS is a modern approach that uses four factors to explore areas, namely: communication, control, organisation, and decision-making. MRS allows researchers to develop traditional exploration methods into more modern methods and implement them for a group of robots. For example, the Trapezoid exploration strategy is a grid method application consisting of neighbouring local grids that shape a global grid. Each robot has its local grid to cover, and by sharing the coverage status with neighbouring robots, each robot can decide what to cover next (Sharma and Tiwari, 2016). For the path planning approach, an alternative method called Gap-Navigation Tree (GNT) (Nasir and Elnagar, 2015) is being used for MRS, where robots are constructed to form a vital link based on their distance between each other, aiming to reach the maximum depth inside an unknown area. The usage of MRS improves the

robot's performance as each robot is assigned a specific task, and in this way, the overall exploration time is reduced. Furthermore, the research on MRS is expanding to include methods on swarm control, swarm formation, swarm communication, etc.

While each exploration method controls the swarm's organisation and communication, choosing the proper method increases the swarm's chance to cover more areas and minimises the overall exploration time. Therefore, having a decision-making method is crucial for the MRS. For instance, a simple decision-making method is presented by Kernbach et al. (2013) where a swarm of micro robots, with limited sensing capabilities and no direct communication, decide their spatial location using a thermal tactic aggregation of bees. Although this simple method proves that a decision made by limited-capability robots can be achieved, it does not provide a decision when facing an obstruction forcing the robot to hold on to its last position. Additionally, the method is a hardware-based decision in which the swarm uses its sensing capability, such as a thermal detector, to attract the swarm to a thermal source, such as heat, in the absence of cooperative decision making. Therefore, this method is unreliable for exploring unknown areas as each robot will collide with either an obstacle or another robot. An advanced cooperative decision approach proposed by Marjovi et al. (2009) uses both frontier-based exploration and A-Star searching techniques to localise a fire source and minimise the overall exploration time, which relies on the cost-gain ratio as a decision evaluation. This advanced method uses exploration methods that need a map of the area to function, which is impossible when exploring unknown areas. In other words, the cost-gain ratio cannot be established without a pre-made map. In addition, as the structure of an environment changes over time, the pre-made map will be outdated and invalid to use, leading to an incorrect evaluation of a decision-making process.

More recent research by Faria Dias et al. (2021) reported many drawbacks in collision avoidance between robots and obstacle detection, especially when exploring unknown areas, and emphasised the importance of using swarm formation. Swarm formation methods, e.g. biological-inspired (Oh *et al.*, 2017) and pattern formation (Xu *et al.*, 2010), cannot change the swarm's formation as they are designed to create a fixed shape. As a swarm formation affects the robots' task distribution, synchronisation and decision-making when exploring areas, it is necessary to review the latest methods of SR formations. Reviewing SR formations will help locate the point where the swarm

will need to change its formation, thus, facilitating the task of exploring an unknown area.

2.2 Swarm Robotic Formations

Swarm formation defines the group organisation between different robots within the swarm. When a swarm performs an exploration task, choosing a suitable swarm formation is essential to adapt to the area being explored. Therefore, determining the method of developing a swarm formation when exploring unknown areas is critically important to ensure the swarm's discovery process. Consequently, studying different swarm formation methods is important to understand the choice of setting certain formations for specific robotic tasks. Also, to investigate the possibility of applying a swarm transformation when facing an obstruction for exploration purposes.

This research theme contains two sections, one section which reviews different swarm formation methods named section 2.2.1, and the other section reviews swarm transformation models for exploration purposes named section 2.2.2.

2.2.1 Swarm Formation Approaches

Swarm formation describes the organisation of autonomous robots while structuring themselves to a certain formation. Swarm formations can be classified into two categories: biologically based or mathematically based formations. Additionally, this review presents several swarm formation approaches depending on the swarm's control type, which can use either a distributed or a centralised controller.

The source of biologically based formations lies in formations seen in creatures such as flocking birds, school of fish, ant colony, bees, etc. (Olaronke *et al.*, 2020). The typical applications used by biologically based formations are rescuing human beings and object-searching missions (Xiong *et al.*, 2009). For the mentioned applications, a swarm must explore the area assigned for its mission, especially when the area's structure is unknown. For example, an experiment described by Cheng, Wang and Dasgupta (2009) mined a flocking-bird formation, shown in Figure 2.3, to explore area shapes such as a square and a triangle. The experiment compares the coverage percentage between different flocking-bird formations, including line-flocking, V-flocking, and hybrid-flocking. Additionally, the experiment compares the flocking-bird

formations with a stochastic swarm formation, where the distribution of the swarm formation is not predicted. The results show that the stochastic swarm formation can cover about 20% more area than the flocking-bird formation types. This is also confirmed in Stachniss and Burgard (2003b); Ahmadi and Stone (2006), which show that flocking-bird formations need improvements for area coverage applications, especially when compared to other biological formations.

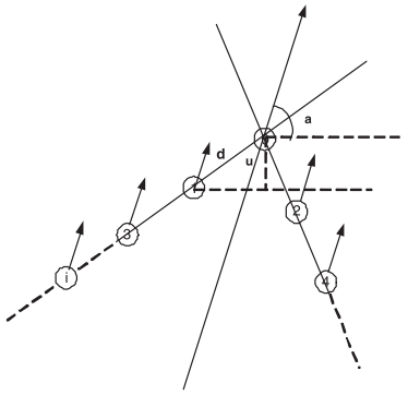


Figure 2.3 The designed formation of the V-flocking birds used to cover a specific area. Each unit (robot) is separated by distance *d* and with a separation angle of *a* (Cheng, Wang and Dasgupta, 2009) - Used with permission.

Ant and bee formations have several useful features to improve area coverage. For example, ants use pheromones that leave marks to guide the swarm while exploring their colony using Ant Colony Optimisation (ACO) (Hunt, Jones and Hauert, 2019). Bees memorise an explored area by sending scouting bees to explore, for example, a rose field, and performing a dancing process called the waggle dance to describe both the direction and the level of suitability taken by a bee to the swarm (Karaboga and Akay, 2009). The reviewed features are mainly used for foraging purposes and can therefore also be applied to explore unknown areas.

An example of ant formations is shown in a study that compares several methods of creating ant formations, including node-counting and Learning Real-Time A Star (LRTA*) while covering an area (Koenig, Szymanski and Liu, 2001). Although the study shows good coverage efficiency for ant formations, the total coverage time and the number of steps applied by a robotic swarm are relatively high compared to the use of other biological formations i.e. bees (Gordon, Wagner and Bruckstein, 2003). That

is because some of the ant formations rely on central swarm control, which affects the time for formation creation. Another decentralised ant-based formation named Stochastic Diffusion Search (SDS) is an intelligent algorithm that mimics the recruitment of a specific ant behaviour (Majid-al-Rifaie and Bishop, 2020). SDS manages to distribute useful information to the rest of the swarm in an effort to locate a target e.g. metastasis (Majid al-Rifaie, Aber and Raisys, 2013); however, the swarm formation is stochastic and does not control the distribution of a robotic swarm. Additionally, SDS terminates at a fixed activity rate, which may not be suitable for an unknown area exploration.

The bee algorithms are mainly used for foraging applications, and therefore, a little work focuses on using bee formations for area-coverage applications. An example of a foraging application is described by (Efremov and Kholod, 2020), where a non-pheromone algorithm inspired by bee behaviour shows a lower foraging time compared to a pheromone-based algorithm inspired by ants. However, as bee algorithms use a stochastic swarm formation, the total area coverage time is unstable and therefore, it is difficult to determine the total coverage time (Jevtić *et al.*, 2012).

A school-of-fish formation features a formation called the aggregation formation and attempts to cover a specific area. For example, an experiment conducted by (Yang and Tian, 2007) aimed to develop an aggregation formation using the school-of-fish model to surround a target. As the model shows a good performance in avoiding obstacles within an unknown area. The model is randomly searching for a target rather than exploring the unknown area, leading to inefficient use of the model and the exploration time. Other fish school formation methods such as Spatio-temporal (Vedachalam *et al.*, 2020) and self-organised school of fish (Thrun and Ultsch, 2021) are used for navigation applications. Like bee formation, there are only a limited number of studies that use school-of-fish formation for area-coverage applications because of the fixed formation design.

Overall, there are several biologically based formations used for area exploration. However, biologically based formations are robust and cannot be adjusted due to their design restriction, e.g. flocking birds formation is fixed to a V-shape design, leading to a non-adjustable swarm formation which can be inadequate in exploring or covering unknown areas. Therefore, biologically based formations have been optimised to

improve their exploration performance by using features from other swarm creatures. For example, Particle Swarm Optimisation (PSO) is a metaheuristic optimisation method that uses flocking bird behaviours to explore areas (Dadgar, Jafari and Hamzeh, 2016), while ACO is improved by combining different ants' behaviours such as carrying, foraging, etc. for each robot within the swarm (Lima and Oliveira, 2017). A hybrid algorithm combining ACO and SPO is used to control the swarm's ant formation and the communication between neighbouring robots (Amar and Jasim, 2021). Although these optimised formations might improve the total time of covering an area, the designed formations remain fixed, leading to inadequate use of biological-based formations to explore an unknown area.

The review of some biologically based formations raises an important question: is it possible to build a flexible swarm formation that can restructure its current formation to explore unknown areas? Some biologically based formations, such as flocking birds, can be mathematically built using geometric shapes such as a square, a rectangle, a circle, etc. Therefore, it is necessary to review some of the mathematically based formations used to cover unknown areas. Within the SR field, examples of mathematically based formations are self-organisation, pattern formation, and reconfigurable robots.

Self-organisation is a process of interaction between autonomous robots to build a formation that can be used for a specific purpose (Ashby, 2004; Trianni, Nolfi and Dorigo, 2008). The concept of self-organisation is based on a random distribution of a robotic swarm, where there is no leader, and each robot acts individually to interact with the other robots and construct a swarm formation suitable for completing the assigned task. Self-organisation is used mainly in mimicking biologically based formations because it functions spontaneously, and therefore, there is no particular formation for self-organisation. Self-organisation is a type of organisation that allows a robotic swarm to restructure itself to a different formation.

Pattern formation describes the interaction between robots to construct a specific shape in an organised procedure, as shown in Figure 2.4. Both pattern formation and self-organisation have the same approach of restructuring certain formations for a robotics application (Varghese and McKee, 2009). Several pattern formation algorithms have been developed for searching and covering techniques such as grading and path-

planning. Like self-organisation, pattern formation does not have a specific formation structure and needs to build a suitable pattern for an exploration application. For example, a control law system was made for a group of Unmanned Aerial Vehicles (UAVs) to implement the desired swarm formation during area discovery (Koo and Shahruz, 2001). The system involves the use of a central UAV that builds a suitable pattern formation. Pattern formation relies on swarm control, and therefore, without having information about an unknown area, pattern formation will not be appropriate. Other examples of the use of pattern formation were described in the swarm formation approaches section 2.21 (Malchow *et al.*, 2000; Gordon, Wagner and Bruckstein, 2003; Xu *et al.*, 2010).

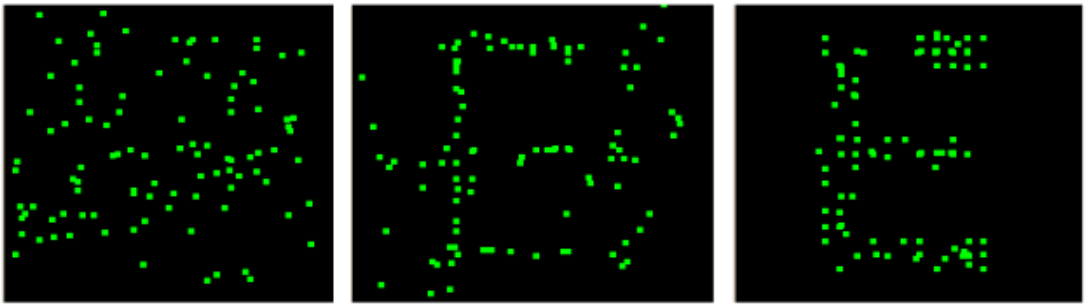


Figure 2.4 A group of robots using self-organisation to form a pattern of the letter (E) (Xu *et al.*, 2010) - Used with permission.

Reconfigurable robots, also known as modular robots, are a group of autonomous units that can mechanically attach together to form a specific shape (Kurokawa *et al.*, 2008). Modular robots do not need formation algorithms but simple instructions to form a shape. These units can either have the same structure units known as homogenous modular robots or have different structural units known as heterogeneous modular robots. Modular robots are primarily used in search and rescue tasks due to their capability to structure different shapes. For example, a framework is made for a team of modular robots that can compensate for lost or failed units when searching within an unknown environment (Gunna and Anderson, 2013). Also, modular robots can perform a task using either a centralised or a decentralised swarm control (Abukhalil and Sobh, 2013; Saldana *et al.*, 2017). As modular robots are hardware-based control, each unit relies on one another to construct a formation. Therefore, modular robots can perform only a limited number of formations depending on the number of units, which is not suitable for unknown area exploration.

2.2.2 Transformation Models

Swarm transformation is a process of restructuring an existing formation by either changing between different swarm formations or modifying the same formation. Commonly, the transformation process requires each robot to change its location so the swarm can establish a new formation. In SR field, researchers have proposed several swarm-transformation models such as: composition and decomposition (Jermann *et al.*, 2006), self-organisation (Ashby, 2004), and repositioning (Varghese and McKee, 2009) in order to improve the swarm's abilities to surpass exploration challenges, such as avoiding obstacles, SR needs to determine the subsequent formation to change Additionally, the process of swarm transformation shows significant difficulties in repositioning each robot within the swarm. Therefore, this section discusses swarm issues while performing a transformation process.

The major problem of swarm transformation lies in the determination of a suitable new formation. While performing a swarm transformation, the swarm's target is to establish a formation that helps to explore an area. Swarm formations can be classified into two types: filling and non-filling formations. Filling formations can be applied to covering an area of a determined shape, while the non-filling formations describe the coverage of the perimeter of a determined shape (Cheng, Cheng and Nagpal, 2005). The benefit of a filling formation lies in covering a simple area, but for a random shape and a finite number of robots, there is no guarantee of covering an area completely. Thus, the filling formation does not help cover an unknown area. Determining a particular formation is made possible using the non-filling type because of its simplicity in performing using a robotic swarm (Varghese and McKee, 2009). However, neither the filling nor the non-filling swarm transformation types can trace complex area shapes because of the difficulty of controlling each robot, resulting in robotic collisions (Cai *et al.*, 2007). As a result, research work on determining a feasible formation using either filling or non-filling types is limited to geometric shapes (Jermann *et al.*, 2006).

Another problem with swarm transformation lies in the repositioning of the robots. When performing a swarm transformation, a robot's primary role is to change its position to support/create the newly desired swarm formation. However, a part of the robotics swarm might fail to reposition itself on a particular occasion. For instance, during a decomposition process, where all robots are separated from each other, a

communication failure could lead to an enormous collision and breakdown of the swarm formation, resulting in a failed transformation. Also, for a centralised swarm, an error caused by the leader robot will result in a repositioning failure and a termination of the transformation process. Researchers are working on integrating transformation models with supported robotic mechanisms, such as path planning and collision avoidance, to prevent problems caused by repositioning (Y. C. Chen and Wang, 2007). Still, the work on a method for repositioning robots is not being addressed. Additionally, other swarm transformation problems affect formation stability (Varghese and McKee, 2010), the robot's role assignment (Y. G. Chen and Wang, 2007), and the coordination of multiple swarms (Hsu and Liu, 2004).

In conclusion, the current work on swarm transformations reveals many deficiencies in functionality and performance. The current swarm transformation models for exploring unknown areas cannot deal with complex areas and recognise complex shapes. Hence, swarm transformation needs a development process by building a modern model to form flexible shapes. Using flexible shapes will improve the swarm performance in exploring unknown areas and allow a robotic swarm to adapt to complex environments. Therefore, this research aims to develop a swarm transformation model that uses shapes inspired by nature.

2.3 Fractals for Swarm Formations

Restructuring a flexible shape is an essential swarm process to enable exploring complex areas. Therefore, this research attempts to understand the process of creating flexible shapes by observing natural phenomena. The concept behind the natural phenomenon is the growth of simple shapes to form complex shapes. These natural phenomena are known as fractals. For instance, the structure of a fractal tree is based on the growth of several line branches, while the structure of a water vortex is based on the growth of different-sized circles.

A fractal is a recursive decomposition process of a basic shape into scaled patterns (Peitgen et al, 2004). Fractals can either be found in nature, i.e. snowflakes and clouds, or mathematically formed, i.e. Sierpinski triangle as shown in Figure 2.5.

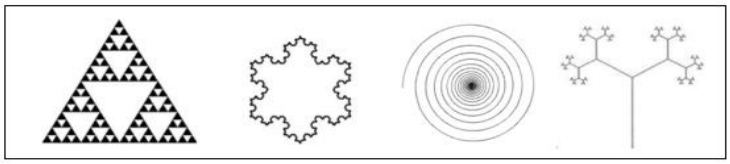


Figure 2.5 Common fractal shapes (from left to right): Sierpinski triangle, Koch snowflake, water vortex, and 2-branch tree (Peitgen *et al*, 2004) - Used with permission.

This section presents a classification of fractals and any work reflected in using fractals in swarm robotics.

2.3.1 Fractal Classification

Fractals can be classified based on self-similarity, shape, dimension, and flow direction. Self-similarity is the process of developing patterns that repeats the same fundamental shape known as an exact-similar shape. Patterns can also represent part of the decomposed shape's properties, which is known as a quasi-similar shape. The decomposition of an original shape into patterns with a different structure and property is known as a statistical shape (Falconer, 1990).

The definition of a shape classification is a presentation form of a particular object (Peitgen et al, 2004). A fractal shape can have either a deterministic or a random formation. The first type can be a geometric or an algebraic shape, while the second is stochastic. A deterministic formation is easy to decompose into smaller patterns and to calculate the formation's dimension. For random formations, several stochastic creation methods can be used to analyse the rule behind the creation of the random shape, such as Brownian motion, percolation, Levy process(Yang *et al.*, 2017), and chaos theory (Mandelbrot, 1983).

The dimension classification uses the parameters of a different shape, such as perimeter, area, or volume. The purpose of dimension classification lies in understanding the creation of a random formation. Dimension classification presents a complexity index of different fractal shapes and measures their roughness and regularity levels (Cheng *et al.*, 2012). Several methods are estimating the dimension of complex shapes, such as Hausdorff, box-counting, euclidean, and topological (Gneiting, Ševčíková and Percival, 2012; Balka, Buczolich and Elekes, 2015). Each of these dimension methods can simplify the complexity of different random or deterministic formations.

In the field of swarm robotics, there is limited research that attempts to use fractals. For example, a swarm of robots was used to form shape-based fractals such as space filling, tree and curve-based fractals, but these shapes have not been used in area exploration applications (Zhou and Goldman, 2017). Another attempt of using fractals is by structuring a robotic vortex formation to understand the aggregation behaviour of a school of fish (Yang and Tian, 2007).

The review of fractals and their properties shows that the work on integrating fractals within the SR field is at an early stage. Further, the review demonstrates the swarm's ability to build fractal formations. As it is possible to form different fractal formations, this research aims to use fractal properties to build different fractal formations with dynamic changeability towards exploring unknown areas.

2.4 Research Gap

Recent work on developing swarm transformations does not consider exploration applications as it shows several problems regarding swarm organisation. While MRS provides a feasible approach to exploring unknown areas compared to path planning and grid patterns, conventional swarm formations are inadequate for exploring unknown areas due to the swarm fixed design. Finally, based on reviewing fractals, it is possible to structure fractal formations using SR.

Fractal formations have typically not been used in area exploration applications. Also, applying fractals to swarm transformation methods has not been reported. Using fractal formations for exploring unknown areas will allow the swarm to control the number of robots used, based on the fractal type. Also, fractal formations will increase the overall area coverage by adjusting different parameters, described in chapter 4, to allow for a flexible distribution of the robots. Hence, the research gap is to determine the benefits of applying fractal formations in exploring unknown areas.

Therefore, the research question is formulated as follows: "What are the advantages/disadvantages of using different fractal swarm formations to explore and cover unknown areas?" Answering the research question requires answering sub research questions: the first sub research question is as follow: "How can a swarm of robots apply a fractal formation to unknown areas". The second research question is as

follow: “What are the pros and cons of changing parameters of a fractal formation towards covering unknown areas”.

As this research aims to determine both the advantages and disadvantages of using each fractal formation to cover an unknown area, Chapter 3 demonstrates the growth rule of different fractal formations by a swarm of robots by developing a mathematical formula that a robotic swarm can implement. Chapter 4 shows both the pros and cons of changing each parameter of a fractal formation class when covering an unknown area by a robotic swarm. Chapter 5 presents a reflection on the outcomes obtained by the previous chapters leading to present the contribution made to the SR field and considered for future work.

Chapter 3: The Mathematical Modelling of Fractals for a Swarm of Robots

This chapter describes four fractals named: N-Branched Tree, Vicsek, Julia set, and Reverse Julia set, implemented as a robotic swarm formation. A demonstration of developing each fractal formation is made using the MATLAB platform, and implementation of each fractal formation is made using Virtual Robot Experimentation Platform (V-REP). This chapter contains four sections: the first section classifies the fractals according to their development process into two types, namely: line-based formation and curve-based formation. The second section shows the growth rule for line-based fractal formations resulting in a mathematical formula for each formation adequate for a robotic swarm. The third section shows the growth rule for curve-based formation. Additionally, a simulation of how each fractal formation is made, alongside improvements in the robotic swarm's distribution while implementing a fractal formation. The final section summarises the overall work made as well as clarifies the contribution to swarm robotics research.

3.1 Fractal Classes

A fractal is a representation of objects, e.g., geometric shapes, which inherently has the ability to recure and has similar patterns to an original object. Observing natural fractals, such as plants, clouds, crystals, etc. shows a common growth concept of recuring a shape in a self-similar structure. However, the mathematical representation for each fractal can be different according to the original shape, which can be as simple as a line segment to a complex shape. Therefore, it is necessary to classify fractals based on the development of similar shapes. For instance, a line segment can be presented using a linear formula, while a curve shape can be presented using a non-linear formula. Therefore, fractals are classified according to their mathematical development into two types named: line-based fractals, where a fractal formation is developed linearly depending on one or two variables, and curve-based fractals, where a fractal formation is developed depending on a polynomial function.

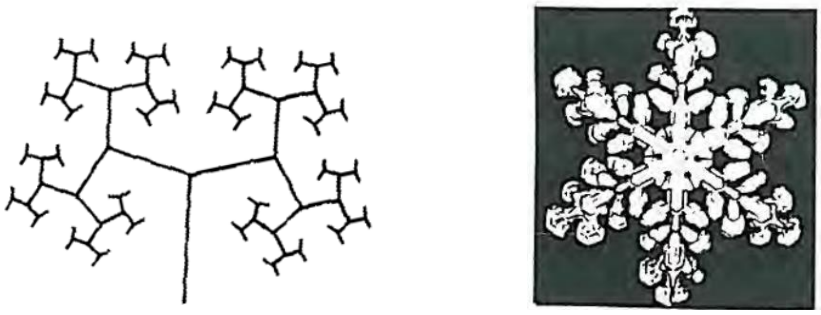
The purpose of classifying fractals is to develop a mathematical formula for each fractal, to be implemented by a robotic swarm. For the swarm robotics research, each fractal class presented with two fractals that have the same original shape, such as

straight line or curved line, and can be mathematically modelled and implemented by a swarm of robots. The selection of a fractal was inspired by observing the changes in plants and climate. Plants can have different structures such as trees, grass, flowers, etc., which recured themselves in identical patterns. Plants can adapt to the change in the environment by changing their structure frame when growing. For instance, planting two seeds of an apple in different environmental conditions can result in two different apple trees with different branches and leaves. A tree skeleton is seen as a number of line segments grown as branches, which can be described using a linear formula, and therefore, a tree fractal can be classified as a line-based fractal class.

From a fractal perspective, climate represents the regular pattern of weather for a period of time, resulting in unusual weather behaviour. For example, having snowy weather results in snowflakes that are a fractal shape, and the low air pressure can result in a cyclone, and thunderstorms, which are also a fractal shape (Falconer, 1990). Unlike plants, different climate behaviours can be seen as a complicated shape, which cannot be represented using a linear formula. For example, cyclone behaviour has a vortex shape that revolves around itself in a curve shape. A cyclone can be seen as a curved line, which can be structured using a non-linear formula, and therefore, can be classified as a curve-based formation.

A tree fractal consists of a main branch called a trunk that develops a smaller pattern of branches about the same shape as the trunk. Each tree's branches can be seen in Figure 3.1 (a) (Vicsek and Gould, 2013). In addition, the linking point, where two or more branches are joined together, can be modified. Considering the observation made on a tree formation in nature, structuring a tree fractal formation for a robotic swarm required a mathematical formula that controls the number of robots used and their position through the fractal formation.

The Vicsek fractal is a snowflake-inspired formation where all the branches grow symmetrically in all directions from a single point, as shown in Figure 3.1 (b) (Vicsek and Gould, 2013). The simplest form of a Vicsek fractal formation is by having four branches grown in a cardinal direction, and these four branches continue recurring to their cardinal direction as patterns. Each of the Vicsek branches can be developed as a straight line that a robot can follow, and therefore, can be described as a line-based formation.



(a) (b)

Figure 3.1 (a) A skeleton of a tree fractal (b) A structure of a snowflake which the Vicsek fractal is inspired from (Vicsek and Gould, 2013) - Used with permission.

Another fractal which can be described in a curve-based class is Julia Set. Julia set is a fractal shape produced using a polynomial of complex values. Recuring a polynomial of complex numbers can result in complicated yet organised cyclone shapes. The Julia set fractal can be represented in different shapes by changing the values of the complex numbers (see chapter 4 Section 4.3), and therefore, it can have different shapes. Figure 3.2 shows a number of developed cyclone shapes in different location points (Falconer, 1990). These location points can be used to guide a swarm while discovering unknown areas.

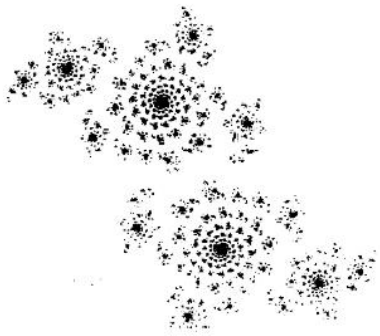


Figure 3.2 One set of the Julia fractal sets resembles different cyclone shapes (Falconer, 1990) - Used with permission.

The reverse Julia set is a novel creation of a fractal formation inspired by the Julia set fractal. A polynomial function is modified to develop location points in a reverse growth direction of the Julia set. The reverse Julia set expands its curved lines outwards by reversing the Julia set formula, which helps the swarm explore unknown areas. Like the Julia set fractal formation, the reverse Julia set forms part of the curve-based class. The structure of the reverse Julia set resembles the shape in Figure 3.2.

The following section presents a growth rule formula for each fractal that describes its development process mathematically. Two elements were considered when creating a growth rule formula. The first element is the number of robots needed for developing a fractal formation. The second element is the direction of the swarm's movement as the fractal develops. For the line-based formation, the growth rule consists of two formulas, one for determining the number of robots needed for developing a fractal formation, and the other for determining the direction of each participating robot within the fractal development. For the curve-based formation, only one formula is needed to assign the direction of the robots, while the fractal formation can be developed using a minimum of two robots.

3.2 Line-Based Fractals

This section presents a detailed description of structuring two line-based fractal formations, namely: the N-Branch Tree Fractal Formation and the Vicsek Fractal Formation, by creating a growth rule formula that can be implemented using a robotic swarm. In addition, a demonstration of each fractal formation is made using MATLAB, and a robotic simulation tool called Virtual Robot Experimentation Platform V-REP.

3.2.1 N-Branch Tree Fractal Formation

The structure of an N-branch tree fractal formation is based on line segments. Taking that a swarm requires at least two robots, one robot will create the first line segment, which sets the trunk of the tree for the first iteration. Based on selecting the number of branches (N), the next robots will follow the first robot to form the next line segment, (branches). The tree fractal formation is developed using the fractal’s properties of self-similarity and recursiveness. Each robot moves a specific distance (d) with an angle of separation (α) for each iteration, as shown in Figure 3.3.

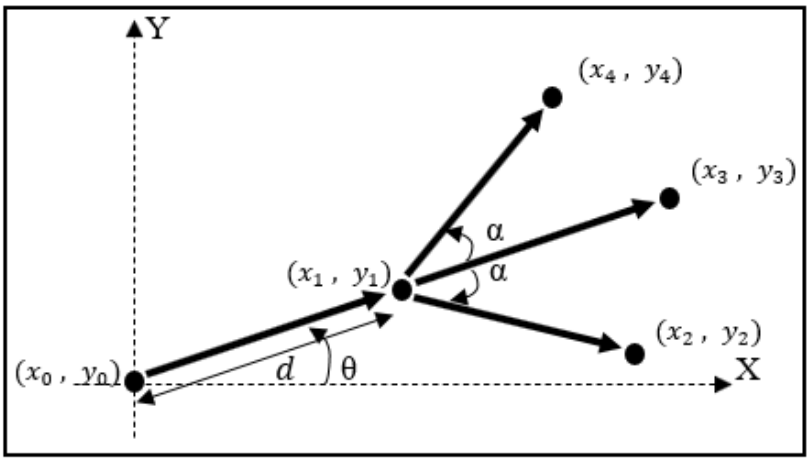


Figure 3.3 Structure of the first iteration of a 3-branch tree fractal formation with N=3.

Taking a 3-branch tree fractal formation as an example, the structure of developing a 3-branch tree fractal formation goes as follow: the main tree line (trunk) is directed by the formation direction angle (θ), and each line must branch out exactly three segments of lines. The middle branch continues from the tree trunk, and the side branches are symmetrically separated by angle (α). Each robot is travelling at a distance (d). Determining the location of each robot in the formation is based on the cartesian coordinates, where each robot has its own specific location in every developing process. Figure 3.3 illustrates the growth process of the 3-branch tree fractal formation, including the location of each robot.

Determining the exact Number of Robots (NoR) needed for the N-branch fractal tree formation is described in Equation (3.2), where (N) is the number of branches. While the 0th iteration (i=0) requires only one robot, the first iteration (i=1) is expressed as ($NoR_1 = N^1$). The second iteration (i=2) is expressed as follow ($NoR_2 = N^2$). The third iteration (i=3) is expressed as ($NoR_3 = N^3$). Overall, the i^{th} iteration for a tree fractal formation is expressed in Equation (3.2) as follow:

$$NoR_{i^{th}} = N^i \quad for\ i > 0\ , N > 1 \qquad (3.1)$$

Therefore, the total number of robots needed for all iterations is:

$$Total\ NoR_{i^{th}} = \sum_{p=1}^{i} N^p \qquad for\ p > 0\ , N > 1 \qquad (3.2)$$

Where (p) is an iteration counter ranges between (1 and *i*). Applying a 3-branch tree fractal formation, (N) = 3, and the first three iterations, according to Equation (3.1), resulted into the following number of robots respectively: 3, 9, and 27 robots.

Assuming the starting points x_0 and y_0 are the origin points of the formation, the robots will travel to the next coordinates x_1 and y_1 using the below movement formulas:

$$x_1 = x_0 + d\cos\theta \tag{3.3}$$

$$y_1 = y_0 + d\sin\theta \tag{3.4}$$

At this coordinate point, the swarm will apply the growth rule for a symmetric tree branch. The swarm will then move to the next coordinate point counted as i according to the following formulas:

$$x_{i+1} = x_i + d\cos(\theta + \beta) \qquad for\ i = 0,1,2\ldots \tag{3.5}$$

$$y_{i+1} = y_i + d\sin(\theta + \beta) \qquad for\ i = 0,1,2\ldots \tag{3.6}$$

Where β depends on the number of branches and the respective branch within the iteration's set:

$$\beta = \begin{cases} 0,\ \pm m\alpha \quad For\ m{:}\ 1 \rightarrow \left\lfloor \frac{N}{2} \right\rfloor & When\ N\ is\ odd \\ \pm m\alpha \quad For\ m{:}\ 1 \rightarrow \left\lfloor \frac{N}{2} \right\rfloor & When\ N\ is\ even \end{cases} \tag{3.7}$$

To verify the function of the tree fractal formation, the growth rule process for the 3-branch tree formation is simulated using MATLAB. The formula for calculating the total number of robots needed as well as the formula for determining their next location, is added to MATLAB to produce the tree structure in which the swarm will follow. The number of branches is 3, a fixed travelled distance is given as (d = 53 units), and the separation angle is (α = 35°). The MATLAB will use the given values in the designed formulas to display a tree structure suitable for each robot to follow. A MATLAB growth rule simulation for the first three iterations for a 3-branch tree fractal formation is shown in Figure 3.4. The development code for this section is presented in Appendix A.1.

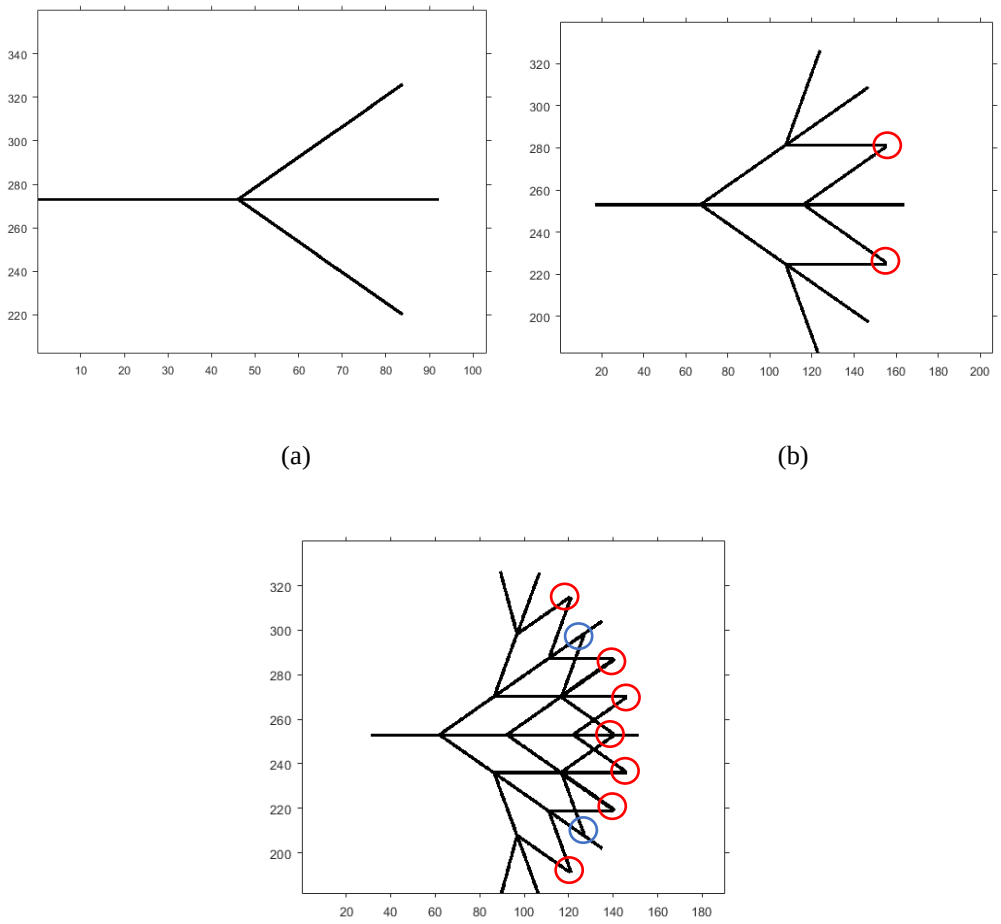


Figure 3.4 The growth rule of a 3-branch tree formation: (a) first iteration, (b) second iteration with overlaps (red circles), (c) and third iteration with high overlaps (red and blue circles).

The MATLAB simulation in Figures 3.4 (b) and (c) show a noticeable overlap between two end line segments from the second iteration and an intersection between lines from the third iteration. The overlap occurs due to the self-similarity feature, where the distribution of the line segments is symmetric, causing an issue where two robots will be meeting at the exact location. One of the two interfering line segments must be removed to overcome this issue.

To determine the amount of overlap within an iteration, an additional formula is required. Figure 3.4 (b) shows two overlapping location points in the structure of the tree formation, while it increases to six location points for the next iteration. The formula in Equation (3.8) counts the number of overlapping line segments while

preserving the overall structure of the tree formation. The overlap formula is expressed as follow:

$$Overlaps\ for\ each\ iteration = \sum_{i=2}^{n}[N-1]^{i-1} \tag{3.8}$$

Where i is the number of iterations for the tree fractal formation. The overlap formula is functional for all possible angles of separation (α) starting from the second iteration and can determine the number of overlaps occurring for each iteration. The total overlaps are then subtracted from the total number of robots needed is shown in Equation (3.9).

$$Total\ NoR_{i^{th}} = \sum_{p=1}^{n} N^{p} - Overlaps\ for\ each\ iteration \tag{3.9}$$

To demonstrate that equation (3.8) is adequate to use by a robotic swarm, an integrated robotic demonstration tool named V-REP is used. The robotic demonstration tool is linked to MATLAB as a bi-directional communication. MATLAB sends the number of robots needed and their assigned location for one iteration. The V-REP tool receives these values and sets a unique number for each robot in numerical order. Each robot will travel to the assigned location, and once a robot reaches the assigned location, the next robot will travel to the next assigned location and so on. When all robots have reached their assigned location, V-REP will send a confirmation signal to MATLAB indicating the successful operation of distributing the robots, and the robots will receive their assigned location in the next iteration by MATLAB.

A simple area design in V-REP tool is a flat surface of square blocks with 11 blocks long, and 11 blocks width, and each block has an area size of $1m^2$. The area contains walls at the edges of the area, acting as the boundary for the robots to explore and one entrance as an origin point. A group of quadcopter robots were used to implement a fractal formation, which contains a built-in camera pointing to the ground that counts the number of blocks passed while the copter moves. Figures 3.5, 3.6, and 3.7 show a demonstration process for quadcopter robots travelling to their assigned location according to the values received by MATLAB. The development code for this section is presented in Appendix A.1.

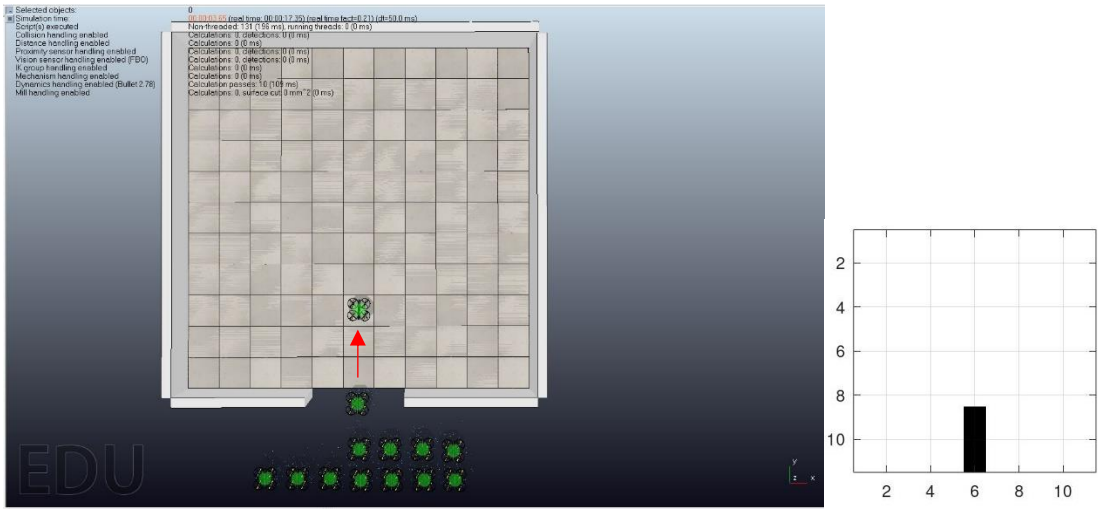


Figure 3.5 A swarm of quadcopters mimics a 3-branch tree fractal formation. The left-hand image shows the first robot moving to the assigned location. The right-hand side image shows the follow-up progress of each robot sent by V-REP to MATLAB.

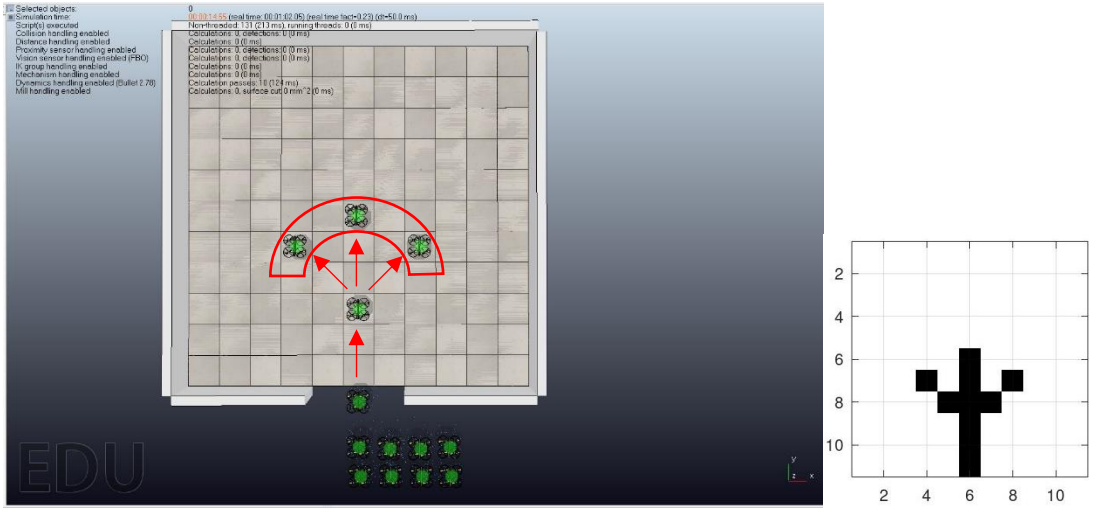


Figure 3.6 A complete first iteration of a 3-branch tree fractal formation with a separation angle of 45° made by a swarm of 3 quadcopters (circled in red) shown in the left-hand side image. Alongside the follow-up progress of each robot sent by V-REP to MATLAB shown in the right-hand side image.

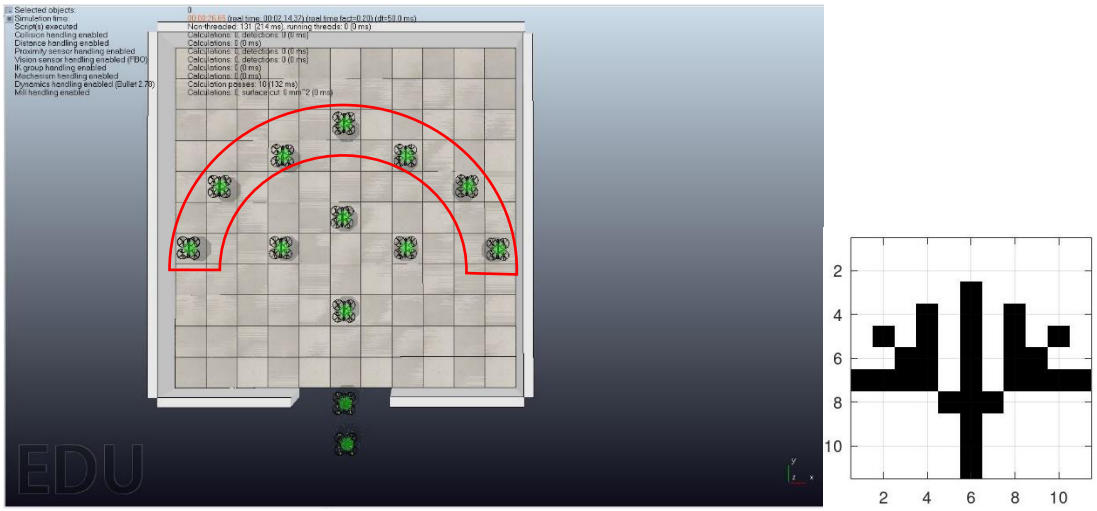


Figure 3.7 A complete second iteration of the 3-branch tree fractal formation with a separation angle of 45° made by a swarm of 7 quadcopters (circled in red) is shown in the right-hand side image. Alongside the follow-up progress of each robot sent by V-REP to MATLAB shown on the right-hand side image.

To validate the use of line-based fractal formations in exploring an unknown area, each fractal is implemented by a robotic swarm to explore a real unknown area of the Tabon cave (Choa *et al.*, 2016). The structure of the cave discovered so far was taken and used to distribute a robotic swarm using both 3-branch tree fractal formation and Vicsek fractal formation. MATLAB is used to simulate the distribution process and present the amount of area covered compared to a traditional exploration method named the Probabilistic Roadmap (PRM). As shown in Figure 3.8 (a), each robot has a diameter size of *D*, can move up to a distance of *d*, and has a distance measurement sensor covering a range of 180° with a length of *S*.

While each robot is scanning the unknown area, an overlap could occur due to the repetition of scanning part of the area by neighbouring robots when developing particularly the tree fractal formation. Figure 3.8 (b) shows that the repetition of scanning part of an area. Overall, there are 4 areas, of which 2 are identical in shape (named OL_{12} and OL_{13}) both found at the bottom and then there is the half circle that forms the front of the scanning of 1 and overlaps with the new branches (named OL_{123}) and the overlap between the 2 new branches (named OL_{23}). While the latter 2 areas are a half circle and square respectively, the former (OL_{12} & OL_{13}) are a set of triangles. The overlaps can be calculated as follow:

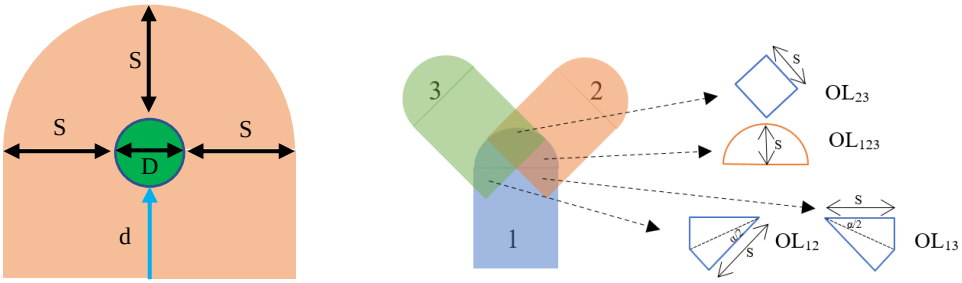
$$OL_{12} = OL_{13} = 2 \times \left[\frac{S^2 \times \sin\frac{\alpha}{2}}{2}\right] \quad (3.10)$$

$$OL_{123} = \frac{\pi S^2}{2} \quad (3.11)$$

$$OL_{23} = S^2 \quad (3.12)$$

Resulting in a total overlap of:

$$Total\ Overlap = OL_{12} + OL_{13} + OL_{23} + OL_{123} \quad (3.13)$$



(a) (b)

Figure 3.8 (a) represents a robot (green circled size) moving with a distance of *d* and scanning the unknown area using a distance sensor with a maximum distance of *S* (b) An overlap occurred when neighbouring robots rescanned part of an area scanned by another robot, resulting in four shapes: one square shape (named OL_{23}), one half circle shape (named OL_{123}), and a set of triangle shapes (named OL_{12} and OL_{13}).

By calculating the overlap that occurred due to the rescanned area, it is possible to obtain the exact amount of area covered from MATLAB by subtracting the overlapped area from the total amount of area covered by the robotic swarm. Figure 3.9 (a) shows the area to be explored and the line where the robots can be distributed, and Figure 3.9 (b) shows the swarm distribution of 100 robots using PRM, while Figure 3.10 shows the swarm distribution of 4 robots using 2-branch tree fractal formation. The formations in Figures 3.9 (b) and 3.10 initiated from the red entrance line, and the total amount of area obtained from these formations had their overlaps excluded. The tree fractal formation terminates it iteration process once a branch faces an obstacle, and this condition applies for the rest of the fractals in this chapter.

Figure 3.9 (a) The assigned area to be explored and the red line is where the robots can be distributed (b) The distribution of a robotic swarm using PRM in an unknown area.

Figure 3.10 The distribution of a robotic swarm using a 2-branch tree formation in an unknown area.

The simulation result in Figure 3.9 (b) shows the distribution process of 100 robots using PRM with an average covered area of 283.095m^2 across 30 simulations. While Figure 3.10 shows the distribution process of distributing 4 robots using 2-branch tree formation with a branch distance of 3m, a scanning radius of 2m, and a total covered area of 143.414m^2. It is noted that PRM covered a larger area size compared to the 2-branch tree fractal formation, however, the tree formation uses only 4 robots to cover about half the area size compared to the use of 100 robots by PRM. Also, the swarm's distribution of PRM forms a stochastic formation, which results in massive overlaps that are difficult to remove. On the other hand, the swarm's distribution of the tree

fractal formation is organised, allowing for a minimum overlap that is manageable to remove. The organised fractal formation led to the swarm being distributed inside the unknown area, while using PRM led some robots to explore outside the unknown area.

3.2.2 Vicsek Fractal Formation

Like the N-branch tree fractal formation, the structure of a Vicsek fractal is also based on line segments. a Vicsek fractal develops in multiple directions of a 2D plane, which could be particularly handy for situations where the robots are dropped into an unknown area. Similar to the N-branch tree formation, the Vicsek formation has a number of branches (N) for this particular fractal, although the most commonly used Vicsek has N=4, which also forms the main focus for this development. Nevertheless, all formulas have been generalized to allow for more generic use in the future.

The Vicsek fractal formation requires an initial number of robots of N, to allow development in each of the N directions and each one of these branches would then be separated by 360°/N. The next iteration then starts from the end points of the first one, developing again in N directions similar to the original development, as shown in Figure 3.11. It is assumed that all developments are oriented in the same direction as the first development, hence parallel to the X-axis. However, due to symmetry around the Y-axis when N is even, there will be overlap between a previously discovered track and a new direction, which impacts on the number of robots, and so this overlap needs to be discounted by using N-1 rather than N for each new iteration, if N is even. The growth rule of the Vicsek fractal formation is expressed as follow:

$$NoR_i = N \times \prod_{p=1}^{i}(N-1) \qquad For\ even\ N \tag{3.14}$$

$$NoR_i = N \times \prod_{p=1}^{i}(N) \qquad For\ odd\ N \tag{3.15}$$

Where (p) is an iteration counter ranges between (1 and *i*). Taking that the Vicsek formation develops in all directions, the origin point will always be the centre of the fractal, and is therefore also chosen as (x_0, y_0). The next branches develop based on the number of branches and consequently their angle of separation as well as the distance to be travelled. There are as many branches to be calculated as N. So, for each iteration, one needs to add a factor of $\mathrm{d/px}\cos(\mathrm{m}\,(360°/\mathrm{N}))$ for the x-coordinate while a sine

is used for the y-coordinate. Within this formula $d/(px)$ is the distance travelled and that normally reduces per iteration (p), while one needs to ensure that x is larger than 1. While m (360°/N) needs to go through all possible m, starting at 0 and up to N-1 to deal with the different branches. As the fractal develops one needs to add additional factors as to calculate the new location points respectively starting from the end point of the previous iteration. Hence, for every *i*, one needs to run through m=0 to N-1. Figure 3.11 shows an illustration of the swarm movements towards developing the Vicsek fractal formation.

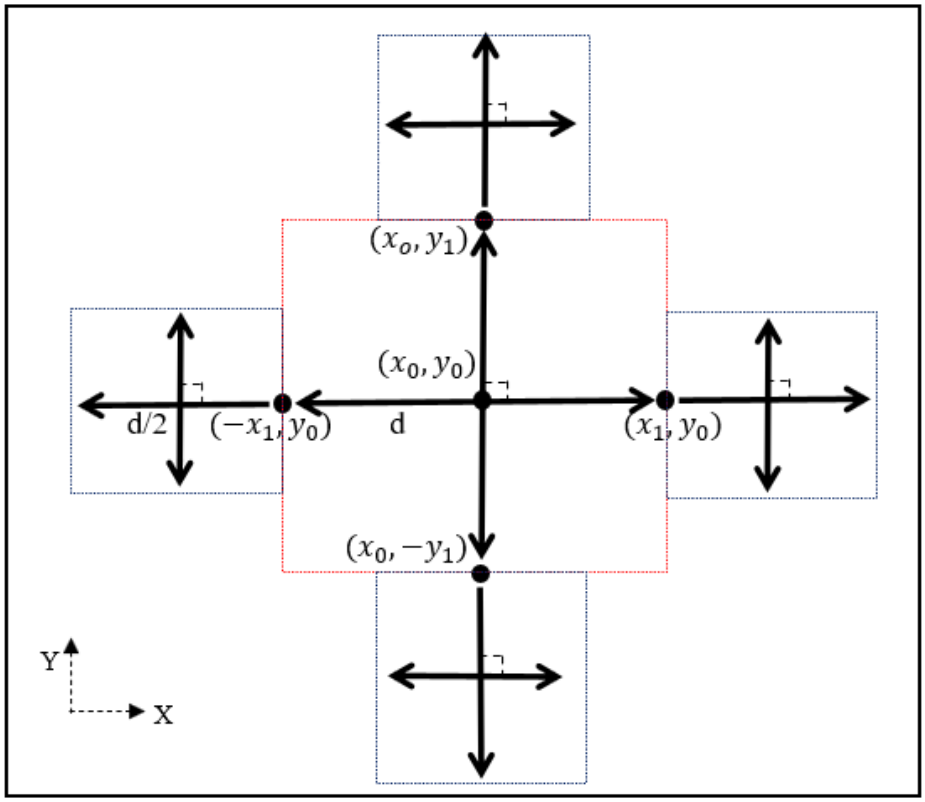


Figure 3.11 Structure of the first iteration of the Vicsek fractal formation. The initial Vicsek structure (highlighted in a red square) has four patterns (highlighted in a blue square) in each cardinal direction.

Consequently, the location point formula that can be used for any Vicsek is:

X-coordinate:

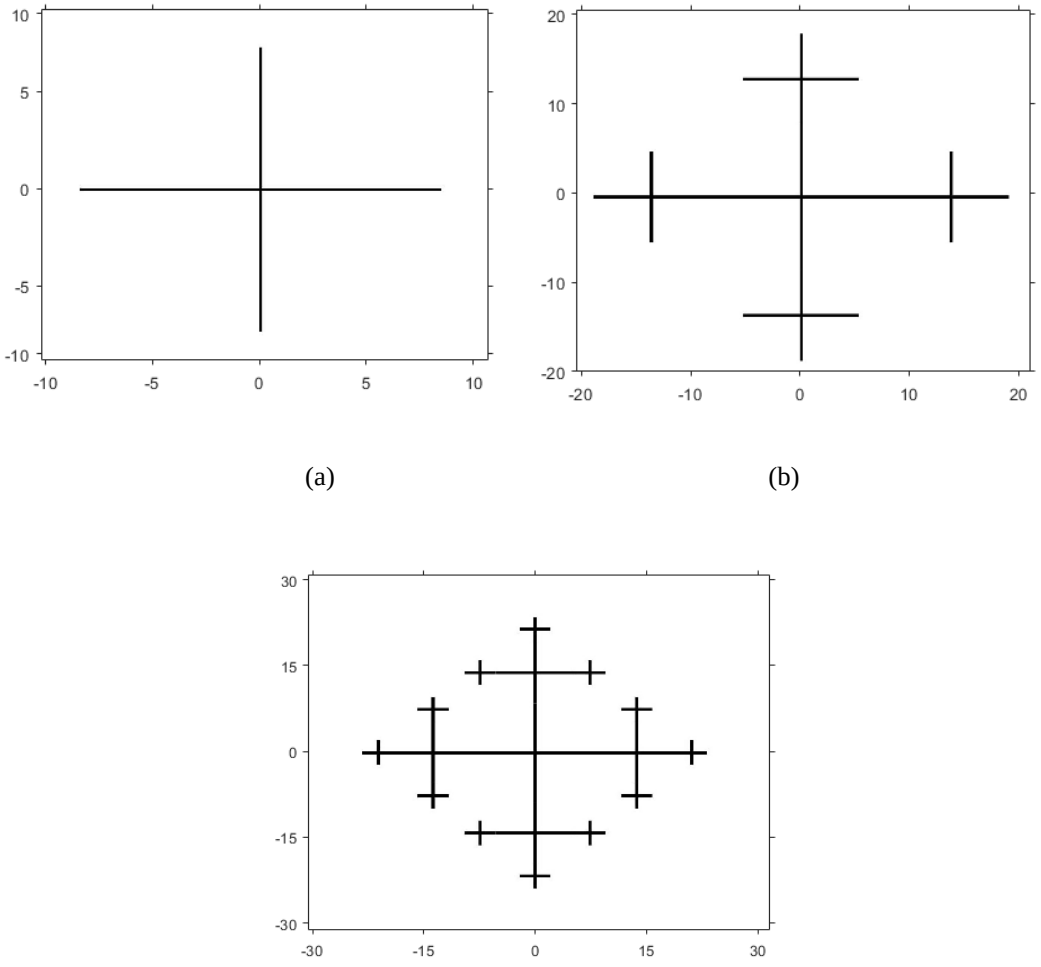
$$x_{\mathrm{i,m}} = x_0 + \sum_{p=1}^{i+1} \frac{d}{px} \cos\left(m \times \frac{360^\circ}{N}\right) \tag{3.16}$$

Y-coordinate:

$$y_{\mathrm{i,m}} = y_0 + \sum_{p=1}^{i+1} \frac{d}{px} \sin\left(m \times \frac{360^\circ}{N}\right) \tag{3.17}$$

Where: $x > 1$, and for each iteration, one needs to recursively work through all possible m, from 0 to N-1.

Verifying the functionality of the Vicsek formulas for determining the number of robots needed and their travel locations required computational software. Like the N-branch tree formation, MATLAB is used to implement the formulas of the Vicsek. For this simulation, the initial number of robots X_0 is set to four robots, the separation angle between line segments is 90°, the formation direction angle is set according to the cardinal directions into the following values: (0°, 90°, 180°, 270°), and the initial travel distance is set to 8-unit length. The images in Figure 3.12 show the simulated Vicsek formation via MATLAB for the first two iterations.



(a) (b)

(c)

Figure 3.12 The growth rule of the Vicsek fractal formation: (a) initial structure, (b) First iteration, (c) Second iteration.

To verify that a robotic swarm can mimic the Vicsek fractal formation. The V-REP tool is integrated with MATLAB. Each robot receives location points according to the Equations (3.16) and (3.17), observing the progress made by the robotic swarm and confirming the completion of developing the Vicsek fractal formation. The designed area in V-REP is the same as the one made for mimicking the N-branch tree fractal formation in MATLAB, but the origin location point is at the centre of the area. All the participating robots have the exact specifications used in developing the 3-branch tree formation adding a compass meter to support the robots in determining the current cardinal direction. Each robot moves to the first location point in the following directions: east, west, north, and south. To reach the next location point, each robot summoned two new robots to move towards the next location points, and so on. Figures 3.13, 3.14, and 3.15 shows a real-time robotic swarm mimicking the Vicsek fractal formation for two iterations.

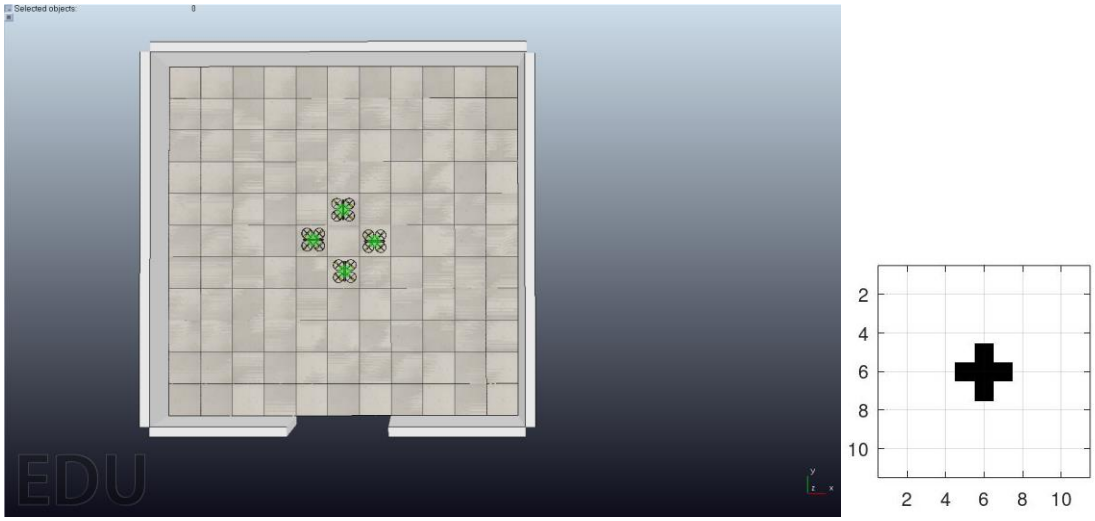


Figure 3.13 A swarm of quadcopters mimics the Vicsek fractal formation. The left-hand image shows four robots moving to their assigned location. The right-hand side image shows the follow-up progress of each robot sent by V-REP to MATLAB.

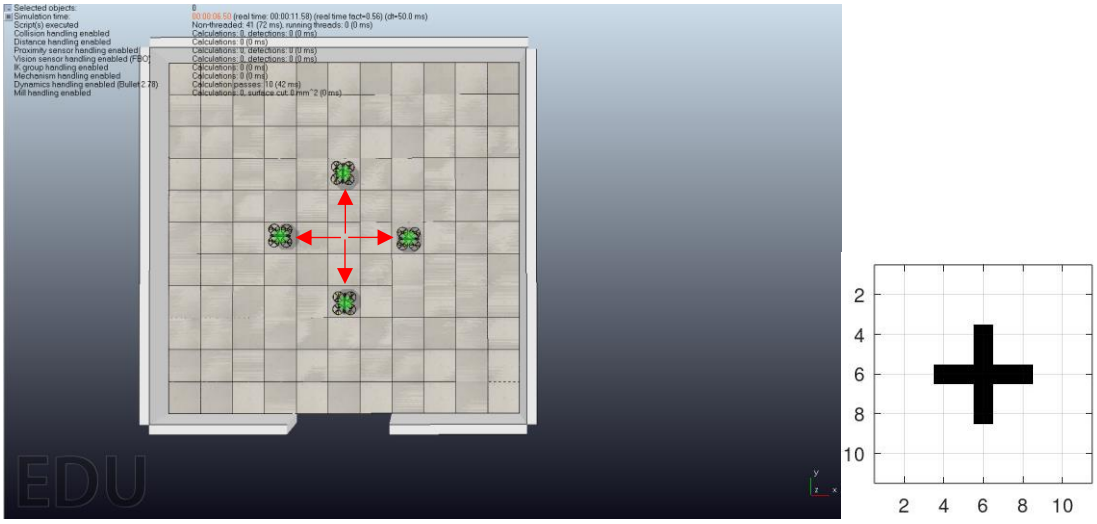


Figure 3.14 The swarm has reached their initial location points preparing to develop the first iteration of the Vicsek fractal formation.

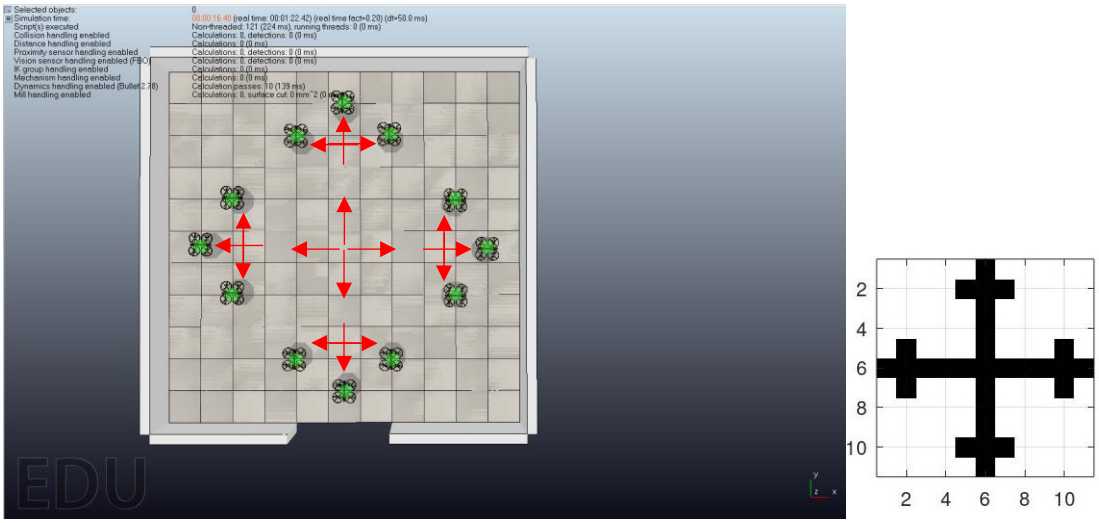


Figure 3.15 A complete first iteration of the Vicsek fractal formation

To validate the function of using the Vicsek fractal formation to cover an unknown area, the area of the Tabon cave is used as an unknown area for distributing the Vicsek formation. The formation starts from the centre of the cave's entrance and can develop until a robot sense an obstacle, as shown in Figure 3.16.

Figure 3.16 The distribution of a robotic swarm using Vicsek fractal formation on an unknown area.

The simulation in Figure 3.16 shows the distribution of 12 robots covering an area of 111.513m^2. The Vicsek formation covers about the same amount of area as the tree fractal formation but uses a higher number of robots. Also, the west part of the fractal formation attempted to cover areas outside the unknown area as the Vicsek formation distributed the robotic swarm in a cardinal direction. For the best use of the Vicsek fractal formation, the formation should be developed from the centre of the area investigated.

3.3 Curve-Based Fractals

This section presents a detailed description of Julia set and reverse Julia set as curve-based fractals. Each fractal is developed using a growth rule formula built as a polynomial equation. MATLAB is used to demonstrate the function of both Julia Set and reverse Julia. While V-REP is used to illustrate the implementation process of mimicking the reverse Julia Set formation.

3.3.1 Julia Set Fractal Formation

Unlike the line-based fractal formations, the structure of the Julia Set is based on recuring a complex number using a quadratic polynomial function. The variables of the

quadratic function are Z and C, where $(Z = a_z \pm b_z i)$ is a complex number representing a cartesian form as a location point for robots. $(C = a_c \pm b_c i)$ is a constant complex parameter that determines the specific set of the Julia fractal shape. As Julia fractal contains a set of different shapes, this research considers resembling the cyclone shape, which according to (Falconer, 1990; Heinz-Otto Peitgen, Hartmut Jürgens, 2004), represents the constant values of $[C = 0.1 + 0.6i]$.

The complex quadratic polynomial equation for the Julia Set is expressed as follow:

$$Z_{n+1} = {Z_n}^2 + C \qquad n = 0,1,2, \ldots\ldots \qquad (3.18)$$

By applying C as mentioned above, and the initial location point Z_0 as an origin point, a first location point is generated as Z_1. The next location point Z_2 can then be generated by applying the previous location point Z_1 and the same constant parameter, and so on. Considering R is a set of complex values generated by the quadratic polynomial equation where $R_i = \{Z_1, Z_2, Z_3, \ldots, Z_i\}$, a complete iteration for the Julia Set is accomplished only when the following condition $(Z_n \approx Z_{n+i})$ is satisfied. This condition means that a complete cycle of a cyclone shape is generated, and accordingly, a new cycle will be generated as a new iteration. Therefore, the number of robots needed for one iteration depends on the condition of completing one iteration. While applying the Equation (3.18) with $C = 0.1+0.6i$, it is noticeable that one iteration is accomplished when $(Z_1 \approx Z_8)$. The experiment shows that a complete iteration/cycle consists of seven points starting from Z_1 to Z_7, while Z_8 is identical to the start point of a new iteration. Figure 3.17 (a) shows one complete iteration of a cyclone shape as the location points Z_1 and Z_8 are almost the same value. Table 1 shows a list of complex values Z_n generated via MATLAB using the Julia Set Equation (3.18) for three complete iterations.

The distribution process of the swarm for the Julia set formation is different from the line-based fractal formations. Unlike the line-based formation, where all the robots start travelling from the origin point, each location point in the first iteration of the Julia Set is a start point for one robot. For example, the location point Z_1 is assigned for the first robot as a start location point, while the start location point for the second robot is Z_2, and the process goes on until the last location point Z_7 is assigned for the seventh robot.

As the swarm aims to mimic the cyclone shape, each robot travels to the nearest location point of the next iteration as its next location point. Each robot must follow the same condition $(Z_n \approx Z_{n+i})$ to ensure the correct selection of the next location point. Accordingly, the next location of the first robot is Z_8, and the next location point for the second robot is Z_9. Generally, each robot has a set of location points to follow, which can be described as follow:

$$R_i = \{Z_n, Z_{n+i}, Z_{n+2i}, \dots, Z_{n+xi}\} \qquad n = 1,2,3, \dots \qquad (3.19)$$

Where (Z_{n+xi}) is the last location point for a robot to reach. Figure 3.17 (b), (c), and (d) show a cyclone shape generated using Equation (3.18) for a period of 85 iterations, alongside with lines linking each location point to another location point using Equation (3.19). It is noticeable that the distance between two location points is named the displacement.

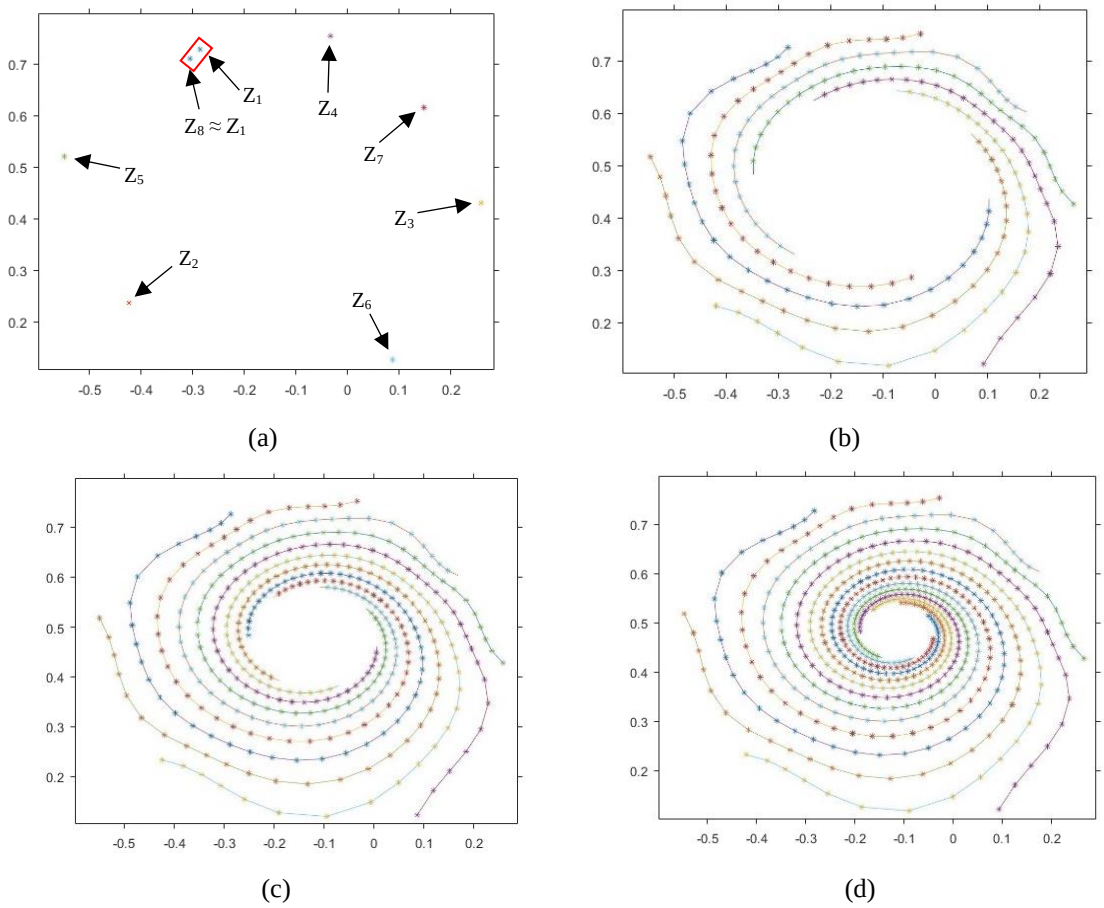


Figure 3.17: A cyclone shape generated using the Julia set formula (a) first iteration that shows the first 7 location points, and an 8th location point near to the first location point (b) A cyclone shape generated by 28 iterations of Julia Set (c) A cyclone shape generated by 57 iterations (d) A cyclone shape generated by 85 iterations.

3.3.2 Reverse Julia Set Fractal Formation

The purpose of using fractals in the context of this research as swarm formations is to explore an unknown area. A fractal formation must be expandable for further iterations so the swarm can cover the area. The cyclone shape of the Julia set formation is developing in an inward direction. According to Table 1, the location point values highlighted in the same colour are getting smaller due to the recurring process of the quadratic polynomial equation. In order for the reverse Julia set fractal to extend its formation, quadratic polynomial Equation (3.18) must be reversed, hence the name. The reverse Julia Set fractal formation can be expressed as follow:

$$Z_{n+1} = \sqrt{Z_n - C} \qquad n = 0,1,2, \ldots\ldots \qquad (3.20)$$

Where all the parameters are the same as in Equation (3.18). To validate the functionality of the reverse Julia Set equation, Table 3-1 shows two highlighted columns of generated values for both the Julia set formula (left side) and the reverse Julia set (right side) for three complete iterations. The left highlighted column shows 21 generated values (from Z_1 to Z_{21}) using Equation (3.18). The last generated value Z_{21} was used as Z_0 for the Equation (3.20) while keeping the same value of C as mentioned in the Julia Set formation section. The right highlighted column shows another 21 generated values using the Equating (3.20). It is noticeable that the values generated by both the Julia set and the reverse Julia set have resembled. The last 4 values generated by the reverse Julia Set Z_{18} to Z_{21} are however different from the first 4 values of the Julia Set Z_0 to Z_3 as the reverse Julia Set formula does not reach absolute zero during the recuring process.

Table 3-1 A list of complex numbers generated by the Julia Set formula (left column) and the reverse Julia Set formula (right column) for three iterations. The values generated by the reverse Julia Set formula (e.g. Z_1) match with the corresponding values generated by the Julia Set formula (e.g. Z_{20}).

Iterations	Julia Set $Z_{n+1} = {Z_n}^2 + C$	Iterations	Reverse Julia Set $Z_{n+1} = \sqrt{Z_n - C}$
		Initial "Z"	Z_0 = 0.086858 + 0.21822i
3rd Iteration	Z_{21} = 0.086858 + 0.21822i	1st Iteration	Z_1 = -0.42946 + 0.44449i
	Z_{20} = -0.42946 + 0.44449i		Z_2 = -0.10575 + 0.73528i
	Z_{19} = -0.10575 + 0.73528i		Z_3 = 0.14229 + 0.47539i
	Z_{18} = 0.14229 + 0.47539i		Z_4 = -0.29485 + 0.21131i
	Z_{17} = -0.29486 + 0.21131i		Z_5 = -0.28214 + 0.68881i
	Z_{16} = -0.28214 + 0.68881i		Z_6 = 0.071359 + 0.62228i
	Z_{15} = 0.071359 + 0.62228i		Z_7 = 0.061836 + 0.18018i
2nd Iteration	Z_{14} = 0.061832 + 0.18018i	2nd Iteration	Z_8 = -0.43783 + 0.47943i
	Z_{13} = -0.43783 + 0.47944i		Z_9 = -0.081695 + 0.7379i
	Z_{12} = -0.081693 + 0.7379i		Z_{10} = 0.15233 + 0.45266i
	Z_{11} = 0.15233 + 0.45265i		Z_{11} = -0.32302 + 0.22807i
	Z_{10} = -0.32302 + 0.22807i		Z_{12} = -0.26481 + 0.70225i
	Z_9 = -0.26481 + 0.70225i		Z_{13} = 0.083836 + 0.60979i
	Z_8 = 0.083837 + 0.60979i		Z_{14} = 0.03697 + 0.1324i
1st Iteration	Z_7 = 0.036964 + 0.1324i	3rd Iteration	Z_{15} = -0.4521 + 0.51714i
	Z_6 = -0.4521 + 0.51714i		Z_{16} = -0.055605 + 0.74511i
	Z_5 = -0.055602 + 0.74511i		Z_{17} = 0.16906 + 0.42917i
	Z_4 = 0.16906 + 0.42917i		Z_{18} = -0.10575 + 0.73528i
	Z_3 = -0.3559 + 0.24i		Z_{19} = -0.42946 + 0.44449i
	Z_2 = -0.25 + 0.72i		Z_{20} = 0.086858 + 0.21822i
	Z_1 = 0.1 + 0.6i		Z_{21} = 0.059925 + 0.63791i
Initial "Z"	Z_0 = 0 + 0i		

To verify that a robotic swarm can mimic the cyclone shape generated by the reverse Julia set, MATLAB was used to implement the formula of the reverse Julia set. During this simulation, the initial value (Z_0) is as origin, while the constant value (C) is set to 0.1+0.6i, allowing the reverse Julia set formula to generate a cyclone shape. The images in Figure 3.18 show the growth process of a cyclone shape that takes an outwards flow direction. Each image shows the growth of a cyclone shape from the first iteration all the way till the 30th iteration.

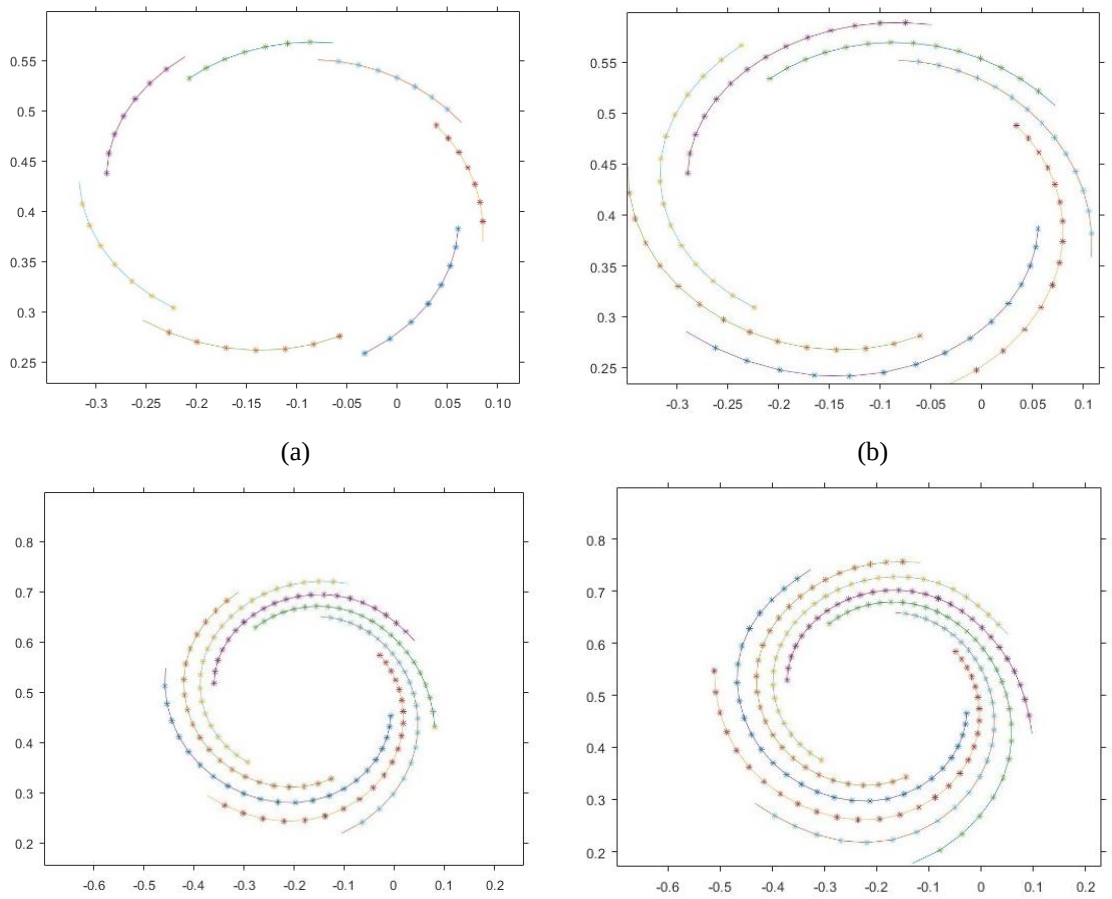


Figure 3.18 The cyclone shape generated using the reverse Julia Set formula (a) Cyclone shape after 8 iterations (b) Cyclone shape after 15 iterations (c) Cyclone shape after 20 iterations. (d) Cyclone shape after 30 iterations.

Like the line-based formation, MATLAB and V-REP were used for implementing the cyclone shape of the reverse Julia Set formation. The characteristic of the area used to implement the cyclone shape is the same as the one used in the line-based formation. Each robot will receive the location point from MATLAB using Equation (3.20). Because of the small values generated by the reverse Julia Set formula, both a_z and b_z will be scaled up as well as the area. The first component a_z is scaled 100 times, while the second component is scaled 10 times for a better distribution of the robotic swarm. Figures 3.19 and 3.20 show the process of distributing a swarm of copters inside an area where the swarm is performing a cyclone shape for 2 iterations.

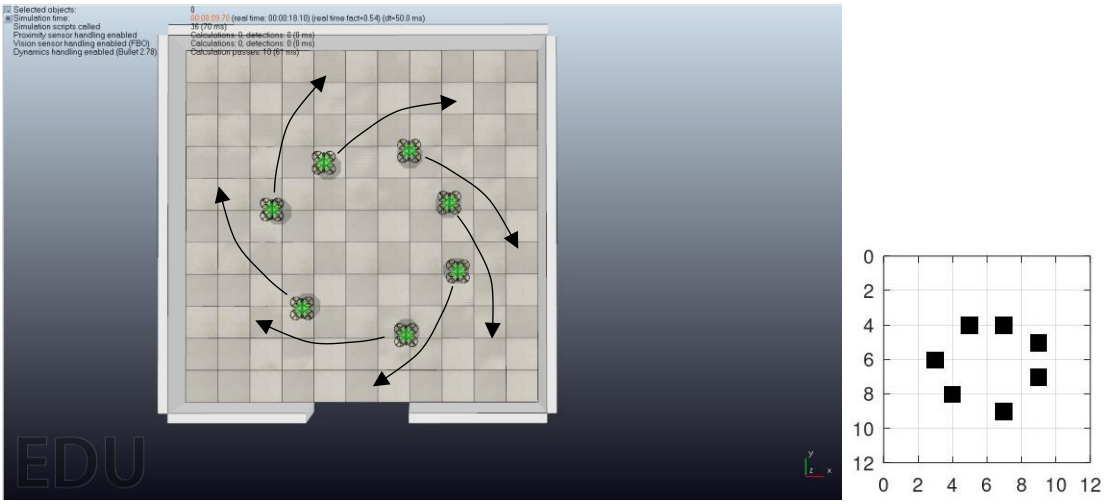


Figure 3.19 A robotic swarm mimicking the first iteration of a cyclone shape generated by the reverse Julia Set, where each robot travelled to the assigned location point, preparing to travel to the next location point.

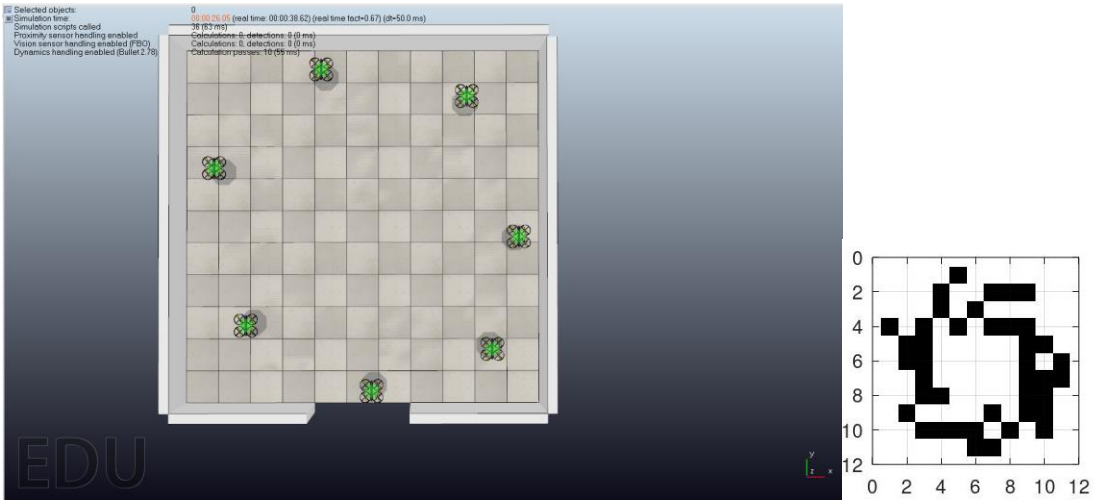


Figure 3.20 A swarm of robots completed its first iteration of a cyclone shape using the reverse Julia set formula.

To validate the function of using the reverse Julia set formation towards covering an unknown area, the reverse Julia set is used to distribute a robotic swarm on the Tabon cave as an unknown area. The reverse Julia set has a C component of 0.1+0.6i while the Z component of zero. The formation is developed until a robot faces an obstacle. Figure 3.21 presents a robotic swarm distribution that mimics the cyclone shape of the reverse Julia set.

Figure 3.21 The distribution of a robotic swarm using reverse Julia set fractal formation on an unknown area.

The simulated result in Figure 3.21 shows the distribution of 7 robots covering an area of 81.097m^2. It is noted that the formation expands at a slow rate which covers less amount of area compared to the line-based formations. However, the reverse Julia set is the only formation that uses a fixed number of robots for every incremented iteration compared to line-based formations.

To analyse the outcomes e.g. number of robots used and the amount of area covered, from using different fractal formations, Table 3-2 shows the total amount of area covered and the used number of robots using both line and curve based formations compared to PRM. It is noted that although using PRM had covered more areas than using fractal formations, the number of robots needed by each fractal formation to cover about 45% of the area covered by PRM is 88% less than the PRM. As all fractals terminates their development process when a robot faces an obstacle, all the formations could not reach the same covering value obtained by PRM. One solution to overcome this obstruction is by changing one or more parameters of a fractal formation, which is discussed in chapter 4.

Table 3-2 A list of the number of robots used and the amount of area covered using different fractal formations and PRM.

Formation type	PRM	Tree	Vicsek	R. Julia Set
No. of robots used	100	4	12	7
Covered area (total = 785.1m^2)	283.095m^2	143.414m^2	111.513m^2	81.097m^2
Average Time Cost (Seconds)	54.1	17.5	22.8	14.5
Average and Stand. Dev.	Avg.: 283.095 Std. Dev.: 8.862	Not Applicable	Not Applicable	Not Applicable
Percentage of covered area	36.1%	18.3%	14.2%	10.3%

3.4 Summary

This chapter presents a detailed description of generating four fractal formations, where each fractal is mathematically modelled and implemented by a robotic swarm in an unknown area. Each fractal formation is classified according to its development process into either a line-based formation or a curve-based formation. Each fractal class contains two fractal formations, and each fractal formation had its growth rule formula built to determine the number of robots needed and their location point within the swarm. Table 2 shows a summary list of the amount of area obtained by a robotic swarm when using both PRM and different formations of fractals.

The mathematical formula of the line-based fractals is based on the line segment, while the curve-based fractals use the quadratic polynomial function for complex numbers. Both MATLAB and V-REP are used to verify and implement the growth rule formulas of each fractal formation using a robotic swarm. The simulation results show the ability of a robotic swarm to implement a fractal as a swarm formation and use each formation for exploring and covering an unknown area with a limitation when facing an obstacle.

This chapter answers the sub-question “How can a swarm of robots apply a fractal formation to cover unknown areas?” However, while a robot faces an obstruction, a change in fractal’s parameters could help the swarm to continue exploring and covering

an unknown area. Therefore, it is important to clarify the advantages/disadvantages of using fractal formations with different parameters by a robotic swarm to cover an unknown area. The next chapter describes, with extensive detail, both the pros and cons of using fractal formations by analysing a number of parameters related to the fractal class.

Chapter 4: Studying the Parameters of Fractal Formations for Covering Unknown Areas

As fractals were used as a swarm formation to cover an unknown area, the parameters of fractals controlling the distribution of robots were fixed to a specific value, leading to a limited swarm distribution due to facing obstructions/boundaries. This chapter focuses on understanding the effect of these parameters, as these parameters linked to the growth rule formulas, on the distribution of a robotic swarm and covering an unknown area. This effect is investigated by changing each parameter to different values and observing the swarm behaviour's effect when using fractal formations towards covering an unknown area. The outcome of investigating each parameter is an analysis of the formation distribution on a pre-designed area and the amount of area covered when distributing the changed parameter of a fractal formation on an unknown area.

Therefore, this chapter presents a detailed study of the parameters affecting the robotic swarm distribution for each fractal formation class. This study leads to an extraction of the advantages and disadvantages of changing different parameters towards distributing a robotic swarm and covering an unknown area. Each parameter is simulated using a fractal formation on two different areas. The first area is a rectangular shaped area with a known dimension to observe the impact of changing each individual parameter when developing a fractal formation. The second area is a real unknown area taken from the Tabon Cave (Choa *et al.*, 2016). For both areas, fractal formations terminate their development process once a branch faces an obstacle. MATLAB is used to simulate each changing parameter when developed in both known and unknown areas.

This chapter presents four sections: the first section describes the techniques of decomposing a complex area into simple shapes used as a basis for all the case studies. The second section presents a detailed description of changing parameters of the line-based fractals to different values and simulated each change on a rectangle-size area and an unknown area. Part of the investigated parameters, such as obstacle existence, are included in the study. The third section presents a detailed description of changing parameters of the curve-based fractals. The simulation process used in the line-based fractals is also applied for the curve-based fractals. The fourth section presents an

optimisation process of combining all the parameters together and obtaining the best parameter values that provide the highest area coverage. The last section summarises the work achieved and describes the impact of changing the parameters of fractal formation towards covering an unknown area.

4.1 Analysis of an Unknown Area

As a robotic swarm uses a fractal formation to explore an unknown area, the swarm needs to identify the structure of the area being explored to make the necessary adjustment to one or more parameters and continue exploring the unknown area. While it is impossible to identify the size of an unknown area without exploring this area, a robotic swarm could partially estimate the shape of an unknown area by discovering the elements that construct this unknown shape. According to (Hoffman, Lomonosov and Sitharam, 2001; Jermann et al., 2006), a shape is constructed from joining line segments to create a perimeter, and an area, exploring the area inside the shape requires an entrance, and the area may contain obstacles. For the research purpose, the elements affecting the construction of a shape are: boundary, entrance, and obstacle. Defining each element helps the robotic swarm recognise these elements and determine the suitable change of a fractal's parameter to explore the unknown area. Each element is defined as follow:

Boundary: "*A barrier that defines the area of a shape, which could limit the swarm from developing further formation iterations*".

Entrance: "*An access point located at the boundary of an unknown area allowing a robot to enter and explore*".

Obstacle: "*A solid object preventing a robot from moving forward*".

As the area to be discovered is unknown, it is a challenging task for the robotic swarm to identify the total size of the shape being discovered. Having a discovered area helps the swarm match the discovered area to the nearest geometric shape, and therefore, has an estimated size of the discovered area. Consequently, the robotic swarm will be able to adjust parameters related to the selected fractal formation to increase the coverage of the unknown area.

In real life, an unknown area is seen as a complex shape that can be divided into a finite number of geometric shapes. One approach to analysing a complex shape is using geometric constraint solving, an approach where a complex shape is assembled into simple shapes using constraint solver algorithms such as graph reduction and recognition using region division (Brüderlin, 1998; Gu *et al.*, 2009). For an unknown area, the geometric constraint solving method can facilitate the robot's task to identify a discovered object by dividing an area into regions, as shown in Figure 4.1, where a division region approach is used to divide a swan into regions of head, body, and tail.

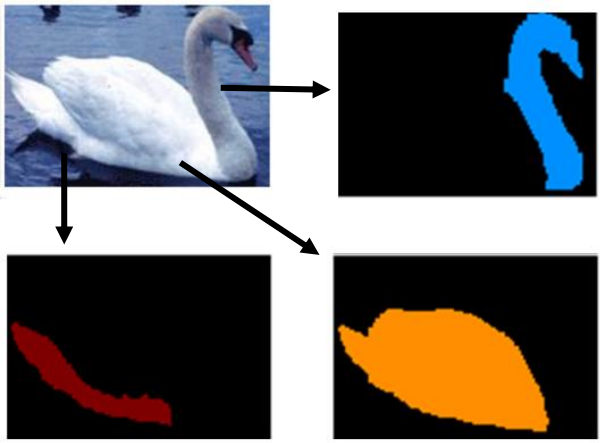


Figure 4.1 A swan image is identified using the recognition of regions (head, body, and tail) (Gu et al., 2009). It is noted that the approach did not efficiently separate the body and the tail (shown in the right-downside image) due to the similarity in colouring - Used with permission.

Using the division region method by a robotic swarm to explore complex areas can help the swarm assemble a complex shape into several geometric shapes using a suitable fractal formation. However, this method requires the swarm to have a pre-made map of the area to be analysed by a constrain solver algorithm. Therefore, the division region method is unsuitable for dividing an unknown area unless the perimeter is determined.

A similar approach for analysing a complex shape is decomposing a complex shape into simple geometric shapes using a decomposition method such as a tree decomposition and a division analysis (Hidalgo and Joan-Arinyo, 2015; Kapoutsis, Chatzichristofis and Kosmatopoulos, 2017). This approach decomposes a complex shape into more minor forms of a basic shape, as shown in Figure 4.2. This approach is famous for the swarm robotics field when analysing and recognising areas (Jermann *et al.*, 2006). However, for this approach to be practical, the swarm must identify the

boundaries of the complex area using the traditional exploration methods described in Chapter 2. In addition, having an effective swarm exploration relies on understanding each exploration method's features so the swarm can choose the most effective method for efficient exploration.

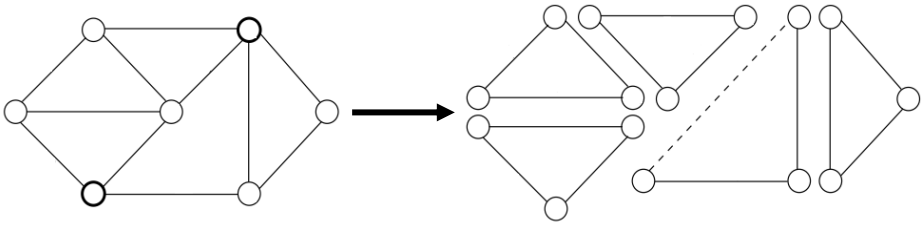


Figure 4.2 Decomposing a complex shape (left side) into a number of triangle shapes (right side) using the division analysis method (Jermann et al., 2006) - Used with permission.

The above approaches show the importance of understanding the features of each fractal formation for better covering an unknown area. Realising the effect of changing a parameter on a fractal formation when covering an unknown area requires examining these changes inside a rectangular shape with the criteria mentioned at the beginning of the chapter. As shown in Figure 4.3, a simple rectangular shape with a width of 2 m and a length of 4 m is used as a geometric area for each case study. The shape size was selected to fit the developed fractal formation.

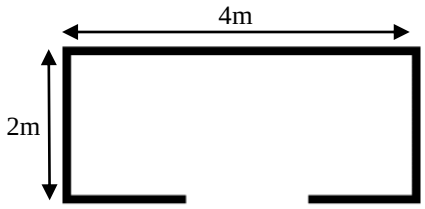


Figure 4.3 The proposed geometric shape used as an area to be explored.

4.2 Line-Based Fractal Formation

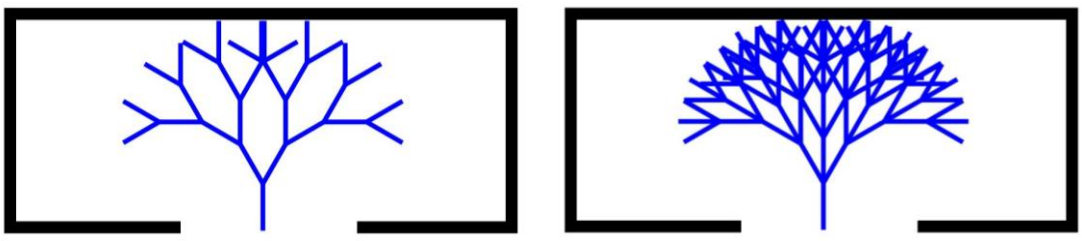
For the line-based formation, there are four algorithmic parameters related to the change of the line fractals, and they are: number of branches (N), branch length (d), initial formation direction (θ), and the separation angle between branches (α), while there are three non-algorithmic parameters for the rectangular shape, namely: multiple entrances, obstacle existence, and non-linear areas.

4.2.1 Case Study 1: Number of Branches (N)

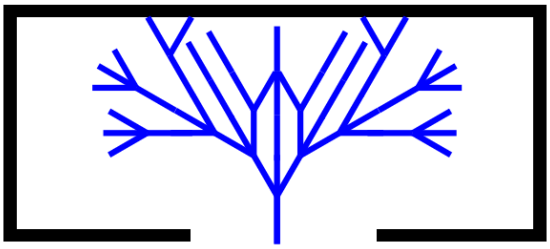
Increasing or decreasing the number of branches can impact the amount of area to be covered. Theoretically, the more branches the formation has, the higher the chance of exploring more areas as each branch is developed in a different direction from neighbouring branches. Demonstrating the effect of increasing the number of branches towards covering an area requires selecting a fractal formation of which 'N' can be adjusted. For this study, the tree fractal formation is used with a minimum number of two branches.

The experiment conducted for this case study is to distribute a tree fractal formation with two branches and three branches per iteration. As the area structure used is a rectangular shape, shown in Figure 4.3, this experiment's assumption is as follows: the area has only one entrance and no obstacles. Other parameters, including branch length, initial formation direction, and the separation angle, are fixed to a constant value. As for the tree formation, the angle separating the branches is fixed to ($\alpha = 30°$), the tree fractal formation can develop up to four iterations, the length of each branch ($d = 0.5m$), and the start iteration direction ($\theta = 90°$) perpendicular to the angle of the entrance.

The simulated images in Figure 4.4 shows both two and three branches of tree fractal formation, including the removal of the overlapping branches, distributed inside a rectangular shape. Each image is a simulation result of the tree formation behaviour using MATLAB.



(a) (b)



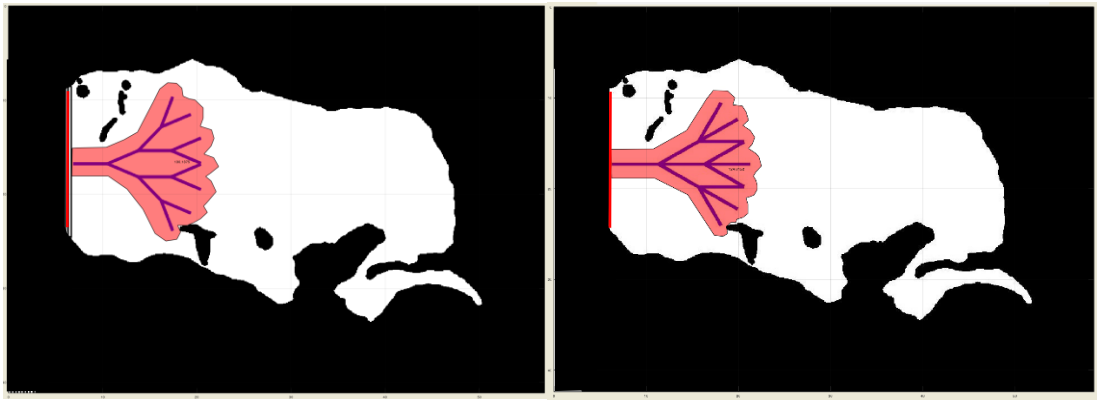
(c)

Figure 4.4 The distribution behaviour of N-branch tree fractal formation inside a rectangular shape for (a) two (b) three including overlaps (c) three excluding overlaps.

The simulation results in Figures 4.4 (a) and (b) show a noticeable increase in the area covered with an increase in the number of branches. However, both far-right and far-left sides of the rectangle area are not covered. The higher the number of branches used, the more likely overlap of the branches will occur.

Figure 4.4 (c) shows a simplified distribution of a 3-branch tree formation after removing the overlapping branches, allowing for even a higher area coverage and a smaller number of robots to use. For this experiment, it is suitable to an extend to increase the number of branches to cover more of the area's surface, but it is not clear what the impact of increasing the number of branches is when exploring the Tabon Cave area.

A robotic simulation is conducted to observe the coverage of an unknown area when changing the number of branches for a tree fractal formation, where each robot is distributed, within the swarm, using the N-tree branch formulas (3.3) and (3.4). Each robot has a sensing ability that covers a distance of 1m with a range of 180°. The robotic swarm is not aware of the total size of the Tabon Cave (Choa *et al.*, 2016). Each robot can develop a branch with a distance of 5m. The simulated images in Figure 4.5 demonstrate the amount of area covered using both 2-branches and 3-branches, respectively.



(a) (b)

Figure 4.5 The robotic swarm distribution of N-branch tree fractal formation inside the Tabon Cave for (a) two branches with three iterations (b) three branches with two iterations.

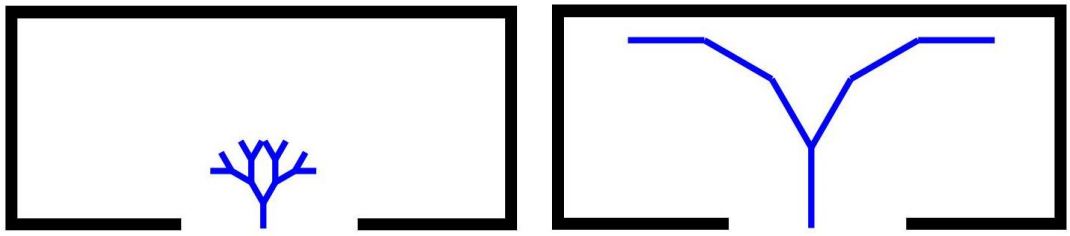
Figure 4.5 (a) shows a total distribution of 7 robots using 2-branch tree formation covering an area of 124.81m^2, while Figure 4.5 (b) shows a total distribution of the same 7 robots but using 3-branch tree formation covering an area of 121.34m^2. Although the 2-branch tree formation has a slightly higher area coverage, the formation needed 3 iterations, while the 3-branch tree formation required only 2 iterations to reach about the same area coverage. This experiment shows that, for the Tabon cave environment, it is preferable to increase the number of branches to increase the amount of area covered.

4.2.2 Case Study 2: Branch Length (d)

Increasing the length of all the branches can increase the amount of area covered in fewer iterations while decreasing the length of the branches can decrease the amount covered without affecting the number of robots used for both cases. From another perspective, each branch can have a different length, which allows a fractal formation to cover more of the unknown area in a particular direction than another direction. Two experiments are conducted to verify the effect of changing the branch length on the swarm distribution and the amount of area covered. The first experiment aims to change the length of all the branches, while the second experiment aims to change the length of each branch individually.

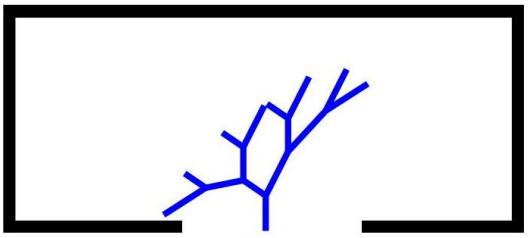
The first experiment's assumption is as follows: the exact characteristics of the area applied in the first case study also apply to this experiment. A tree fractal formation is used with the minimum number of branches N=2, the separation angle between the branches is fixed to ($\alpha = 30°$), the tree fractal formation can develop up to four iterations, the length of each branch is double the length of the first case study (d = 1m), and the start iteration direction ($\theta = 90°$) perpendicular to the angle of the entrance. The second experiment's assumption is the same as in the first experiment, but the branch length is changed as follow: the length of the left branch is 0.25m, the length of the middle branch is 0.5m, and the length of the right branch is 1m.

For all the experiments below, the tree fractal formation is constrained with a fixed number of iterations (i=3) for an equal comparison with the different changes of the branch length. The images in Figure 4.6 shows the results for developing a tree fractal formation inside a rectangular area, where the branches of the tree formation are short (d = 0.25m) as shown in Figure 4.6 (a), while the branches are long (d = 1m) as shown in figure 4.6 (b). Figures 4.6 (c) show different lengths of branches developed in order.



(a)

(b)

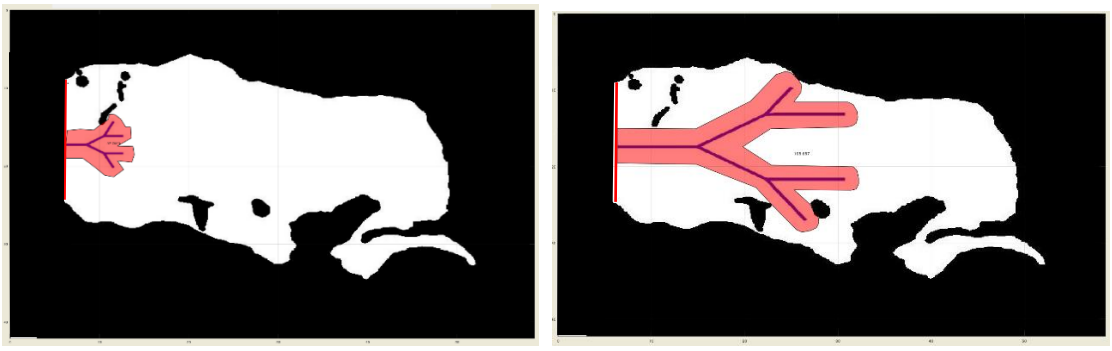


(c)

Figure 4.6 The distribution behaviour of 2-branch tree fractal formation inside a rectangular shape for (a) short length branches (b) long length branches (c) different lengths of branches.

The simulation result in Figure 4.6 (a) shows that the tree formation uses all the available robots to form 3 iterations and cover as much area as possible, while Figure 4.6 (b) shows a fewer number of developed iterations as part of the formation reached to the boundary of the area. However, because of the small branch length in Figure 4.6 (a), the tree fractal is able to expand its formation compared to the formation with long branches. As for the simulation results in Figure 4.6 (c), the formation tends to expand more at the longest branch side while narrowing at the shortest branch side, allowing the formation to expand towards the longer branches. It is noted that the increase of the branch length depends on the shape of the area, as some areas have a wide-size shape that requires the swarm to increase the branch length in a horizontal direction, and vice versa for the long shapes.

The parameter (d) is adjusted and applied in the Tabon cave by a robotic swarm to observe the effect of changing a branch length when covering an unknown area. Each robot can travel up to a distance of d = 2.5m for the short branch and a distance of d= 10m for the long branch. The characteristic of the robots used in this experiment is the same as in the first case study. The simulated images in Figure 4.7 show the amount of area covered using both short length branches and long length branches.



(a) (b)

Figure 4.7 The robotic swarm distribution of 2-branch tree fractal formation inside the Tabon Cave for (a) short length branches (b) long length branches.

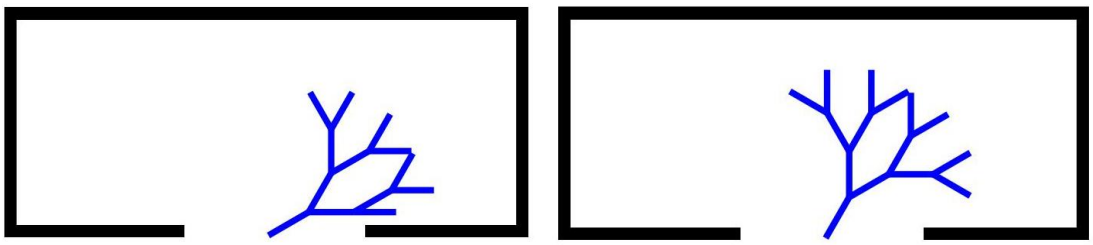
The simulated fractal formation image in Figure 4.7 (a) shows a total distribution of 4 robots covering a total area of 37.766m^2. Figure 4.7 (b) shows a total distribution of 4 robots covering an area of 195.657m^2. The results show that for the selected environment, the increase in the branch's length allows the robotic swarm to cover more

area over the use of shorter branches. However, the long-length branch formation could not develop more branches if a robot faces an obstruction, as shown in Figure 4.7 (b).

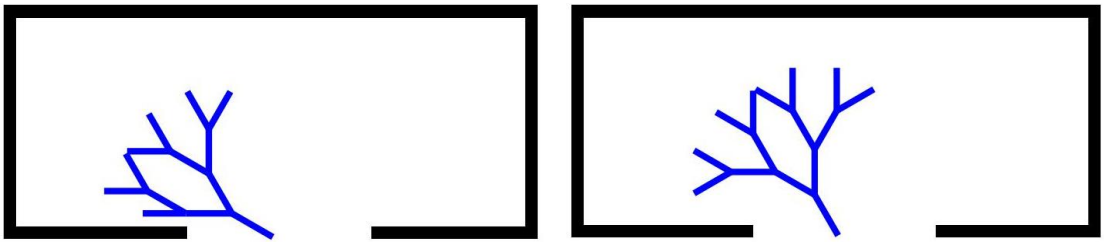
4.2.3 Case Study 3: Initial Formation Direction (θ)

The third parameter affecting the behaviour of a fractal formation is the initial formation direction. The initial formation direction (θ) sets the direction of the formation to develop. The initial formation direction can help the swarm prevent any obstruction limiting the fractal development, and cover more areas in a specific direction. In the previous case studies, the initial formation direction was set to be perpendicular to the angle of the entrance (θ = 90°). For this case study, the initial formation direction is changed to four different angles ranging from zero to 180° as follow: 30°, 60°,120° and 150°.

The experiment's assumption is as follow: the characteristics of the area applied in the previous case study apply for this case study, a tree fractal formation is used with three branches development, the angle separating the branches is fixed to (α = 30°), the tree fractal formation can develop up to four iterations, and the branch length is fixed to (d = 0.5m). The initial formation direction is set to four angles of 30°, 60°, 120°, and 150°. Each value of the initial formation direction is applied as a separate experiment. The tree formation is developed inside the rectangular shape until reaching the boundary of the area. Images in Figure 4.8 show the simulation results of developing a tree fractal formation with different initial formation directions.



(a) (b)



(c) (d)

Figure 4.8 The distribution behaviour of 2-branch tree fractal formation with different initial formation directions inside a rectangular shape for (a) 30° (b) 60° (c) 120° (d) 150°.

The simulation results show that the initial formation directions of 30° and 60° cover more areas towards the right side of the rectangular shape, while the initial formation directions of 120° and 150° tend to cover more towards the left side of the rectangular shape. |The initial formation direction helps the swarm cover more areas in a certain direction than the default initial formation direction (90°). However, the total area covered is about half of the rectangular shape, as shown in Figure 4.8 (a), where the right side of the rectangular shape is mainly covered but not for the left-side and vice versa for Figure 4.8 (d). Like the branch length case study, for this experiment, the change in the initial formation direction depends on the shape of the area, which could be a useful feature for the robotic swarm to avoid obstacles and cover a certain part of the unknown area.

Observing the initial formation direction change in an unknown area required a simulation experiment. An experiment is conducted where a robotic swarm covers the Tabon cave but with an initial formation direction of 45°. The robotic sawm develops two branches using the tree fractal formation. The maximum distance for a branch is d = 5m, and the characteristic of the robots used in this experiment is the same as in the first case study. The simulated outcome in Figure 4.9 shows the amount of area covered when changing the initial formation direction to 45°.

Figure 4.9 The robotic swarm distribution of 2-branch tree fractal formation inside the Tabon Cave with a start point in the middle of the entrance, and for an initial formation direction of 45°.

The simulated result shows the distribution of 4 robots with a covered amount of area of 74.235m^2. Like the analysis made for Figure 4.8, the initial formation direction can either aid the robotic swarm in covering more areas or prevent the swarm from exploring the unknown area. In the case of exploring the Tabon cave, one side of the tree formation faces the boundary of the cave, preventing two robots from continuing exploring the area. To overcome the obstruction issue, cooperative decision-making is made to adjust two parameters: the distance of the branch (d) and the separation angle (α). The decision was made by all the robots to prevent any overlaps between neighbouring robots, avoid the obstruction caused, and continue exploring more areas. The separation angle (α) is increased from 30° to 110°, while the distance of the branches where left variable until each robot faces another obstacle. The decision allows the obstructed robots to continue exploring the unknown area until the boundary of the unknown area obstructs the robots.

4.2.4 Case Study 4: Separation Angle between Branches (α)

The last parameter that affects the line-based fractals is the separation angle (α), where a fractal can either expand or compress its formation by changing the separation angle between its branches. The wide fractal formation allows for a wide distribution of the

swarm, which can help explore wide areas, while the narrow fractal formation allows for a narrow distribution of the swarm, which helps explore narrow areas. In this case study, there are four experiments for different separation angles. The first experiment examines a narrow separation angle of $\alpha = 10°$. The second experiment examines a regular separation angle of $\alpha = 45°$. The third experiment examines a wide separation angle of $\alpha = 90°$. The final experiment examines a wider separation angle of $\alpha = 135°$.

In this case study, a tree fractal formation is used in all the experiments, with two branches developed in each iteration. The initial formation direction is set to 90°, and the length of each branch is set to d = 0.5m. The characteristics of the area to be explored are the same as in the previous case studies. The procedure of each experiment is to use the tree fractal formation to distribute the robotic swarm but with different separation angles. Each experiment is conducted using MATLAB, and the tree formation is terminated when completing its fourth iteration. The images below in Figure 4.10 shows a demonstration of a tree formation distribution with different separation angles.

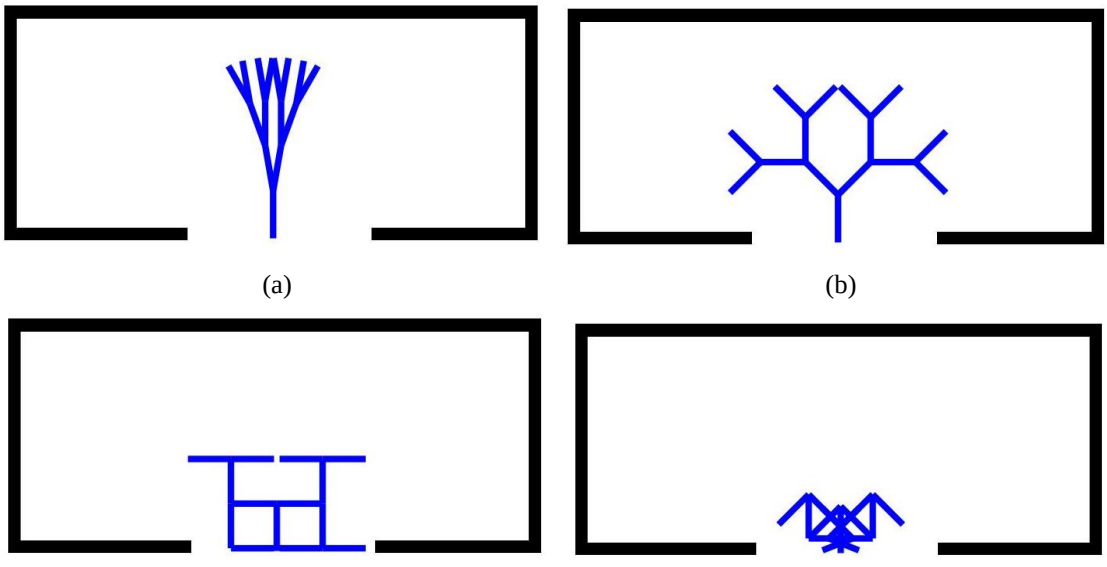


Figure 4.10 The distribution behaviour of 2-branch tree fractal formation with different separation angles of (a) 10° (b) 45° (c) 90° (d) 135°.

The simulation results show a narrowed formation distribution using a narrower separation angle, as shown in Figure 4.10 (a), while the formation distribution is wider from Figure 4.10 (b) to Figure 4.10 (d). It is noticeable that the separation angle of $\alpha = 45°$ gives the swarm the maximum distribution using the tree formation, while the separation angle of $\alpha = 135°$ reverses the formation distribution to grow inwards at one

iteration, then grows outwards in the next iteration as shown in Figure 4.10 (d). The reverse growing behaviour is caused due to the separation angle of 135° being higher than the maximum separation angle of distributing the swarm, which leads the branches to grow in the reverse direction on every iteration of the formation development. In addition, the same figure shows a high interference in branches caused by the reverse formation growth. The change in the separation angle helps the swarm to adjust the size of a fractal formation. Narrowing a fractal formation can facilitate the swarm's task to cover narrow areas, e.g. hallway, while expanding a fractal formation can aid the robotic swarm in covering wide areas.

To verify the above analysis, the 2-branch tree fractal formation is applied on the Tabon cave with a separation angle of 45°. The parameters of the tree fractal formation are fixed to the following values: the distance for each branch is set to 5m, the initial formation direction is set to zero, and the characteristic of the robots used in this experiment is the same as in the first case study. The simulated outcome in Figure 4.11 shows the amount of area covered when changing the separation angle to 45°.

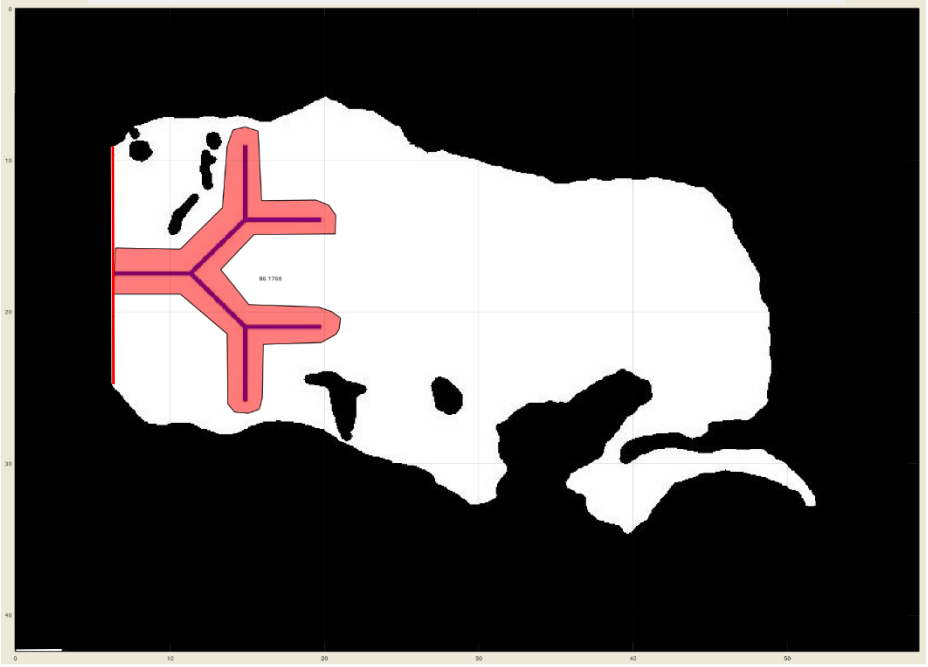


Figure 4.11 The robotic swarm distribution of 2-branch tree fractal formation inside the Tabon Cave with a separation angle of 45°.

The simulated formation shows the distribution of 4 robots covering a total amount of area of 86.177m^2. The observation made for Figure 4.10 (b) matches the distribution of

the robotic simulation in Figure 4.11 as the robotic swarm gain the highest distribution and covers as much area as possible. Compared to the robotic simulation shown in Figures 4.9, 4.7 (a), and 4.5 (a), the separation angle change allows the robotic swarm to obtain the highest amount of area covered. However, compared to the robotic simulation shown in Figures 4.7 (b) and 4.5 (b), it is noted that changing both features of branch length and the number of branches has shown a higher area coverage compared to the change of the separation angle. However, the comparison applies to the Tabon cave area and may not provide the same notice for different areas. Verifying the impact of changing the angle of separation requires an optimisation process that is made in section 4.4.

4.2.5 Case Study 5: Multiple Entrances

Like the parameters of the line-based formation, an area has parameters that could affect the formation distribution during its development process. One parameter is the existence of multiple entrances, where a swarm can use one or more entrances as a start location point to develop a fractal formation. Two experiments are conducted to observe the effect of a fractal formation when developed from multiple entrances. The first experiment applies a fractal devolvement from all the existing entrances simultaneously. The second experiment applies a fractal development only from one entrance, putting the rest of the entrances without a fractal development, and this process is repeated for each entrance individually.

All the experiments' assumptions are as follows: A 2-branch tree fractal formation is used to develop with a separation angle of 30°. The initial formation direction is set to 90°, and the length of each branch is set to d = 0.5m. The area to be explored is a rectangular shape with three entrances. One entrance is located at the width of the rectangle shape, a second entrance is located at the length of the rectangle shape, and a third entrance is located at the corner. In the first experiment, 2-branch tree fractal formation is developed in all the existing entrances at the same time, while in the second experiment, the 2-branch tree fractal formation is developed at one entrance, in which the fractal is developed at the entrance of the long side and repeating the same development process at the entrance of the width side until a branch overlaps with another branch. Figure 4.12 shows the simulation of the 2-branch tree fractal formation inside a rectangle area from different entrances.

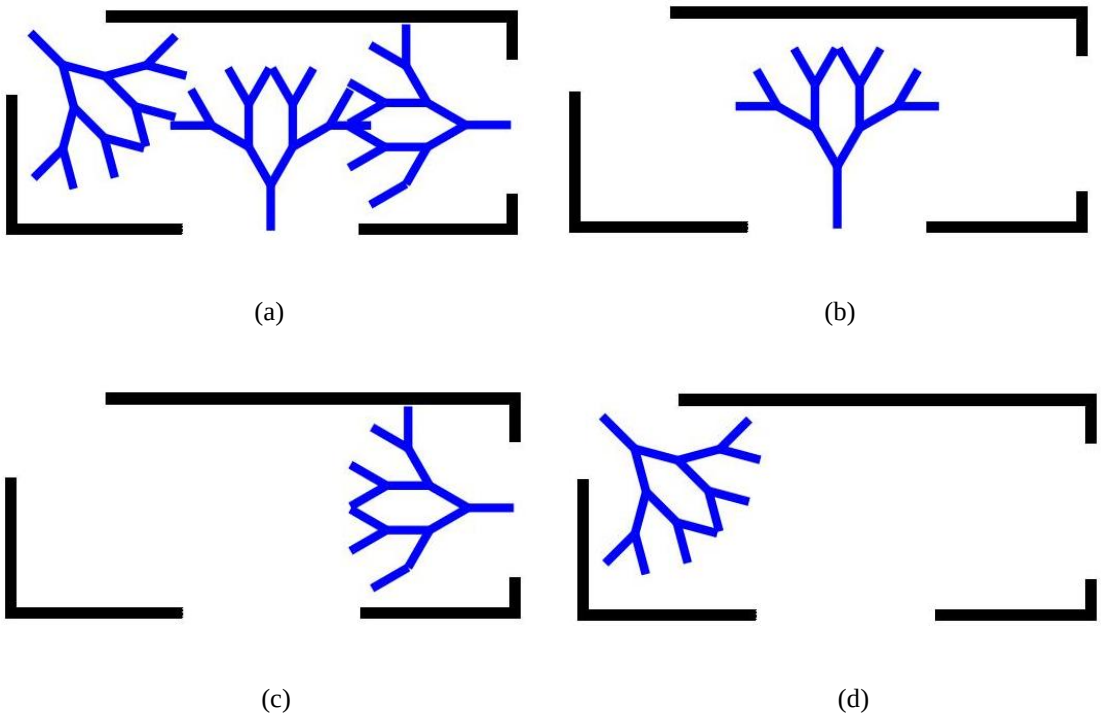


Figure 4.12 The distribution behaviour of 2-branch tree fractal formation which applies on (a) all the existing entrances (b) the entrance of the long side (c) the entrance of the width side (d) the entrance at the corner side.

The simulation result in Figure 4.12 (a) shows a higher coverage of the area compared to the Figures 4.12 (b), (c), and (d), as all the entrances were used to develop a fractal formation, leading to an overall time reduction in covering the rectangular area. However, this comes with overlaps between the distributed formations. Two approaches to address the overlap issue are to reduce the number of branches or the number of iterations used. The simulation results in Figures 4.12 (b) and (c) show that a robotic swarm used one entrance to develop a fractal formation to cover part of the area, which may not be covered when developing the formation from a different entrance. However, using one formation covered less area than using two fractals at the same time from different entrances. One approach that allows the swarm to increase their area coverage is by increasing the number of branches, as shown in Figure 4.4.

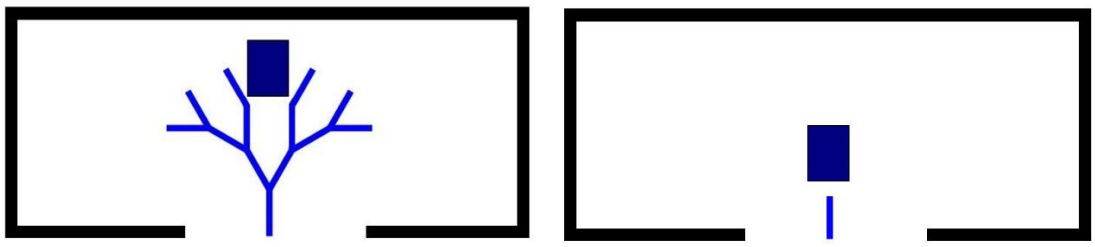
For the select rectangle shape, it is noted that the entrances located on the long side of the rectangle shape and in the corner are the best locations to explore and cover the shape. Additionally, adjusting a certain parameter, such as the initial formation direction, for a fractal formation can reduce the overlap that occurs when developing another fractal to cover an area.

4.2.6 Case Study 6: Obstacle Existence

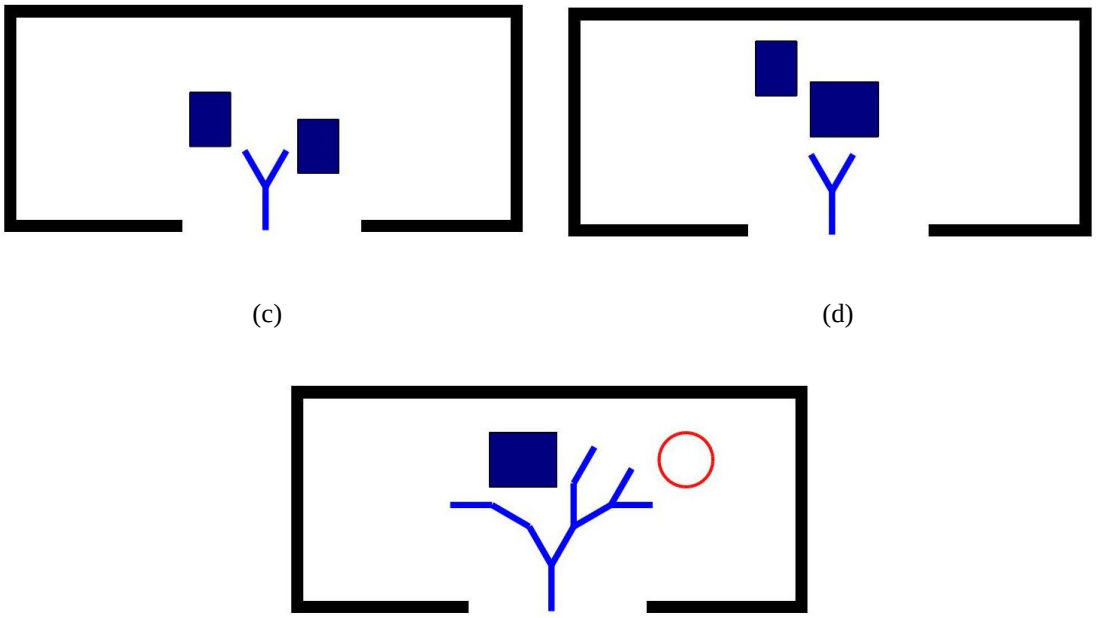
Another parameter related to the designed area is the presence of obstacles. Obstacles are considered the most significant obstruction for a fractal formation because they prevent the formation from developing, consequently affecting the progress of covering an unknown area by a robotic swarm. It is impossible to determine an obstacle inside an unknown area unless it is detected and recognised by the swarm. The parameters which describe an obstacle are size, shape, and location, and therefore, this subsection presents three experiments showing the effect of fractal development when facing an obstacle.

The first experiment shows the development of a fractal formation when facing one obstacle in different locations. The second experiment is divided into two parts, where one part shows the development of a fractal formation when facing multiple obstacles of the same shape and size but in different locations, while the second part contains multiple obstacles with the same shape but with a different size and in different locations. The third experiment shows the development of a fractal formation when facing multiple obstacles with different shapes, sizes, and locations.

All the experiments' assumptions are as follows: A 2-branch tree fractal formation is used with a separation angle of 30°. The initial formation direction is set to 90°, and the length of the fractal's branch is set to d = 0.5m. The area to be explored is rectangular with one entrance and contains one or more obstacles placed in a random location. A 2-branch tree fractal formation is developed inside the rectangle area where obstacles are randomly positioned for all the experiments. The branch which faces an obstacle terminates its development process, and the formation will terminate its development process when completing its fourth iteration. The simulation outcomes in Figure 4.13 show the effect of the tree fractal formation's development when facing obstacles.



(a) (b)



(c) (d)

(e)

Figure 4.13 The distribution behaviour of 2-branch tree fractal formation inside a rectangular shape which contains obstacles (a) far from the entrance (b) nearby the entrance (c) same size obstacles (d) different size obstacles (e) different shapes of obstacles.

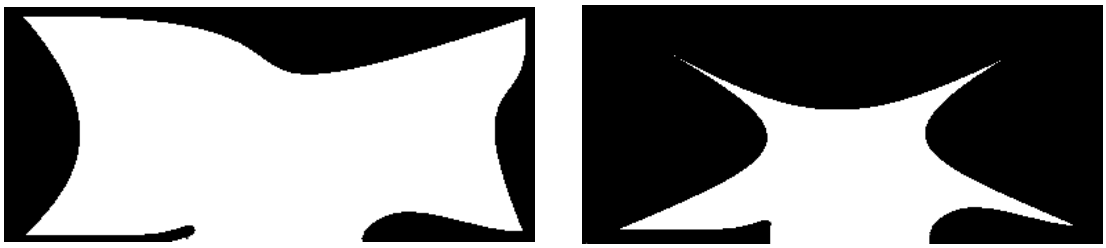
The simulation results in Figures 4.13 (a) and (b) show that the closer the obstacle to the entrance, the difficult the formation grows as an obstacle blocks some branches. One approach to resolve this issue is changing the separation angle to allow the branches to continue developing. Figures 4.13 (c) and (d) show that the higher the number of presented obstacles, the higher number of obstructed branches occur. However, the non-obstructed branches iterate around an obstacle, which compensates the obstructed branches from the inability to iterate further and helps the overall formation cover as much area as possible. Figures 4.13 (e) and (f) show that the larger the obstacle's size, the higher the number of obstructed branches occurs. Figures 4.13 (g) and (h) show that having different obstacles prevents current branches from developing further branches.

Having obstacles inside an area can have a major impact on developing a fractal formation, especially when these obstacles are nearby the start location point of the fractal formation. However, as the line-based fractal formation contains a number of flexible parameters, fractal formations can be adjusted to overcome the obstruction made by obstacles. Additionally, the compensation made by the unobstructed branches

allows the formation to surround obstacles and approximately identify their size and shape.

4.2.7 Case Study 7: Non-linear Area

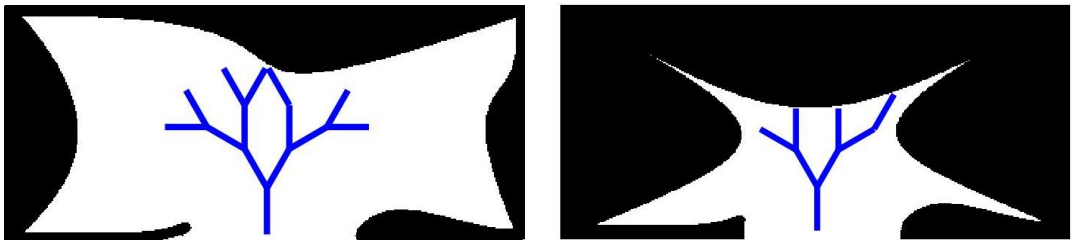
The last parameter related to the designed area is the surface of an area, where the internal dimension of the area to be discovered does not represent a basic shape, but a random shape. As the formation aims to cover an area completely, the non-linear structure of the internal area surface can be considered an obstruction, preventing a fractal formation from continuing to develop more branches. To observe the effect of a fractal formation development towards covering a non-linear area, two random shapes are presented in Figure 4.14. Each random shape is considered a separate experiment covered by a fractal formation based on the assumption made. Two experiments were conducted; the first experiment aimed to develop a selected fractal formation inside the large area of a random shape shown in Figure 4.14 (a), and the second experiment aimed to develop a selected fractal formation inside the small area in a random shape as shown in Figure 4.14 (b).



(a) (b)

Figure 4.14 Two random-dimension shapes to be covered by a line-based fractal formation (a) first random shape (b) second random shape.

The assumptions for both experiments are as follows: a 2-branch tree fractal formation is used to develop up to 4 iterations with a separation angle of 30°. The initial formation direction is set to 90°, and the length of the fractal's branch is set to d = 0.5m. The area to be explored is a random shape with one entrance, and no existing obstacles. The 2-branch tree fractal formation is developed inside each random area, and the formation will terminate its development process after four iterations. The simulation outcomes in Figure 4.15 presents the formation development on random shapes.



(a) (b)

Figure 4.15 The distribution behaviour of 2-branch tree fractal formation inside a random-dimension shape (a) first random shape (b) second random shape.

Figure 4.15 (a) shows that the tree fractal formation was partially affected for some branches, while other non-affected branches could continue developing more branches. As a result of the partial obstruction, both large and small areas were not completely covered. One approach to overcoming this issue is changing the separation angle between branches, as shown in Figure 4.15 (a). The simulation result in Figure 4.15 (b) shows a severe effect on the formation development, where all the branches are obstructed. The same approach used for the previous experiment of Figure 4.15 (a) applies to this experiment, where the formation will continue developing more branches, but at the cost of overlapping.

It is expected to have a complex random shape as an unknown area in real life, which requires a careful selection of a fractal formation and its parameter values. Therefore, the experiments for distributing a robotic swarm when changing a particular fractal parameter were applied on a real map of the Tabon cave to ensure the reliable use of different parameters towards exploring an unknown area.

4.3 Curve-Based Fractal Formation

Similar to the previous section, this section presents the effect of changing parameters but then for the curve-based formations. Each parameter is discussed and analysed as a separate case study where a brief introduction about the experiment is provided alongside the respective assumption and the experimental procedure. Simulation experiments are conducted using MATLAB, and result analysis is discussed at the end of each section. As the purpose of each experiment is to observe the expansion of the curve-based formation inside an area, all the experiments are executed using the reverse

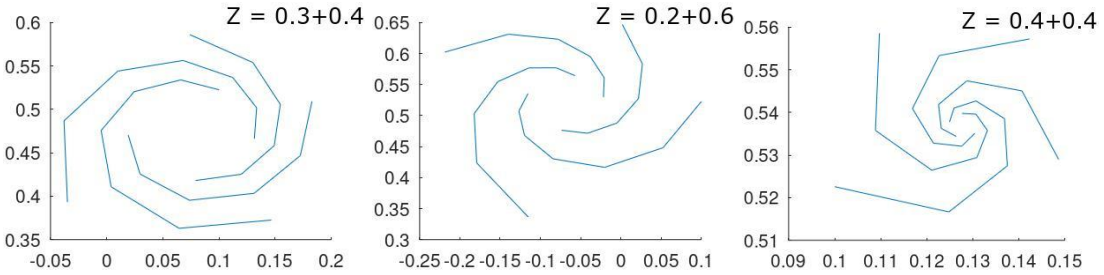
Julia set formula. The curve-based formation contains five parameters, two are related to the change of the curve fractals, and three are related to the change of the rectangular shape.

4.3.1 Case Study 1: Changing the *Z* Value

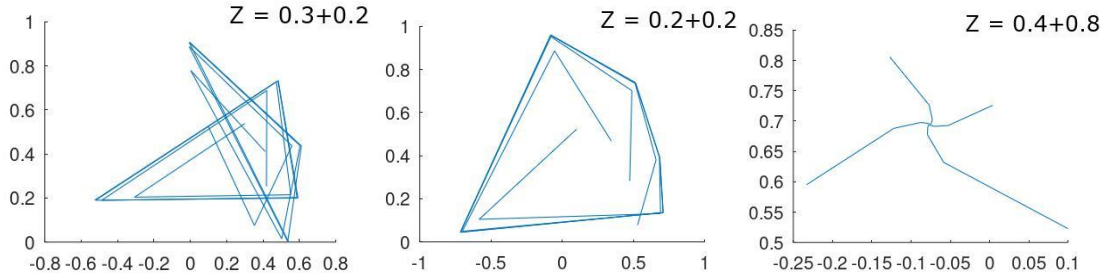
As the principle of developing curve-based fractal formations is recuring the complex number Z using a quadratic polynomial function, this case focuses on observing the effect of changing the element *Z* towards developing a curve-based fractal formation in covering an unknown area. According to Section 3.3.1 in Chapter 3, the Z value represents the location points for the robots to follow. While $Z = a_z \pm b_z i$ contains both real and imaginary parts, setting specific values for each part and recuring them, using Equation 3.22 or 3.24, produces a certain shape related to the Julia Set fractal formation.

To observe the effect of changing the *Z* value towards discovering an unknown area, the C value is set to a fixed value of (0.1+0.6i) to resemble the cyclone shape, while the Z is changeable. For this section, two experiments are conducted in which the first experiment demonstrates the change of the cyclone shape affected by the change of the parameters of *Z*, while the second experiment applies some of the results from the first experiment into an area and observes the amount of area covered for each result. The area's characteristics to be explored are the same shape applied in the line-based formation, namely a rectangular shape with one entrance.

In the first experiment, the values of the parameters a_z and b_z should have a similar structure to the cyclone shape, and therefore, are both ranged from 0.2 to 0.8. b_z is fixed to a certain value while a_z is changing its value in a step of 0.1. It is expected that some results can produce different formations that suit the swarm to mimic and apply for the second experiment, and some can produce overlapped formations that are difficult to use by the swarm. A total of 49 simulation images show the output shapes when changing the *Z* value, some of these simulated shapes are similar, and therefore, the three simulated images that resemble the cyclone shapes are presented in Figure 4.16 (a), while the three simulated shapes that show irregular shapes are presented in Figure 4.16 (b). The complete set of 49 simulation images is presented in Appendix A.2.



(a)



(b)

Figure 4.16 The distribution behaviour of the reverse Julia set fractal formation when changing the *Z* values (a) Shapes which resemble a cyclone the closest (b) Shapes which are far from resembling a cyclone shape.

The simulation results in Figure 4.16 (a) show that while changing the *Z* for both real and imaginary values can produce a different structure of cyclone shapes, it also can result in irregular shapes, as shown in Figure 4.16 (b), which is difficult to mimic by a robotic swarm due to the overlap. One can clearly observe that the flow direction of the cyclone shape can be either anti-clockwise (left side of Figure 4.16 (a)) or clockwise (right side of Figure 4.16 (a)). The simulated outcome is useful as it allows the robotic swarm to decide which cyclone shape is best to mimic based on the information obtained while exploring an unknown area. Based on the observed simulation, the selected formations in Figure 4.16 (a) are applied on a rectangle area where the core of each formation is centred at the middle of the entrance. Figure 4.17 shows the distribution of the selected formations towards covering a rectangular shape.

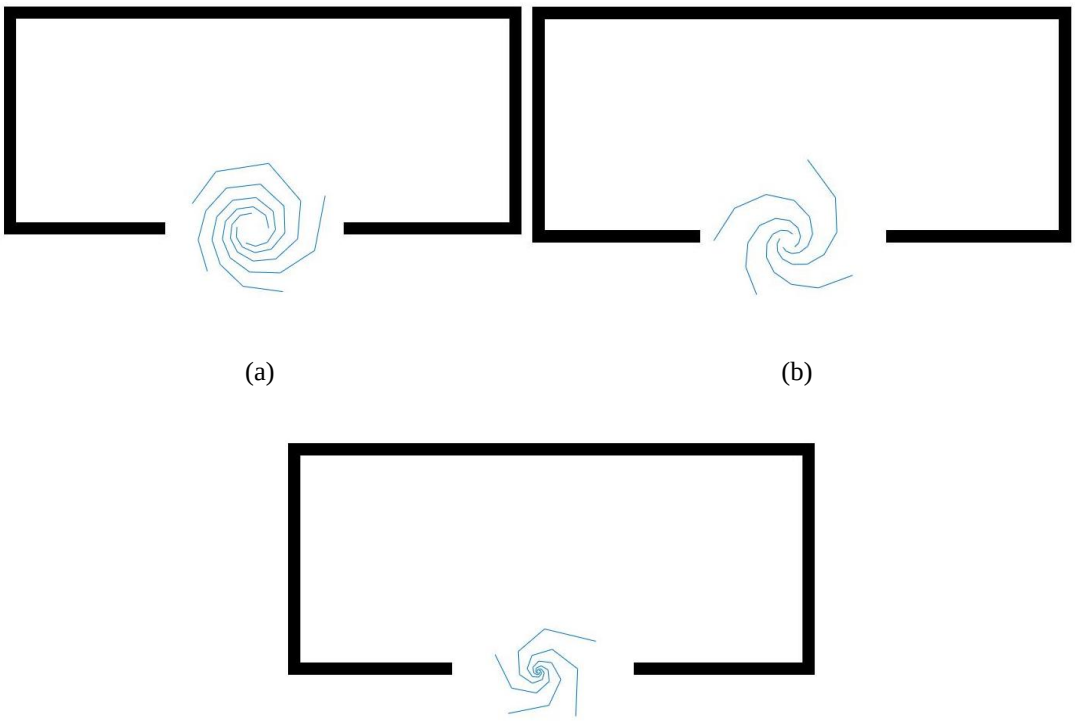
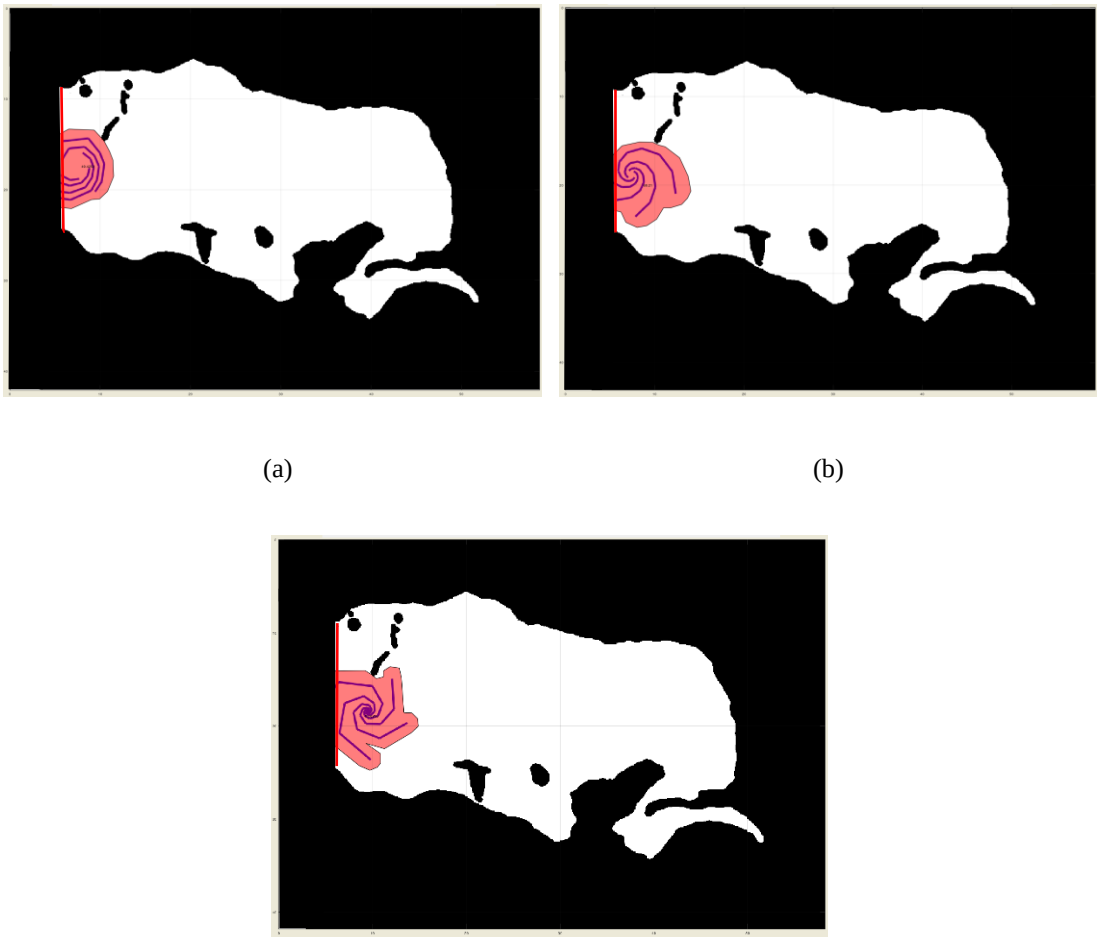


Figure 4.17 The distribution behaviour of different cyclone shapes of the reverse Julia set fractal formation inside a rectangle area (a) when $Z = 0.3+0.4i$ (b) when $Z = 0.2+0.6i$ (c) when $Z = 0.4+0.4i$.

The simulation results in Figure 4.17 shows that the separation between the cyclone branches is wide (Figure 4.17 (b)), while it is narrow for other cyclone branches (Figure 4.17 (a)). It is also notable that while all the cyclone shapes have the same iteration level, some of these shapes have covered more area than other cyclone shapes. Depending on the sensing capability of a robotic swarm, the swarm can select or adjust the separation of the current cyclone shape to allow for more expansion and more covering of an unknown area.

To realise the effect of changing the value of Z towards covering an unknown area, the cyclone shapes presented in Figure 4.17 are applied to the Tabon cave as an unknown area. Three experiments are conducted where each cyclone formation holds a unique Z value and is distributed using a robotic swarm. The robot's sensing ability as well as the speed have the same values as the in the first case study in the line-based formation. The simulated formation in Figure 4.18 shows the amount of area covered when changing the value of Z to 0.3+0.4i (a), 0.2+0.6i (b), and 0.4+0.4i (c), respectively.



(a) (b)

(c)

Figure 4.18 The robotic swarm distribution of different cyclone shapes of the reverse Julia set fractal formation inside the Tabon cave (a) when $Z = 0.3+0.4i$ (b) when $Z = 0.2+0.6i$ (c) when $Z = 0.4+0.4i$.

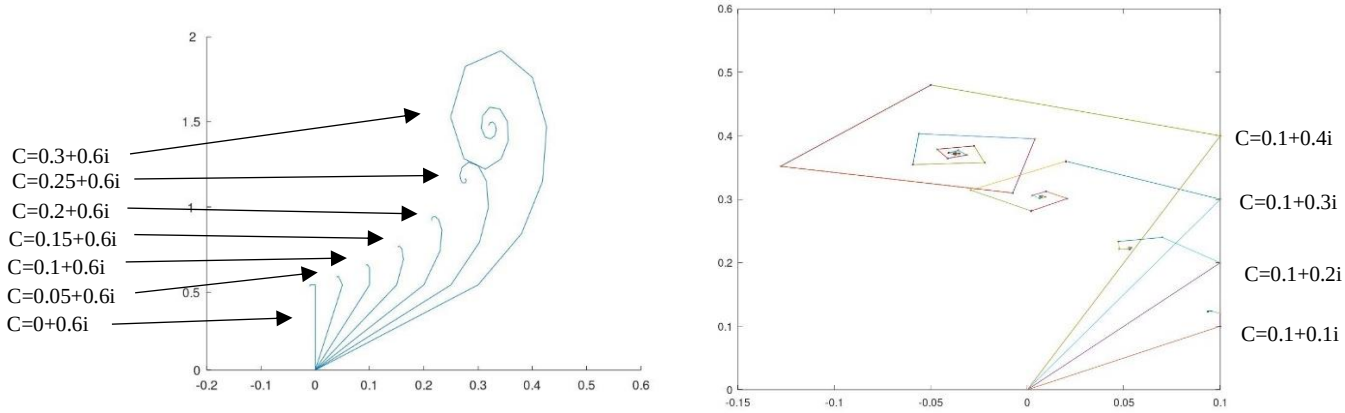
The simulated images in Figures 4.18 (a), (b), and (c) show the distribution of 4 robots covering a total area of 69.514m^2, 79.980m^2, and 74.648m^2, respectively. It is noticeable that applying different values of Z show different values of covered areas; more specifically, increasing the real part and decreasing the imaginary part improves the chances for the robotic swarm to cover more areas as Figure 4.18 (b) shows the highest value of the covered area.

4.3.2 Case Study 2: Changing the *C* Value

The second parameter which affects the behaviour of the Julia set fractal formation is the C value. Like the Z value, the C value contains both real and imaginary values $C = a_c \pm b_c i$ which can change the Julia set shape based on the change of both complex

parts. This section focuses on investigating the changing behaviour of the cyclone shape, presented the value of $C = 0.1+0.6i$ of the Julia set formation by changing the values of both complex parts.

The experiment's assumptions are as follows: the reverse Julia set formula is applied where the Z value is set to zero, while the C value is changeable. One part of the first experiment contains a fixed imaginary part $b_C = 0.6i$ while the real part is changed between zero and 0.3 with a step change of 0.05 for detailed results. The other part contains a fixed real part $a_C = 0.1$ while the imaginary part is changed between 0.1 and 0.4 with a step change of 0.1. Figure 4.19 shows the changing behaviour of the cyclone shape when changing both real and imaginary values in Figures 4.19 (a) and (b), respectively.



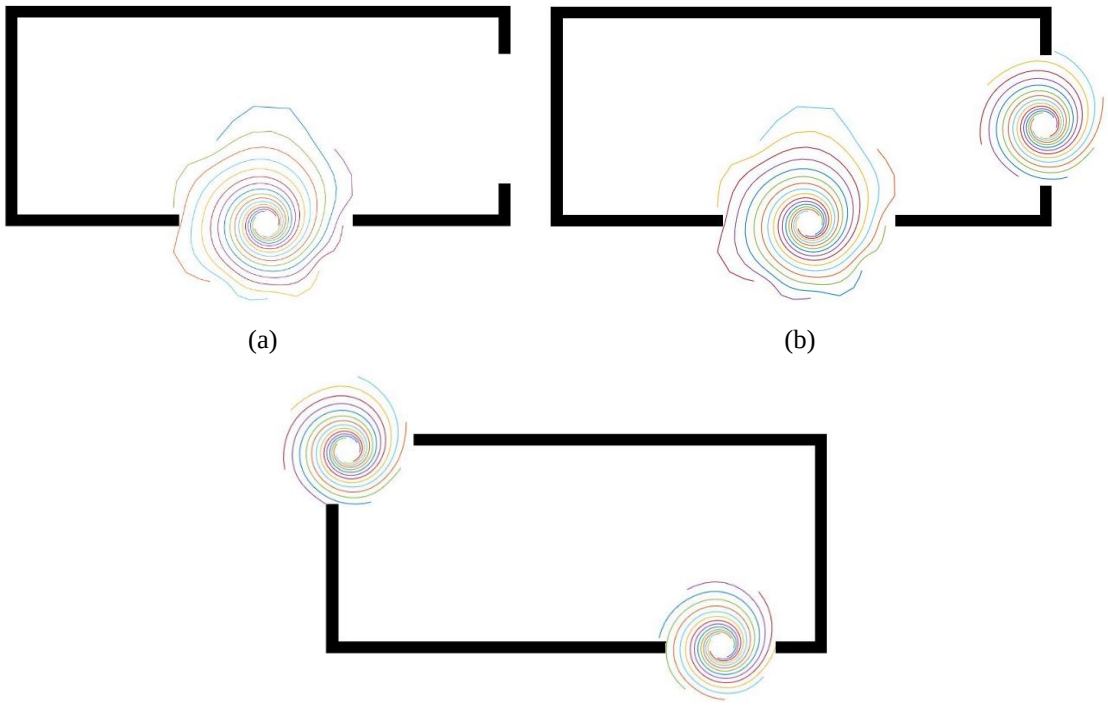
(a) (b)

Figure 4.19 The distribution behaviour of the reverse Julia set fractal formation when changing the C value (a) changing the real part of the C value (b) changing the imaginary part of the C value.

Figure 4.19 (a) shows that increasing the real part enlarge the cyclone formation, while increasing the imaginary part results in the cyclone expanding its formation vertically. Changing the parameters of the C value allows the robotic swarm to adjust its current cyclone formation to increase its size for a higher chance of covering more areas. Overall, the change in the C values shows the ability to resize the cyclone formation without the need to change the shape itself.

4.3.3 Case Study 3: Multiple Entrances

Similar to the line-based formation, one needs to consider changes to the unknown area, such as the existence of multiple entrances. Having multiple entrances allows one cyclone formation to be developed at each entrance, increasing the overall coverage of the unknown area. To observe the effect of developing a cyclone formation at each entrance, two experiments are conducted. The first experiment presents a cyclone development in one entrance, while the second one presents cyclone development in all entrances. For both experiments, two entrances exist, one located at the width of the rectangle perimeter and another entrance located at the length of the perimeter. The formula used to develop a cyclone shape is the reverse Julia Set with Z value of zero, and C value of $0.1+0.6i$. The reverse Julia Set will terminate its development at the fourth iteration or when any developed branches hit a boundary. Figure 4.20 presents the effect of developing multiple cyclone shapes on multiple entrances when covering a rectangle area.



(a) (b)

(c)

Figure 4.20 The distribution behaviour of the reverse Julia Set fractal formation inside a rectangle area (a) Using one entrance (b) Using all the available entrances (c) Using two entrances, one which located at the corner of the rectangular shape.

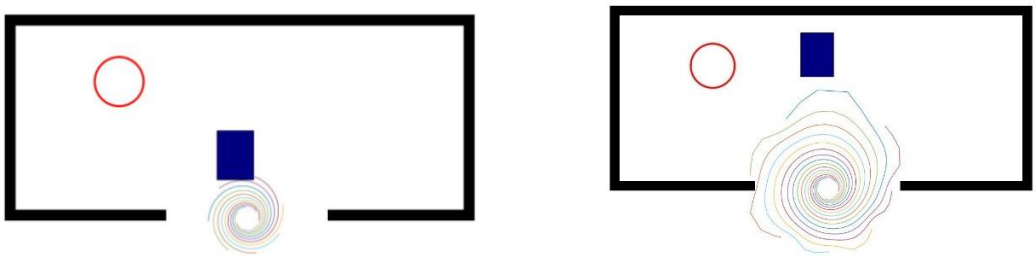
Figure 4.20 shows that the upper part of the cyclone formation covered the area inside the rectangular shape, while the lower part covered some area outside of the area of

exploration. Having multiple cyclone shape helps the robotic swarm to cover more areas, but as part of the cyclone covers an area outside the rectangle, the overall coverage is only partially efficient. A suggested solution is to redevelop different fractal shapes from the last location points of the cyclone shape, if the last location points are inside the area. Unlike the cyclone development from both the width and length, the developed cyclone located at the corner entrance covers less area as most of the cyclone shape covers the outside area.

4.3.4 Case Study 4: Obstacle Existence

Another parameter that affects the development of a curve-based formation is the presence of obstacles. Obstacles can be either major or minor obstructions on developing a curve-based formation depending on their location inside the area. To observe the effect of developing a cyclone formation when facing obstacles, two experiments are conducted. The first experiment contains two obstacles, of which one is near the entrance, and the same applies to the second experiment, but with an obstacle far from the entrance of an area. The assumption for both experiments are as follow: the characteristics of the area to be explored is a rectangular shape with one entrance and contains two obstacles with different shapes, and the formula used for developing a cyclone shape is the reverse Julia Set with a Z value of zero and a C value of $0.1+0.6i$.

The procedure of both experiments is as follows: a cyclone shape is developed at the centre of the entrance until the formation reaches either an obstacle or a boundary. Figure 4.21 shows the impact of obstacle existence on the formation development and covering an area.



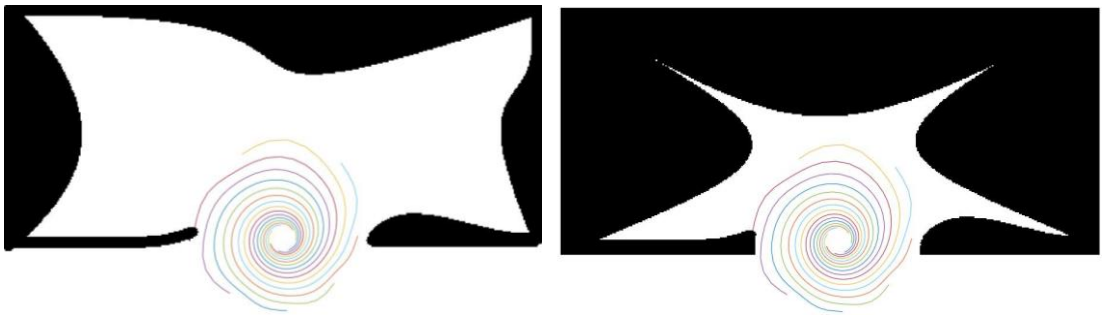
(a) (b)

Figure 4.21 The distribution behaviour of the reverse Julia set fractal formation inside a rectangle area (a) Two obstacles, one which is nearby the entrance (b) Two obstacles, one which is far from the entrance.

Figure 4.21 shows that the impact of the nearby rectangle obstacle prevents the cyclone from expanding (Figure 4.21 (a)). While the rectangle obstacle located far from the entrance allows the cyclone shape to complete its development (Figure 4.21 (b)), increasing the swarm's ability to cover more of the area. Like the line-based formation, the curve-based formation needs to adjust its formation to prevent obstacles. One approach to overcoming the obstruction issue is changing the *Z* value to convert to a suitable shape that develops a formation away from obstacles. Also, changing the current formation to line-based formation can aid the robotic swarm towards covering more parts inside an area.

4.3.5 Case Study 5: Non-linear Area

The last parameter that affects the development of a curve-based formation is the change of the internal shape of an area. Similar to the line-based formation, the same shapes shown in Figure 4.14 are considered to observe the effect of developing a cyclone shape on covering a non-linear area. Therefore, two experiments were conducted. The first experiment is meant to apply a cyclone shape to be developed inside a large-space shape, and the second experiment applies the development of a cyclone shape inside a narrow shape. The assumptions made for both experiments are as follow: The cyclone formation is generated using the reverse Julia set with *Z* value set to zero, and *C* value is set to 0.1+0.6i, which resembles the cyclone shape. The cyclone formation is to develop until it hits the surface of the non-linear area. Figure 4.22 shows the effect of developing a cyclone shape at the entrance of a non-linear area for large and narrow spaces.



(a) (b)

Figure 4.22 The distribution behaviour of the reverse Julia set fractal formation inside non-linear areas (a) large space non-linear area (b) narrow space non-linear area.

The simulation results in Figure 4.22 show that the upper part of the cyclone shape is able to cover inside the non-linear area, while the lower part covers the outside area. As the cyclone formation grows slowly, having a complex area surface with non-linear boundaries did not affect the development process of the cyclone formation. However, with the centre of the entrance being the start location for the cyclone formation, the formation did not fulfil the task of covering an unknown area. Therefore, the cyclone formation is useful when it develops inside the unknown area to ensure all the formation parts cover the unknown area.

Overall, the change in any parameter of a fractal formation can affect the distribution of a robotic swarm towards covering an unknown area. The effect can be in the amount of area covered and the number of robots needed to develop certain fractal formations. To summarise the results obtained from the case studies, the table below lists all the robotic simulation results obtained using both line and curve-based formations and compare them with a traditional exploration method called PRM. Table 4-1 shows that although most of the changed parameters did not reach the amount of area covered by PRM, all the fractal formations use 88% less number of robots to reach about 50% of the area covered by PRM.

Table 4-1 Lists all the results obtained from the robotic simulation experiments when changing each parameter and compared them to PRM.

Formation type	PRM	Tree	Tree	Tree	Tree	Tree	Tree	R. Julia Set	R. Julia Set	R. Julia Set
Adjusted Parameters	-	N = 2 d = 5m θ = 0° α = 30°	N = 3 d = 5m θ = 0° α = 30°	N = 2 d = 10m θ = 0° α = 30°	N = 2 d =2.5m θ = 0° α = 30°	N = 2 d = 5m θ = 45° α = 30°	N = 2 d = 5m θ = 0° α = 45°	Z = 0.4+0.4i C = 0.1+0.6i	Z = 0.3+0.4i C = 0.1+0.6i	Z = 0.2+0.6i C = 0.1+0.6i
No. of robots used (Max = 100)	100	7	7	4	4	4	4	4	4	4
Percentage of used robots	100%	7%	7%	4%	4%	4%	4%	4%	4%	4%
Covered area (total = 785.1m^2)	283.095m^2	124.810m^2	121.340m^2	195.657m^2	37.766m^2	74.253m^2	86.177m^2	74.648m^2	69.514m^2	79.980m^2
Percentage of covered area	36.1%	15.8%	15.5%	24.9%	4.8%	9.5%	10.9%	9.5%	8.9%	10.2%

4.4 Optimisation of Fractal Formations' Parameters

Parameters of a fractal model are individually investigated but with a set of selected values, and these values may not show the highest area coverage when combined with the rest of the parameters. The previous section did not investigate the possible combination of the parameters as it requires an optimisation process. Therefore, this section presents an optimisation for parameters of a fractal model to obtain the maximum area coverage a fractal model can reach.

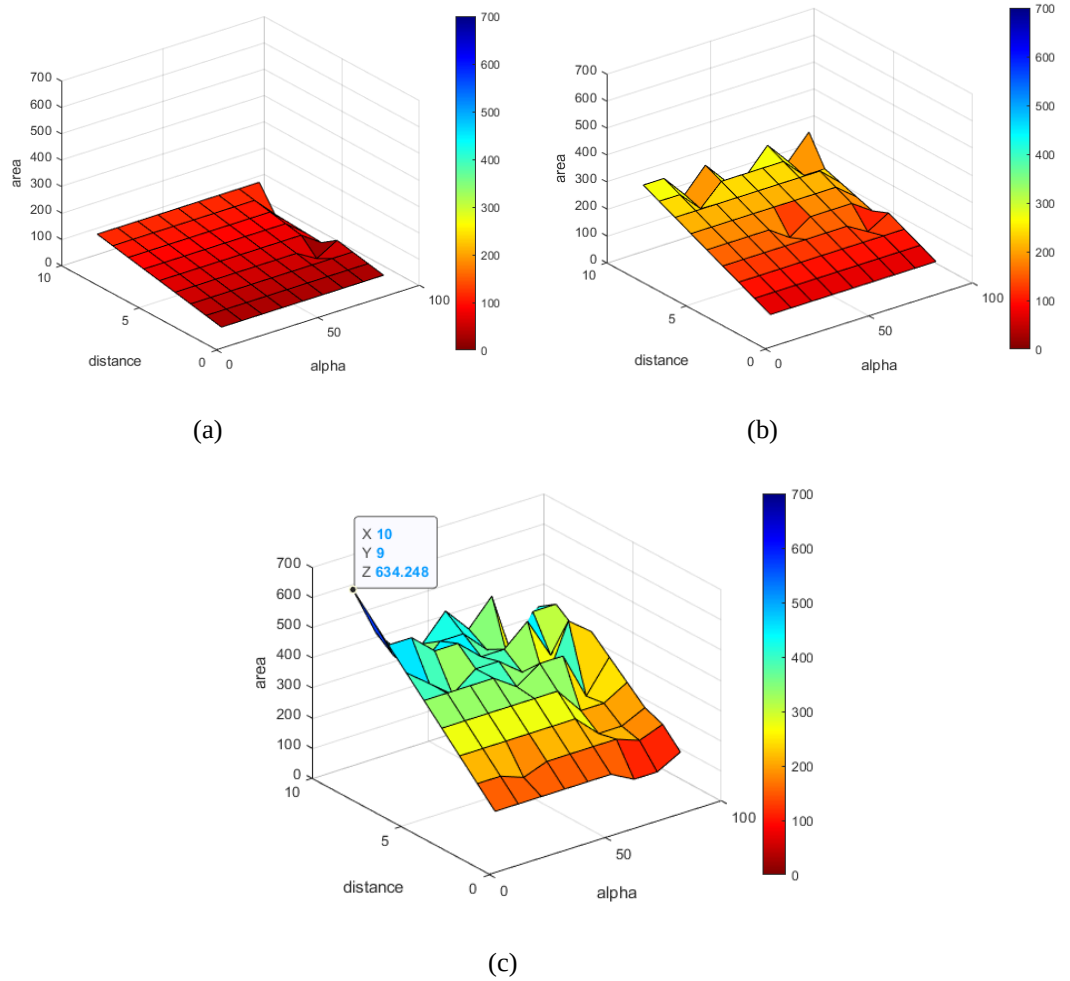
In this section, one line-based fractal model named the N-branch tree fractal formation is optimised using the gradient descent algorithm (Hooman Oroojeni, Majid Al-Rifaie and Nicolaou, 2018). Gradient descent is a first-order optimisation method that obtains the maximum cost value by adjusting the entered parameters to reach the best value compared to a default one. This section focuses on optimising linear parameters, such as in the tree fractal formation, while the non-linear parameters of the curve-based formation require further steps of selecting the cyclone shape that expands[2] and converting the complex numbers to new parameters suitable for optimisation. These further steps are out of the scope of the research, and therefore, gradient descent is a convenient method to optimise the tree fractal formation.

To simplify the optimisation process, the simplest form of tree formation is the 2-branch tree fractal model, which constrains the number of branches to N=2. Therefore, the parameters to be optimised are the separation angle (α), the branch distance (d), and the initial formation direction (θ). Each parameter is set to the maximum range to ensure that all the possible combinations are processed, and the maximum area coverage is obtained. The separation angle parameter is ranged from 10° to 90° with 10° step size, the branch distance is ranged from 1m to 9m with 1m step size, and the initial formation direction is ranged from -90° to 90° with a 20° step size. It is noted that the step-size is adjusted to allow all the parameters to have the same size-length for the optimisation to perform[3]. The selected environment to use is the Tabon Cave area mentioned in Figure

[2] For more details on the reverse Julia set's cyclone shapes, please check Appendix A.2 on page 97

[3] A simplified code illustrating the process of optimising the parameters of the N-branch Tree fractal formation is shown in Appendix A.3 on page 100.

3.9 (a). The images in Figure 4.23 show the results obtained after implementing the optimisation method for the combined parameters.



(a) (b)

(c)

Figure 4.23 High-dimensional representation of the area covered by optimising the parameters of (α, d, θ) for (a) the first iteration (b) the second iteration, and (c) the third iteration with the maximum area coverage shown on the top-left side.

For the first iteration, Figure 4.23 (a) shows a linear relationship between the parameters and the amount of area covered. However, the graph shows a degradation of the covered area when the separation angle is higher than 70°, and the branch length is higher than 5m. The degradation is presented because the formation faces obstacles when a wide separation angle is tested. For the second iteration, Figure 4.23 (b) shows a bit more degradation in area coverage compared to the first iteration when the branch length is equal to or higher than 5m. The degradation happens due to having more developed branches, allowing the formation to cover different areas and facing more obstacles. Figure 4.23 (c) shows even more degradation in area coverage for the third iteration

compared to the previous two iterations. However, the gradian descent successfully obtained maximum area coverage in the third iteration with a value of 634.248m^2.

The conducted optimisation process shows that a tree fractal formation with parameter values of α = 10°, d = 9m, and θ = 0° can cover 80.1% of the total area and 44.6% more covered area compared to PRM. It is noted that by optimising the parameters of a fractal formation, a robotic swarm can obtain the maximum possible area to cover with a minimum number of robots needed. The optimisation of fractal parameters shows the benefit of using fractals as a swarm formation by having the best possible parameter values deployed to the available number of robots and maximising the coverage of a particular area. To observe the overall iterations of the optimised tree fractal formation, Figure 4.24 represents the combination of the tree fractal parameters and their related area covered as a circle shape distributed in a horizontal sheet representing the iterations.

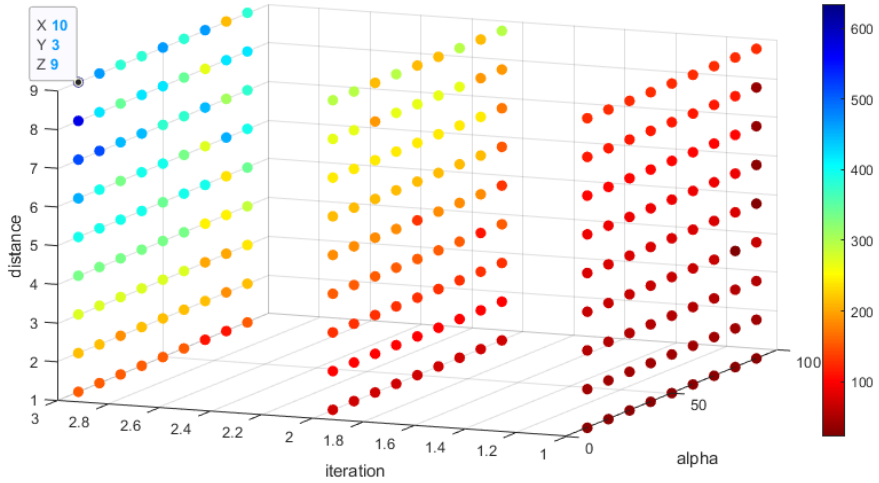


Figure 4.24 The overall representation of the optimised parameters for the first 3 iterations with the maximum area coverage detected at the top-left side.

4.5 Summary

This chapter presented an analysis of fractal classes by understanding the effect of adjusting parameters of a fractal formation toward covering an unknown area. Each fractal class contains a number of parameters that can partially change the fractal formation without affecting the development process. Each parameter is studied separately and simulated on a rectangle size area and an unknown area. All the experiment outcomes are compared to the PRM in terms of the number of robots used

and the amount of area covered on a fixed timeframe. In order to understand the effect of changing multiple parameters simultaneously, optimisation of a tree fractal formation is made by combining the parameters “separation angle (α), the branch distance (d), and the initial formation direction (θ)” to determine the maximum area coverage.

It is noticed that some parameters, such as the number of branches *N*, or changing the value of constant *C*, can aid the robotic swarm in increasing the chance of covering an unknown area. In contrast, some non-linear parameters, such as having obstacles inside an area, prevent the robotic swarm from covering an unknown area. The robotic swarm can adjust a part of the fractal formation to allow the robots to develop more branches, increasing the possibility of discovering different areas without changing the current formation. Optimising the parameters of the tree fractal model provides a substantial chance of obtaining the maximum area coverage by adjusting these parameters together and determining the best combination of the parameters leading to the highest area coverage a tree fractal model can get.

Unlike the transformation from one formation to another, which requires decomposing and composing shapes, a robotic swarm is not required to go through these stages to change a specific formation as fractal formations are adjustable. Observing the change in a particular fractal parameter reveals that a robotic swarm can better cover certain areas where changes in fractal parameters are required. Therefore, the robotic swarm can choose which parameter to change to prevent obstruction and increase the ability to cover areas. The common effect in the swarm formation is reducing the number of robots used compared to the PRM exploration method, where PRM uses a higher number of robots to cover as much area as a fractal formation, but with 88% fewer number of robots.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

This thesis presents an implementation of four fractal formations by a robotic swarm to explore and cover an unknown area. These fractals are classified, modelled, and verified to be used as a robotic swarm formation. The effect of changing a fractal parameter on SR when covering an unknown area shows an increase in the area coverage changing some parameters, such as branch length, and an obstruction avoidance when changing other parameters, such as the separation angle.

Overall, the work presented in this thesis shows that a contribution is made to the SR field by introducing fractals as a swarm formation approach. Fractals show the balance of using a particular number of robots when developing to a certain iteration. SR shows the ability to explore an unknown area using fractal formations with about 88% less use of robots and 3 to 10 times more efficiency than PRM, according to, when changing the parameters, giving the SR the flexibility to extend the exploration of an unknown area.

As fractal formations have advantages when used by SR to explore an unknown area, there also shows limitations with regards to facing obstruction and overlapping. Changing the parameters of a fractal formation can help the swarm to overcome obstacles, however, some of these parameters, such as the Z value, are not effective for certain fractal formations. One presented fractal, tree formation, shows overlaps between its branches when developing. This overlap is addressed by adding an overlap formula detecting and removing these overlapping branches. Still, the current development structure of the tree formation can be improved in future work.

In conclusion, it is confirmed that a swarm of robots can use fractals as a swarm formation to explore an unknown area. The research question “What are the advantages/disadvantages of using different fractal swarm formations to explore and cover unknown areas?” is answered by presenting both the advantages and disadvantages. Consequently, a number of suggested future work is presented in the next section.

5.2 Future Work

A number of suggested future work are described as follow: explore more fractals to improve the structure of the current fractal formations and obtain new advantages and parameters, and also, develop a decision-making process for a robotic swarm to decide which fractal to use.

As the tree fractal formation shows overlap in its branch, future work should consider looking at a similar structure to tree fractals whose branches do not overlap with each other. An example of such a plant is a fern, whose branches do not overlap and show the fractal properties of self-similarity and recursiveness. A fern fractal formation will overcome the overlap issue and may provide new parameters helping the SR cover more areas.

While a large number of fractals exist, future work should consider the exploration of new fractals for implementation by SR. This implementation will present new fractal classifications and discover new parameters. Examples of fractals to explore are: Sierpinski triangle, Koch curve, Apollonian gasket, etc (Bandt, Mörters and Zähle, 2009). Other natural fractals such as crystals, lightning bolts, clouds, etc. can also be considered and may need a growth rule to be implemented by SR.

Selecting a suitable fractal formation to explore an unknown area requires swarm detection, and therefore, future work should consider creating a decision-making process for a robotic swarm. This decision-making process would allow the swarm to make a cooperative decision and ensure that the selected fractal formation and the changing parameter effectively cover the unknown area. Building an independent decision-making process, particularly for fractal formations, allows for adding more fractals/parameters to be used by SR.

References

Abukhalil, T. and Sobh, T. (2013) *Survey on Decentralized Modular Swarm Robots and Control Interfaces, Madhav Patil & Tarek Sobh International Journal of Engineering (IJE)*.

Ahmadi, M. and Stone, P. (2006) "A Multi-Robot System for Continuous Area Sweeping Tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1724–1729.

Ahmed, J.U. *et al.* (2021) *Rescue Mission in the Tham Luang Nang Non Cave, Thailand*. 1st edn, *Rescue Mission in the Tham Luang Nang Non Cave, Thailand*. 1st edn. London, United Kingdom: SAGE Publications.

Amar, L.B. and Jasim, W.M. (2021) "Hybrid metaheuristic approach for robot path planning in dynamic environment," *Bulletin of Electrical Engineering and Informatics*, 10(4).

Anguera, J. *et al.* (2020) "Fractal Antennas: An Historical Perspective," *Fractal and Fractional*, 4(1), pp. 1–26.

Ashby, W.R. (2004) "Principles of the Self-Organizing System," *ECO Emergence: Complexity and Organization*, 6(1–2), pp. 102–126.

Ashton, C. *et al.* (2015) "The Search for MH370," *Journal of Navigation*, 68(1), pp. 1–22.

Balka, R., Buczolich, Z. and Elekes, M. (2015) "A New Fractal Dimension: The Topological Hausdorff Dimension," *Advances in Mathematics*, 274, pp. 881–927.

Bandt, C., Mörters, P. and Zähle, M. (2009) *Fractal Geometry and Stochastics IV*, *Progress in probability*. Edited by P. Mörters. Berlin: Birkhäuser Verlag AG.

Brüderlin, B. (1998) *Geometric Constraint Solving and Applications*, *Geometric Constraint Solving and Applications*. Springer Science & Business Media.

Cabrera-Mora, F. and Xiao, J. (2012) "A Flooding Algorithm for Multirobot Exploration," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3), pp. 850–863.

Cai, C. *et al.* (2007) "Collision Avoidance in Multi-Robot Systems," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007*, pp. 2795–2800.

Cardona, G.A. and Calderon, J.M. (2019) "Robot Swarm Navigation and Victim Detection Using Rendezvous Consensus in Search and Rescue Operations," *Applied Sciences (Switzerland)*, 9(8), pp. 1–23.

Chen, Y.C. and Wang, Y.T. (2007) "Obstacle Avoidance and Role Assignment Algorithms for Robot Formation Control," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1–6.

Chen, Y.G. and Wang, Y.T. (2007) "Dynamic Role Assignment Algorithm for Robot Formation Control," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1–6.

Cheng, C. *et al.* (2012) "Outdoor scene image segmentation based on background recognition and perceptual organization," *IEEE Transactions on Image Processing*, 21(3), pp. 1007–1019.

Cheng, J., Cheng, W. and Nagpal, R. (2005) "Robust and self-repairing formation control for swarms of mobile agents," *AAAI'05 Proceedings of the 20th national conference on Artificial intelligence*, 1, pp. 59–64.

Cheng, K., Wang, Y. and Dasgupta, P. (2009) "Distributed Area Coverage Using Robot Flocks," in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 678–683.

Choa, O. *et al.* (2016) "Stable Isotopes in Guano: Potential Contributions Towards Palaeoenvironmental Reconstruction in Tabon Cave, Palawan, Philippines," *Quaternary International*, 416, pp. 27–37.

Dadgar, M., Jafari, S. and Hamzeh, A. (2016) “A PSO-based multi-robot cooperation method for target searching in unknown environments,” *Neurocomputing*, 177.

Dorigo, M., Theraulaz, G. and Trianni, V. (2021) “Swarm robotics: Past, present, and future,” *Proceedings of the IEEE*.

Efremov, M.A. and Kholod, I.I. (2020) “Swarm Robotics Foraging Approaches,” in *Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020*.

Eissa, H. *et al.* (2018) “Enhancing Robotic Swarms With Fractal Behaviours to Explore Unknown Enclosed Areas,” in *3rd Medway Engineering Conference: Systems: Efficiency, Sustainability and Modelling*. Chatham Maritime, United Kingdom, pp. 1–6.

Falconer, K. (1990) “Fractal Geometry: Mathematical Foundations and Applications.,” *John Wiley and Sons*, p. 886.

Faria Dias, P.G. *et al.* (2021) “Swarm robotics: A perspective on the latest reviewed concepts and applications,” *Sensors*.

Foead, D. *et al.* (2021) “A Systematic Literature Review of A*Pathfinding,” in *Procedia Computer Science*.

Gneiting, T., Ševčíková, H. and Percival, D.B. (2012) “Estimators of Fractal Dimension: Assessing the Roughness of Time Series and Spatial Data,” *Statistical Science*, 27(2), pp. 247–277.

Gordon, N., Wagner, I. a. and Bruckstein, A.M. (2003) “Discrete Bee Dance Algorithm for Pattern Formation on a Grid,” in *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.*, pp. 1–5.

Gu, C. *et al.* (2009) “Recognition Using Regions,” in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pp. 1030–1037.

Gunna, T. and Anderson, J. (2013) “Dynamic Heterogeneous Team Formation for Robotic Urban Search and Rescue,” in *Procedia Computer Science*, pp. 22–31.

le Hardy, P.K. and Moore, C. (2014) “Deep Ocean Search for Malaysia Airlines Flight 370,” in *2014 Oceans - St. John’s, OCEANS 2014*. St. John’s, NL, Canada: IEEE, pp. 1–4.

Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, 4(2), pp. 100–107.

Heinz-Otto Peitgen, Hartmut Jürgens, D.S. (2004) *Chaos and Fractals: New Frontiers of Science*. 2nd edn. NewYork: Springer.

Hidalgo, M. and Joan-Arinyo, R. (2015) “h-graphs: A New Representation for Tree Decompositions of Graphs,” *Computer-Aided Design*, 67–68(2015), pp. 38–47.

Hoffman, C.M., Lomonosov, A. and Sitharam, M. (2001) “Decomposition Plans for Geometric Constraint Problems, Part II: New Algorithms,” *Journal of Symbolic Computation*, 31(4), pp. 409–427.

Hooman Oroojeni, M.J., Majid Al-Rifaie, M. and Nicolaou, M.A. (2018) “Deep neuroevolution: Training deep neural networks for false alarm detection in intensive care units,” in *European Signal Processing Conference*.

Hsu, H.C.-H. and Liu, A. (2004) “Multiple Teams for Mobile Robot Formation Control,” in *Proceedings of the 2004 IEEE International Symposium on Intelligent Control, 2004.*, pp. 168–173.

Hunt, E.R., Jones, S. and Hauert, S. (2019) “Testing the Limits of Pheromone Stigmergy in High-Density Robot Swarms,” *Royal Society Open Science*, 6(11), pp. 1–14.

Ismail, Z.H. and Hamami, M.G.M. (2021) “Systematic literature review of swarm robotics strategies applied to target search problem with environment constraints,” *Applied Sciences (Switzerland)*.

Jermann, C. *et al.* (2006) “Decomposition of Geometric Constraint Systems: a Survey,” *International Journal of Computational Geometry & Applications*, 23(7), pp. 1–31.

Jevtić, A. *et al.* (2012) “Distributed Bees Algorithm for Task Allocation in Swarm of Robots,” *IEEE Systems Journal*, 6(2), pp. 296–304.

Kapoutsis, A.C., Chatzichristofis, S.A. and Kosmatopoulos, E.B. (2017) “DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, 86(3–4), pp. 663–680.

Karaboga, D. and Akay, B. (2009) “A survey: Algorithms simulating bee swarm intelligence,” *Artificial Intelligence Review*, 31(1–4), pp. 61–85.

Kavraki, L.E. *et al.* (1996) “Probabilistic Roadmaps For Path Planning in High-Dimensional Configuration Spaces,” *IEEE Transactions on Robotics and Automation*, 12(4), pp. 566–580.

Kernbach, S. *et al.* (2013) “Adaptive Collective Decision-Making in Limited Robot Swarms Without Communication,” *International Journal of Robotics Research*, 32(1), pp. 35–55.

Koch, M., Manuylov, I. and Smolka, M. (2021) “Robots and Firms,” *The Economic Journal*, 131(683), pp. 2553–2584.

Koenig, S., Szymanski, B. and Liu, Y. (2001) “Efficient and Inefficient Ant Coverage Methods,” *Annals of Mathematics and Artificial Intelligence*, 31(1–4), pp. 41–76.

Koo, T.J. and Shahruz, S.M. (2001) “Formation of a Group of Unmanned Aerial Vehicles (UAVs),” in *Proceedings of the American Control Conference*. Arlington, VA, USA: IEEE, pp. 69–74.

Kurokawa, H. *et al.* (2008) “Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System,” *International Journal of Robotics Research*, 27(3–4), pp. 373–386.

Kwa, H.L., Leong Kit, J. and Bouffanais, R. (2022) “Balancing Collective Exploration and Exploitation in Multi-Agent and Multi-Robot Systems: A Review,” *Frontiers in Robotics and AI*. Frontiers Media S.A.

LaValle, S.M. (1998) “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” in *The annual research report*, pp. 1–4.

Lavalle, S.M. (2006) *Planning Algorithms, Cambridge University Press*. Cambridge, United Kingdom: Cambridge University Press.

Lima, D.A. and Oliveira, G.M.B. (2017) “A Probabilistic Cellular Automata Ant Memory Model for a Swarm of Foraging Robots,” in *2016 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016*. Phuket, Thailand, pp. 1–6.

Majid al-Rifaie, M., Aber, A. and Raisys, R. (2013) “Swarming robots and possible medical applications,” *International Society for Electronic Art*, pp. 1–7.

Majid-al-Rifaie, M. and Bishop, J.M. (2020) “Stochastic Diffusion Search: A Tutorial,” in *Swarm Intelligence Algorithms*.

Makarenko, A.A. *et al.* (2002) “An Experiment in Integrated Exploration,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Lausanne, pp. 534–539.

Malchow, H. *et al.* (2000) “Spatio-Temporal Pattern Formation in Coupled Models of Plankton Dynamics and Fish School Motion,” *Nonlinear Analysis: Real World Applications*, 1(1), pp. 53–67.

Mandelbrot, B.B. (1983) *The Fractal Geometry of Nature*. San Francisco: W.H. Freeman and Company.

Marjovi, A. *et al.* (2009) “Multi-Robot Exploration and Fire Searching,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 1929–1934.

Moravec, H.P. and Elfes, A. (1986) “High Resolution Maps from Wide-Angle Sonar,” in *IEEE International Conference on Robotics and Automation*, pp. 116–121.

Nasir, R. and Elnagar, A. (2015) “Gap Navigation Trees for Discovering Unknown Environments,” *Intelligent Control and Automation*, 6, pp. 229–240.

Oh, H. *et al.* (2017) “Bio-inspired self-organising multi-robot pattern formation: A review,” *Robotics and Autonomous Systems*, 91, pp. 83–100.

Olaronke, I. *et al.* (2020) “A Systematic Review of Swarm Robots,” *Current Journal of Applied Science and Technology*, pp. 79–97.

Pang, B. *et al.* (2021) “Effect of Random Walk Methods on Searching Efficiency in Swarm Robots for Area Exploration,” *Applied Intelligence*, 51(7), pp. 5189–5199.

Roy, D., Maitra, M. and Bhattacharya, S. (2021) “Exploration of Multiple Unknown Areas by Swarm of Robots Utilizing Virtual-Region-Based Splitting and Merging Technique,” *IEEE Transactions on Automation Science and Engineering*, pp. 1–12.

Saldana, D. *et al.* (2017) “A decentralized algorithm for assembling structures with modular robots,” in *IEEE International Conference on Intelligent Robots and Systems*.

Schranz, M. *et al.* (2020) “Swarm Robotic Behaviors and Current Applications,” *Frontiers in Robotics and AI*, 7(April), pp. 1–20.

Schultz, a. C. and Adams, W. (1998) “Continuous Localization Using Evidence Grids,” in *Proceedings - IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, pp. 2833–2839.

Sharma, S. and Tiwari, R. (2016) “A survey on multi robots area exploration techniques and algorithms,” in *2016 International Conference on Computational Techniques in Information and Communication Technologies, ICCTICT 2016 - Proceedings*, pp. 151–158.

Stachniss, C. and Burgard, W. (2003a) “Exploring Unknown Environments With Mobile Robots Using Coverage Maps,” in *IJCAI International Joint Conference on Artificial Intelligence*.

Stachniss, C. and Burgard, W. (2003b) “Exploring Unknown Environments with Mobile Robots Using Coverage Maps,” in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1127–1132.

Stentz, A. (1994) “Optimal and Efficient Path Planning for Partially-Known Environments,” in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3310–3317.

Thrun, M.C. and Ultsch, A. (2021) “Swarm intelligence for self-organized clustering,” *Artificial Intelligence*, 290.

Trianni, V., Nolfi, S. and Dorigo, M. (2008) “Evolution, Self-organization and Swarm Robotics,” in *Swarm Intelligence, Natural Computing Series*, pp. 163–191.

Varghese, B. and McKee, G. (2009) “A Review and Implementation of Swarm Pattern Formation and Transformation Models,” *International Journal of Intelligent Computing and Cybernetics*, 2(4), pp. 786–817.

Varghese, B. and McKee, G. (2010) “A mathematical model, implementation and study of a swarm system,” *Robotics and Autonomous Systems*, 58(3), pp. 287–294.

Vedachalam, N. *et al.* (2020) “Design considerations for strategic autonomous underwater swarm robotic systems,” *Marine Technology Society Journal*, 54(2).

Vichalai, C. (2019) “Geophysics Under Stressed: a Case Study of Tham Luang Nang Non Cave,” *RMUTSB Academic Journal*, 7(2), pp. 247–258.

Vicsek, T. and Gould, H. (2013) *Fractal Growth Phenomena*. 2nd edn, *Computers in Physics*. 2nd edn. World Scientific Press.

Wang, L. and Tang, S. (2021) “Editorial: An introduction to Fractals in Construction Materials,” *Fractals*, 29(2), pp. 1–5.

Xiong, N. *et al.* (2009) “Decentralized Flocking Algorithms for a Swarm of Mobile Robots: Problem, Current Research and Future Directions,” in *2009 6th IEEE Consumer Communications and Networking Conference, CCNC 2009*, pp. 1–6.

Xu, H.X.H. *et al.* (2010) “A Multi-robot Pattern Formation Algorithm Based on Distributed Swarm Intelligence,” in *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, pp. 71–75.

Yamauchi, B. (1997) “A Frontier-Based Exploration for Autonomous Exploration,” in *IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA*. Monterey, CA, USA, pp. 146–151.

Yamauchi, B. (1998) “Frontier-Based Exploration Using Multiple Robots,” in *Proceedings of the International Conference on Autonomous Agents*, pp. 47–53.

Yamauchi, B., Schultz, A. and Adams, W. (1998) “Mobile Robot Exploration and Map-Building with Continuous Localization,” in *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3715–3720.

Yamauchi, B., Schultz, A. and Adams, W. (1999) “Integrating Exploration and Localization for Mobile Robots,” *Adaptive Behavior*, 7(2), pp. 217–229.

Yang, X.-S. *et al.* (2017) “Swarm Intelligence: Past, Present and Future,” *Soft Computing*, 22(1), pp. 5923–5933.

Yang, Y. and Tian, Y. (2007) “Swarm Robots Aggregation Formation Control Inspired by Fish School,” in *2007 IEEE International Conference on Robotics and Biomimetics, ROBIO*, pp. 805–809.

Zhou, Y. and Goldman, R. (2017) “Building Fractals with a Robot Swarm,” in *International Conference in Swarm Intelligence*, pp. 185–198.

Appendix A

A.1 MATLAB and V-REP Coding for fractal construction and implementation.

Below is a simplified code illustrating the process of developing the N-branch Tree fractal formation.

```
img = imread('The image of the area to be explored');
map = im2bw(img,0.5);     // convert to binary image
imshow(map)               // prepare the map for exploration
x-axis =                  // set x-axis as a start point in the map
y-axis =                  // set y-axis as a start point in the map
theta =                   // set the initial direction formation
distance=                 // set the robot's travel distance
i =                      // counter to count the number of iterations
a = b =                   // output locations to be used by the swarm
 axis(gca,'equal')
   [a,b,i] = rotate(x-axis,y-axis,theta,distance,map,0,0,1);

function [a,b,i] = rotate(x1,y1,th,dis,map1,a,b,i)
## this section is related to placing obstacles in case of having a
rectangle area ##
viscircles([300 60],20);
## Circle obs.
rectangle('Position',[184,60,50,40],'FaceColor',[0       0       .5]);
##center obs.
rectangle('Position',[144,30,30,40],'FaceColor',[0       0       .5]);
##side obs 1
rectangle('Position',[240,20,60,50],'FaceColor',[0       0       0]);
##side obs 2
## End of obstacle section ##

## This section sets the limit of the image ##
if y2 >= y-max-area
   y2= y-max area;
endif

if y2 <= y-min-area
   y2 = y-min-area;
endif

if x2 >= x-max area
   x2= x-max area;
endif

if x2 <= x-min area
   x2 = x-min area;
endif
## End of limit section ##
```

```
## This section shows the development of the tree fractal formation##
x2=x1+cosd(th)*(dis+70);
 y2=y1+sind(th)*(dis+70);
 a(i)=x1; b(i)=y1; a(i+1)=x2; b(i+1)=y2; i+=2;
## End of development section ##

## This section ensures the location is collected and not set out of
the range of the map ##
if dis~=0
     xa=round(x1); ya=round(y1);
     xb=round(x2); yb=round(y2);
     val1 = map1(ya,xa);
     val2 = map1(yb,xb);

als=sqrt(power(a[32]-a[22],2)+power(b[32]-b[22],2));  ##(need  arrays
to store data)

     if val1 == 0 || val2 == 0
         children = get(gca, 'children');
         delete(children(1));
          x2=x1; y2=y1;
     endif

line([x1 x2],[y1 y2],'Color','b', 'LineWidth',3)  // The path line for
the robots to follow

## End of obstacle section ##

## This section the recurring process for N number of iterations ##

     [a,b,i] = rotate(x2,y2,th+60,dis-1,map1,a,b,i);
     [a,b,i] = rotate(x2,y2,th+30,dis-1,map1,a,b,i);
     [a,b,i] = rotate(x2,y2,th,dis-1,map1,a,b,i);
     [a,b,i] = rotate(x2,y2,th-30,dis-1,map1,a,b,i);
     [a,b,i] = rotate(x2,y2,th-60,dis-1,map1,a,b,i);
     pause(0.1);
endif
endfunction
## End of the function ##
```

The below simplified function illustrates the process of developing the Vicsek fractal formation.

```
function [a,b,c] = Iter(a,b,c)
x=[]; y=[];
z1=[0 0 0 0]; z2=[0 0 0 0];
th = 0;
for n = 1:1:4
x(n)=(a*cosd(th))+b;
y(n)=(a*sind(th))+c;
line([b x(n)],[c y(n)],'LineWidth',3,'Color','blue')
th=th+90;
```

```
end
 end
```

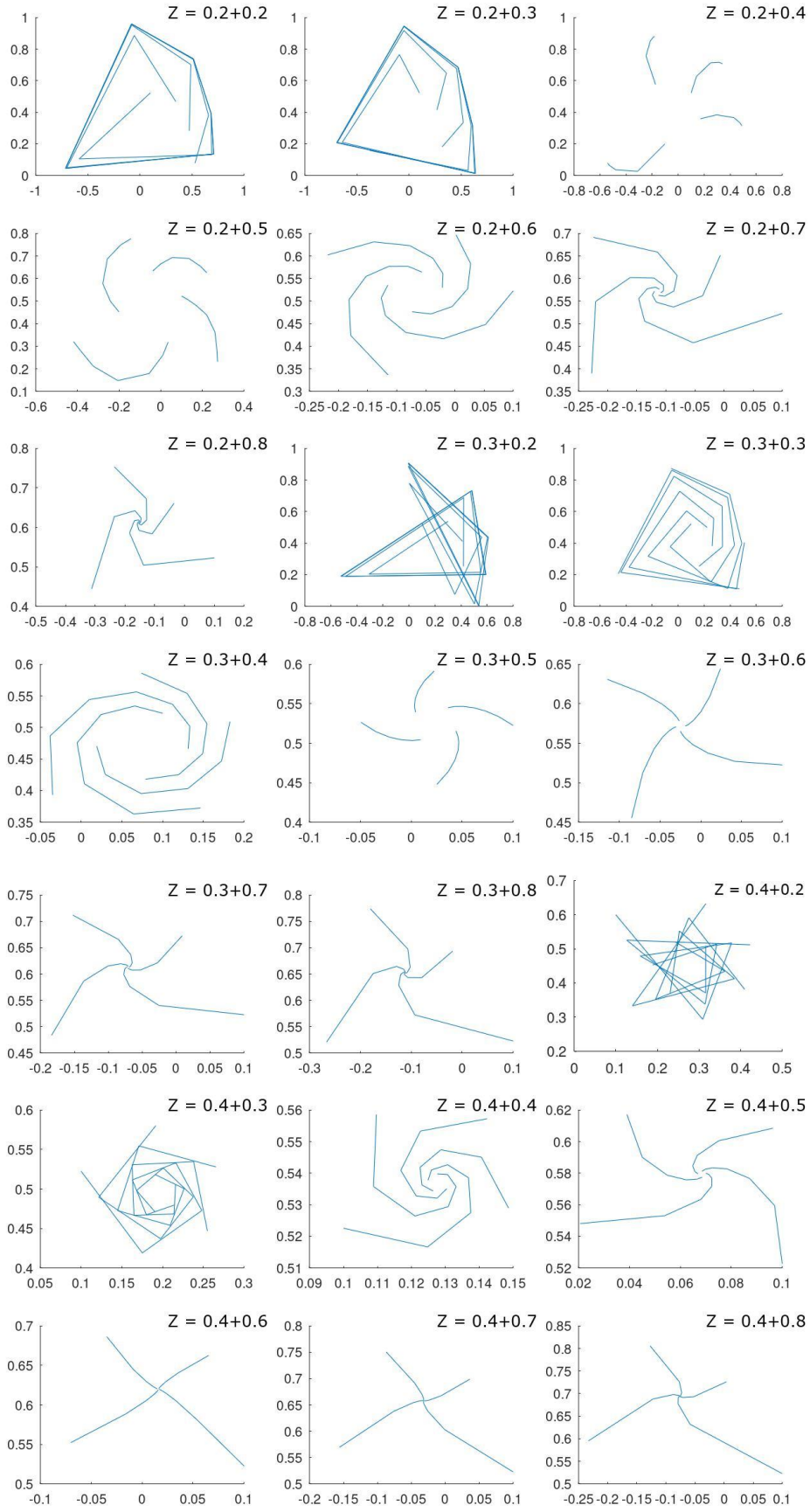
Below is a simplified function illustrating the process of developing both Julia set and reverse Julia set fractal formation.

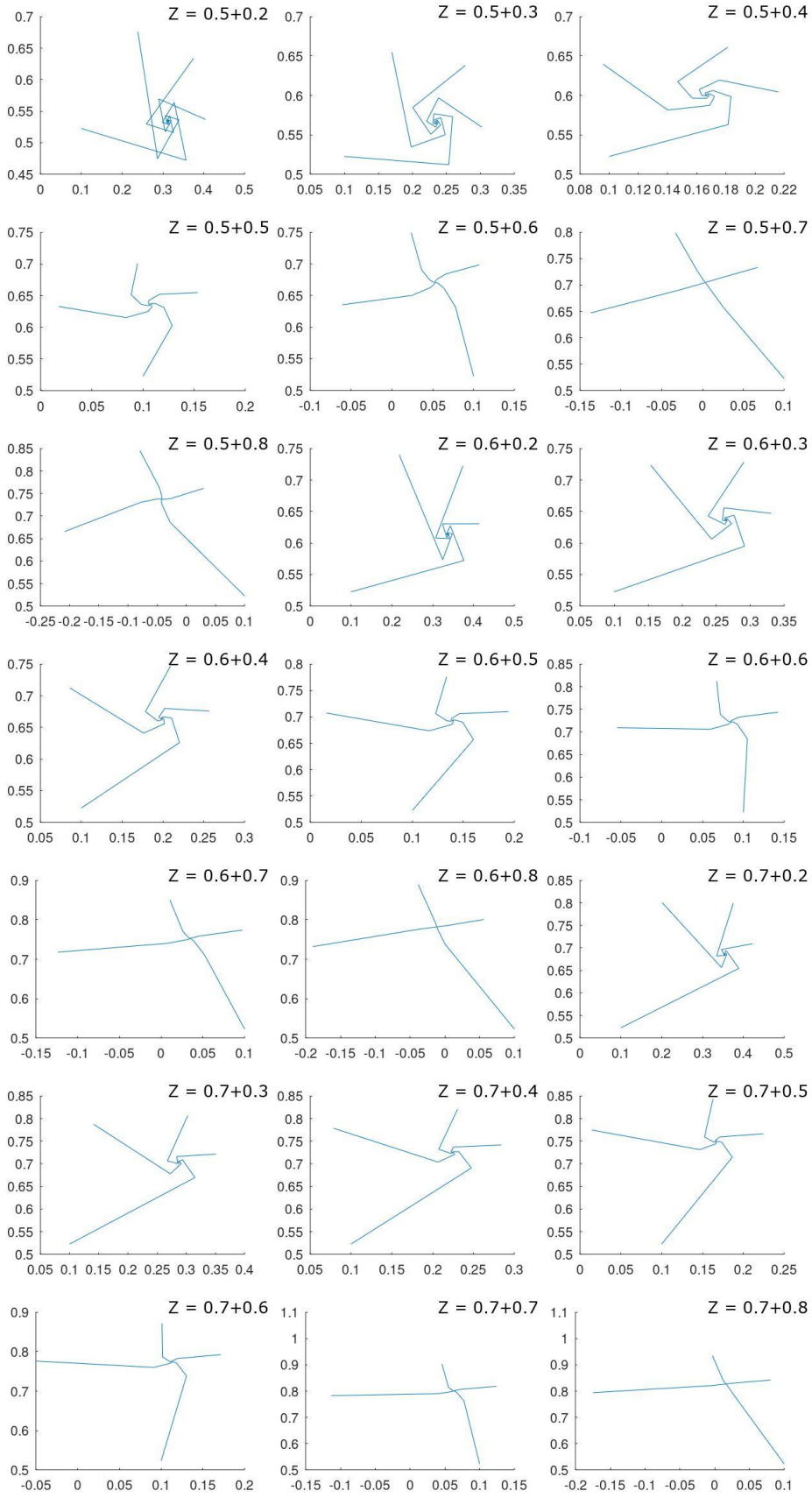
```
zx(1)=0;
ax(1)=real(zx(1)); ax(1)=(ax(1)*1510)+15;
bx(1)=imag(zx(1)); bx(1)=(bx(1)*1510)-640;

c=complex(0.1,0.6);


for i=1:1:N
   zx(i+1)=sqrt(zx(i)-c);    // Reverse Julia set
   zx(i+1)=(zx(i))^2+c;      // Julia set
    endif

ax(i+1)=real(zx(i+1)); ax(i+1)=(ax(i+1)*1510)+15;
bx(i+1)=imag(zx(i+1)); bx(i+1)=(bx(i+1)*1510)-640;

end
for i=1:1:N
    pause(0.001);
     line([ax(i) ax(i+1)],[bx(i)
bx(i+1)],'LineWidth',2,'Color','blue')
     hold on
end
```

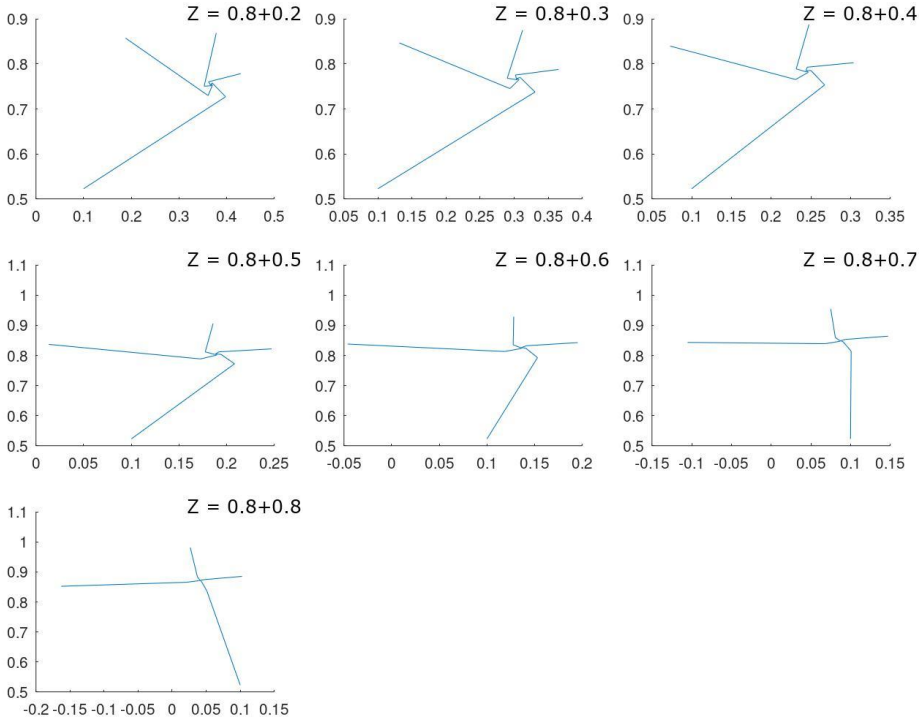
Below is a simplified function illustrating the process of transmitting location points from MATLAB to V-REP for the robots.

```
vrep=remApi('remoteApi');
vrep.simxFinish(-1);

clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)
        disp('Connected');
        % set for devoloping a fractal formation
for i=1:1:N                              // N is the number of robots
[returnCode,copN]=vrep.simxGetObjectHandle(clientID
,'Quadricopter_target' ,vrep.simx_opmode_blocking);
        x(1:3)=[x, y, hight];
        [returnCode]=vrep.simxSetObjectPosition(clientID,copN  ,-1  ,x
,vrep.simx_opmode_oneshot_wait);
pause(3);
end
vrep.delete();
test1;
```

A.2 All the 49 sets of the output shapes when changing the *Z* value for the reverse Julia set.




Z = 0.5+0.2
Z = 0.5+0.3
Z = 0.5+0.4
Z = 0.5+0.5
Z = 0.5+0.6
Z = 0.5+0.7
Z = 0.5+0.8
Z = 0.6+0.2
Z = 0.6+0.3
Z = 0.6+0.4
Z = 0.6+0.5
Z = 0.6+0.6
Z = 0.6+0.7
Z = 0.6+0.8
Z = 0.7+0.2
Z = 0.7+0.3
Z = 0.7+0.4
Z = 0.7+0.5
Z = 0.7+0.6
Z = 0.7+0.7
Z = 0.7+0.8


Z = 0.8+0.2
Z = 0.8+0.3
Z = 0.8+0.4
Z = 0.8+0.5
Z = 0.8+0.6
Z = 0.8+0.7
Z = 0.8+0.8

A.3 The optimisation process for N-Branch tree fractal formations' parameters.

Below is a simplified code illustrating the process of optimising the parameters of the N-branch Tree fractal formation.

```
img = imread('TabonCaveImp.PNG');
map = im2bw(img,0.5);
imshow(map)

m=9;
itration=linspace(1,3,3);
alpha=linspace(10,90,m);
distance=linspace(1,9,m);
[X,Y,Z] = meshgrid(alpha,itration,distance);
x = X(:);
y = Y(:);
z = Z(:);
for itr = 1:length(itration)
    for al=1:length(alpha)
        for dist=1:length(distance)
            x0    =    [X(itr,al,dist),Y(itr,al,dist),Z(itr,al,dist)];
[a,b,i,N]=rotate(106,250,0,itration(itr)+1,distance(dist),alpha(al),m
ap,0,0,1,0);
%area(itr,al,dist)=((distance(dist)*4)+(0.5*pi*(2)^2))+((distance(dis
t)*4)+(0.5*pi*(2)^2))*(itration(itr)*(itration(itr)-1));
area(itr,al,dist) = ((distance(dist)*4)+(0.5*pi*(2)^2))*(N);
        end
    end
end
figure(2);clf(2)
pp=scatter3(x,y,z,50,area(:),'filled');
colorbar;colormap('jet');
xx = squeeze(X(1,:,:));
%yy = squeeze(Y(1,:,:));
zz = squeeze(Z(1,:,:));
area1 = squeeze(area(1,:,:));
figure(3);clf(3)
surf(xx,zz,area1)
colorbar;colormap('jet');
```