EXCAVATION USING DECENTRALIZED SWARM ROBOTS FOR OFF-EARTH HABITATION

L. K. I. K. Gunasekara

199469A

Degree Of Master Of Science In Artificial Intelligence

Department Of Computational Mathematics,
Faculty Of Information Technology

University Of Moratuwa
Sri Lanka

March 2022

Excavation using Decentralized Swarm Robots for off-earth Habitation

Loku Kaththotage Isuru Kalhara Gunasekara

(199469A)

Thesis Submitted In Partial Fulfillment Of The Requirements For The Degree Of Master Of Science In Artificial Intelligence

Department Of Computational Mathematics,
Faculty Of Information Technology

University Of Moratuwa
Sri Lanka

March 2022

DECLARATION

I declare that this is my own research proposal and this proposal does not incorporate without acknowledgement any material previously published submitted for a Degree or Diploma in any other university or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Signature: ..... Date:

I have read the proposal and it is in accordance with the approved university proposal outline. I am willing to supervise the research work of the above candidate in the proposed area.

Signature of the supervisor:..... Date:

Abstract

Civilization has progressed over time as a result of exploration of previously undiscovered portions of the Earth and the subsequent exploitation of new resources. Humans will now explore and exploit new places beyond Earth, i.e. in space, as a natural continuation of this process. The Moon has been partially investigated, but it still needs to be fully explored and colonized. Mars and beyond will be the next phase. The establishment of outposts on these new space bodies will be necessary before they can be habited. But these extraterrestrial constructions are expensive, time-consuming, and risky. In this article building habitations by excavation is suggested and proven to be more convenient. Having many challenges in the physical excavation tasks, only a few researchers have been trying to innovate and improve excavation-related technologies. Amongst these, some have proposed adopting robotics in the aspects of excavation in the construction process but none has been tested practically. In our work, we try to introduce a novel approach for excavation using swarm robotics.

Behaviour of a robotic swarm is collective and aimed at solving a problem using the collective conduct. This is similar to the natural animal swarm behaviour of bees/ ants/ termites...etc. Even though there are many researches and developments done in the field of swarm robotics, the concept has not yet made its way into industrial environments. In order to colonize planets and moons, it is required to build a surface structure which needs to be a few meters thick to protect living beings from solar/cosmic radiation, meteoroid impacts, and extreme temperature variations. However, creating a structure with thickness of a few meters has been a research challenge for many decades due to practical limitations of developing them on off-earth.

This research proposes our approach **C**ave **C**onstruction using **S**warm **R**obots acronymed as CCSR, a practical method to excavate the ground to create subsurface habitats using decentralized robot swarms. Our CCSR design uses vision, RFID, and orientation sensor data to decide actioneeded to be taken by the robot. Robots can do basic actions such as traversing, removing, and dumping regolith.

The swarm consists of a set of robots that are practical to implement, with limited visibility and limited communication skills. Having only the local view of the terrain, robots in the swarm excavate a given shape in 3D in collaboration with the other robots in the swarm. With the application of swarm concepts in an improved manner, the swarm is able to construct the given shape through excavation, displaying true parallelism which in turn will improve the construction time.

DEDICATION

To my grand parents for their dedicated partnership in the success of my life

ACKNOWLEDGEMENTS

I would like to send my deepest thanks to Dr. K.S.D. Fernando, who ignited my passion for research and provided help and wisdom at every step of the way of this research. Her guidance was unequivocally crucial to this work, and I owe a great deal of its success to her.

I would like to extend my thanks and appreciation to the staff members of the Department of Computation Mathematics, for their interest in, and consideration of, my research work.

Lastly, I would also like to express my heartfelt appreciation to all my family and friends, who supported me in this long endeavor and through all my life. Though they may not know it, the impact they have had upon me is substantial, and I would have never made it to where I am now without them.

TABLE OF CONTENTS

DECLARATION **2**

Abstract **3**

DEDICATION **4**

ACKNOWLEDGEMENTS **5**

LIST OF FIGURES **9**

LIST OF TABLES **11**

CHAPTER 1 - INTRODUCTION **1**

1.1 Prolegomena 1

1.2 Aim And Objectives 2

1.3 Background And Motivation 2

1.4 Research Scope 3

1.5 Problem Definition 3

1.6 Novel Approach To Swarm Robots Based Excavation 3

1.7 Outline Of The Thesis 4

1.8 Summary 4

CHAPTER 2 - SWARM ROBOTICS IN OFF-EARTH CONSTRUCTION **5**

2.1 Introduction 5

2.2 What Are Swarm Robots? Why Are They Applicable Here? 5

2.3 Early Developments (Gestation) In Swarm Robotics And Concepts 6

2.3.1 Swarm intelligence 6

2.3.2 Basic swarm behaviors for swarm robotics 7

2.4 Breakthroughs And Trends In Swarm Robots 10

2.4.1 Concepts in collective constructions 10

2.5 Off-Earth Constructions 18

2.6 Challenges In Swarm Robot Based Constructions 22

2.7 Problem Definition 25

2.8 Summary 25

CHAPTER 3-TECHNOLOGIES USED FOR SWARM BASED EXCAVATION 26
3.1 Introduction 26
3.2 Swarm Robot Design 26
3.3 Technologies Used In Swarm Robot Implementation 27
3.3.1 Stigmergic cooperation 27
3.3.1.1 Stigmergy from the local view of the construction 27
3.3.1.2 Stigmergy from the environment 28
3.3.2 Collective construction 29
3.3.3 Rule based controllers 29
3.4 Application Of Swarm Robots 30
3.5 Summary 30
CHAPTER 4-APPROACH 31
4.1 Introduction 31
4.2 Hypothesis 31
4.3 Input 31
4.4 Output 32
4.5 Process 32
4.6 Users 33
4.7 Features 33
4.8 Summary 33
CHAPTER 5 - DESIGN 34
5.1 Introduction 34
5.2 High Level Architecture 34
5.2.1 Localization 35
5.2.2 Motion planner 35
5.2.3 Shape map 36
5.3 Design of swarm robots 36
5.4 Simulation design 38
5.5 Summary 38
CHAPTER 6 - IMPLEMENTATION 39

6.1 Introduction 39
6.2 Simulation Framework 39
6.3 Simulation Environment 39
6.4 Robot Controller 40
6.4.1 Navigation 40
6.4.2 Sensor data processing 41
6.4.3 Calculating the next excavation location 41
6.4.4 Control algorithm 43
6.5 Simulation Outcome 44
6.6 Summary 49
CHAPTER 7 - EVALUATION **50**
7.1 Introduction 50
7.2 Evaluation Procedure 50
7.2.1 Parallel behavior in construction 51
7.2.2 System redundancy 51
7.2.3 Final outcome of the system 51
7.3 Performance Vs Number Of Bots 52
7.4 Use Cases 53
7.5 Summary 53
CHAPTER 8 - CONCLUSION **54**
8.1 Introduction 54
8.2 Conclusions 54
8.3 Future work 55
Appendix **58**

LIST OF FIGURES

	Page
Figure 2.1: Basic swarm Behaviours	8
Figure 2.2: Nests constructed by blind bulldozing by one, two and four robots	11
Figure 2.3: Leader robot managing other robots using light source	13
Figure 2.4: Some shapes constructed by rule-based lattice swarms	14
Figure 2.5: Termites robots create 3D structure	14
Figure 2.6: Global map of construction (left) and perceived local map (right) of a robot (marked x) at a particular time. The red color circle shows the maximum sensory distance of the robot	16
Figure 2.7: Structure proposed for off-earth habitats	19
Figure 2.8: Adding regolith layer top of the habitat module	19
Figure 2.9: Modular heterogeneous swarm robots that can form and 3D print	20
Figure 2.10: Structure made using sulfur concrete	21
Figure 2.11: NASA's RASSOR robot	21
Figure 2.12: Various types of off-earth construction	22
Figure 2.13: Swarm robot based underground structure proposed in 2021 by Bier et al	23
Table 2.14: Summary of benefits and challenges	24
Figure 5.1- Top level architecture of CCSR	35
Figure 5.2 - Example of a shape map	37
Figure 5.3: Swarm bot inputs and outputs	38
Figure 6.1a/b: View of the graphical user interface and the view of the simulation environment. Here 30x30x8 block terrain is used with 8 robots. Gray area is the starting position of those robots. Orange spheres shows robots.	41
Figure 6.2: (a) and (b) shows a robot removing bricks from the layer its on	43

and (c) and (d) shows a robot removing the front layer thats one step higher than its position

Figure 6.3: Shape Map of the structure 45

Figure 6.4: After excavating few blocks 45

Figure 6.5: bricks that are above excavated bricks are made transparent for easy visualisation 46

Figure 6.6: After 80% completion 46

Figure 6.7: After the final shape map is obtained 47

Figure 6.8: performance of each robot 47

Figure 6.9: Three robots are killed while the excavation happens and the given shape map 48

Figure 6.10: Performance of each robot 48

Figure 7.1: Time taken to finish a given shape vs number of bots in the swarm system 51

LIST OF TABLES

Table 2.1: Tabulation of existing research
Table 2.2: Summary of benefits and challenges

CHAPTER 1 - INTRODUCTION

1.1 Prolegomena

Mars and Lunar are currently within reach for interplanetary habitation, and technology readiness is predicted to be achieved in the near future. Previous study suggests that regolith, crushed rock, and dust found on Mars might be used as a construction material, and that regolith structures could shield people from enormous levels of radiation. But the structure thickness should be several meters thick to make the environment habitable [1]. To achieve this thickness heavy construction machineries and large human labour is required. Moreover, transportation of these resources and the process of construction is expensive and risky.

As a solution to these challenges, our research proposes a practical method to excavate the ground using decentralized swarm robots to create subsurface habitats. Because the temperature underneath is more steady, excavation provides natural protection from radiation as well as thermal insulation. At the same time, important resources can be mined for In-Situ Resource Utilization (ISRU) through excavation.

The performance of a collective excavation task done by a robot swarm depends mainly on the systematical order of excavating the ground and the task distribution within the swarm. A robot in such a multi-robot system needs to fetch excavated blocks from the environment and place them in a suitable position in the desired structure, so that the structure is built collectively within a minimum time and a minimum distance traveled.

This chapter gives an overall idea about the research we have done in the subject area of swarm robots in excavation to promote extraterrestrial habitation. In doing so, we have structured the introduction chapter under several subheadings, aims and objectives, background and motivation, problem definition, and our solution.

1.2 Aim And Objectives

The aim of this research is to develop a simulation of a robot swarm for underground cave construction. In order to reach this aim, we have defined the following objectives.

- Critically review existing approaches for extraterrestrial construction methods using swarm robots and define the research problem.
- Study on how swarm excavation is advantageous in outer space construction that's suitable for habitation.
- An in-depth study on the swarm, multi-agent, and multi-robot technologies and analysis.
- Design and develop an algorithm to control robots to achieve the final task.
- Evaluation of the approach on a simulation and compare and contrast with existing approaches.

1.3 Background And Motivation

Swarm robots have numerous applications in the real world. Many researches have been carried out related to swarm robotics where they have proposed methods for performing various tasks using various algorithms. Space exploration is a key area that considers using robotic swarms for various tasks since it is easier to get a large number of small structures into space than sending a small number of large machinery. There are many proposals for colonizing Mars and the moon which require building large structures on them Ruess et al [1]. Huge automatic construction tasks like that may require a swarm robotic construction approach since it is not practical to use one large 3D printer of the size of the building.

Proposals for constructing habitable buildings on mars and the moon includes creating shelter structures at first for protection from radiation and small meteorites. Rather than building a structure, digging underground to have a cave structure for living is proposed by Bier et al [2] in 2021. In this research, we will focus on methods to dig 3D cave structures as an initiative to such tasks.

1.4 Research Scope

- Project is focused to develop a control and decision algorithm for swarm robots
- Project is not going to discuss the hardware of the robots, localization or mapping.
- Project is not going to discuss how the construction site is selected or the cave plan is designed
- 3D Simulation with block-based environment will be developed such that simulated robots can manipulate those blocks.
- Decision and control algorithm of swarm robots will be developed to construct a practical subsurface structure

1.5 Problem Definition

To make the surface of the Moon/Mars habitable, the living area should be covered from at least a 2.5m regolith layer to mitigate hazards such as solar/cosmic radiation, sand storms and high-temperature variation (-150C to 100C). Many types of research are focused on this area where they propose different methods to build structures that will help living beings to survive. But these proposed solutions are expensive, inefficient, and not practical because of the need of having heavy machines, established communication, and the requirement of global view.

Therefore, building a robust structure that separates living beings and the planet's surface before sending astronauts has been identified as a research problem.

1.6 Novel Approach To Swarm Robots Based Excavation

This research proposes construction of an underground cavity for human habitation. Underground habitation offers the advantage of natural protection from radiation exposure and storms while also being less affected by thermal stresses as the temperature is more stable underground Bier et al [2]. Expansion of the cave system is easier and less expensive than structures on the surface. Stigmergy based localization and control can be implemented which will adapt to the environment.

1.7 Outline Of The Thesis

Chapter 1 provides an overview of the research which covers a brief introduction to the background of the problem domain, identified problem, solution, and resource requirements of the project.

Chapter 2 consists of a critical review of previous related work and challenges identified in their solutions.

Chapter 3 presents a deep analysis of related technologies and past work done related to the technology.

Chapter 4 explains the approach taken in intended research and inputs, outputs, process, and features of the outcomes.

Chapter 5 gives a step-by-step description of our simulation design.

Chapter 6 includes the implementation details of the project.

Chapter 7 discusses the evaluation of the proposed system.

Chapter 8 Finally, includes the concluding remarks with implications while future work is discussed.

1.8 Summary

In this chapter we discussed the overall idea about the research we have done in the subject area of swarm robots in excavation. In the next chapter, a detailed discussion on the literature review on swarm robots is given.

CHAPTER 2 - SWARM ROBOTICS IN OFF-EARTH CONSTRUCTION

2.1 Introduction

In the previous chapter, we provided an overall picture of the research presented in this thesis. This chapter gives a critical review of the literature on Swarm Robots and application of them in extraterrestrial constructions.In doing so, we have structured the literature review under several aspects, where we discuss early developments in swarm robots and concepts, collective construction with swarm robots, Off-earth constructions, Trends and challenges in swarm robots,etc. This chapter also summarizes the challenges in swarm robots and defines the research problem.

2.2 What Are Swarm Robots? Why Are They Applicable Here?

To understand our work properly it is necessary to understand what exactly is a swarm robot and why it is advantageous to apply them in our work.

Swarm robotics is a new discipline that aims to apply the natural phenomenon of swarming to robotics. It's a study of robots designed to emulate natural swarms, such as ants and birds, in order to create a scalable, adaptable, and robust system. Self-organization, autonomy, cooperation, and coordination are all demonstrated by these robots. The goal is to keep the cost and design complexity as low as possible, resulting in systems that are extremely similar to natural swarms. The robots are controlled by no central entity, and communication between them might be direct (robot-to-robot) or indirect (robot-to-environment). From mundane home duties to military operations, swarm robots offers a wide range of applications. [3].

Swarms were interesting as robotic systems because they featured very basic components compared to centralized systems developed for the same objective. As a result, the robotic units may theoretically be modularized, mass-produced, replaceable, and possibly disposable. The second main (promised) benefit was reliability: because the swarm was highly redundant in general, it could be designed to withstand a wide range of interruptions (possibly more intense than those

considered in regular control systems); redundancy also allowed the swarm to adapt dynamically to the working environment, which is another feature required for high reliability.It was also feasible to imagine the swarm working as a vast parallel computational system, capable of doing tasks that were previously impossible for other types of robotic systems, such as complicated single robots or centralized groups of robots.Below we gonna elaborate swarm robotics early developments, intelligence and new trends.[4]

2.3 Early Developments (Gestation) In Swarm Robotics And Concepts

G. Beni [5] and Fukuda[6] applied the term 'swarm' in the context of robotics for the first time in 1988. Cellular robotics, according to G. Beni, is a system made up of autonomous robots that function in an n-dimensional cellular area without the need for a central controller. They also communicate seldom amongst themselves, and they coordinate and cooperate to achieve common objectives. Fokuda, on the other hand, use swarms as a collection of robots that can collaborate in the same way as human cells can, allowing them to achieve complex goals. In the context of cellular robotic systems, G. Beni and J. Wang [5] introduced the phrase "swarm intelligence." They claimed that by coordinating their operations, cellular robotic systems can demonstrate "intelligent" behavior.

In 1993, Gregory Dudek et al. [7] defined swarm robotics in terms of size, communication range among swarm robots, communication topology, communication bandwidth, swarm reorganization rate, swarm member capacities, and swarm homo- or heterogeneity. The term'swarm robotics' is a metaphor for multi-robot systems, according to the authors, which is why it was unclear what qualities distinguished it from other robotic systems.

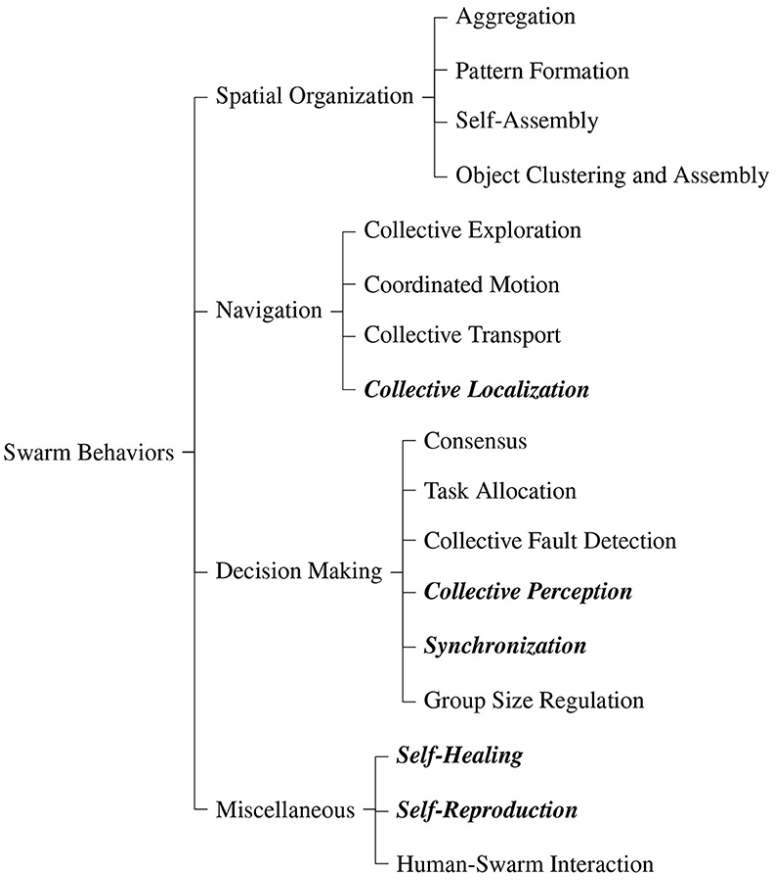
2.3.1 Swarm intelligence

The emergence of swarm intelligence (SI) is a difficult process, and the mechanisms required to assure the establishment of collective intelligence are unknown.
Individual agents in a swarm follow simple rules and act on local information, with no central control [8]. Such rule-based interactions can result in self-organisation, resulting in higher-level structures and traits. Individuals in the system are not

intelligent, but the system as a whole can act intelligently, to the point where it can be deemed collective intelligence. The essential swarming activities are depicted in this growing self-organization.

2.3.2 Basic swarm behaviors for swarm robotics

Individuals in most swarm algorithms follow local rules, and the overall behavior arises spontaneously through the interactions of the swarm's members. In other words, individual robots in the robotics domain exhibit behavior that is based on a local rule set that can range from a simple reactive mapping between sensor inputs and actuator outputs to complex local algorithms. Local behaviors usually include interactions with the physical world, such as the environment and other robots [9]. Each contact entails reading and interpreting sensory data, processing it, and controlling actuators as needed. Basic behavior that is repeated endlessly or until a desired state is obtained is characterized as such a sequence of interactions.



2.1 Basic swarm Behaviors

Spatial Organization-behaviors that control the motion of the swarm robots spatially. Divided into 4 behaviors, Aggregation - control the motion of individual robots to congregate spatially, Pattern formation - moves the swarm to achieve a specific pattern, Self assembly - join the robots with other robots to create structures, Object clustering and assembly - robots will spatially manipulate distributed objects in the environment

Navigation-behaviors that allow the swarm robots to have a coordinated movements in the environment which includes Collective exploration (cooperative navigation of robots), Coordinated motion (moves robots in a formation),Collective transport (enables to collectively move objects), Collective localization (find robots orientation and the position).

Decision Making-behaviors that allow the swarm robots to take a common selection to a given issue that includes Consensus (allows the swarm to converge or agree on to a single choice), Task allocation(dynamically assigns arising tasks to the individuals in swarm), Collective fault detection (determines deficiencies of individual robots),Collective perception (fuses the locally sensed data to form a big picture) ,Synchronization (aligns frequency and phase of oscillators),Group size regulation (allows forming groups)

Miscellaneous-behaviors like Self-healing (recover from faults),Self-reproduction (allow swarm to create new robots or replicate the pattern),Human-swarm interaction (allows humans to control the robots)

The focus of early swarm robotics research remained on the study of swarming behaviors in many species, such as ants, birds, fish, and others. The researchers looked at these tendencies and looked into how they could be used in robotic systems (Cheraghi et al [3]). Various swarming behaviors, such as foraging, flocking, sorting, stigmergy, and cooperation, have been replicated in several studies and researches. The term stigmergy first appeared in the 1990s.

Swarm robots can achieve their aims thanks to two ideas used by social insects: convention and stigmergy. All agents (insects or robots) can be confident that the

other members of the swarm obey the same rules as them thanks to social convention. This promise allows them to take certain things for granted without having to communicate them explicitly Werfel et al [10].

Stigmergy is the act of storing information in the environment in order to communicate indirectly. Termites use stigmergy in their construction by leaving building materials and chemical pheromones in the environment, which might elicit specific responses in other termites who visit the site afterwards. Robots can utilize stigmergy to evaluate if and where to add more material by looking at the local arrangement of the construction material Theraulaz et al [11]. Alternatively, in extended stigmergy, the more sophisticated construction material can be utilized to store extra data, such as a shared coordinate system's location or even directions to a location where more material is required Werfel et al [10].

G. Beni [4] on the other hand, attempted to characterize a swarm more precisely in 2004. He claims that swarm robots are basic, identical, and self-organizing, that the system must be scalable, and that swarm members can only communicate locally. These are the characteristics that are still used to define and differentiate swarm robotic systems from other robotic systems.

According to Dorigo and Sahin's criteria, we regard this multi-robot system to be a robot swarm (2004). These criteria were created to determine the degree to which a study may be classified as "swarm robotic." According to Dorigo & Sahin (2004), the following criteria should be met [12]:

(i) the study should be relevant to the coordination of large numbers of robots;

(ii) the robotic system being studied should be made up of a small number of homogeneous groups of robots, with a large number of robots in each group;

(iii) The robots in the study should struggle to carry out or complete the tasks on their own, and their performance should increase when they collaborate;

(iv) the robots in the study should only have local and restricted sensing and communication capacities.

2.4 Breakthroughs And Trends In Swarm Robots

Foraging, box pushing, clustering, sorting, and formation forming are all common problem domains for the research of swarm-based robotic systems. Many of the characteristics common to collective (or swarm) intelligent systems in general may be found in swarm robotic systems. Robustness, distributedness, adaptability, flexibility, self-organization, and minimalism are some of these characteristics. Another problem domain is collective construction, which has recently sparked renewed interest. Multiple robots assemble or dump building material at discernible and worldwide relevant spatial places in collective construction tasks. Russell et al [12]

A system of cooperative swarm robots has the advantage of operating locally with minimum resources and without the need to comprehend the system's complexity. Because no single failure can make the entire system or task fail, such decentralization of work gives a high level of robustness and flexibility.

With the downsizing of robots equipped with construction capabilities, new opportunities are expected to emerge. Micro or nanorobots, for example, might mend human tissue or build chemicals into minuscule structures. The urge for robot swarms to create buildings that correspond to a design of our choosing is similar to many of these projected uses. [12]

2.4.1 Concepts in collective constructions

Blind Bulldozing

The concept of using swarms of robots to accomplish construction chores isn't new.

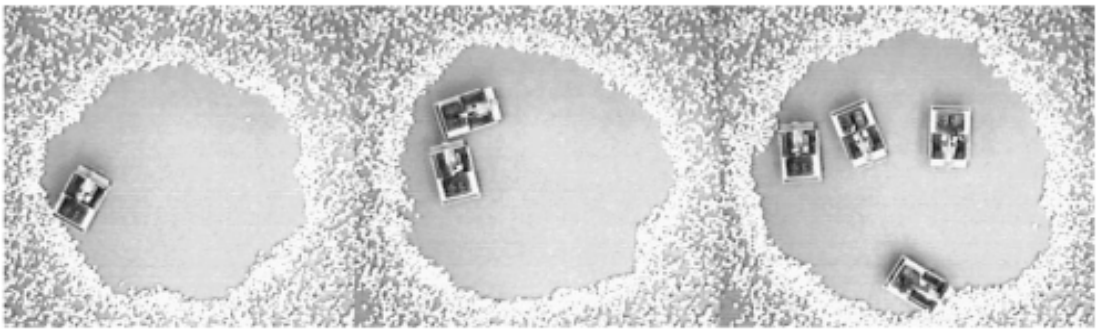


Figure 2.2: Nests constructed by blind bulldozing by one, two and four robots

Brooks et al. proposed in 1990 that using basic guidelines, several robots might level the ground at a lunar construction site Russell et al [12]. However, the viability of

ideas like this has only lately been shown in practice. Parker, Zhang, and Kube [13] demonstrated that simple bulldozer robots could clear rubble to build a "nest" that was roughly circular in shape Figure 2.2

Parker et al [13] uses a blind bulldozing method to construct a nest-like structure by removing rubble on the environment using a robot swarm. The robots in this case are equipped with basic pressure sensors attached to their fronts. Once the task is started, they try to push the material outside of their initial position. The pushing stops once a pressure threshold is detected. With evenly distributed gravel on the ground, a swarm of robots can clear a circular area in this manner.

Collective construction can have two main strategies. Construction by accumulating material and construction by removing material. Blind bulldozing is a proposed swarm robot construction method for the second strategy. A swarm of robots having minimal sensory capabilities can adopt this method. With very simple instructions, the robot swarm manages to construct a structure in this method. The circular nest shape emerges from the simple act of pushing out material until a pressure threshold is measured. This kind of robot swarms can easily display true parallel construction behavior because they have a very simple instruction set.

Leader-based construction

A robot swarm can have a leader to help guide the task that it is intended to do. In nature, some swarm-forming animal behavior can be seen centered around leaders. Ants store their food and keep the eggs close to their leader. Bees create hives centering on their leaders [14]. A similar approach is taken when a leader-based robot swarm is designed [15].

Russell et al. [12] proposed a method that uses moving light fields to direct the creation of walls using swarms Figure 2.3. The swarm-bots follow the light beam emitted by the leader and do the construction at places where the light intensity is at a particular level. With this method, a stationary leader with light can guide a swarm

to construct a wall around the light at a distance where the intensity is at a specific value. The leader can move while swarm robots do the construction and guide the robots to build arbitrary-shaped walls.

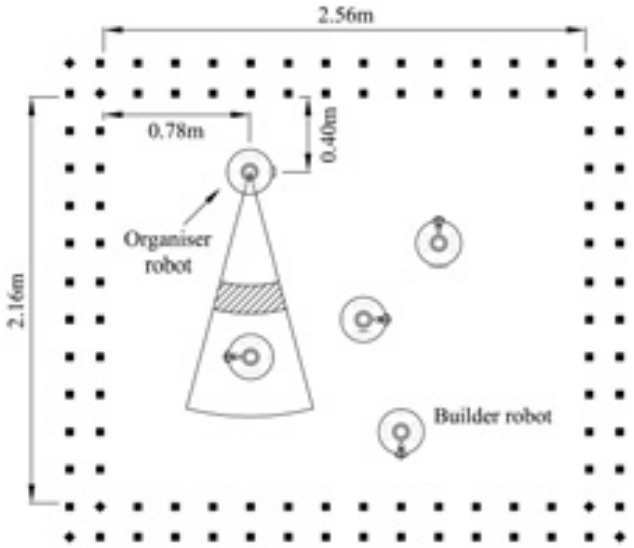


Figure 2.3: Leader robot managing other robots using light source

An extended research on the above by Werfel et al [16] gives a better insight at the two approaches and shows mathematically that using communicative blocks is more efficient. However, in this setup, the robots do not have a communication system among them. Robots figure out the real-time status of the construction using the data broadcasted by the building blocks. Another research by Werfel et al [17] evaluates how a robot swarm can build predefined patterns of blocks using communicating blocks and multiple simple robots.

Rule-based construction

Some insects that show swarm behavior are observed to be following simple rules for building structures. Inspired by wasps like insects, some researches are done with swarms to build coherent nest-like structures. However, with only rules to guide the swarm robots, there are no predetermined shapes fed to the swarm. Instead, the building shape emerges from the rules

Theraulaz et al. [18] shows that such rule-based construction systems can produce

coherent biological-like architectures. Furthermore, they show that the swarm entities need specific coordination properties to produce such architectures.

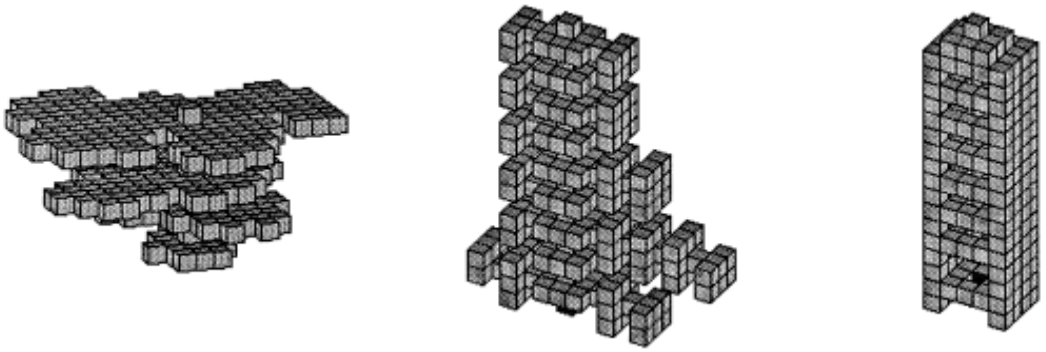


Figure 2.4: Some shapes constructed by rule-based lattice swarms

Pollack et al [19] combine rule-based approach with stigmergic cooperation between the robots to produce more complex constructions. However, this still does not aim at creating pre-determined shapes.

Swarm robotics at Harward [20] shows how a multi-robot construction system inspired by the behavior of termites mound-building capabilities Figure 2.4. They try to understand and mimic the low-level rules that govern the behavior of termites resulting in coordinated construction of very large structures compared to the size of the individual building entities.

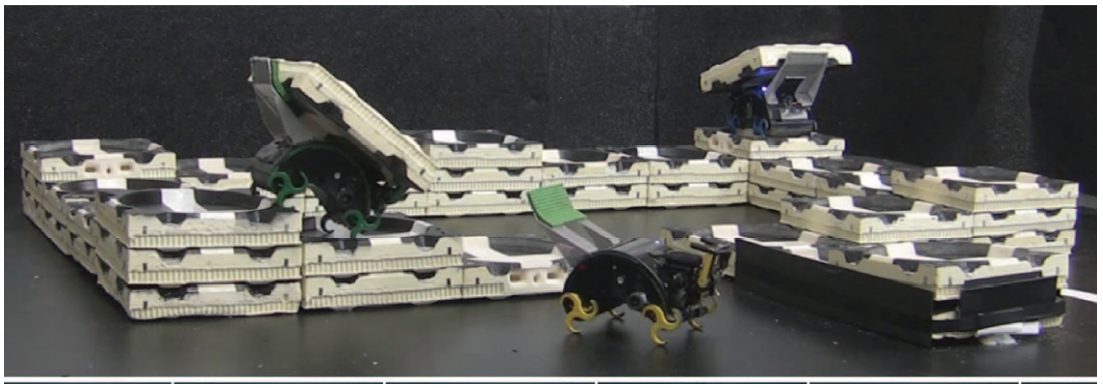


Figure 2.5: Termites robots create 3D structure

Research by Justin Werfel [10] compares and contrasts multiple approaches for collective constructions with robot swarms. It shows that simple algorithmic approaches with robots having only basic senses will result in incomplete constructions. The proposed solution in the paper is having advanced building blocks with programming and communication capabilities. It has demonstrated how such blocks can be used to build 3D structures successfully by a robot swarm.

A more practical approach has been suggested and tested by Stewart et al [12] for wall construction. The research paper has suggested an algorithm based on a distributed feedback mechanism to regulate the construction and the researchers have tested it successfully in the real world using a set of small bots.

Stigmergy Cooperation

Swarms observed in nature coordinate their behavior stigmergically. Communication and coordination between entities of the swarm happen through the changes in the local environment they observe. This is adapted heavily for swarm robotic behavior in various ways [19]. Stigmergic cooperation allows robots in a swarm to have information without heavy communications between robots. This allows simplistic swarm robot designs while having coordination between the robots and the robot task of the swarm.

In a construction task, stigmergy happens from the changes in the structure that is being constructed by the swarm. Small changes one makes in the construction is new information to the other robots when they perceive the change. Then they can modify their next behaviour for the task completion depending on this stigmergic information.

Without direct interactions between the robots, stigmergy makes the construction task independent of each individual entity, which means an increased number of entities only help speed up the construction but not the outcome of the process. This

in turn introduces a massive redundancy to the system since the failure of one/ few entities does not affect the task of the swarm system [19]

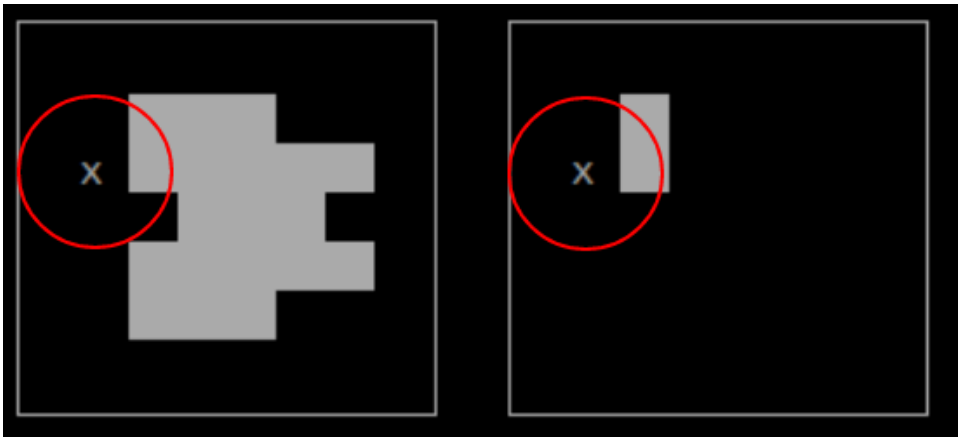


Figure 2.6: Global map of construction (left) and perceived local map (right) of a robot (marked x) at a particular time. The red color circle shows the maximum sensory distance of the robot

The popular Kilobot swarm robots [21] are capable of self-assembling into given 2D shapes. The robots can perceive the immediate environment and decide on where to assemble themselves in the given shape in order for the collective robot swarm to construct the shape. The state of the local environment which gives robots a partial view of the state of the construction is used for the stigmergy in this research

Some researchers have used extended stigmergic techniques by enhancing the perception of nature by various means. Werfer et al [16] has proposed an extended stigmergic method that uses building blocks that have communication capabilities. This allows robots to have a global view of the construction in real-time without directly perceiving the entire construction. This is achieved by enabling the building blocks to communicate with each other and the robots. As building blocks can determine their relative position in the construction, robots can use information from the blocks to determine the real-time state of the construction. However, this is not practical in constructing large structures with hundreds or more building blocks since

the cost can get very high and the failure in building block communication can lead to large errors.

Table 2.1: Tabulation of existing research

Research	**Brief**	**Features**
Blind bulldozing-multiple root nest construction [13]	Constructing a nest like structure with robots with bulldozer like behavior	1. Minimum robot sensory and processing capabilities 2. Building by clearing out material
Controllability Characterizations of Leader-Based Swarm Interactions [15]	Investigation into network topologies for controlling swarms of mobile robots	1. Swarms controlled by humans; commands are given to the leader 2. A networked command and control system for robots
A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm [12]	Construction of walls with a guided leader in the swarm to mark the area where the construction needs to happen	1. Leader guides the swarm using light as a signal 2. Feedback between robots and the leader to understand the state of the construction
Modeling the Collective Building of Complex Architectures in Social Insects with Lattice Swarm [18]	Building 3D structures with swarm robots governed by simple rules	1. Rule based construction 2. No predetermined shape of construction
A Stigmergic Cooperative	A robot swarm architecture for	1. Started from a seed block

Multi-Robot Control Architecture [19]	stigmergically construct 3D shapes initiated from a seed block	2. Limited sensory capabilities 3. No predetermined shape of construction
Construction by Robot Swarms Using Extended Stigmergy [16]	A robot swarm constructing using building blocks that have communication capabilities	1. Building blocks communicate with robots 2. Robot have a real time view of the construction
Kilobot: A low cost robot with scalable operations designed for collective behaviors [22]	A low cost small robot capable of swarm behaviours such as self assembly	1. Very small build 2. Simple algorithmic approach for self assembly
Designing Collective Behavior in a Termite-Inspired Robot Construction Team [20]	Construct structures by pre determining control rules for the swarm depending on the shape of the construction	1. A compiler creates instructions for robots prior to the construction 2. Robots use local sensing to coordinate activities to a limited extent
Optimizing Robotic Swarm Based Construction Tasks[23]	Constructing a given 2D shape by placing building blocks in parallel manner using decentralized swarm robots	1. Decentralized swarms having local view of the environment 2. Parallel behavior with stigmergy based communication

2.5 Off-Earth Constructions

Space exploration is a key area that is considering using robotic swarms [24] (Rouff et al, 2006) for various tasks since it is efficient, redundant with reduced cost and risk. And also it is easier to get a large number of small items into space than sending a small number of large machinery. Thangavelautham et al [25] described that sending autonomous robots instead of astronauts can reduce the cost by 50% for lunar missions.

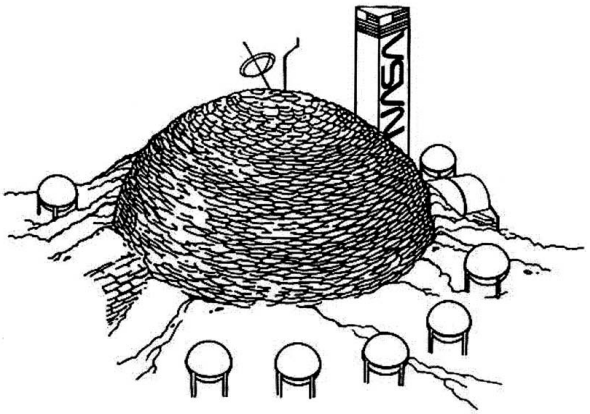


Figure 2.7: Structure proposed for off-earth habitats

There are many proposals for colonizing planets and moons which require building large structures on them to mitigate damages that can happen to living beings due to solar/cosmic radiation, meteoroid impacts, and extreme temperature variations (-150C to 100C on the Moon and -153C to 20C on Mars). Ruess et al [1] proposes to have a 2.5m regolith layer on top of the habitation to remove those hazards Figure 2.8. The strength of this structure should be high because the structure weighs about 5,100 kg/m2 on the Moon since the lunar regolith is about 1.7 g/cm3 [1].

Figure 2.8: Adding regolith layer top of the habitat module

Wilkinson et al [26] proposes a method to create a regolith layer over the habitat module by using heterogeneous robot swarms to collect regolith, place them over the habitat module and sinter them Figure 2.8. The method proposed by Irawan et al [27] has modular heterogeneous swarm robots that can change their configuration to collect regolith, sinter them, and 3D print to create structures. This proposed method comes with number of challenges and problems. As heterogenous swarm robots are used separately for various tasks the algorithm and mechanism to built them is complex which requires lots of energy. And even if one robot breaks down the whole system gets affected as each robot is assigned a separate task and the work is done collectively.

Figure 2.9. Khoshnevis et al [28] describe the way of using sulfur concrete to make structures using 3D printing Figure 2.10. Since sulfur is abundant on the Moon and Mars the raw materials can be found easily. To built structures using 3D printing heavy machinery is required and the maintenance of it and transportation of the machine is a huge issue.

And none of the above mentioned researches have presented a specific algorithm yet they only suggest the idea. But in our work excavation is suggested for extraterrestrial construction and the algorithm to built habitations is provided.

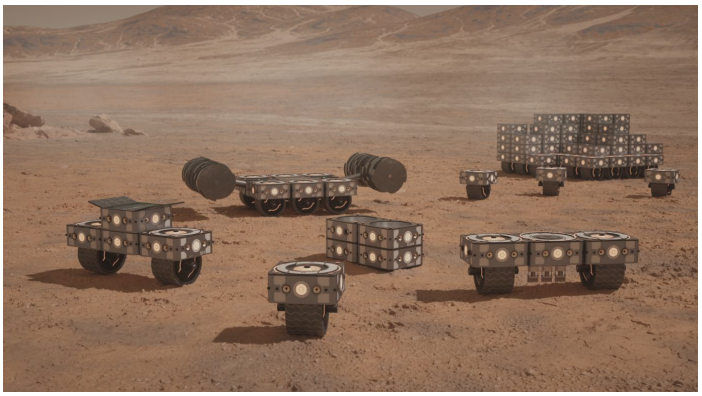


Figure 2.9: Modular heterogeneous swarm robots that can form and 3D print

Not even a single paper has been written on off-earth excavations, Yet Artificial Neural Tissue (ANT) based centralized bulldozing multi-robot application is proposed by Thangavelautham et al [25]. After training the system these robots will be able to do the preparation of the lunar land using bulldozing to a given plan. But here the robots do not try to build a habitations instead they just level the lunar ground.

Even though making habitat underground is a valid solution, most of the researchers did not focus on it because grading and excavation are difficult and expensive due to the requirement of heavy machinery and lack of traction and locking nature of the regolith [1].

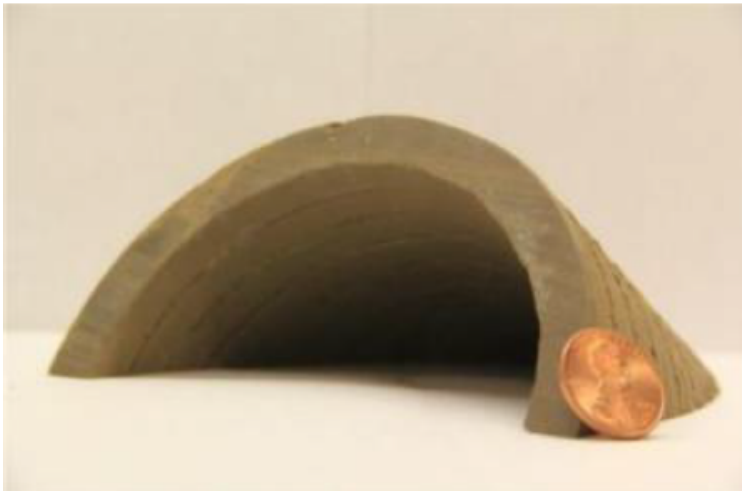


Figure 2.10: Structure made using sulfur concrete.

Recent researches have come up with different excavation techniques and machines that can be used on Mars and Lunar applications. Just et al [29] reviewed existing regolith excavation techniques. RASSOR is a robot platform that is capable of doing excavations on low gravity planets Figure 2.11.

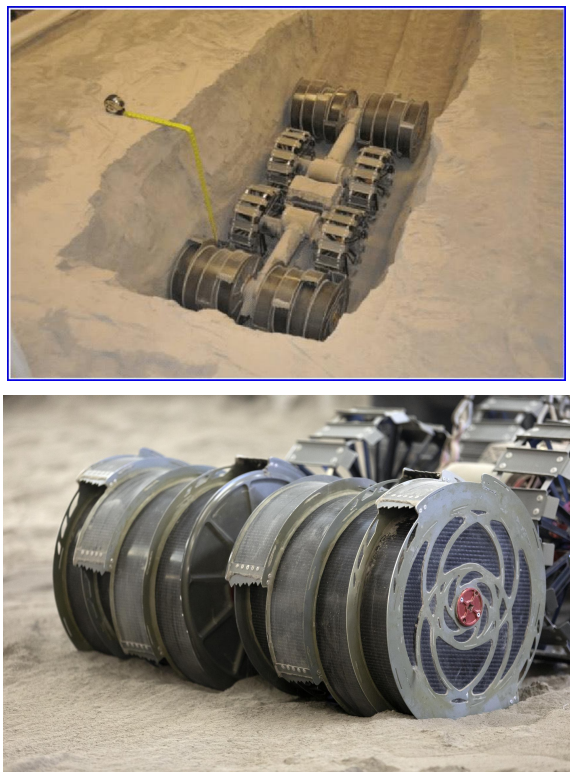


Figure 2.11: NASA's RASSOR robot

Swarm robotics based underground habitation approach is primarily identified by Bier et al [2]. In their method, they use both subtractive and additive methods to address tunneling and reinforcement problems respectively. After the structure is excavated by tunneling and reinforcing the remaining regolith needs to be removed.

The chart below depicts various ways of planting extra terrestrial habitations and different technologies that can be adopted for the process of construction.

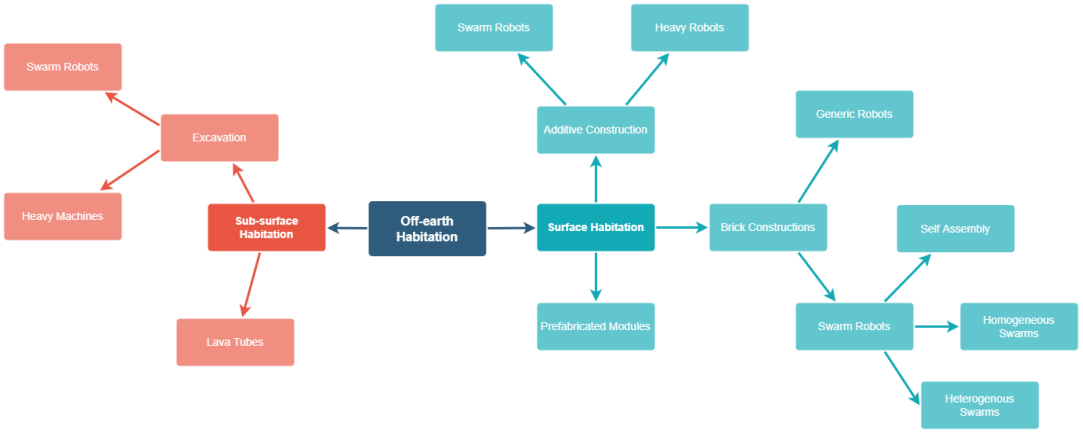


Figure 2.12: Various types of off-earth construction

2.6 Challenges In Swarm Robot Based Constructions

A number of variables make collaborative construction particularly difficult. The constraints of mobile robots, the complexity of a swarm, and the challenge of global-to-local compilation are among them. The limits that mobile robots encounter are especially important for robots that are designed to be basic and disposable [10]:

• Localization—In general, robots have a hard time figuring out where they are in any global coordinate system. GPS is costly, not available in all locations, and not always accurate even when it is. Odometry (trying to predict position by integrating estimated velocity) is incredibly inaccurate, especially in clumsy environments like construction sites, where wheels are prone to slipping and other disruptions are common. However, if a swarm of robots is attempting to construct a single structure, they must all agree on a same coordinate system, or their efforts will be incompatible.

• Communication—For mobile robots, establishing and sustaining ad-hoc communication networks is an open research subject. As robots travel, the network topology changes; messages may be lost, individual robots may lose touch with the rest of the network, or the network may break into several isolated pieces.

• Manipulation—Another key open study topic for robots is manipulating physical items. In controlled environments such as factories, the environment can be managed

to the point where robots only meet a few known circumstances; but, in real-world, uncontrolled contexts, even advanced robots struggle to manipulate objects. Of course, exact alignment of building components is likely to be essential for construction tasks.

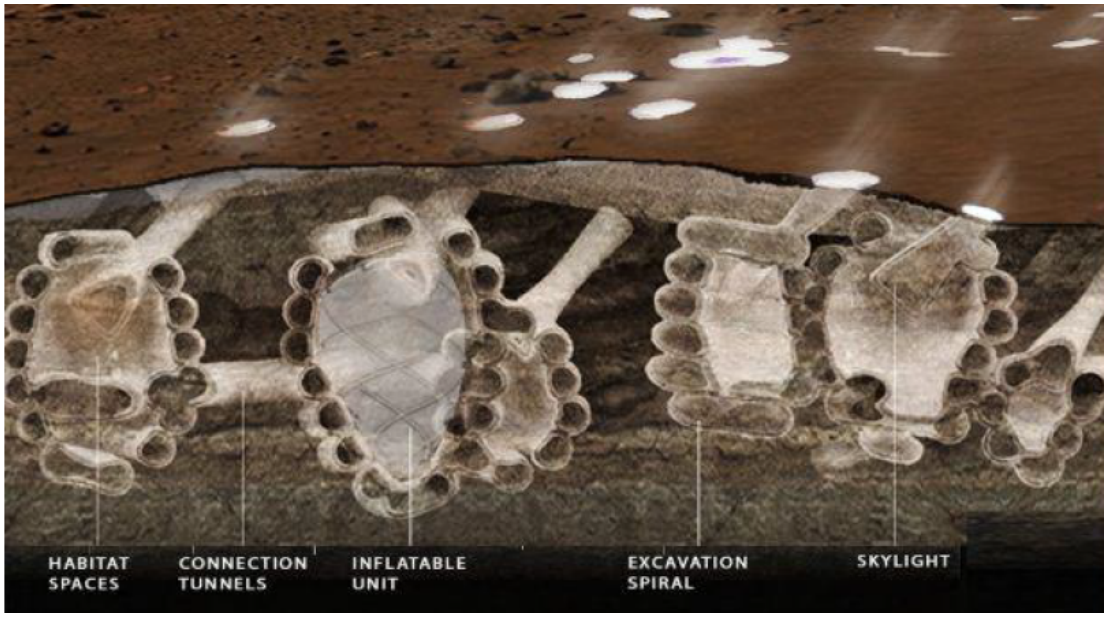


Figure 2.13: Swarm robot based underground structure proposed in 2021 by Bier et al

In off-earth constructions, sending high volume machines will cost you 50% more than sending small machines. Building constructions on top of the surface of Mars/Lunar need in situ resource utilization (ISRU), here concrete and required raw materials will be generated by the machinery we sent there. It is again a challenge because the regolith composition can vary from one location to another.

Having underground caves for habitation will help you to reduce the building materials needed. Bier et al [2] proposed a method that uses swarm robots to excavate the ground in a spiral way Figure 2.13. After the swarms make a tunnel structure underground a separate mechanism can be used to remove the remaining regolith from the structure. That paper does not discuss how the robots are controlled, algorithm, or how to remove the remaining regolith and again its a

complex mechanism that requires different kinds of robots with different algorithms which is expensive and time consuming. The chart below tabulates all the problems and advantages of the researches we discussed above.

Table 2.2: Summary of benefits and challenges

Research	Benefits	Challenges
Ruess et al [1]	The proposed 2.5m Structure will protect living beings from hazards.	Sending bigger machines to construct these structure is difficult
Wilkinson et al [26]	Creating a regolith layer over the habitat module is easier than building a construction	Heterogeneous swarms is needed with mechanisms for sintering regolith
Irawan et al [27]	Robots will complete the structure themselves without needing of separate stations	Heavy communication is needed for reconfigurable robots. Redundancy is low. Heterogeneous swarms.
Khoshnevis et al [28]	System will create sulfur concrete and 3D print the structure. Sulfur is freely available on Mars/Lunar	Heavy machinery is needed. Low redundancy. High transportation cost.
Thangavelautham et al [25]	Those swarm robots will learn the optimum behavior and do construction.	Centralized localization is needed. Scaling up the structure will increase the time exponentially.
Bier et al [2]	Underground structure is designed using swarm robots which will give natural protection and good temperature stability	After excavating the spiral structure, a separate mechanism is needed for removing the remaining regolith.

2.7 Problem Definition

To make the surface of the Moon/Mars habitable, the living area should be covered from at least a 2.5m regolith layer to mitigate hazards such as solar/cosmic radiation, sand storms, and high-temperature variation (-150C to 100C). As discussed above many researches are focused on this area where they propose different methods to build structures that will help living beings to be placed. But these proposed solutions are expensive, inefficient, and not practical because of the need of having heavy machines, established communication, and the requirement of global view.
Therefore, building a robust structure that separates living beings and the planet's surface before sending astronauts has been identified as a research problem.

2.8 Summary

This chapter presented a critical review of literature in Swarm robotics and off-earth constructions. Here the research problem has been defined as, how to build a robust structure on the planet's surface to create a hazard-free environment for living beings. In the next chapter, we give a detailed discussion on the technology adopted for implementing swarm robots.

CHAPTER 3-TECHNOLOGIES USED FOR SWARM BASED EXCAVATION

3.1 Introduction

In chapter 2, we presented a comprehensive critical review of literature on the use of Swarm robotics and off earth construction methods and defined our research problem as, how to build a robust structure on Mars/Lunar planet surface to create a hazard free environment for living beings. In order to solve this problem, we have recognized the need for research into developing swarm robotic excavation systems. This chapter describes the design of swarm robots, technologies used, and their application used in this work.

3.2 Swarm Robot Design

Swarm robots are designed to have low cost hardware which is readily available at the time of doing this research. A small robot in a swarm can have a single board computer running with its controlling software and a set of low cost sensors.

The swarm robots have sensory inputs such that they can perceive their immediate environment. This limits their visibility to a local area and thus not having a global view of the real time construction which is the case for a real world robot. Robots are assumed to be able to identify other robots (for collision avoidance while navigation) and identify the environment depicted using blocks that can be excavated. It is practical to initially allocate the dumping locations for the robots (ISRU locations) since they are stationary.

With the current technology, pheromone trails are not practical to be used by robots. Though we can implement communication and localization capabilities through building blocks in swarm construction tasks, we can't do the same with excavation.

In the real world, it is safe to assume that the inter robot communication has a limited range. Therefore, having a global view of the real time construction is not possible.

3.3 Technologies Used In Swarm Robot Implementation

Swarm robot systems can be implemented easily due to their decentralized features. In off-earth situations, humans cannot establish a communication network that would let robots communicate with each other and this may lead to failures in robots. So a loosely coupled system should be used with a lot of redundancy Werfel et al [10]. Stigmergy helps us in such a situation because robots will communicate through the environment or building materials, this is kind of a simulation of pheromone used by insects Theraulaz et al [11]. Furthermore, the robots must be implemented to work collectively and abide by the rules.

Here, to simplify the excavation problem, this research ground is considered a set of blocks, and excavation is depicted using the removal of those blocks.

3.3.1 Stigmergic cooperation

In nature, animals that show swarm behavior communicate using changes they do to the environment. Changes one entity of the swarm does is observed by others and that information helps them adjust/ reorganize their tasks.

3.3.1.1 Stigmergy from the local view of the construction

The sensory view of robots is information for the robot to determine the state of the construction. Even though this makes the robots have a partial view of the construction, it is practical in the real world to have such a view instead of a global view [30].A robot can not have a global up-to-date map in this manner. Therefore, a local map is maintained by each robot. As the robot explores new areas, the map is

updated.

When the robot needs to excavate, it determines the excavating location based on the data it has on the environment which is its local map. However, once the determined place is reached, it might be already removed by another robot since the robots are working independently. As the local map gets updated with this new information, now the robot can determine a new place to excavate.

3.3.1.2 Stigmergy from the environment

When multiple robots work together in the environment, they can cross each other's paths. Practically, the robots should not collide with each other since it can be damaging the hardware of the robot.

Once a robot in the simulation sees another robot in its vicinity, the new robot is perceived as an obstacle in the map. With the robot's capability of differentiating between blocks and robots, it should know that this obstacle is not a permanent one. This information is used to not include the detection in the local map for calculating the next block excavation.

Furthermore, a path planned to excavate can be obstructed when the robot returns to a particular place since it has previously been there. The excavation by other swarm bots has been going on while the robot was gone to drop the block.

When a robot sees another robot obstructing its path, or when the planned path is obstructed by the latest building blocks placed, a new path is planned again using the local map. This time, the local map is updated with the new data before planning the path and hence it will avoid the newly identified obstacles.

3.3.2 Collective construction

Collective construction can have two main strategies. Construction by accumulating material and construction by removing material. In our problem we need the second strategy. As blindbuldozing Parker et al [13] removes items from the floor, our robots should remove regolith until the final structure emerges. However other construction methods can be identified as an inverse problem of excavating. Termite robots in Harward could construct a 3D structure using thin blocks Werfel et al [20], such that each robot can travel over one block up or down. Even though this method lets the robots move up and down through the structure, practical structure development is difficult with this method. But using the same method for excavation can help us to have a practical cave.

3.3.3 Rule based controllers

Inspired by wasps and similar insects, underground excavation can be done with excavation entities having simple rule sets to determine how to excavate blocks relative to the environment and the current state of the construction. In such systems, the final shape of the construction is an emergent feature of the rules the swarm robots have. Such a system does not need to possess large computational capabilities or communication capabilities between robots. No global view of the final outcome is needed for the robots to do the construction either.

With the added stigmergy, rule based controllers can be improved. The rules are introduced to the swarm robots so that,

1. They avoid deadlocks where two or more robots can get stuck, being unable to move in their preferred path.
2. Do not harm each other by colliding while in operation
3. Update tasks based on the new information from the environment

3.4 Application Of Swarm Robots

Amongst many applications of Swarm robots including monitoring an environment, establishing communication network, search for objects, in our work we apply them for excavation. Excavating swarm is not a popular research area, but few researches have happened including the blindbuldozing Parker et al [13]. Artificial Neural Tissue (ANT) based centralized bulldozing multi robot application that is proposed by Thangavelautham et al [25] is also an excavating swarm use case.

3.5 Summary

This chapter presented a brief introduction to swarm robots design and their purpose. Furthermore,a detailed discussion on the technology adopted for implementing swarm robots and the application of swarm robots to our work as excavating swarms are given. In the next chapter we will discuss the approach taken to meet the challenges of our problem of building extraterrestrial habitations.

CHAPTER 4-APPROACH

4.1 Introduction

In chapter 3, we discussed the technologies that can be used to solve our research problem of having habitations in off-earth satellites. We have come up with a swarm robot approach to address the above problem which is named as CCSR, an acronym for **C**ave **C**onstruction using **S**warm **R**obots. This new approach proposes to have an underground cavity to have human habitat which is constructed using swarm robots. This approach is inspired by numerous collective construction applications of swarm robots. Underground habitation has the advantage of natural protection from radiation and storms while also being less affected by thermal stresses because the temperature is more stable underground [11]. Expansion of the cave system is easier and less expensive than structures on the surface. The rest of the chapter has been structured with subheadings hypothesis, input/output, process, and features.

4.2 Hypothesis

The hypothesis of this research is that the swarm robots can be used in an optimized simple way to construct an underground cave using less power, resources, and time compared to the existing approaches.

4.3 Input

There are four main inputs required for a single robot which are surrounding data including other robots and regolith layers in the vicinity, shape map, and the localization data of the robot. Surrounding data can be extracted using the vision based sensor(s) of the robot. A shape map that contains the data of the final construction is uploaded to the robots at the beginning.

Shape map is a 3D grid that contains the final excavated shape. Layers that need to be removed are marked in the grid with a weight value corresponding to the height of the layer.Shape map is designed in a way that all the locations of the excavation can be traversed by robots with its movement constraints.

4.4 Output

Output of the system comes in the form of action performed by the robots. There are three basic actions that each robot can perform. They are, traversing around the surface or to a layer one step below or above to the current layer, remove the regolith from the layer it is on or from the front layer which is one step higher than the current one and dump regolith. After the process is finished simulated underground structure to the given shape map is developed.

4.5 Process

In order to get actions from the robots we have developed a model which has two parts, namely, localization and motion planer. In this research problem we suppose that localization data is available for each robot, This data is directly used by the motion planner with surrounding data to plan the next movement. Plan of the final construction, which is called a shape map, is designed such that a robot can travel from the bottom to the top easily.

All the robots can do three simple actions, traverse through the environment, remove bricks and dump them in a specific location.

- Robots can only traverse around the same surface or to a layer one step below or above the current layer
- A robot can remove bricks from the layer it's on or from the front layer which is one step higher than the one where the robot is on
- In the excavation process, robots needs to go back to the starting position after one brick is removed to excavate again

Collective behavior and communication of the swarm system are achieved using stigmergy based on the passive environment. The system is completely decentralized. No robot to robot direct communication. Dijkstra's shortest path algorithm is used for path planning

- Each robot has its own local map which gets updated as it moves around.
- Using the given shape map and its own local map, a robot will select a brick that is nearest to it and will traverse to that bricks location

- If the path is obstructed with another robot or changed due to other excavations, the robot will replan the path using his updated local map
- If the brick is already removed, the robot will select another brick using the shape map and will go to it
- After a brick is excavated robot will plan the path and will move to its starting location
- If the path cannot be calculated to the given location, robot will plan the path to a near location which will help him update it's map.
- If a robot is moving back and forth in the same location or stuck in the same location, it will change the destination to the starting position after a random no of tries to mitigate deadlocks
- Even remaining one robot can finish the excavation. Number of robots will speed up the process

4.6 Users

Users of this proposed solution can be identified as organizations/governments that are conducting and funding space missions. And also this solution can be applied to on earth applications too which will help the mining and manufacturing industry.

4.7 Features

Features can be seen from this solution are low cost implementation due to use of small robots rather than heavy robots, resources can be collected from the excavated regolith, expandability of the structure and also basic swarm features that are identified as low cost, adaptability, scalability and redundancy.

4.8 Summary

In this chapter we discussed the approach of our CCSR solution by identifying inputs/outputs, process, users and features. Each module in this process part is discussed later in the Design chapter.

CHAPTER 5 - DESIGN

5.1 Introduction

In the previous chapter the approach for this research is described. In the chapter before, a brief introduction to swarm robot design and a detailed description of the technological concepts that are adapted for this research are discussed. This chapter extensively dives into the design of the research implementation using the discussed technologies and the approach.

Chapter 4 presented our approach CCSR (Cave Construction using Swarm Robotics). This chapter presents the top level architecture of CCSR. Our swarm control system consists of three main modules, namely, Localization, Motion planner, and Shape map. The rest of the chapter describes each module in detail in separate sections. Top level architecture of CCSR is shown in Figure 5.1.

5.2 High Level Architecture

The simulation environment consists of swarm robots, dumping areas and the environment consists of blocks that can be removed by robots as mentioned in the previous section. The modules in the system are connected as follows

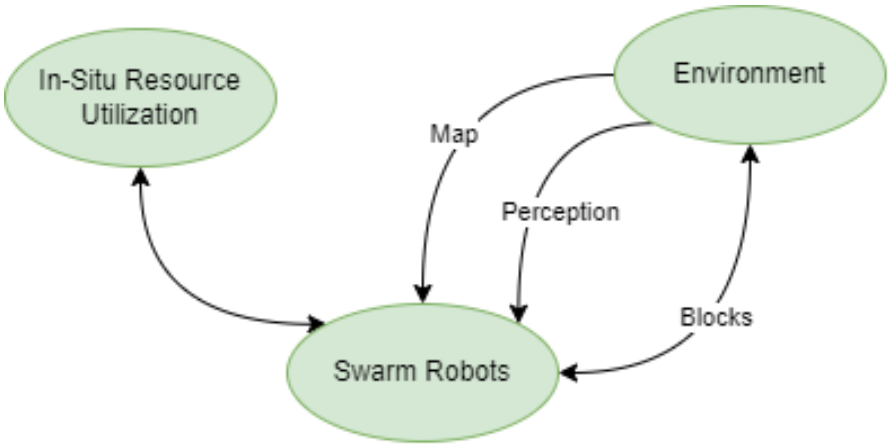


Figure 5.1- Top level architecture of CCSR

1. **Swarm robot**

   Swarm robots will have sensory inputs such that they can perceive their immediate environment as described in the previous sections. This limits their visibility to a local area and thus not having a global view of the real time construction which is the case for a real world robot.

2. **Environment**

   To simplify the excavation problem, the ground is depicted in the simulation using layers of blocks. Each block can be removed by swarm robots. Environment contains all the modules and enables perception for the robot sensors in the simulation environment

3. **In-Situ Resource Utilization Stations (ISRU)**

   Robots need to dump the excavated blocks to these stations before starting excavation again. ISRU stations can collect, process and store valuable materials found in the other planets.

5.2.1 Localization

Theoretically this module extracts information from orientation sensor, visual data and shape map to calculate the position of the robot according to the given shape map. Localization is very much needed in any robotics application to identify where the robots are and plan the next steps according to it. In this project localization data is given to each robot by the simulation environment since this project is more focused on the swarm behaviors. Localization data will be directly used by the Motion planner in CCSR.

5.2.2 Motion planner

Motion planner decides the actions that the robot should take in each step. It uses visual data, localization data and shape map to decide next movements. Basic actions that robots can do are identified as, traversing around the surface or to a layer one step below or above to the current layer, remove the regolith from the layer it is on or

from the front layer which is one step higher than the current one and dump regolith. Path planning is also a part of this motion planner A* algorithm is used to do path planning for swarm robots.

5.2.3 Shape map

This is the 3D map which has the complete construction plan of the intended structure. This map consists of blocks representing regolith on Mars/Lunar. Each block is given priority value at the beginning which can be used to decide which block needs to be removed first or last. After the construction site is selected, the design of the structure is planned in a layer by layer way so that
any bottom position of the cavity can be accessed by robots. Refer to the Figure 5.2

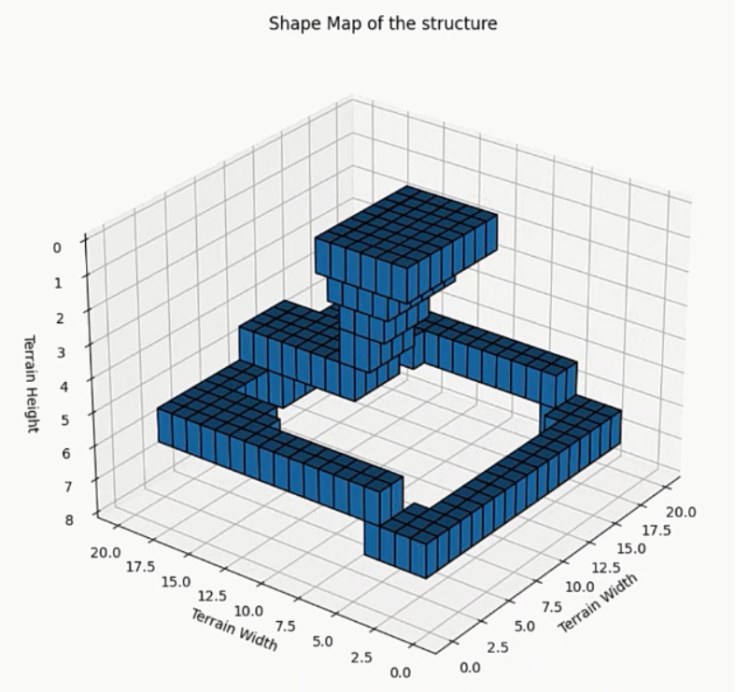


Figure 5.2 - Example of a shape map

5.3 Design of swarm robots

In order to test a construction robot task, a swarm of robots is needed with defined

capabilities and limitations. Due to the practical limitations, the research will be done using a simulation ensuring a minimal reality gap. In the scope of this research, we assume a bot in the swarm to have specific abilities and limitations which are practical for small low cost robots with the current technologies

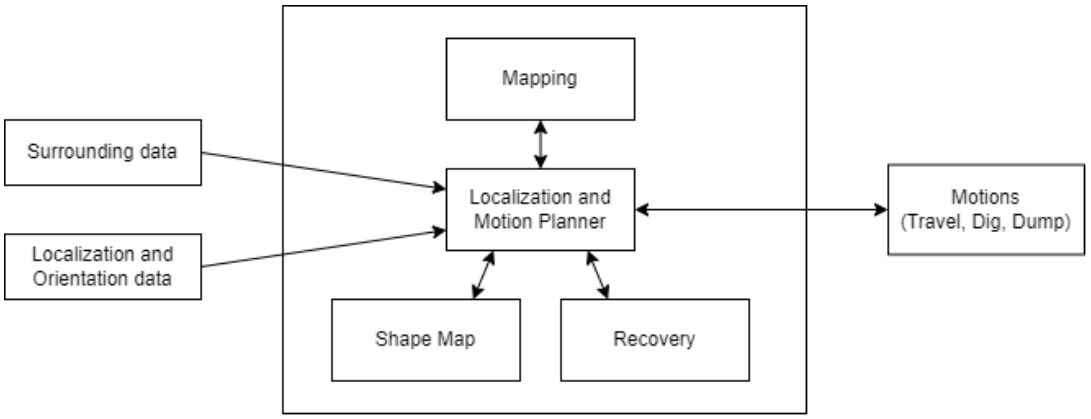


Figure 5.3: Swarm bot inputs and outputs

With the existing technologies, a swarm robot created for an excavation task will be a small unit capable of self navigation while avoiding obstacles including other swarm bots. It will be capable of detecting, picking up, carrying and placing a block. The bot will have a simple communication method to exchange information with other robots in its neighborhood. There will be a GPS receiver on board each swarm robot so that the robots know their location on the global map. These can be listed as follows

1. Able to carry one building block at a time
2. Able to navigate avoiding other robots and obstacle
3. Able to move one step up and one step down
4. Have a GPS receiver enabling to know one's location on global map
5. Can identify and differentiate objects in its vicinity (blocks/ bots)

Given a shape to excavate, the robots in the swarm basically have to remove blocks in the ground according to the shape. The challenge in this research is to make the

robots remove the blocks in an optimized pattern making the excavation time minimal.

Removed regolith can be used to extract minerals and important resources. The task of the swarm of the robots will be to excavate a predefined structure inside the ground on a predefined location. Furthermore, the structure might change from one construction site to another.

5.4 Simulation design

The 3D simulation is designed to simulate the evolution of the robot swarms in a computationally optimized way, making the progress faster. The environment will basically consists of the terrain with few block layers, swarm bots and ISRU stations
The simulation environment implementation is straightforward. Environment keeps track of all the robots and blocks. When a robot asks about a position in the environment, it replies with the information of the occupancy of the position. (Occupied by a block, a bot...etc)

As the excavation goes on, the progress is monitored by the environment. After the construction is finished the simulation will end.The grid environment will help robots map the terrain and determine the block positions for the shape that needs to be build

5.5 Summary

This chapter described the design of the simulation environment for testing our hypothesis. The architectural designs and mathematical details were discussed extensively. In the next chapter we go into the details on the implementation of this design.

CHAPTER 6 - IMPLEMENTATION

6.1 Introduction

Implementing the simulation for testing the hypothesis is described in this chapter. We go into the details of the concepts and mathematics used for implementing the proposed design in this chapter

6.2 Simulation Framework

There are many frameworks present for 3D environment simulations. URSINA[31] is an open source python based game engine which can be used to design 3D games and visualizations. This engine has all the built in shapes and functionality to design our swarm based simulation.

6.3 Simulation Environment

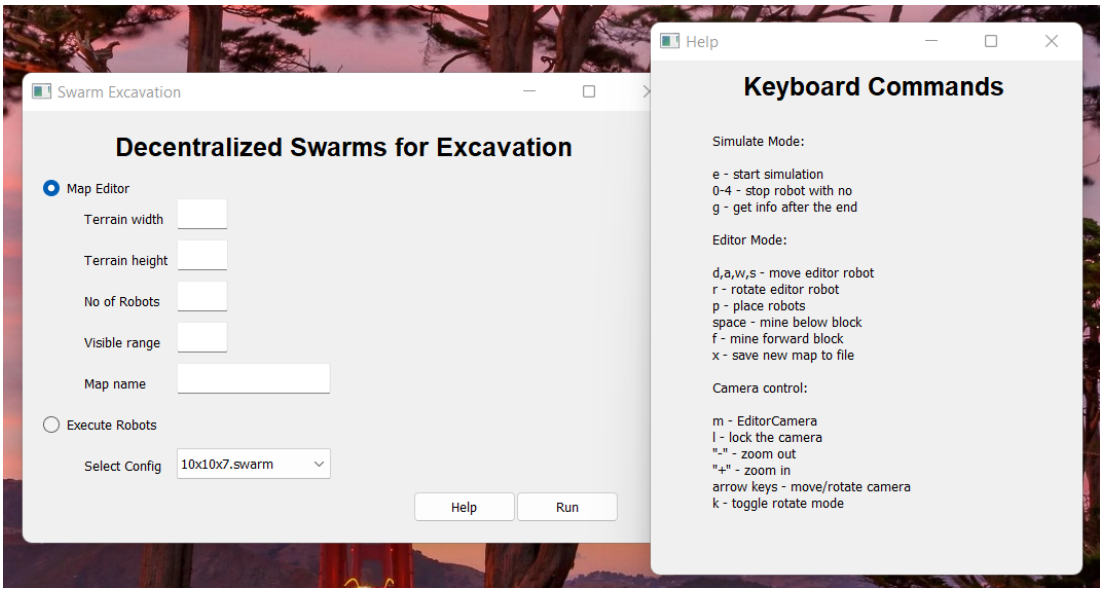


Figure 6.1a: User interface of the Simulator.

Simulation environment is controlled using a graphical user interface implemented using Qt5[32]. This interface will let users create new shape maps on a given sized environment or execute swarm based excavation using an already created map.

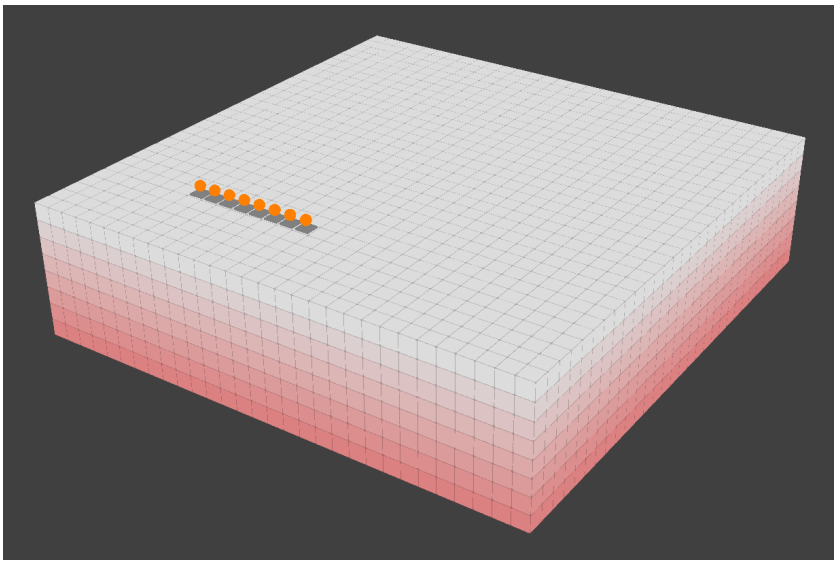


Figure 6.1b: This is a view of the simulation environment. Here 30x30x8 block terrain is used with 8 robots. Gray area is the starting position of those robots. Orange spheres show robots.

The terrain is represented by a 3D grid of size $W \times L \times H$ where W is the width, L is the length and H is the height in simulation environment $units$. In the grid, we define a building block size of $1 \times 1 \times 1\, units$. The terrain has a height(H) where it defines how many layers of blocks it has.

6.4 Robot Controller

6.4.1 Navigation

The robot work environment is the 3D grid terrain. Robots can move on top of the terrain and one step up and down between layers. Each robot keeps it's own map which updates with environment data.

Robots need to navigate to and fro dumping locations and the excavation site. At the beginning, each robot finds the nearest excavation location and goes there to excavate. After the excavation is done robots need to move back to the starting dump location to dump the excavated regolith. Dumping locations are fixed and each robots dumping location is the starting position.
Robots use Dijkstra path planning for planning the navigation path

6.4.2 Sensor data processing

Robots perceive only their immediate environment by their sensors. This data is used to update each robot's local map at each step of the simulation. Until new information comes on a particular location on a map, the old data is used for calculations

6.4.3 Calculating the next excavation location

The most critical task for a successful excavation is the systematic order of block removal of the shape for excavation. We have created an algorithm where robots try to remove the blocks giving priority to the top layers, which will keep robots accessing all the required blocks. The steps of this algorithm can be broken down as follows;

1. Set next excavation position to the nearest excavatable location to the robot

   From the shape map robot will check available excavatable blocks considering the local updated map and the height. Highest layer is selected first and then checks whether there are any excavatable blocks. If any, Euclidean distance

to each block is calculated from the robot using the XY plane. Nearest block is selected and using the local map, Dijkstra's algorithm is used to find the shortest path to the block. Then execute the path step by step

2. In each step of the movement, the robot will observe the surrounding blocks and update its local map which saves the occupation of each block and other robots.

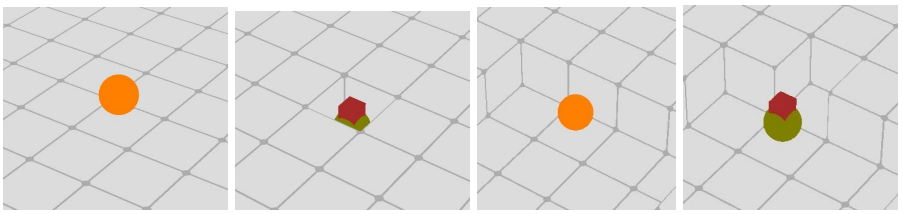
   This encourages bots to spatially spread through the excavation site based on the dumping locations. Also height priority based excavation promotes efficient local map update since robots will move and update the full map before moving into the next layer.

3. If the path is obstructed with another robot or changed due to other excavations, the robot will replan the path using his updated local map

4. If the selected block is already excavated after coming to the block location, redo the 1st step again using the current location and select another block to excavate.

   This will happen when another robot has already excavated the selected block by the robot. Here the robot's current location is used rather than the dumping location used when the algo runs after the dumping.

5. After coming to the excavation location, the robot will remove the block using it's two methods which are,
   - A robot can remove bricks from the layer it's on
   - Or from the front layer which is one step higher than the one where the robot is on



(a) (b) (c) (d)

Figure 6.2: (a) and (b) shows a robot removing bricks from the layer its on and (c) and (d) shows a robot removing the front layer thats one step higher than its position

6. If the path cannot be calculated from the given location to the intended location, the robot will plan the path to a near location to its destination which will help him update it's map.

7. If a robot is moving back and forth in the same location or stuck in the same location, it will change the destination to the starting position after a random no of tries to mitigate deadlocks

6.4.4 Control algorithm

```
While not all bots finished:
      For each bot:
            Update local_map with sensor data
            If no block:
                  execute next step from the movement_steps
            Elif path blocked:
                  Reroute
                  If no path can be calculated to the
destination:
                        calculate path to the start position
                  add path to the movement_steps
            Elif build_shape finished:
                  Set bot finished
```

```
            Elif deadlock_condition:
                  Set goal to start_position after random time
            Elif reached excavate location:
                  excavate block
                  Calculate path to start
                  add path to the movement_steps
            Elif reached start position:
                  Calculate path to the nearest excavate
location
                  If no path can be calculated to the
destination:
                        calculate path to a near destination
point
                  add path to the movement_steps
```

6.5 Simulation Outcome

Here a simulation of the swarm system is demonstrated further describing how the swarm concepts and the implemented control algorithm works to create a given underground shape. In this example 5 robots are used to do the excavation in a 15x15x7 brick terrain. Each robot has 2 bricks in the vicinity range.

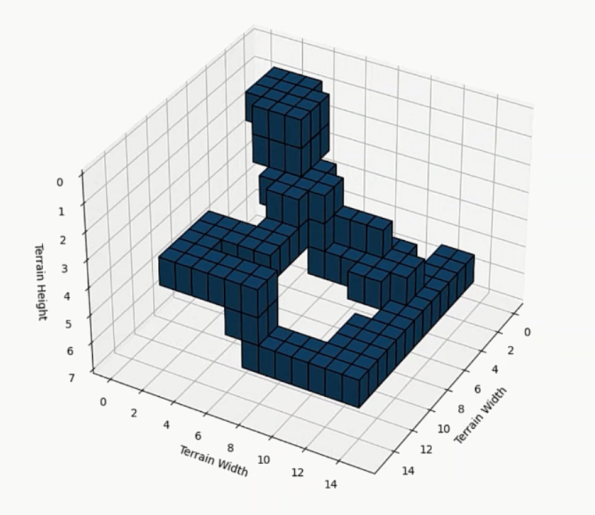


Figure 6.3: Shape Map of the structure

1. Start position of the each robot is depicted using gray pads. Free robots can be identified as orange color spheres and excavated ones as olive colored spheres with box on top of it.

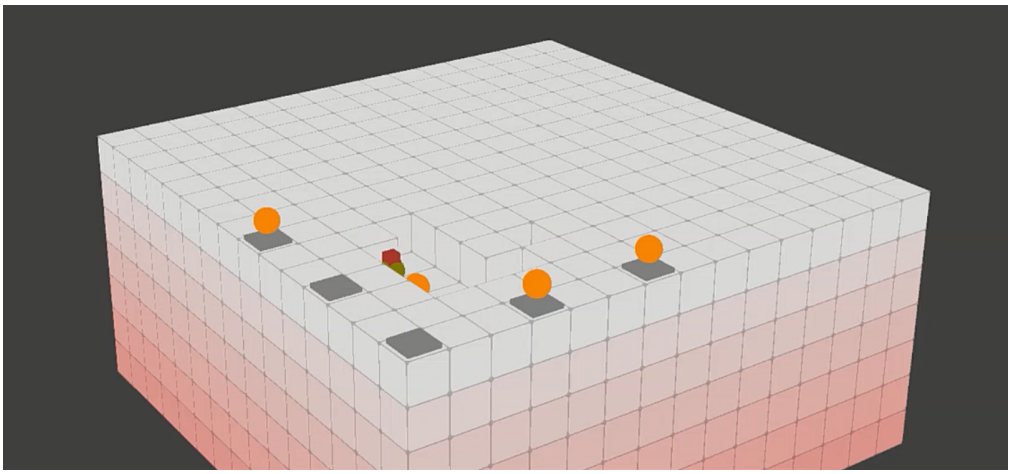


Figure 6.4: After excavating few blocks

2. After removing a few blocks. To visualization purposes the bricks that are above the bricks that are removed are made transparent.

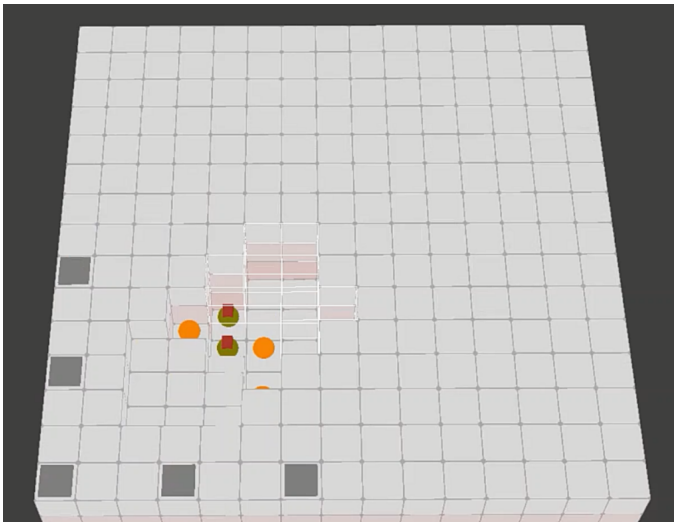


Figure 6.5:bricks that are above excavated bricks are made transparent for easy visualisation

3. After 80% of the structure is excavated

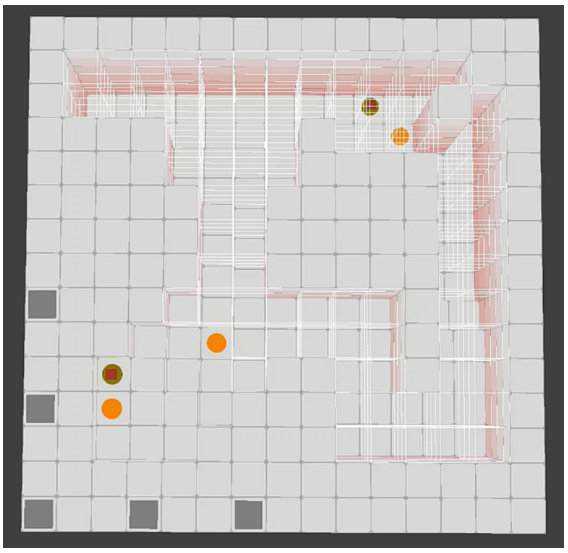
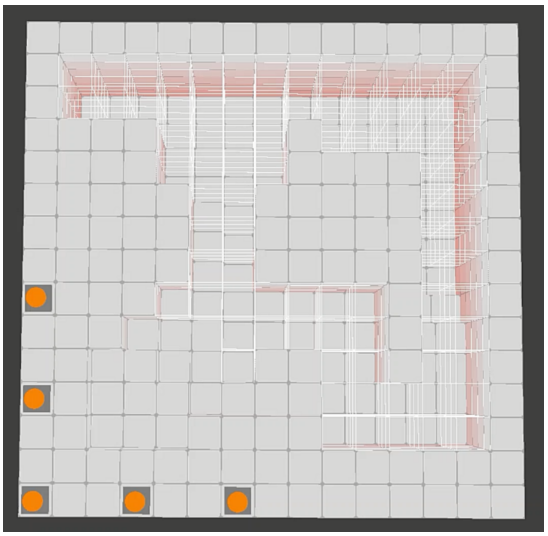
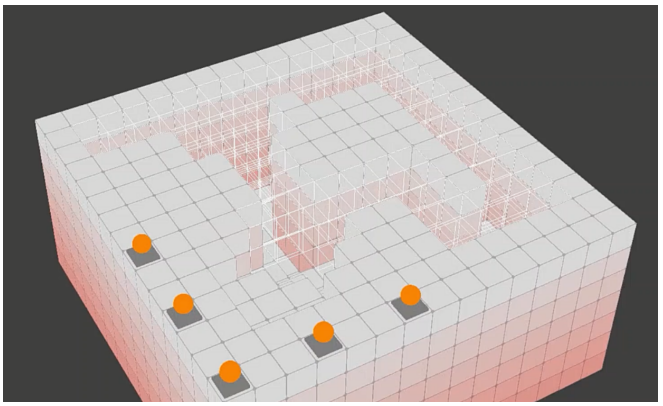


Figure 6.6: After 80% completion

4. After all the blocks are excavated as to the shape map, robots will come back to the starting positon



(a)



(b)

Figure 6.7: (a) and (b) after the final shape map is obtained

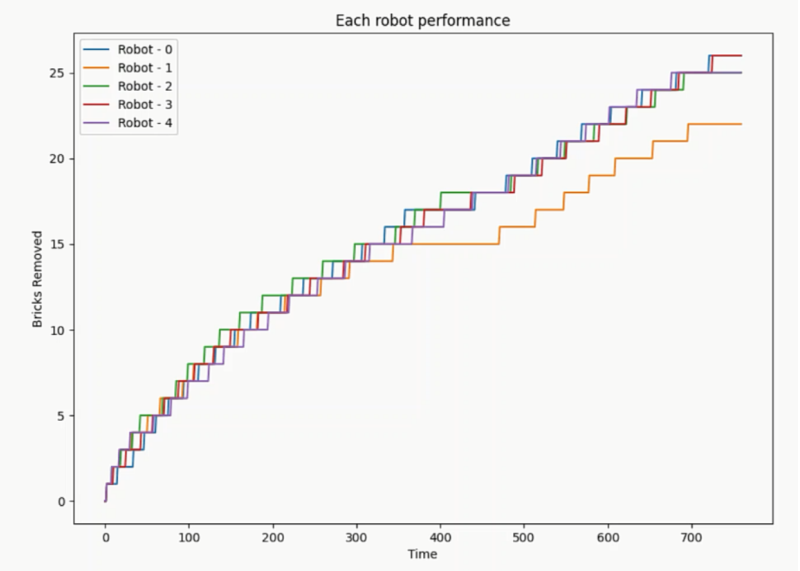
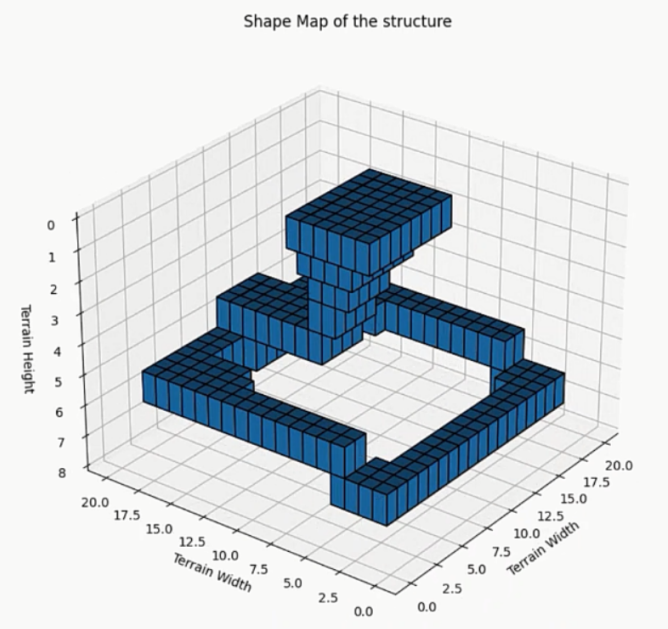
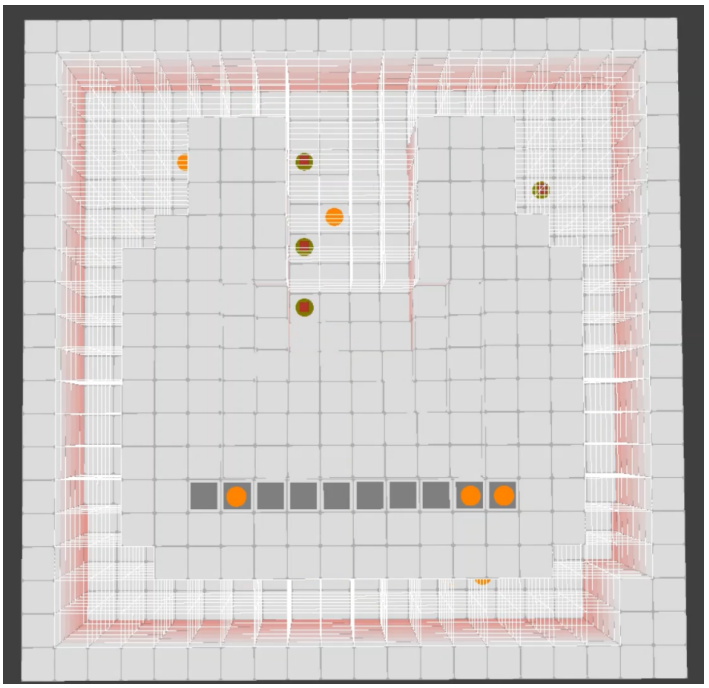


Figure 6.8: performance of each robot

Here to show redundancy, 3 robots are manually killed at the starting position while the excavation happens. This example uses 10 robots and excavation happens in a 20x20x8 terrain.



(a)



(b)

Figure 6.9: Three robots are killed while the excavation happens and the given shape map

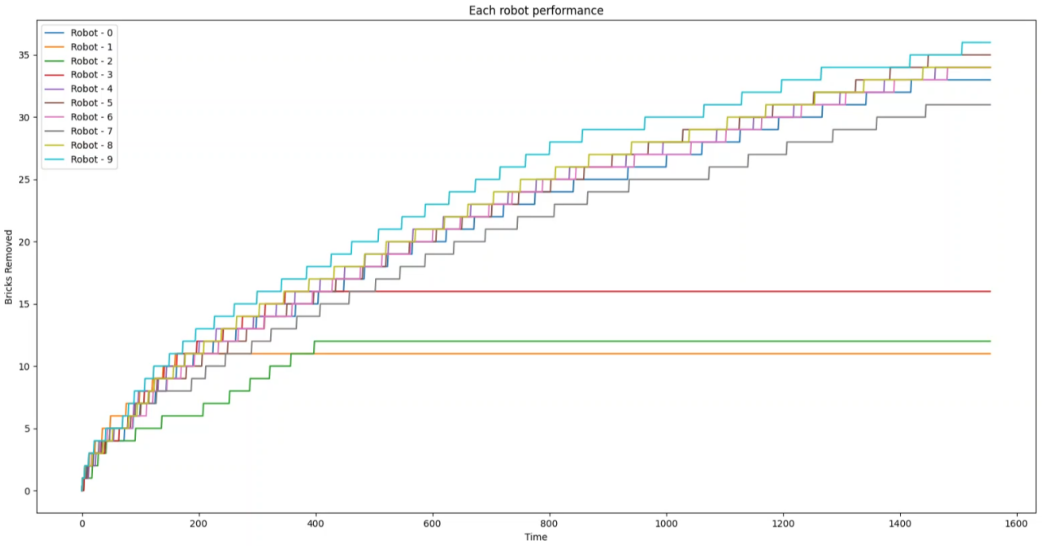


Figure 6.10: Performance of each robot

6.6 Summary

This chapter described how the robot controllers and environment are implemented for simulating a real world 2D swarm construction task. The developed control algorithm alongside with the adapted swarm concepts are evaluated in the next chapter.

CHAPTER 7 - EVALUATION

7.1 Introduction

In this chapter we evaluate the swarm robot controllers designed and developed by this research. Here we discussed whether the objectives mentioned in earlier chapters are met and to what extent.

7.2 Evaluation Procedure

The aim of this research was to extend parallel behavior of swarms for 3D excavation. However, extremely similar studies are required for a direct quantitative comparison, and none were discovered in the literature review. Each study proposal tries out a distinct set of swarm robotics concepts and aims to improve/explore a different part of the discipline. A summary of the most related research can be found in table 1 on section 2.5. These vary within self assembly, 2D construction, wall/ nest construction and beyond

Major difference between the most similar (construction related) researches and our one is that they have done the construction without a predetermined shape, allowing the emergent property of a swarm to determine the shape. Our aim in this research was to construct a given shape.

Taking into account the foregoing, a qualitative comparison is made, and the proposed approach is assessed based on the following factors, which are limitations highlighted in existing work:

1. True parallel behavior in construction.
2. Redundancy of the system.
3. Final outcome of the system.

7.2.1 Parallel behavior in construction

The method of approach demonstrated true parallel construction, with many swarm robots digging the shape and removing building pieces at the same time. This has not been noticed in previous studies that creates [16], [20], [33] or self-organizes into specific shapes [21], [22], [34]. Simple methods, such as blind bulldozing [13], demonstrate real parallelism but have limited decision-making skills, making them unsuitable for complicated building projects like the ones we're pursuing in this study. The idea of exploiting parallel behavior in 2D creation was first proposed by Liyanage et al.[23] This concept was expanded to 3D excavation in our research.

7.2.2 System redundancy

The use of stigmergy has aided in the independence of the robots in our proposed system. As a result, the construction process is unaffected by the number of robots on the job. The number of robots has no effect on the construction speed. There are several ways with redundant swarms in both construction and self-organization in other existing 3D construction methods. Those, on the other hand, do not exhibit parallel building behavior. Some methods, such as Werfer et al [15], lack redundancy since they rely heavily on precompile instructions for each robot in the swarm.

7.2.3 Final outcome of the system

Our proposal has demonstrated that it can produce the desired result. It can excavate a 3D shape with the help of several robots. In rule-based form generation [23,24], similar studies that modify swarm behavior for construction have been effective. Their issue, though, is that the conclusion is not predetermined. Blind bulldozing [13] is a simple procedure that is extremely dependent on the environment and cannot be used to create random shapes. Others that satisfy the final result, namely

self-organizing in a kilobot swarm [22], do not exhibit at least one of the properties we analyze in this comparison.

7.3 Performance Vs Number Of Bots

In a swarm system, performance is increased with the number of swarm entities in general. However, overcrowding can cause performance to drop as the interactions between swarm units increase too much.

In our proposed system, the swarm bots take time to resolve inter robot interactions. Robots have to take longer paths to their goals to avoid collision with other bots which are obstructing its pre planned path. With this, after a certain number of bots in the system, the swarm system loses performance. This behavior can be observed in the graph below

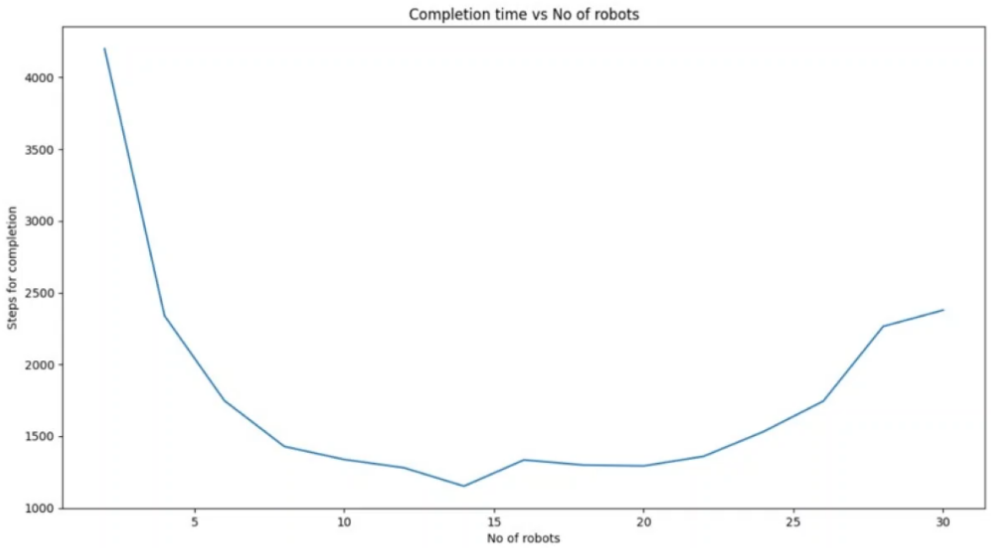


Figure 7.1: Time taken to finish a given shape vs number of bots in the swarm system

The graph shows the time steps that the swarm system spent on constructing a given shape vs the number of bots used in the system. For the particular shape used, the best performance can be observed at around 40 robots in the swarm.

The number of bots for best performance is dependent on the shape constructed, building block size, terrain and the bot interaction handling criteria. However, it would be possible to pre-determine this number by realistic enough simulations in the real world application.

7.4 Use Cases

As explained in section 1.3, swarm construction is heavily discussed in the field of space exploration. With the existing technologies, swarm construction is financially sound considering the fact that the payload that is sent from earth is considerably smaller compared to the payload needed for any other means of construction.

Construction of mines and underground tunnels with human labor can be life risky and time consuming. 3D excavating swarms can be applied to such constructions including ground formation and leveling

If a robot swarm is less costly than a conventional excavation crew/ machines, a robot swarm is more advantageous for 3D construction tasks in the real world such as digging, drainage systems and underground structures.

7.5 Summary

In this chapter we have evaluated the developed robot controllers using proper experiment mechanisms. We have qualitatively compared the success of the approach and presented the results. The results shows that indeed, the proposed method is optimal compared to the existing 2D construction methods

CHAPTER 8 - CONCLUSION

8.1 Introduction

This thesis presents results that show that it is possible to introduce swarm robotics to optimise excavation tasks. Even though there is research done on construction using decentralised swarm bots, not much interest was paid on excavation. We expect to suggest that it is possible to use swarm robots for excavation and present the algorithm for it in this research.

8.2 Conclusions

We examined the limitations of previous methodologies given for swarm based extraterrestrial habitat construction. The goal of our suggested strategy was to introduce decentralised swarm based excavation algorithm to improve off-earth constructions, In doing so we have shown that swarm robotics can be effectively used for excavation tasks.

We created a simulation of a swarm robotic system that performs 3D excavation tasks. For the robots in the swarm, we employed stigmergy and rule-based control principles and devised a control algorithm for the robots to fulfil their respective duties so that the shape supplied is excavated collectively.

The observation with all the existing research for constructing space habitations is that they are expensive and require heavy machinery that's difficult to transport. Though swarm robotics have been suggested for excavation tasks no proper algorithm was introduced. Our solution has overcome the problem of building extraterrestrial habitats by introducing swarm based excavation. Furthermore, we have achieved all five objectives set at the start of this research.

Finally, we may infer that combining stigmergy with rule-based control in a swarm robotic system with a well-designed control algorithm can result in a robotic swarm that can be effectively used for excavating activities.

8.3 Future work

The study focuses on developing a simulated algorithm for decentralised swarm excavation which can be implemented and used in the actual world. Furthermore, the same algorithm can be extended to optimise existing centralised 3D construction tasks.

In our research the considered environment is depicted as blocks, but when implementing this in the real world, actual environment features should be considered. To do so this suggested algorithm should be improved.

The robots could be implemented in a way to dynamically alter the given shape map when met with obstructions.

When implementing this in the real world, localization is a necessity, but doing this in outer space can be challenging, therefore new concepts and methods need to be implemented to overcome this.

References

[1] F. Ruess, J. Schaenzlin, and H. Benaroya, “Structural Design of a Lunar Habitat,” p. 25.

[2] H. Bier, H. Vermeer, A. Hidding, K. Jani, and T. Delft, “Design-to-Robotic-Production of Underground Habitats on Mars,” p. 8.

[3] A. R. Cheraghi, S. Shahzad, and K. Graffi, “Past, Present, and Future of Swarm Robotics,” *arXiv:2101.00671 [cs]*, Jan. 2021, Accessed: Jun. 26, 2021. [Online]. Available: http://arxiv.org/abs/2101.00671

[4] G. Beni, “From Swarm Intelligence to Swarm Robotics,” in *Swarm Robotics*, vol. 3342, E. Şahin and W. M. Spears, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–9. doi: 10.1007/978-3-540-30552-1_1.

[5] G. Beni, “The concept of cellular robotic system,” in *Proceedings IEEE International Symposium on Intelligent Control 1988*, Arlington, VA, USA, 1989, pp. 57–62. doi: 10.1109/ISIC.1988.65405.

[6] T. Fukuda and S. Nakagawa, “Approach to the dynamically reconfigurable robotic system,” *J Intell Robot Syst*, vol. 1, no. 1, pp. 55–72, 1988, doi: 10.1007/BF00437320.

[7] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “A taxonomy for swarm robots,” in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ’93)*, Yokohama, Japan, 1993, vol. 1, pp. 441–447. doi: 10.1109/IROS.1993.583135.

[8] B. Webb, “Swarm Intelligence: From Natural to Artificial Systems,” *Connection Science*, vol. 14, no. 2, pp. 163–164, Jun. 2002, doi: 10.1080/09540090210144948.

[9] J. Chappell, “Book review: Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies,” *Am. J. Hum. Biol.*, vol. 21, no. 5, pp. 713–714, Sep. 2009, doi: 10.1002/ajhb.20948.

[10] J. Werfel, “Collective Construction with Robot Swarms,” in *Morphogenetic Engineering*, R. Doursat, H. Sayama, and O. Michel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 115–140. doi: 10.1007/978-3-642-33902-8_5.

[11] G. Theraulaz and E. Bonabeau, “Coordination in Distributed Building,” *Science*, vol. 269, no. 5224, pp. 686–688, Aug. 1995, doi: 10.1126/science.269.5224.686.

[12] R. L. Stewart and R. A. Russell, “A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm,” *Adaptive Behavior*, vol. 14, no. 1, pp. 21–51, Mar. 2006, doi: 10.1177/105971230601400104.

[13] C. A. C. Parker, Hong Zhang, and C. R. Kube, “Blind bulldozing: multiple robot nest construction,” in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Las Vegas, NV, USA, 2003, vol. 2, pp. 2010–2015. doi: 10.1109/IROS.2003.1248950.

[14] S. Garnier, J. Gautrais, and G. Theraulaz, “The biological principles of swarm intelligence,” *Swarm Intell*, vol. 1, no. 1, pp. 3–31, Oct. 2007, doi: 10.1007/s11721-007-0004-y.

[15] de la Croix, Jean-Pierre and Egerstedt, Magnus B., “Controllability Characterizations of Leader-Based Swarm Interactions,” *AAAI Fall Symposium: Human Control of Bioinspired Swarms*, Nov. 2012, [Online]. Available: http://hdl.handle.net/1853/46177

[16] J. Werfel, Y. Bar-Yam, and R. Nagpal, “Construction by Robot Swarms Using Extended Stigmergy,” p. 19.

[17] J. Werfel, Y. Bar-Yam, and R. Nagpal, “Building Patterned Structures with Robot Swarms,” p. 8.

[18] G. Theraulaz and E. Bonabeau, “Modelling the Collective Building of Complex Architectures in Social Insects with Lattice Swarms,” *Journal of Theoretical Biology*, vol. 177, no. 4, pp. 381–400, Dec. 1995, doi: 10.1006/jtbi.1995.0255.
[19] J. Pollack, M. A. Bedau, P. Husbands, R. A. Watson, and T. Ikegami, Eds., *Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*. The MIT Press, 2004. doi: 10.7551/mitpress/1429.001.0001.
[20] J. Werfel, K. Petersen, and R. Nagpal, “Designing Collective Behavior in a Termite-Inspired Robot Construction Team,” *Science*, vol. 343, no. 6172, pp. 754–758, Feb. 2014, doi: 10.1126/science.1245842.
[21] Gebhardt, Gregor H.W. and Neumann, Gerhard, “The Kilobot Gym,” *ICRA 2018 Workshop*, 2018.
[22] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal, “Kilobot: A low cost robot with scalable operations designed for collective behaviors,” *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 966–975, Jul. 2014, doi: 10.1016/j.robot.2013.08.006.
[23] T. Liyanage and S. Fernando, “Optimizing Robotic Swarm Based Construction Tasks,” p. 4.
[24] C. A. Rouff, M. G. Hinchey, W. F. Truszkowski, and J. L. Rash, “Experiences applying formal approaches in the development of swarm-based space exploration systems,” *Int J Softw Tools Technol Transfer*, vol. 8, no. 6, pp. 587–603, Oct. 2006, doi: 10.1007/s10009-006-0027-5.
[25] J. Thangavelautham, K. Law, T. Fu, N. Abu El Samid, A. D. S. Smith, and G. M. T. D’Eleuterio, “Autonomous multirobot excavation for lunar applications,” *Robotica*, vol. 35, no. 12, pp. 2330–2362, Dec. 2017, doi: 10.1017/S0263574717000017.
[26] S. Wilkinson, J. Musil, J. Dierckx, I. Gallou, and X. de Kestelier, “Autonomous Additive Construction on Mars,” p. 12.
[27] J. Irawan, X. De Kestelier, N. Argyros, B. Lewis, and S. Gregson, “A Reconfigurable Modular Swarm Robotic System for ISRU (In-Situ Resource Utilisation) Autonomous 3D Printing in Extreme Environments,” in *Impact: Design With All Senses*, C. Gengnagel, O. Baverel, J. Burry, M. Ramsgaard Thomsen, and S. Weinzierl, Eds. Cham: Springer International Publishing, 2020, pp. 685–698. doi: 10.1007/978-3-030-29829-6_53.
[28] B. Khoshnevis, X. Yuan, B. Zahiri, J. Zhang, and B. Xia, “Construction by Contour Crafting using sulfur concrete with planetary applications,” *RPJ*, vol. 22, no. 5, pp. 848–856, Aug. 2016, doi: 10.1108/RPJ-11-2015-0165.
[29] G. H. Just, K. Smith, K. H. Joy, and M. J. Roy, “Parametric review of existing regolith excavation techniques for lunar In Situ Resource Utilisation (ISRU) and recommendations for future excavation experiments,” *Planetary and Space Science*, vol. 180, p. 104746, Jan. 2020, doi: 10.1016/j.pss.2019.104746.
[30] D. Miner, “Swarm Robotics Algorithms: A Survey,” p. 15.
[31] “Ursina Engine.” https://www.ursinaengine.org/ (accessed Jul. 17, 2022).
[32] “Python UI | Design GUI with Python | Python Bindings for Qt.” https://www.qt.io/qt-for-python (accessed Jul. 17, 2022).
[33] F. Rossi, S. Bandyopadhyay, M. Wolf, and M. Pavone, “Review of Multi-Agent Algorithms for Collective Behavior: a Structural Taxonomy,” *IFAC-PapersOnLine*, vol. 51, no. 12, p. 112—117, 2018.
[34] D. A. Feshbach and C. Sung, “Reconfiguring Non-Convex Holes in Pivoting Modular Cube Robots,” *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6701–6708, Oct. 2021, doi: 10.1109/LRA.2021.3095030.

Appendix

Python Application Code

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QRadioButton, QLabel,
QLineEdit, QComboBox, QMainWindow, QVBoxLayout
from PyQt5.QtGui import QIcon, QFont, QIntValidator
from PyQt5.QtCore import pyqtSlot
from os import listdir
from os.path import isfile, join

from ursina import *
from ursina.prefabs.first_person_controller import FirstPersonController
from math import hypot
import random
from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
import time
import pickle

terrainWidth = 0
terrainHeight = 0

robot_cnt = 0
visibleRange = 0

subsets = []
subCubes = []
sci = 0 # subCube index.
currentSubset = 0

prevTime = None
env = None
shape_map = None
rotateCam = False

completed = []
moveSequence = []
subjects = []
graph = []

robotLocations = []

terrainGenerated = False
counter = 0
started = False
filename = 'newMap1.swarm'
editor_camera = None
editor_mode = False
```

```
## Classes

class HelpPage(QWidget):

    def __init__(self):
        super().__init__()
        self.title = 'Help'
        self.left = 700
        self.top = 240
        self.width = 420
        self.height = 500
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        title = QLabel('Keyboard Commands', self)
        myFont = QFont('Arial',15)
        myFont.setBold(True)
        title.setFont(myFont)
        title.move(90,10)

        text = '''Simulate Mode:\n\ne - start simulation\n0-4 - stop robot with no\ng - get
info after the end\n\nEditor Mode:\n\nd,a,w,s - move editor robot\nr - rotate editor
robot\np - place robots\nspace - mine below block\nf - mine forward block\nx - save new map
to file\n\nCamera control:\n\nm - EditorCamera\nl - lock the camera\n"-" - zoom out\n"+" -
zoom in\narrow keys - move/rotate camera\nk - toggle rotate mode'''
        info = QLabel(text, self)
        info.move(60,70)


class StartPage(QMainWindow):

    def __init__(self):
        super().__init__()
        self.title = 'Swarm Excavation'
        self.left = 500
        self.top = 240
        self.width = 640
        self.height = 420
        self.mapPath = 'Maps'
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        title = QLabel('Decentralized Swarms for Excavation', self)
        myFont = QFont('Arial',15)
        myFont.setBold(True)
        title.setFont(myFont)
        title.resize(600,50)
        title.move(90,10)

        terrainWidth = QLabel('Terrain width', self)
        terrainWidth.move(60,90)

        terrainHeight = QLabel('Terrain height', self)
```

```
terrainHeight.move(60,130)

noRobots = QLabel('No of Robots', self)
noRobots.move(60,170)

visibility = QLabel('Visible range', self)
visibility.move(60,210)

filenameLable = QLabel('Map name', self)
filenameLable.move(60,250)

config = QLabel('Select Config', self)
config.move(60,330)

self.terrainW = QLineEdit(self)
self.terrainW.move(150, 85)
self.terrainW.resize(50,30)
self.terrainW.setValidator(QIntValidator())

self.terrainH = QLineEdit(self)
self.terrainH.move(150, 125)
self.terrainH.resize(50,30)
self.terrainH.setValidator(QIntValidator())

self.robotCount = QLineEdit(self)
self.robotCount.move(150, 165)
self.robotCount.resize(50,30)
self.robotCount.setValidator(QIntValidator())

self.visibileRange = QLineEdit(self)
self.visibileRange.move(150, 205)
self.visibileRange.resize(50,30)
self.visibileRange.setValidator(QIntValidator())

self.fileNameEdit = QLineEdit(self)
self.fileNameEdit.move(150, 245)
self.fileNameEdit.resize(150,30)

self.radiobutton1 = QRadioButton("Map Editor", self)
self.radiobutton1.setChecked(True)
self.radiobutton1.move(20,60)

self.radiobutton2 = QRadioButton("Execute Robots", self)
self.radiobutton2.resize(150,30)
self.radiobutton2.move(20,290)

self.configSelect = QComboBox(self)
availableMaps = [f for f in listdir(self.mapPath) if isfile(join(self.mapPath, f))]
self.configSelect.addItems(availableMaps)
self.configSelect.resize(150,30)
self.configSelect.move(150,328)

button = QPushButton('Run', self)
button.setToolTip('Run the simulation on the selected mode')
button.move(480,370)
button.clicked.connect(self.on_click)

buttonHelp = QPushButton('Help', self)
buttonHelp.setToolTip('Keyboard inputs')
buttonHelp.move(380,370)
```

```
        buttonHelp.clicked.connect(self.help_click)


        self.show()

    @pyqtSlot()
    def on_click(self):

      if self.radiobutton2.isChecked():
        runSimulator(loadFile=True, filenameInput=self.configSelect.currentText())
      else:
        data = {}
        data['terrainWidth'] = self.terrainW.text()
        data['terrainHeight'] = self.terrainH.text()
        data['robotCount'] = self.robotCount.text()
        data['visibileRange'] = self.visibileRange.text()
        print(data)
        runSimulator(loadFile=False, filenameInput=self.fileNameEdit.text(), data=data)

    def help_click(self):
        self.w = HelpPage()
        self.w.show()



class Robot(Entity):
    def __init__(self, position=(0,1,0),size=(30,20,30),editor_mode=False):
        super().__init__(
        position = position,
        model = 'sphere',
        color=color.orange,
        scale = 0.6
        )
        self.start_pos = position
        self.envMap = np.ones(size, np.int8)
        self.shapeMap = np.zeros(size, np.int8)
        self.removedMap = np.zeros(size, np.int8)
        self.status = None
        self.locationQueue = [Vec3(position)]
        self.queueStart = 0
        self.mineCount = 0
        self.mineGraph = []
        self.printStuff = False

        self.load = Entity(model='cube', color=color.brown, scale=(0.5,0.5,0.5),
position=(0,0.5,0),parent = self)
        self.load.visible = False

        if editor_mode:
          self.arrow = Entity(model='Arrow', color=color.red, scale = (1.2,3,3),parent =
self)
        else:
          self.pad = Entity(model='cube',color=color.gray,scale=(0.8,0.2,0.8),
position=position)
          self.pad.y = 0.5


    def setShapeMap(self,shapeMap):
        self.shapeMap = shapeMap
```

```
    def mine(self,location,env,subsets):

        if env.env_arr[location] == 1:
            for v in subsets[location[0]].model.vertices:
                if (v[0] >= location[0] - 0.5 and
                    v[0] <= location[0] + 0.5 and
                    v[1] >= -location[1] - 0.5 and
                    v[1] <= -location[1] + 0.5 and
                    v[2] >= location[2] - 0.5 and
                    v[2] <= location[2] + 0.5):
                    v[1] -= 1

            subsets[location[0]].model.generate()
            env.remove_voxel(location)
            self.shapeMap[location] = 0
            self.envMap[location] = 0
            self.load.visible = True
            self.color = color.olive
            return True
        return False

    def mineUp(self,location,env,subsets):

        global counter, subjects

        if env.env_arr[location] == 1:
            for i in range(location[1]):
                if env.env_arr[(location[0],i,location[2])] == 1:
                    bud = Entity(model='wireframe_cube',
position=(location[0],-i,location[2]))
                    bud.scale *= 0.9999
                self.mine((location[0],i,location[2]),env,subsets)
            self.mine((location[0],location[1],location[2]),env,subsets)

            return True
        return False

    def graphRemoved(self):
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.set_aspect('auto')
        ax.voxels(np.swapaxes((self.removedMap==1),1,2), edgecolor="k")
        ax.invert_zaxis()

        plt.xlabel("Terrain Width")
        plt.ylabel("Terrain Width")
        ax.set_zlabel("Terrain Height")


class Env:

    def __init__(self, robot_cnt = 0, size=(30,20,30)):
        self.size = size
        self.env_arr = np.ones(size, np.int8)
        self.shape_arr = np.ones(size, np.int8)
        self.robot_location = [None] * robot_cnt

    def update_robot_locations(self,robot_no,location):
        self.robot_location[robot_no] = location
```

```
    def remove_voxel(self,location):
        self.env_arr[location] = 0
        self.shape_arr[location] = 0

def input(key):
    global env, terrainHeight, terrainWidth, graph, subjects, rotateCam, subsets,
robot_cnt, started, filename, editor_camera, editor_mode, robotLocations, shape_map


    if key == 'q' or key == 'escape':
        quit()

    if key == 'e' and not editor_mode:
        createShapeMap()
        started = True
        for i in range(robot_cnt):
            robotController(i)

    if key == 'm':
        editor_camera = EditorCamera()
        camera.position = (0, 0)
        camera.rotation_y = 0
        camera.rotation_x = 0
        # camera.orthographic = True

    if key == 'l':
      if editor_camera is not None:
        editor_camera.ignore = not(editor_camera.ignore)

    ## Mine front
    if key == 'f' and int(subjects[0].y) < 1 :
        subjects[0].mineUp(  (int(subjects[0].x) + int(subjects[0].forward.z*2),
-(int(subjects[0].y)), int(subjects[0].z) - int(subjects[0].forward.x*2) ), env, subsets )
        subjects[0].shapeMap[int(subjects[0].x) +
int(subjects[0].forward.z*2)][-int(subjects[0].y)][int(subjects[0].z) -
int(subjects[0].forward.x*2)] =  -(terrainHeight + int(subjects[0].y))

    ## Mine down
    if key =='space' and -(subjects[0].y - 1) < terrainHeight:
        subjects[0].mine( (int(subjects[0].x), -(int(subjects[0].y) - 1),
int(subjects[0].z)), env, subsets)
        subjects[0].shapeMap[int(subjects[0].x)][-(int(subjects[0].y) -
1)][int(subjects[0].z)] = terrainHeight + (int(subjects[0].y) - 1)

        subjects[0].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] )

    ## Moving the editor robot
    if key == 'd' and (subjects[0].x + 1 < terrainWidth) :
        if abs(np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] -
np.sum(env.env_arr,axis=1)[int(subjects[0].x + 1),int(subjects[0].z)])<=1 :
            subjects[0].x += held_keys['d']
            subjects[0].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] )

    if key == 'a':
        if (subjects[0].x > 0) and
abs(np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] -
np.sum(env.env_arr,axis=1)[int(subjects[0].x - 1),int(subjects[0].z)])<=1 :
            subjects[0].x -= held_keys['a']
```

```
            subjects[0].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] )

    if key == 'w':
        if (subjects[0].z + 1 < terrainWidth) and
abs(np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z + 1)])<=1 :
            subjects[0].z += held_keys['w']
            subjects[0].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] )

    if key == 's':
        if (subjects[0].z > 0) and
abs(np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z - 1)])<=1 :
            subjects[0].z -= held_keys['s']
            subjects[0].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[0].x),int(subjects[0].z)] )

    ## Rotating the editor robot
    if key == 'r':
      subjects[0].rotation += (0,90,0)

    ## Placing pads in the editor mode
    if key == 'p' and editor_mode and len(robotLocations) <= robot_cnt:
      pad = Entity(model='cube',color=color.gray,scale=(0.8,0.2,0.8),
position=(subjects[0].x,0.5,subjects[0].z))
      robotLocations.append((subjects[0].x,1,subjects[0].z))


    ## Move the camera
    if key =='-':
      camera.fov += 0.5

    if key =='=':
      camera.fov -= 0.5

    if key =='right arrow':

        if rotateCam:
          camera.rotation_x += 0.5
          print("rotx",camera.rotation_x)
        else:
          camera.x += 0.5
          print("x",camera.x)

    if key =='left arrow':

        if rotateCam:
          camera.rotation_x -= 0.5
          print("rotx",camera.rotation_x)
        else:
          camera.x -= 0.5
          print("x",camera.x)

    if key =='up arrow':

        if rotateCam:
          camera.rotation_y += 0.5
          print("roty",camera.rotation_y)
```

```
            else:
              camera.y += 0.5
              print("y",camera.y)

        if key =='down arrow':

            if rotateCam:
              camera.rotation_y -= 0.5
              print("roty",camera.rotation_y)
            else:
              camera.y -= 0.5
              print("y",camera.y)

        ## Rotate or move mode of the camera
        if key == 'k':
            rotateCam =  not(rotateCam)

        ## Save to the file
        if key == 'x':
            filePath = 'Maps/'+filename
            print('Saving to file',filePath)
            shape_map = subjects[0].shapeMap
            save_to_file(filePath)


        if key == 'g':
            for i in range(robot_cnt):
                plt.plot(subjects[i].mineGraph,label=('Robot - '+str(i)))

            plt.xlabel("Time")
            plt.ylabel("Bricks Removed")
            plt.title("Each robot performance")
            plt.legend()
            plt.show()

            for i in range(robot_cnt):
                subjects[i].graphRemoved()
                plt.title("Removed Map by Robot - "+str(i+1))
                plt.show()

        if key == '0' or key == '1' or key == '2' or key == '3' or key == '4':
            completed[int(key)] = True
            subjects[int(key)].load.visible = False
            subjects[int(key)].color=color.orange

            # for i in range(robot_cnt):
            #     if subjects[i].position == Vec3(subjects[i].start_pos):
            #         print(i,subjects[i].position)
            #         subjects[i].printStuff = True
            #         moveSequence[i]


    def createGraph(robotNo):
        global subjects, graph, counter
        graph[robotNo] = nx.Graph()
        map = np.sum(subjects[robotNo].envMap,axis=1)

        for i in range(-1,2):
            for j in range(-1,2):
                for h in range(-1,2):
```

```
                if i==0 and j==0:
                    continue
                k = (subjects[robotNo].x + i,subjects[robotNo].y + h,subjects[robotNo].z +
j)

                if Vec3(k) in env.robot_location:
                    map[int(subjects[robotNo].x + i),int(subjects[robotNo].z + j)] = -1

    for i in range(terrainWidth*terrainWidth):
        x = int(i/terrainWidth)
        z = int(i%terrainWidth)


        if (x - 1 >= 0) and abs(map[x,z] - map[x-1,z]) <= 1:
            graph[robotNo].add_edge(str((x,z)), str((x-1,z)))

            if (z - 1 >= 0) and abs(map[x,z] - map[x-1,z-1]) <= 1:
                graph[robotNo].add_edge(str((x,z)), str((x-1,z-1)))

            if (z + 1 < terrainWidth) and abs(map[x,z] - map[x-1,z+1]) <= 1:
                graph[robotNo].add_edge(str((x,z)), str((x-1,z+1)))

        if (x + 1 < terrainWidth) and abs(map[x,z] - map[x+1,z]) <= 1:
            graph[robotNo].add_edge(str((x,z)), str((x+1,z)))

            if (z - 1 >= 0) and abs(map[x,z] - map[x+1,z-1]) <= 1:
                graph[robotNo].add_edge(str((x,z)), str((x+1,z-1)))

            if (z + 1 < terrainWidth) and abs(map[x,z] - map[x+1,z+1]) <= 1:
                graph[robotNo].add_edge(str((x,z)), str((x+1,z+1)))


        if (z - 1 >= 0) and abs(map[x,z] - map[x,z-1]) <= 1:
            graph[robotNo].add_edge(str((x,z)), str((x,z-1)))

        if (z + 1 < terrainWidth) and abs(map[x,z] - map[x,z+1]) <= 1:
            graph[robotNo].add_edge(str((x,z)), str((x,z+1)))


def findPath(robotNo, start, end):
    global subjects, env, graph
    try:
        path = nx.dijkstra_path(graph[robotNo], start, end)

    except nx.exception.NetworkXNoPath:

        return None

    except nx.exception.NodeNotFound:

        return None

    return path[1:]

def robotController(robotNo):
    '''This function will go through each robot map and select nearest high prio location
to mine and send the robot to there with mining method'''
    global env, completed, terrainHeight,subjects, counter, moveSequence

    digLoc = None
```

```
    digFront = False

    if subjects[robotNo].printStuff:
        print("1 controller works",robotNo)

    interactions(robotNo)

    # find excavatable locations layer by layer
    for i in range(terrainHeight,0,-1):
        digLoc=np.where(subjects[robotNo].shapeMap == i)
        if (len(digLoc[0])>0):
            break

        digLoc=np.where(subjects[robotNo].shapeMap == -i)
        if (len(digLoc[0])>0):
            digFront = True
            break

    # Finish if all the locations are excavated
    if (len(digLoc[0])==0):
        completed[robotNo] = True
        print(robotNo,"Finished")
        moveToStart(robotNo)
        return

    locations = []

    for i in range(len(digLoc[0])):
        locations.append((digLoc[0][i],digLoc[1][i],digLoc[2][i]))

    # find the nearest excavation location
    locations.sort(key = lambda p: (p[0] - subjects[robotNo].x)**2 + (p[2] -
subjects[robotNo].z)**2)

    createGraph(robotNo)

    map = np.sum(subjects[robotNo].envMap,axis=1)

    if subjects[robotNo].printStuff:
        print("1 controller",robotNo,digFront,locations)

    if digFront:

        digFront = False

        for i in locations:
            path =
findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]-1,i[2])))
            if i[1] == (terrainHeight - map[i[0]-1,i[2]] - 1) and path != None:
                moveInPath(path, robotNo,'r')
                return

            path =
findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]+1,i[2])))
            if i[1] == (terrainHeight - map[i[0]+1,i[2]] - 1) and path != None:
                moveInPath(path, robotNo,'l')
                return
```

```
            path =
findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0],i[2]-1)))
            if i[1] == (terrainHeight - map[i[0],i[2]-1] - 1) and path != None:
                moveInPath(path, robotNo,'f')
                return

            path =
findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0],i[2]+1)))
            if i[1] == (terrainHeight - map[i[0],i[2]+1] - 1) and path != None:
                moveInPath(path, robotNo,'b')
                return

    else:

        for i in locations:

            path =
findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0],i[2])))
            if path != None:
                moveInPath(path, robotNo,'d')
                return


    for i in locations:
        # if no path found find paths to other available locations
        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]-1,i[2])))
        if i[1] < (terrainHeight - map[i[0]-1,i[2]] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]-1,i[2])))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]+1,i[2])))
        if i[1] < (terrainHeight - map[i[0]+1,i[2]] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]+1,i[2])))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0],i[2]-1)))
        if i[1] < (terrainHeight - map[i[0],i[2]-1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0],i[2]-1)))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0],i[2]+1)))
        if i[1] < (terrainHeight - map[i[0],i[2]+1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0],i[2]+1)))
```

```
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]+1,i[2]+1)))
        if i[1] < (terrainHeight - map[i[0]+1,i[2]+1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]+1,i[2]+1)))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]-1,i[2]-1)))
        if i[1] < (terrainHeight - map[i[0]-1,i[2]-1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]-1,i[2]-1)))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]-1,i[2]+1)))
        if i[1] < (terrainHeight - map[i[0]-1,i[2]+1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]-1,i[2]+1)))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return

        path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((i[0]+1,i[2]-1)))
        if i[1] < (terrainHeight - map[i[0]+1,i[2]-1] - 1) and path != None:
            print(robotNo,"
Worked",subjects[robotNo].status,subjects[robotNo].position,str((i[0]+1,i[2]-1)))
            subjects[robotNo].status = "go_dig"
            moveInPath(path, robotNo)
            return


    if subjects[robotNo].position == Vec3(subjects[robotNo].start_pos) :
        print(robotNo," Robot died. Impossible ******************************** ",digFront)

        print(locations)
        np.set_printoptions(threshold=np.inf)
        print(np.sum(subjects[robotNo].envMap,axis=1))
        print(subjects[robotNo].shapeMap)

    subjects[robotNo].locationQueue.append(subjects[robotNo].position)

    if subjects[robotNo].queueStart <= 2 :
        subjects[robotNo].queueStart += 1
    else:
        subjects[robotNo].locationQueue.pop(0)

    if completed[robotNo] == True:

print("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx")
```

```
        moveToStart(robotNo)

    if moveSequence[robotNo] == []:
        print("Robot Died no movements ", robotNo, subjects[robotNo].position)
        print(locations,subjects[robotNo].status)
        print(np.sum(subjects[robotNo].envMap,axis=1))
        print(env.robot_location)

moveSequence[robotNo].append((int(subjects[robotNo].position[0]),int(subjects[robotNo].posi
tion[1]),int(subjects[robotNo].position[2])))
        return

def moveInPath(path,robotNo,action=None):

    '''
    Update the movesequence so robots will move on that path
    '''
    global env, terrainHeight, moveSequence , subjects

    map = np.sum(subjects[robotNo].envMap,axis=1)

    # if moveSequence[robotNo] == None:
    moveSequence[robotNo] = []

    if path == None:

moveSequence[robotNo].append((int(subjects[robotNo].position[0]),int(subjects[robotNo].posi
tion[1]),int(subjects[robotNo].position[2])))
        return

    for i in path:
        location = i.split(':')[0].strip(')(').split(', ')
        moveSequence[robotNo].append((int(location[0]), 1 - (terrainHeight -
map[int(location[0]),int(location[1])] ), int(location[1]) ))

    if action == 'd' or action == 'r' or action == 'l' or action == 'f' or action == 'b':
        subjects[robotNo].status = "go_dig"
        moveSequence[robotNo].append(action)

def moveToStart(robotNo):

    global subjects

    interactions(robotNo)
    createGraph(robotNo)
    path = findPath(robotNo,str((int(subjects[robotNo].x),int(subjects[robotNo].z))),
str((subjects[robotNo].start_pos[0], subjects[robotNo].start_pos[2])))
    subjects[robotNo].status = "go_back"
    moveInPath(path, robotNo)

def createShapeMap():
    global subjects, env, shape_map

    env.shape_arr = np.copy(shape_map)

    print(env.shape_arr.shape)
    for i in subjects:
        i.shapeMap = np.copy(env.shape_arr)

    fig = plt.figure()
```

```
    ax = fig.gca(projection='3d')
    ax.set_aspect('auto')
    ax.voxels(np.swapaxes((env.shape_arr!=0),1,2), edgecolor="k")
    ax.invert_zaxis()

    plt.xlabel("Terrain Width")
    plt.ylabel("Terrain Width")
    ax.set_zlabel("Terrain Height")
    plt.title("Shape Map of the structure")

    plt.show()



def interactions(robotNo):
    global subjects, robot_cnt, terrainWidth, visibleRange

    x_start = x_end = x = int(subjects[robotNo].x)
    z_start = z_end = z = int(subjects[robotNo].z)

    for i in range(visibleRange,-1,-1):
        if x - i >=0:
            x_start = x - i
            break

    for i in range(visibleRange,-1,-1):
        if z - i >=0:
            z_start = z - i
            break

    for i in range(visibleRange,-1,-1):
        if x + i <=terrainWidth:
            x_end = x + i
            break

    for i in range(visibleRange,-1,-1):
        if z + i <=terrainWidth:
            z_end = z + i
            break

    subjects[robotNo].shapeMap[x_start:x_end,:,z_start:z_end] =
(env.shape_arr)[x_start:x_end,:,z_start:z_end]
    subjects[robotNo].envMap[x_start:x_end,:,z_start:z_end] =
(env.env_arr)[x_start:x_end,:,z_start:z_end]


def moveRobots():
    '''Where to identify and control robot with each step. Change of the path obstacles can
be identified here and do accordingly'''
    global subjects, graph, completed, env, subsets, terrainHeight, counter, moveSequence


    for no,i in enumerate(moveSequence):
        if i==None:
            continue

        # This will update the map data looking at the surrounding
        interactions(no)

        # This is to debug any stucked robots that are not finished :
```

```
        if moveSequence[no] == [] and completed[no] == False:

            #
moveSequence[no].append((int(subjects[no].position[0]),int(subjects[no].position[1]),int(su
bjects[no].position[2])))
            print("+++++++++++++++++",no,subjects[no].position,subjects[no].status)
            print(env.robot_location)

        for index,j in enumerate(i):
            if j == None:
                break
            else:

                if subjects[no].printStuff:
                    print("Robot",no,moveSequence[no])

                #If a robot moving back and forth between same two locations, send it back
to the home. Randomness is added so, only one robot will
                #will start moving home first

                if len(set(subjects[no].locationQueue)) <= 2 and
len(subjects[no].locationQueue) == 4 and random.random() > 0.8:

                    if subjects[no].status == "go_back":
                        moveToStart(no)
                    if subjects[no].status == "go_dig":
                        if completed[no] == False and random.random()>0.5:
                            robotController(no)
                        else:
                            moveToStart(no)

                    break

                # d,r,l,f,b to dipict different excavation operations, with direction
                if j == 'd':
                    subjects[no].mine((int(subjects[no].x), -(int(subjects[no].y) - 1),
int(subjects[no].z)), env, subsets)
                    subjects[no].mineCount += 1
                    subjects[no].removedMap[(int(subjects[no].x), -(int(subjects[no].y) -
1), int(subjects[no].z))] = 1
                    subjects[no].y = 1 - (terrainHeight -
np.sum(env.env_arr,axis=1)[int(subjects[no].x),int(subjects[no].z)] )
                    env.robot_location[no] = subjects[no].position
                    moveSequence[no] = []
                    moveToStart(no)
                    break

                if j == 'r':
                    if(subjects[no].mineUp((int(subjects[no].x)+1, -int(subjects[no].y),
int(subjects[no].z)), env, subsets)):
                        subjects[no].mineCount += 1
                        subjects[no].removedMap[(int(subjects[no].x)+1,
-int(subjects[no].y), int(subjects[no].z))] = 1
                        moveSequence[no] = []
                        moveToStart(no)
                        break

                    robotController(no)
                    break
```

```
                if j == 'l':
                    if(subjects[no].mineUp((int(subjects[no].x)-1, -int(subjects[no].y),
int(subjects[no].z)), env, subsets)):
                        subjects[no].mineCount += 1
                        subjects[no].removedMap[(int(subjects[no].x)-1,
-int(subjects[no].y), int(subjects[no].z))] = 1
                        moveSequence[no] = []
                        moveToStart(no)
                        break

                    robotController(no)
                    break

                if j == 'f':
                    if(subjects[no].mineUp((int(subjects[no].x), -int(subjects[no].y),
int(subjects[no].z)+1), env, subsets)):
                        subjects[no].mineCount += 1
                        subjects[no].removedMap[(int(subjects[no].x), -int(subjects[no].y),
int(subjects[no].z)+1)] = 1
                        moveSequence[no] = []
                        moveToStart(no)
                        break

                    robotController(no)
                    break


                if j == 'b':
                    if(subjects[no].mineUp((int(subjects[no].x), -int(subjects[no].y),
int(subjects[no].z)-1), env, subsets)):
                        subjects[no].mineCount += 1
                        subjects[no].removedMap[(int(subjects[no].x), -int(subjects[no].y),
int(subjects[no].z)-1)] = 1
                        moveSequence[no] = []
                        moveToStart(no)
                        break

                    robotController(no)
                    break

                # if the next location is already excavated plan again or move back home
                if env.env_arr[( j[0], -(j[1]-1), j[2] )] != 1:

                    if subjects[no].status == "go_back":
                        moveToStart(no)
                    if subjects[no].status == "go_dig":
                        if completed[no] == False:
                            robotController(no)
                        else:
                            moveToStart(no)

                    break

                # if a robot is in the next location plan again or move back home
                if Vec3(j) in env.robot_location:

                    if subjects[no].status == "go_back":
                        moveToStart(no)
```

```
                    if subjects[no].status == "go_dig":
                        if completed[no] == False:
                            robotController(no)
                        else:
                            moveToStart(no)

                    break

                # if the next location is far away to move need to debug
                if abs(subjects[no].position[0] - j[0]) > 1 or abs(subjects[no].position[1]
- j[1]) > 1 or abs(subjects[no].position[2] - j[2]) > 1 :
                    print(no,"************ Illegal *************")

                    if abs(subjects[no].position[0] - j[0]) > 1 or
abs(subjects[no].position[2] - j[2]) > 1:
                      print("Moved more away")

                    if abs(subjects[no].position[1] - j[1]) > 1:
                      print("Mined the underneath by another bot")

                    print(subjects[no].position,j)
                    print(moveSequence[no])
                    print(np.sum(subjects[no].envMap,axis=1))

                    print("*************************")

                # move the robot to the next position
                subjects[no].position = j
                env.robot_location[no] = subjects[no].position
                subjects[no].locationQueue.append(subjects[no].position)

                if subjects[no].queueStart <= 2 :
                    subjects[no].queueStart += 1
                else:
                    subjects[no].locationQueue.pop(0)

                del moveSequence[no][index]

                # check the situation of two robots in the same location
                for i in range(robot_cnt):
                    if no == i: continue
                    if subjects[no].position == subjects[i].position:
                        print("*******Same location************")

                # come back to the beginning
                if  moveSequence[no] == [] and completed[no] == False: #TODO change the
identification method of the starting point generic
                    subjects[no].load.visible = False
                    subjects[no].color=color.orange
                    robotController(no)

                break

def update():
    global prevTime,subjects,counter,started

    while terrainGenerated == False:
      generateSubset()
```

```
    if time.time() - prevTime > 0.05:

        prevTime = time.time()
        moveRobots()


        if all(completed):
            # print(counter)
            return

        if started:
            counter += 1
            for i in range(robot_cnt):
                subjects[i].mineGraph.append(subjects[i].mineCount)

def generateSubset():
    global sci, currentSubset, terrainGenerated
    if currentSubset >= len(subsets):
        terrainGenerated = True
        return

    for i in range(terrainWidth*terrainHeight):
        subCubes[i].enable()
        x = subCubes[i].x = floor((i+sci)/(terrainWidth*terrainHeight))
        z = subCubes[i].z = floor((i+sci)%terrainWidth)
        y = subCubes[i].y = -int(i/terrainWidth)
        subCubes[i].color = color.rgb(255,255 + 15*y,255 + 15*y)
        subCubes[i].parent = subsets[currentSubset]
        subCubes[i].disable()


    subsets[currentSubset].combine(auto_destroy=False)
    subsets[currentSubset].enable()
    subsets[currentSubset].texture = "white_cube"
    sci += terrainWidth*terrainHeight
    currentSubset += 1

def load_from_file(filename):

  global terrainWidth, terrainHeight, robot_cnt, visibleRange, shape_map, robotLocations

  print('loading from',filename)
  # Getting back the objects:
  with open(filename,'rb') as f:
    terrainWidth, terrainHeight, robot_cnt, visibleRange, robotLocations, shape_map =
pickle.load(f)

  print('terrainWidth',terrainWidth)
  print('terrainHeight',terrainHeight)
  print('robot_cnt',robot_cnt)
  print('visibleRange',visibleRange)
  print('robotLocations',robotLocations)

def save_to_file(filename):

  global terrainWidth, terrainHeight, visibleRange, shape_map, robotLocations

  if editor_mode:
    robotLocations = robotLocations[1:]
  robotLocations = [tuple(x) for x in robotLocations]
```

```
  robotLocations = [tuple(map(int, tup)) for tup in robotLocations]

  # Saving the objects:
  with open(filename, 'wb') as f:
    pickle.dump([terrainWidth, terrainHeight, len(robotLocations), visibleRange,
robotLocations, shape_map], f)

def runSimulator(loadFile=False, filenameInput = 'TestMap.swarm', data = {}):

  global app, env, subjects, terrainWidth, terrainHeight, robot_cnt, visibleRange,
subCubes, subsets, completed, moveSequence, graph, prevTime, shape_map, robotLocations,
filename, editor_mode

  print('loadFile',loadFile)

  if loadFile:
    editor_mode=False
    load_from_file('Maps/'+filenameInput)

  else:

    editor_mode = True
    robotLocations = [(0, 1, 0)]

    terrainWidth = int(data['terrainWidth'])
    terrainHeight = int(data['terrainHeight'])
    robot_cnt = int(data['robotCount'])
    visibleRange = int(data['visibileRange'])
    filename = filenameInput
    shape_map = np.zeros((terrainWidth,terrainHeight,terrainWidth), np.int8)

  completed = [False] * robot_cnt
  moveSequence = [None] * robot_cnt
  subjects = [None] * robot_cnt
  graph = [None] * robot_cnt


  prevTime = time.time()

  env = Env(robot_cnt=robot_cnt,size=(terrainWidth,terrainHeight,terrainWidth))

  app = Ursina()

  # Instantiate 'ghost' subset cubes.
  for i in range(terrainWidth*terrainHeight):
      bud = Entity(model='cube')
      bud.scale *= 0.9999
      bud.disable()
      subCubes.append(bud)

  # Instantiate empty subsets.
  for i in range(terrainWidth):
      bud = Entity(model=None)
      bud.disable()
      subsets.append(bud)

  for i in range(len(robotLocations)):
    print('>>>>>>',robotLocations[i])
    subjects[i] = Robot(position = robotLocations[i],
size=(terrainWidth,terrainHeight,terrainWidth),editor_mode=editor_mode)
```

```
    camera.position = (9, 42.5)
    camera.rotation_y = 0
    camera.rotation_x = 56.5

    app.run()


if __name__ == '__main__':

    # runSimulator(loadFile=False)
    start = QApplication(sys.argv)
    ex = StartPage()
    ex.show()
    start.exec()
    # sys.exit(start.exec_())
```