
2D Occupancy Grid Mapping in a Dynamic Environment Using Swarm Robotics

MCG 4220: Undergraduate Thesis



uOttawa

Written By:

Salwan Bahia (300025648)

Thesis Supervisor:

Dr. Roland Bouffanais

Submission Date

April 9th, 2022

UNIVERSITY OF OTTAWA

DEPARTMENT OF MECHANICAL ENGINEERING

Abstract

Swarm robotics is becoming an increasingly popular research topic due to its scalability, robustness, and flexibility, which are often used to help complete many tasks such as mapping and tracking. This report delves into 2D occupancy grid mapping using swarm robots in a dynamic, ever-changing environment so as to mimic a real-world situation. While the main development code has already been developed for this project, there are still important research questions that need to be answered to enhance the mapping capabilities of the swarm. Firstly, there is a need to improve the quantification metrics for the response of the system. The improvement of quantification should eventually lead to research which optimizes the performance of the swarm.

Automated tests will be carried out to better understand the relationship between the swarm variables and how they affect the exploration-exploitation spectrum for this complex system. The density variable will specifically be highlighted throughout testing as it becomes clear that an optimal density value does exist. Modifications and additions to the code also allow for a new response metric to be created, which is called the similarity ratio. The similarity ratio will provide more information about the mapping progress for the system and opens up future research opportunities for the quantification of swarm response and the optimization of swarm performance.

Contents

Contents	ii
List of Figures	1
1 Introduction	4
1.1 Motivation	4
1.2 Scope	5
2 Literature Review	6
2.1 Background	6
2.2 Existing Technologies and Related Works	8
3 Methodology	10
3.1 Initial Code	10
3.2 Modifications to Code	13
4 Simulation Results	16
4.1 Testing K and Density With Scaled Pillar Size	16
4.1.1 2-pillar Simulations	19
4.1.2 1-pillar Simulations	21
4.2 Testing K and Density with Non-Scaled Pillar Size	22
4.3 Scaling Repulsion and Fixing K	25
4.4 Testing Similarity Ratio	30
4.5 Comparing Response Metrics	32

5	Future Work	35
6	Conclusion	36
	Bibliography	38

List of Figures

1	Graphical representation of the Central Composite Design method [1]	8
2	Screenshot of simulation video at initial starting positions, where pillars are the square blocks and the orange circles represent robots	11
3	Screenshot of simulation video at iteration 50, where a dynamic change in the environment has suddenly occurred and the collective swarm has not yet registered the change	12
4	Screenshot of simulation video at iteration 150, where the swarm is in the midst of recovering from the dynamic environment change and the collective swarm is beginning to match the true map as it continues its mapping process.	13
5	Screenshot of code which calculates the grid length X and Y given the input parameter <i>density</i>	14
6	Screenshot of code which sends the swarm variables and the response values to a text file	15
7	Screenshot of python function which is called at each iteration to calculate the similarity ratio	16
8	Table for setting the 7 levels in the CCD method	17
9	CCD dataset with the corresponding levels for the 22 trials	18
10	Graphical representation of the modified CCD dataset	19
11	3D graph with results for the 2-pillar environment with 400 iterations	20
12	3D graph with results for the 1-pillar environment with 400 iterations	21
13	Side-by-side screenshots of simulation clips for the 1-pillar and 2-pillar environments	22
14	2-pillar environment with a scaled pillar at a density of 0.000123	23

15	1-pillar environment with a non-scaled pillar at a density of 0.000025	23
16	Dataset for the trial which includes 5 different values of K and 15 different densities (response is in yellow cells)	24
17	Line Graph which depicts results for the dataset with fixed repulsion and different values of K with 15 different densities	25
18	Dataset with scaled repulsion to the area of the grid	26
19	Result for simulation with 75 runs and scaled repulsion	27
20	Box plot for results of 150 simulation runs of swarm mapping with scaled repulsion	29
21	Line graph of simulation results using the new similarity ratio response metric . . .	31
22	Central composite dataset with a constant density	32
23	3D response surface and contour plot where the similarity ratio response is being tracked	33
24	3D response surface and contour plot where the cells discovered response is being tracked	34

Acknowledgments

I would like to thank Dr. Roland Bouffanais for supervising me during this project. I appreciated the opportunity to be able to work on this interesting subject. I also appreciated the guidance and support that was provided by him, without which completing a successful thesis would have been difficult. I also want to acknowledge Jabez Leong Kit from the Singapore University of Technology and Design for the previous research he conducted on swarm intelligence and dynamic swarm mapping. Jabez's work inspired me and was a good reference for me to build upon.

1 Introduction

1.1 Motivation

In the past few years, the scientific world has been fascinated with the concept of decentralized systems. From cryptocurrency, to power grids, to robotics, the value of a system without a centralized controller has been realized. There are several advantages of having a decentralized system. There is no single point of failure with these systems which make them quite reliable and safer to employ. Additionally, a decentralized system such as a swarm of robots can increase the potential for scalability as well as its adaptability to different environments.

Robotic mapping in particular is a useful tool that can be used in a variety of applications. While there has been a fair amount of research done in static environments, there have been very few works done in the presence of dynamic circumstances [2]. In a dynamic, ever-changing, and often unknown environment, the task of mapping becomes more difficult. A multi-robot system (MRS) can accomplish this task more efficiently than a single-robot autonomous exploration system [2], and they can theoretically be scaled to map any size of environment in a variety of conditions. Should one or multiple robots be lost due to unforeseen circumstances, the swarm should still have the ability to adequately map the area with the robots that remain. Autonomous swarm mapping in a dynamic environment will be useful in hazardous situations where it may be unsafe to send a human to perform the task of surveying an area, such as in a first-aid rescue mission in a risky environment. In a case such as that, the swarm would first be deployed to map the area for the rescue team, who can then plan how to approach the situation and determine which type of transportation might be needed. In military or defence applications, the use of robots is already widely used to be the preliminary tool to explore an unsafe environment, such as bomb squad robots

[3]. The addition of swarm robotic mapping could increase their capability to understand unknown environments and in turn could benefit the safety of all involved.

1.2 Scope

The research goal of this thesis is to properly quantify the swarm mapping response and to optimize its performance by carrying out testing with a focus on the adjustment of the swarm variables. There is an existing method of quantifying the swarm mapping performance, however this method is not entirely proven to properly quantify the swarm's mapping ability. Therefore, there is also a need to determine a new method of rating the swarm's performance at any given time during a simulation. Eventually, the goal will be to allow the swarm to assess an environment and automatically adjust the swarm variables to optimize the mapping ability of the complex system. The testing of this experiment will be carried out by using Python simulations which provide video clips should one wish to visually inspect the simulation. It will also have the ability to track results throughout the simulation. The main development of the Python code has already been completed by the research team led by Jabez Leong Kit [2], however there are modifications and additions that can be made to the code so that automated testing can be performed and new Python functions can be added for further data tracking.

It is important to note that for the scope of this experiment, it is not possible to use a brute force approach when executing the test trials. Depending on the computer being used, the simulations can take upwards of 15 minutes per trial. With this fact in mind, it is critical to ensure the efficient use of each simulation. There are certain statistical analysis strategies that can be used to efficiently investigate the impact that controllable factors have on this particular complex system. Using Design of Experiments (DoE) concepts, one can create a dataset that maximizes learning using the minimum resources possible [4].

Once results have been obtained from test simulations, numerical data can be graphed using MATLAB software. The graphs are often visually examined for any trends and MATLAB functions are often used to find useful information that may not be seen by the naked eye.

2 Literature Review

Researchers have been attempting to define the term "Swarm Intelligence" for the past few decades [2]. The definitions for this term often vary depending on which field of study the researcher specializes in. Biologists and robotics engineers usually have differing opinions on how to define the subjective term. Intelligence can be defined as the ability to acquire and apply knowledge and skills [5]. Intelligence is usually marked by one's intellectual capabilities [2], which should have an ability to be evaluated in some way. For example, the intelligence quotient (IQ) test is an assessment that is designed to quantify a human's intelligence through a series of standardized tests. In the same breath, swarm intelligence should also be given an assessment method to quantify and standardize the performance of the swarm itself. This report will heavily delve into the methods of quantification that swarm mapping currently possesses, and what new improvements can be made.

2.1 Background

A common concept that must be researched in MRS is the exploration-exploitation dilemma. These multi-agent, decentralized systems tend to have two mutually exclusive modes which exhibit the opposite behaviour. A system which is in exploration mode will try to gather new information, meaning the agents will be inclined to spread out and explore any unknown areas in the environment, instead of making use of the information that is presently known. A system in exploitation mode will tend to make use of already available information which can result in agents bunching up at

times, instead of exploring unknown areas in the environment [6]. Exploration-exploitation can be visualized as a spectrum that should be fine-tuned depending on the environment that the swarm finds itself in [7]. There are a few variables that affect which type of behaviour the swarm will exhibit.

In this particular dynamic MRS occupancy grid mapping system, the K value denotes the number of neighbouring robot agents that are exchanging sensory and position information. When the K value is 0, it means that agents are not exchanging any information with the surrounding agents in the swarm. The repulsion value is another important factor that can be adjusted according to which end of the spectrum one wishes the system to lean towards. This repulsion value represents the minimum radius between the agents that are part of the swarm [7] at any given moment. Generally speaking, increasing the repulsion value will result in a more explorative system that will be forced to spread out as it maps the environment. The final variable that will have an effect on the exploration-exploitation nature of the swarm is the density. This variable is dependent on the total number of agents in the swarm and the total surface area of the 2D environment which is being mapped.

$$Density = \frac{Total\ agents}{Surface\ Area\ in\ m^2}$$

While the K value and the repulsion have been researched heavily in previous papers, the density is a variable that has very little data on it. To fully optimize the swarm performance, the relationship between the swarm variables needs to be researched further through testing and analysis of the results.

Design of experiments is a branch of applied statistics which uses an organized approach that connects experiments in a rational manner. The influence of interactions between all factors can be estimated by performing a series of experiment trials where the minimum amount of trials are performed with the goal of acquiring the most precise information possible [4]. This allows for multiple input factors to be manipulated, and their effect on a desired output should be determined

thereafter. There are several different methods that can be used to create response surfaces. For example, the central composite circumscribed design (CCD) provides high quality predictions over the entire design space, but require factor settings outside the range of the factors in the factorial part [8].

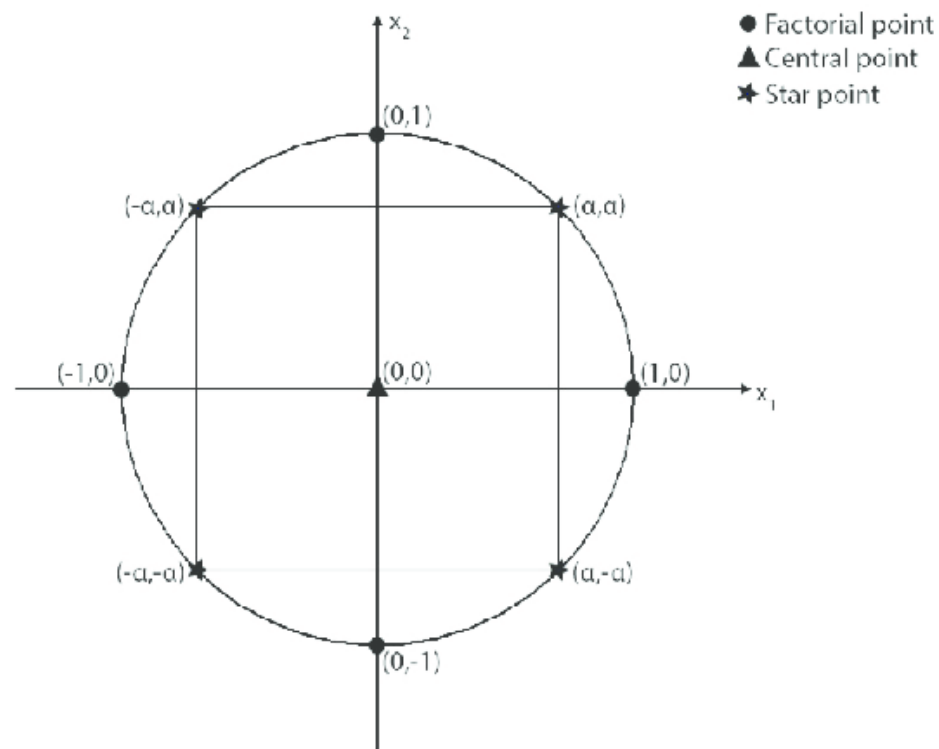


Figure 1: Graphical representation of the Central Composite Design method [1]

The CCD method is often used to create a dataset which can then be graphically analyzed, and in some cases it can also be numerically analyzed.

2.2 Existing Technologies and Related Works

Bio-inspired swarm systems have often been used to create swarm algorithms that are used to complete tasks such as mapping, stocking tracking, etc. These algorithms usually imitate the collective behaviour that can be found in nature, such as worker ants working together to ensure

their colony's survival. There are many biology-inspired swarm behaviours such as aggregation, flocking, exploration, collective transportation, and foraging [9]. These algorithms are normally characterized as distributed, which means that agents only exchange information with other agents when they are neighbouring or are very closeby [9].

In a research project performed at the Université Libre de Bruxelles, researchers compared the different types of algorithms which correspond to the movement of the robots. The most promising method of motion for the robots was the ballistic motion, as it yielded the best maps in the smallest amount of time needed. A robot with ballistic motion moves in a straight line until it detects an obstacle, then changes its direction at random [10].

Li *et al.* proposed a coordinated distributed algorithm called the Brain Storm Optimization (BSO)[11]. In this algorithm, the robots use local perception and communication to decide the next direction in which the robots will turn to, by sharing the data about the environment sensed by each robot.

This thesis paper makes use of the occupancy grid mapping (OGM) technology. Most of the existing OGM technologies that exist in the real world today are meant for static environments. Static applications do not fully realize the potential of this technology and can tend to produce some over-engineered solutions. One of the most difficult aspects of properly implementing swarm technology is to find benchmark metrics to compare two different swarming algorithms [2]. OGM in particular refers to a family of computer algorithms in which the problem of uncertain or noisy sensor data is eliminated. A map of a 2D environment can be represented by an occupancy grid or a Bayes filter. The grid gives you the probability map of the obstacles. If a particular cell has a probability close to 1, then it means that it is very likely that the cell contains an obstacle. Probability close to 0 will indicate a likelihood that the cell is unoccupied, and the 0.5 mark indicates that the contents of the cell are unknown.

3 Methodology

3.1 Initial Code

The dynamic swarm mapping code had been previously developed by a research team led by Jabez Leong Kit from the Singapore University of Technology and Design [2]. The simulations were created to gain response data for a MRS, which can be done by running several automated tests at one time to evaluate the performance of the swarm. Within each simulation, a two-dimensional grid of cells is created. Each cell is characterized as one of the 3 possible states: occupied, empty, or unknown. The determination of the contents of a cell is calculated through probabilistic calculations using the noisy and uncertain LiDAR sensor data. In this report, the number of robot agents will be held at a constant number of 16. There will normally be 2 square-shaped pillars which act as obstacles in the map. The goal of the swarm is to map the obstacles as quickly and accurately as possible. The agents are initially spread out evenly within the grid. The simulations can be exported into video clips which show all the iterations of the mapping process. A screenshot of the initial starting position of the pillars and robots is shown in the following figure.

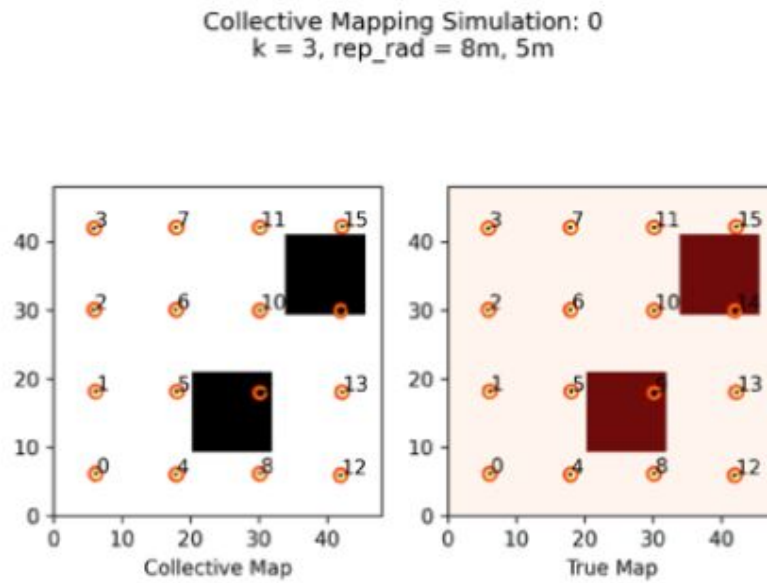


Figure 2: Screenshot of simulation video at initial starting positions, where pillars are the square blocks and the orange circles represent robots

The simulations keep track of two maps, which are the collective map and the true map. The true map displays and keeps track of what the actual pillar positions are at any given moment. The collective map displays what the swarm of robots are tracking as the pillar positions. The pillars are denoted by the black-filled squares. Initially, the collective map and the true map are identical, as it is assumed that the swarm has already perfectly mapped the area. Then, at the 50th iteration, an instantaneous dynamic change in the pillar position will occur. The collective map will not yet have registered the change at this time, so its pillar positions have not changed yet.

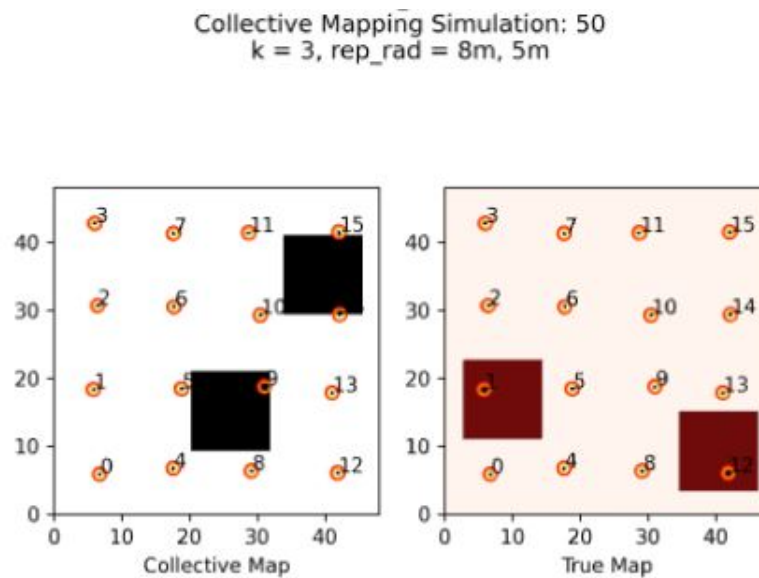


Figure 3: Screenshot of simulation video at iteration 50, where a dynamic change in the environment has suddenly occurred and the collective swarm has not yet registered the change

From iteration 50 and onwards, some agents in the swarm will sense that certain cells are either suddenly empty or suddenly occupied. This should prompt the surrounding agents to either exploit the already available information, or they shall start to explore the unknown areas of the grid. As iterations roll on, eventually progress will start to be made as the swarm attempts to recover from the dynamic change to the environment.

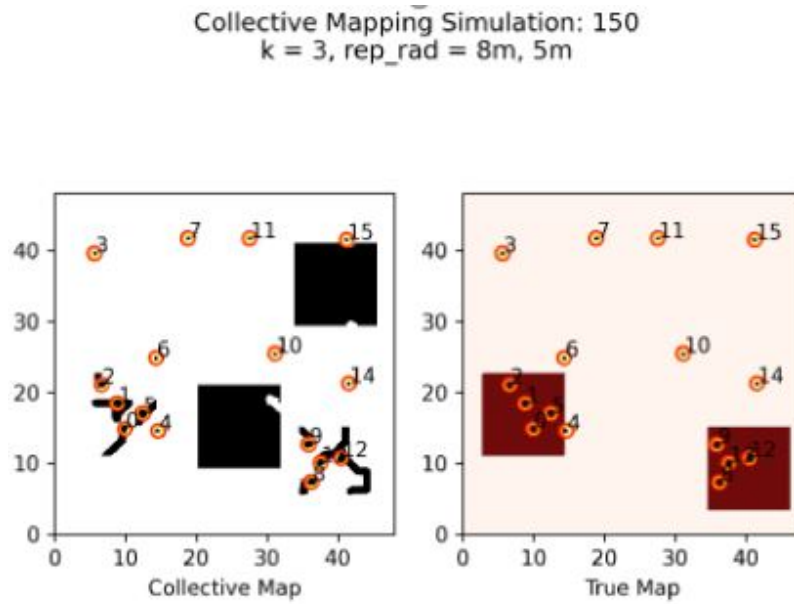


Figure 4: Screenshot of simulation video at iteration 150, where the swarm is in the midst of recovering from the dynamic environment change and the collective swarm is beginning to match the true map as it continues its mapping process.

In the initial code, there is only one response metric that is logged. The metric is known as the count of discovered cells and it represents the number of cells which change their state throughout the simulation.

3.2 Modifications to Code

There were a few changes that were made to the code to automate certain variables and to track some metrics throughout a simulation. In particular, one of the main changes was to use density as a parameter when running a simulation. The code initially sets the number of agents and the grid length as hard-coded values which are usually not meant to change in a batch of simulation experiments. It can become quite simple to make the density a parameter by setting the number of agents to a constant value of 16, and allowing the grid size to vary according to the density value.

A sample calculation of the grid length is provided below, assuming the input parameter of density, D , is given a value of $0.0001 \text{ agents}/m^2$. The number of agents is denoted as N and the grid length of the environment is denoted by L .

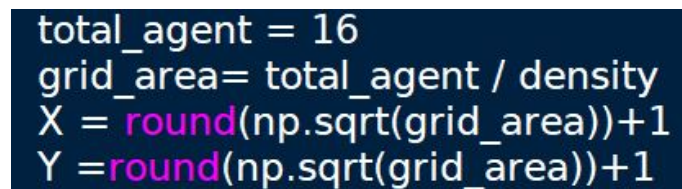
Given:

$$D = 0.0001 \text{ agents}/m^2, N = 16$$

The length of square grid environment would be calculated as follows:

$$L = \sqrt{\frac{N}{D}} = \sqrt{\frac{16 \text{ agents}}{0.0001 \text{ agents}/m^2}} = 400m$$

This modification to the code is shown in the following figure.



```
total_agent = 16
grid_area = total_agent / density
X = round(np.sqrt(grid_area))+1
Y = round(np.sqrt(grid_area))+1
```

Figure 5: Screenshot of code which calculates the grid length X and Y given the input parameter *density*

It also became evident that it would be important to track the swarm variables; density, K value, Repulsion. The response metric, which is the count of discovered cells, should also be logged at the final iteration of each simulation.

A Python script was created to pull the swarm variable values from the file name when each simulation is run. The count of discovered cells was kept track of using an external logging file which was created for each simulation. A screenshot of the python code is shown in the figure below.

```

import os

foldername = '250_reset'
response = open('/home/sbahia/Documents/OGM/Responses/response_%s.txt'%(foldername),"a")

for fole in sorted(os.listdir('/home/sbahia/Documents/OGM/Post_Process/%s'% foldername)):
    lastline= open('/home/sbahia/Documents/OGM/Post_Process/%s/%s/dynamic_change.dat'% (foldername, fole)).readlines()
    dcc_value = lastline[-1].split(":")[1]
    k_value = fole[fole.index("k=")+2:fole.index("k=")+4]
    #r_value = fole[fole.index("r=")+2:fole.index("r=")+4]
    d_value = fole[fole.index("d=")+2:]
    response.write(d_value + ' | ' + k_value+ ' | '+str(dcc_value)+"\n")

```

Figure 6: Screenshot of code which sends the swarm variables and the response values to a text file

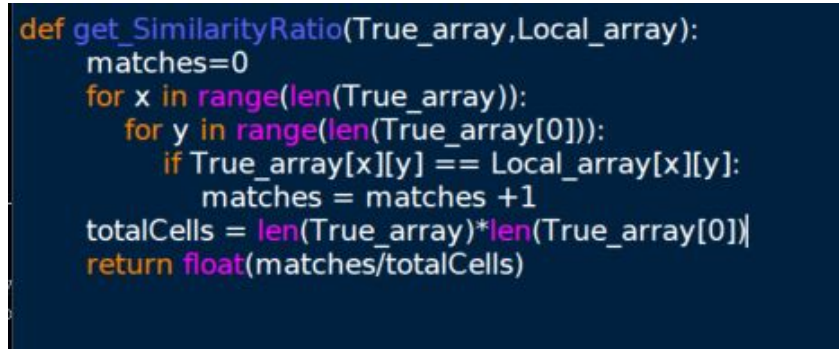
As previously mentioned, the initial code uses one method for quantifying the response of the swarm, which is counting the number of discovered cells. The issue with this metric is that it does not give any indication of how well the environment has actually been mapped at any given time. If one is given the response metric of 15000 total discovered cells, that would not indicate whether the map has been fully mapped or not at all. To workaroud this, a new metric can be created by comparing the collective map and the true map at every iteration, and seeing how identical they are. If perfectly mapped, the collective and true maps would be exactly matching. By calculating how many matching cells and comparing it to the total amount of cells in a grid, one could create a ratio which allows the mapping of the swarm to be tracked in real time. The name of this metric will be called the **similarity ratio**, $R_{similar}$. The formula for this ratio will be defined below in equation 1, when the number of total cells on the grid is denoted as C_{total} . The number of matching cells between the collective and true maps is denoted by $C_{similar}$.

$$R_{similar} = \frac{C_{similar}}{C_{total}} \quad (1)$$

The similarity ratio allows one to better examine the actual mapping progress of the swarm at any given iteration point. The point of maximum mapping potential caps at the value of 1, so the

ratio will always be between 0 and 1. Using a ratio with potential values of only between 0 and 1 benefits the analysis of the results because the response metric will now provide a more relatable quantitative representation of how well the swarm has mapped the environment at any given point.

A Python function was added to the simulation code file which calculates the similarity ratio. The function is called in each iteration loop and it sends the value to a text file after each iteration. Then, this text file can be imported to an excel sheet or to MATLAB where it can be easily plotted in a line graph.



```
def get_SimilarityRatio(True_array,Local_array):
    matches=0
    for x in range(len(True_array)):
        for y in range(len(True_array[0])):
            if True_array[x][y] == Local_array[x][y]:
                matches = matches +1
    totalCells = len(True_array)*len(True_array[0])
    return float(matches/totalCells)
```

Figure 7: Screenshot of python function which is called at each iteration to calculate the similarity ratio

Not all changes in the code were documented in this report, however the important modifications were discussed in this section because they can help explain simulation results.

4 Simulation Results

4.1 Testing K and Density With Scaled Pillar Size

As previously mentioned, it was deemed infeasible to adopt a brute force strategy since a simulation of 250 iterations can take upwards of 15 minutes at a time. Therefore, a small, efficient dataset must be created using a DoE method. In this context, efficiency is defined as using the least amount

of simulations to provide the maximum amount of new information. The central composite method will be used in this dataset. Since repulsion has been tested extensively in other research projects, the density and the K value are the two factors that shall be tested in this trial. Assuming there is a desire to complete the simulations within approximately 5 hours, 22 trials were chosen in the dataset. While there are normally 5 levels in a CCD dataset, 2 more levels (0.5 and -0.5) were added so as to complete a more comprehensive dataset for a total of 7 levels.

	Level	Density	K
Lower	-1	0.0000694	4
Higher	1	0.0001560	13
High Alpha	1.414	0.0001739	15
Low alpha	-1.414	0.0000515	2
zero	0	0.0001127	9
Lower Half	-0.5	0.0000911	6
Higher Half	0.5	0.0001344	11

Figure 8: Table for setting the 7 levels in the CCD method

Run	Density	K	Response
1	-1	-1	
2	1	-1	
3	-1	1	
4	1	1	
5	0	0	
7	-1.414	0	
8	1.414	0	
9	0	-1.414	
10	0	1.414	
11	-1.414	-1	
12	-1.414	1	
13	1.414	-1	
14	1.414	1	
15	-1	-1.414	
16	1	-1.414	
17	-1	1.414	
18	1	1.414	
19	0.5	0.5	
20	-0.5	-0.5	
21	0.5	-0.5	
22	-0.5	0.5	

Figure 9: CCD dataset with the corresponding levels for the 22 trials

The 2D graph can be visualized in the following figure.

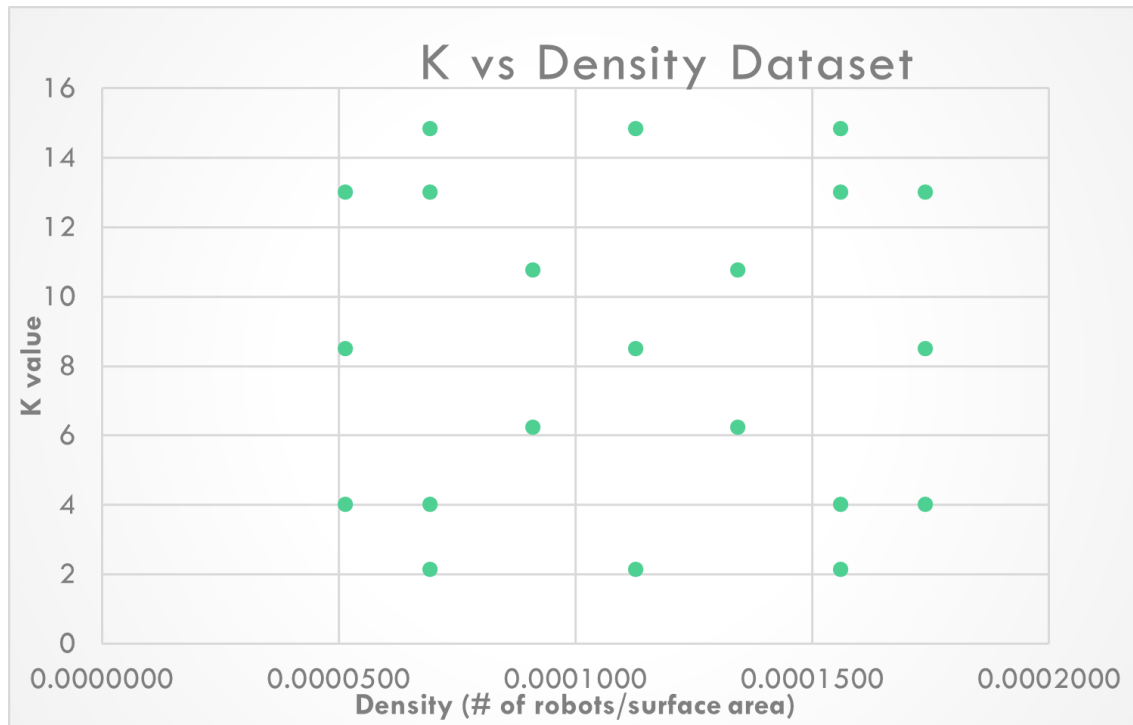


Figure 10: Graphical representation of the modified CCD dataset

These datapoints should be graphed as a 3D surface with the third dimension being the response. The response metric that will be used in this instance is the total number of new cells discovered by the swarm.

It is important to note that when the density is changed in this particular trial, the size of the pillar is being scaled to the environment of which it exists within.

$$L_{pillar} = L_{grid} \times 0.2449 \quad (2)$$

4.1.1 2-pillar Simulations

The program can then be executed with the current dataset. The results were plotted with the MATLAB curve fitting tool to produce the following graph and contour plot.

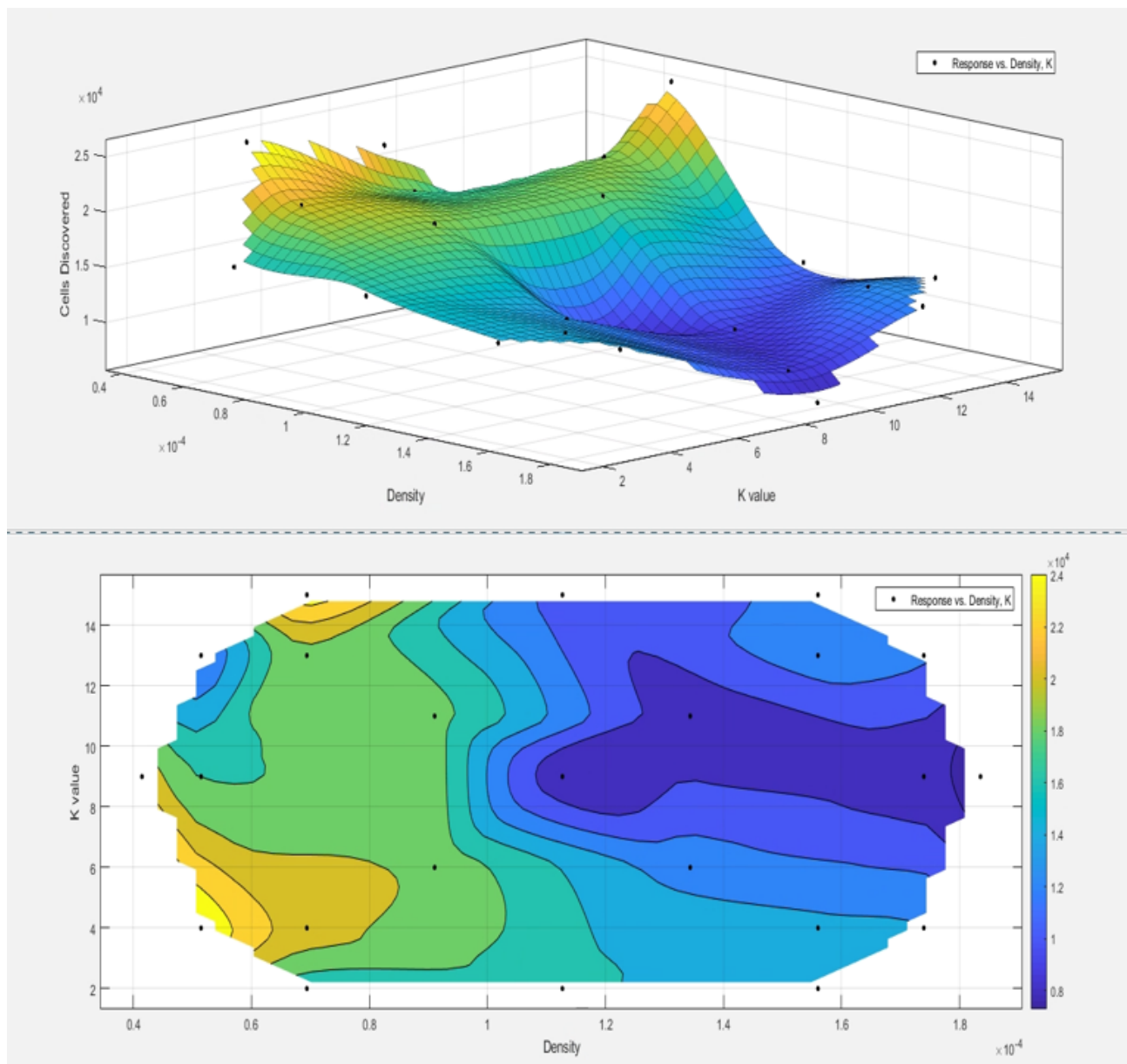


Figure 11: 3D graph with results for the 2-pillar environment with 400 iterations

These results seem to display a fairly consistent pattern, albeit with a small dataset. The results indicate that an increase in the density results in worse performance, according to our response metric. The dark blue strip on the contour plot also indicates that a K value in between 10 and 8 does not yield good results. Overall, not much can be taken away from looking at these results due to the lack of datapoints and the inconsistency of results.

4.1.2 1-pillar Simulations

Since the number of obstacles can also theoretically vary depending on the environment, it was also deemed prudent to perform the same experiment, however this time only with 1 pillar as opposed to 2 pillars in the previous trial. A few simple changes to the code can allow the number of pillars to be a parameter when executing a run of swarm mapping. The dataset for this trial will be identical as the previous, and the differences between the two will be analyzed.

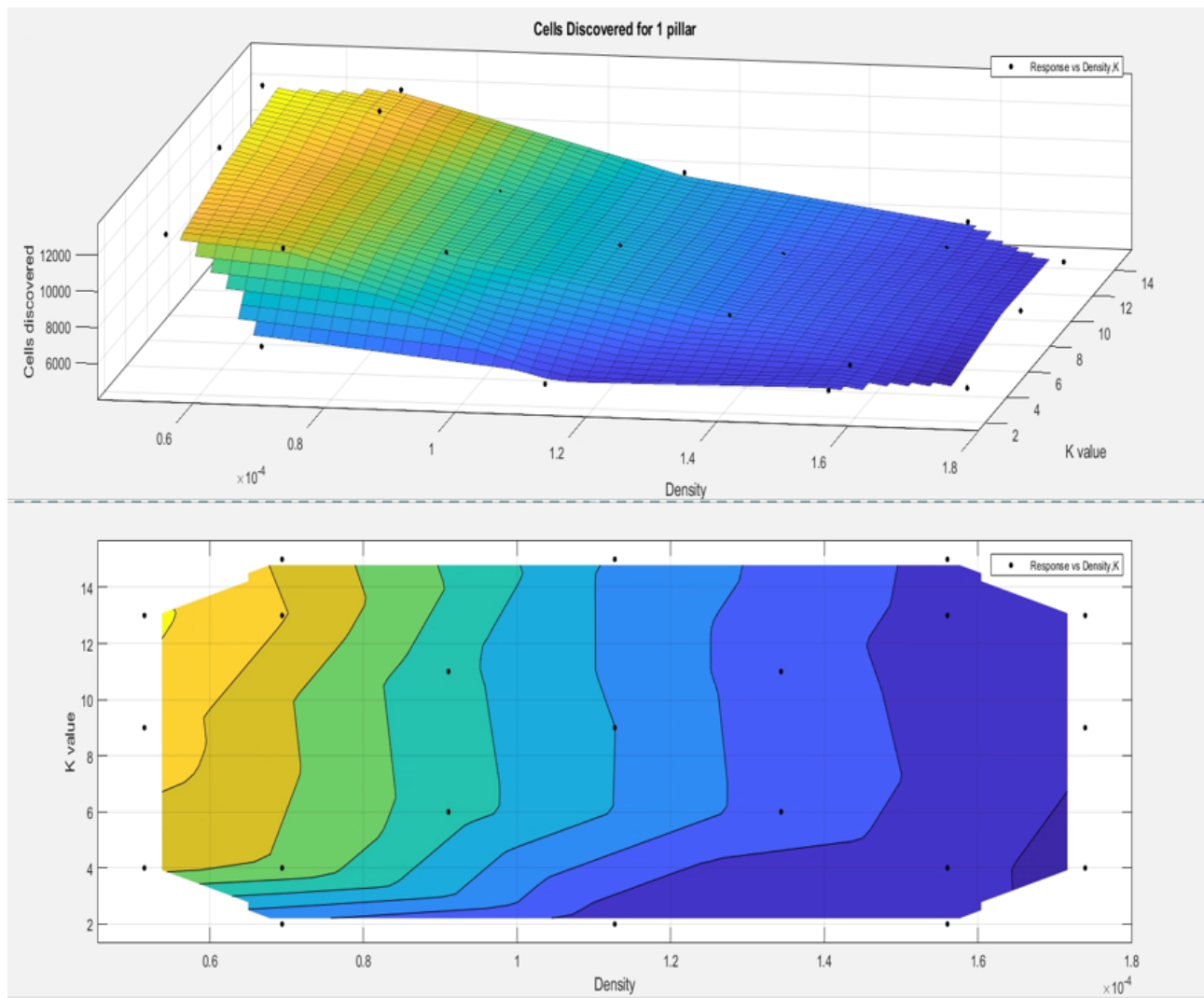


Figure 12: 3D graph with results for the 1-pillar environment with 400 iterations

There is a clear shift in the consistency of these results when there is less pillars (obstacles) in

the environment. In fact, the surface appears to be almost flat, which is different than the concave looking surface in the 2-pillar trial. It is also clear that while results are poor when K is very low, in general the K value seems to have very little effect on the performance of the swarm in this scenario.

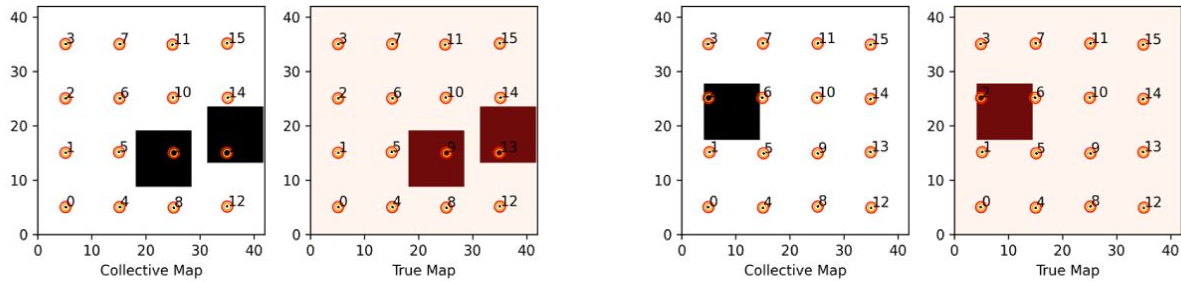


Figure 13: Side-by-side screenshots of simulation clips for the 1-pillar and 2-pillar environments

The results indicate that there is less need for communication between robots when there is less obstacles. Therefore, the density essentially becomes the determining factor for the performance of this trial, so long as the K value does not drop too low.

4.2 Testing K and Density with Non-Scaled Pillar Size

It is important to note that the previous results do not match the theoretical assumption that an increase in density would result in improved performance. As referenced in equation 2, the previous trials used scaled pillar sizes. However, if one wants to accurately depict a change in the density of an environment, the pillar sizes should remain constant throughout, while the size of the grid varies. The pillar sizes can be made constant with a quick adjustment to the code. The simulation clips show what visual effect this new change has on the swarm mapping environment.

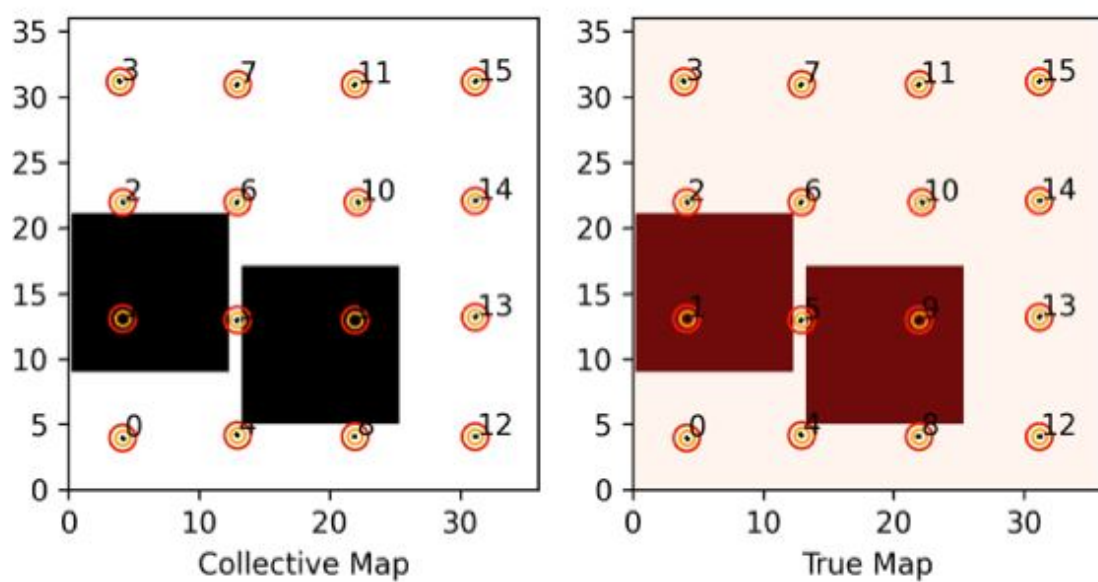


Figure 14: 2-pillar environment with a scaled pillar at a density of 0.000123

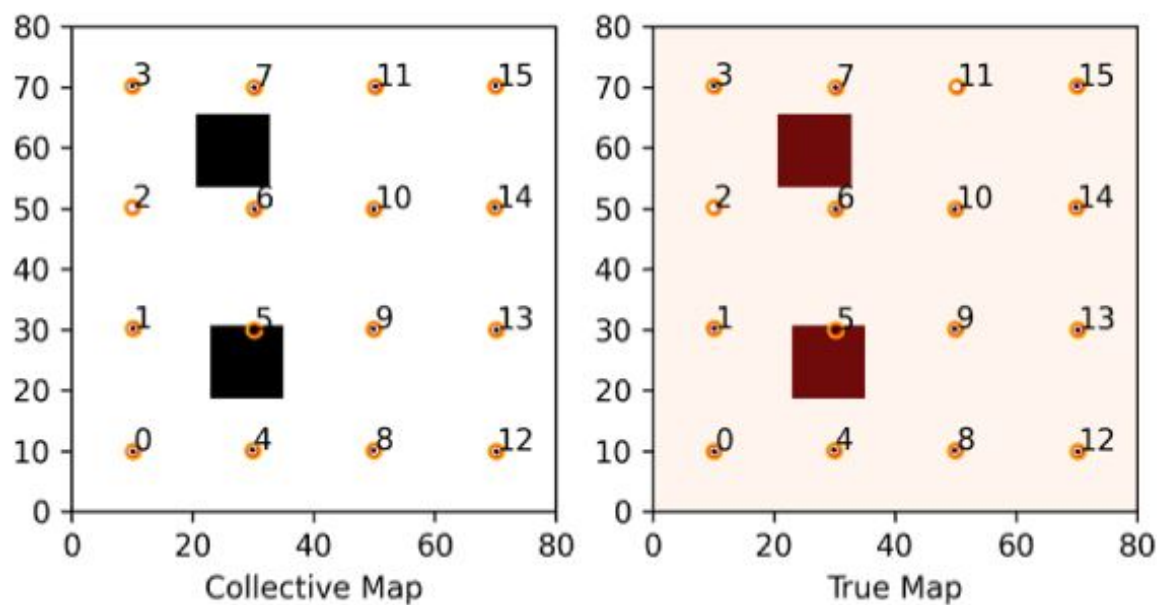


Figure 15: 1-pillar environment with a non-scaled pillar at a density of 0.000025

Next, it was deemed appropriate to run a similar test to the previous, however with slightly different dataset. To counteract the inconsistent results that were present in Figure 12 each datapoint will have 5 executed simulations, each at a different K value. The repulsion will remain fixed at the default value of 8m in this experiment. It is important to note that the pillar size also remains fixed throughout this experiment to ensure an accurate depiction of density throughout testing.

Runs	Density	K Value				
		0	4	8	12	16
1	0.000022145	4477	7151	7483	6305	6240
2	0.000027706	3548	7543	9291	6504	7991
3	0.000033267	2986	11370	8236	6150	8033
4	0.000038828	1762	9600	14577	10754	3916
5	0.000044390	4439	12472	9831	9725	6689
6	0.000049951	5479	17748	5785	14865	15078
7	0.000055512	5429	12042	12541	17749	4668
8	0.000061073	6451	18111	11100	10303	5248
9	0.000066634	4521	12722	9611	11258	12763
10	0.000072195	3649	16714	17082	16481	16519
11	0.000077756	1958	12781	13193	15228	5781
12	0.000083317	4755	10852	5703	16053	9884
13	0.000088878	2762	11721	14541	6850	15006
14	0.000094439	7883	8428	11390	7253	12118
15	0.000100000	2648	13404	10105	7593	7575

Figure 16: Dataset for the trial which includes 5 different values of K and 15 different densities (response is in yellow cells)

The trial was executed and the results are shown in the following figure.

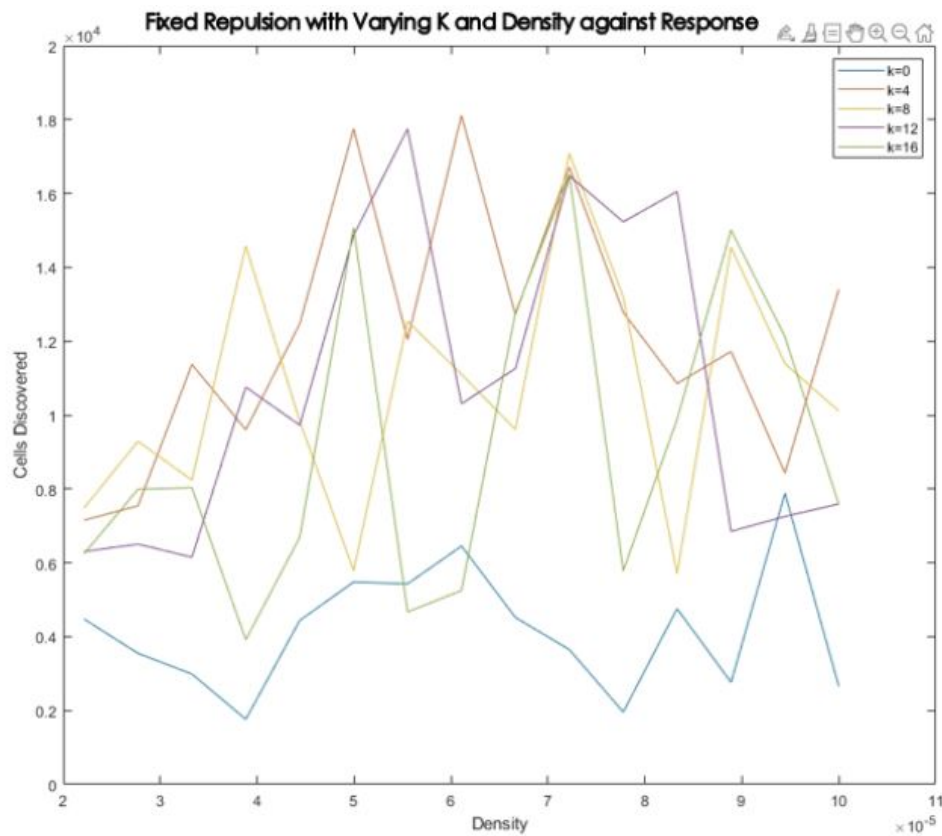


Figure 17: Line Graph which depicts results for the dataset with fixed repulsion and different values of K with 15 different densities

Unfortunately, these results have become a bit scattered and inconsistent that they become slightly difficult to analyze. The only takeaway that can be reinforced from our previous trials is that a low K value such as 0 results in very poor performance since there is little to no communication between agents.

4.3 Scaling Repulsion and Fixing K

There is a strong likelihood that the datasets which have been previously used in this paper were simply too vast to expect consistent results. While still ensuring to avoid a brute-force method, there are still ways to increase the knowledge we gain from these trials. From looking at the results

in this experiment along with previous research done by the others [2], it is generally known that a K value should remain at roughly 50% of the total amount of agents. Therefore, it would be plausible to fix the K value at 8 for a swarm with 16 agents. Additionally, the idea of fixing the repulsion may have to be revisited. If the size of the grid increases (density decreases), yet the amount of agents in the swarm remains constant, then it would not make logical sense to fix the repulsion. The agents will need to explore their environment more if the grid size increases, which means that the repulsion should increase by scaling it proportionally to the grid size. The factor of proportionality, ω , can be created by using the optimal values for the grid length and the repulsion, which are 8m and 480m respectively.

$$\omega = \frac{R_{default}}{L_{default}^2} = \frac{8m}{(480m)^2}$$

If one wishes to find the scaled repulsion of an environment with a given grid area, A_{given} , they can do the following:

$$R_{scaled} = \omega \times A_{given} \quad (3)$$

The following dataset can then be created with a scaled repulsion.

Run	Density	Repulsion (scaled to Area)	Average Response (cells discovered)
1	0.00002215	25	2494
2	0.00002771	20	4902
3	0.00003327	17	5343
4	0.00003883	14	8061
5	0.00004439	13	8582
6	0.00004995	11	10967
7	0.00005551	10	10379
8	0.00006107	9	15751
9	0.00006663	8	14639
10	0.00007219	8	10622
11	0.00007776	7	11629
12	0.00008332	7	11832
13	0.00008888	6	11177
14	0.00009444	6	9501
15	0.00010000	6	10686

Figure 18: Dataset with scaled repulsion to the area of the grid

There are 15 different density runs in this experiment, and each one is executed 5 times. The average response of those 5 simulations will be listed as the average response in the Figure 18 dataset. The results for this dataset are graphed in the figure below.

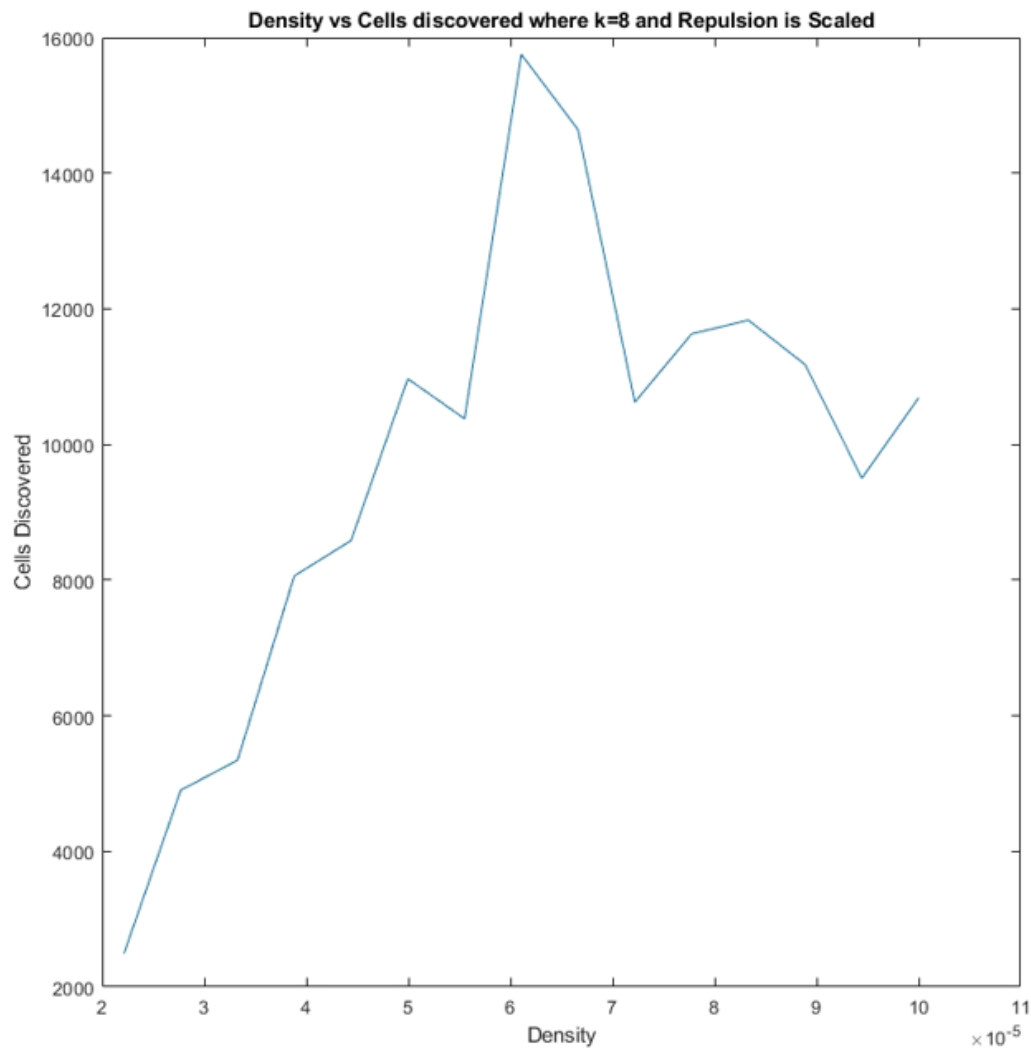


Figure 19: Result for simulation with 75 runs and scaled repulsion

These results provide a much more consistent trend than the previous results and it is more in line with the theoretical assumptions that were made. It demonstrates that as the density gets too low, the performance becomes very poor. This would seem to make sense since one would expect a large surface area to be more difficult to explore for a swarm, as the agents must spread out

to discover new cells. However, there is a noticeable change when the density reaches close the range of 6×10^{-5} to 7×10^{-5} . The performance spike occurs where the number of cells discovered reaches between 15000 and 16000 cells. The trend quickly disappears once the density increases past the aforementioned range as the number of discovered cells drops back down to roughly 10000 cells. This sharp drop-off in performance after a quick spike is an interesting and frankly unexpected result. The explanation for this could be: when there are too many robots in a smaller environment, the lack of space can cause skewed communication between the agents and therefore hinders the performance of the swarm. Before looking too much into these results, it is important to verify this trend with another test to add some more datapoints to the experiment. This time, the results were displayed in a box plot to give a better idea about the distribution of results. Box plots are specifically chosen as an avenue for representing data because they help visualize the distribution of quantitative values in a field of results [12]. It is also for viewing if there are outliers in the results.

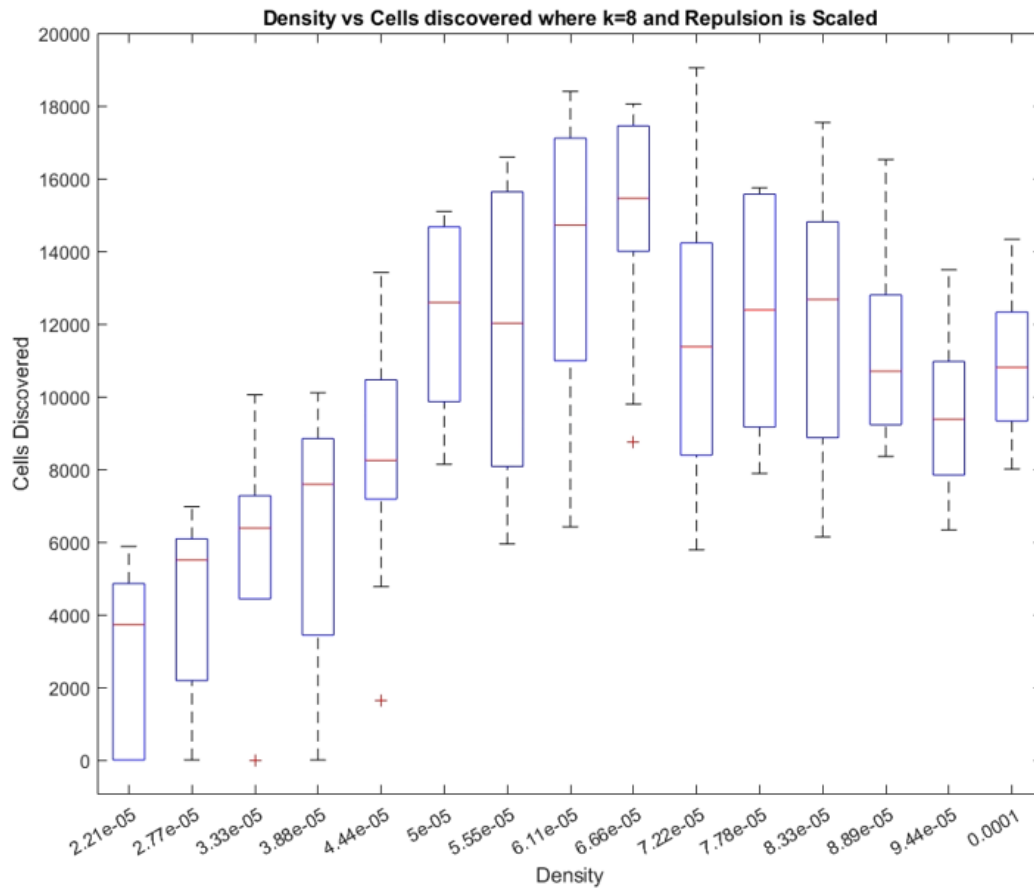


Figure 20: Box plot for results of 150 simulation runs of swarm mapping with scaled repulsion

Interestingly, the trend that was discovered in Figure 19 remains true. There remains a noticeable increase in performance around the density range of 6×10^{-5} to $7 \times 10^{-5} \text{ agents}/m^2$. This would seem to indicate there is an optimal density range which provides peak performance of the swarm at a given K value, while it was previously assumed that once the density reaches a high enough point, the performance would simply level out at its highest possible point.

Unfortunately, the box plot in Figure 20 appears to have quite a wide distribution of quantitative values, which indicates that perhaps the response metric could be too inconsistent. It can be argued that the discovery of new cells is entirely too dependent on the random nature of the dynamic change of the pillar placement in the swarm environment, which explains why the wide range of results occurs. It is also possible that the current response metric gives a better idea of the behaviour of

the swarm instead of how well the swarm is actually mapping the environment.

4.4 Testing Similarity Ratio

As explained further in the methodology section, a new response metric can be created for swarm mapping simulations which involve the idea of comparing the true map with the collective map. One can calculate exactly how many of the cells are identical and create a metric by dividing the identical cells by the total number of cells in the grid, as shown previously in equation 1.

Once the similarity ratio tracking has been added to the code, tests will be executed to see if this similarity ratio provides more consistent results than old metric. Before running a large dataset with several simulations, a singular simulation will be run to see the evolution of the similarity ratio over the course of an entire simulation. The following simulation is run with a K value of 8, the repulsion is set at 9m, and the density is set at $6.94 \times 10^{-5} \text{ agents}/m^2$.

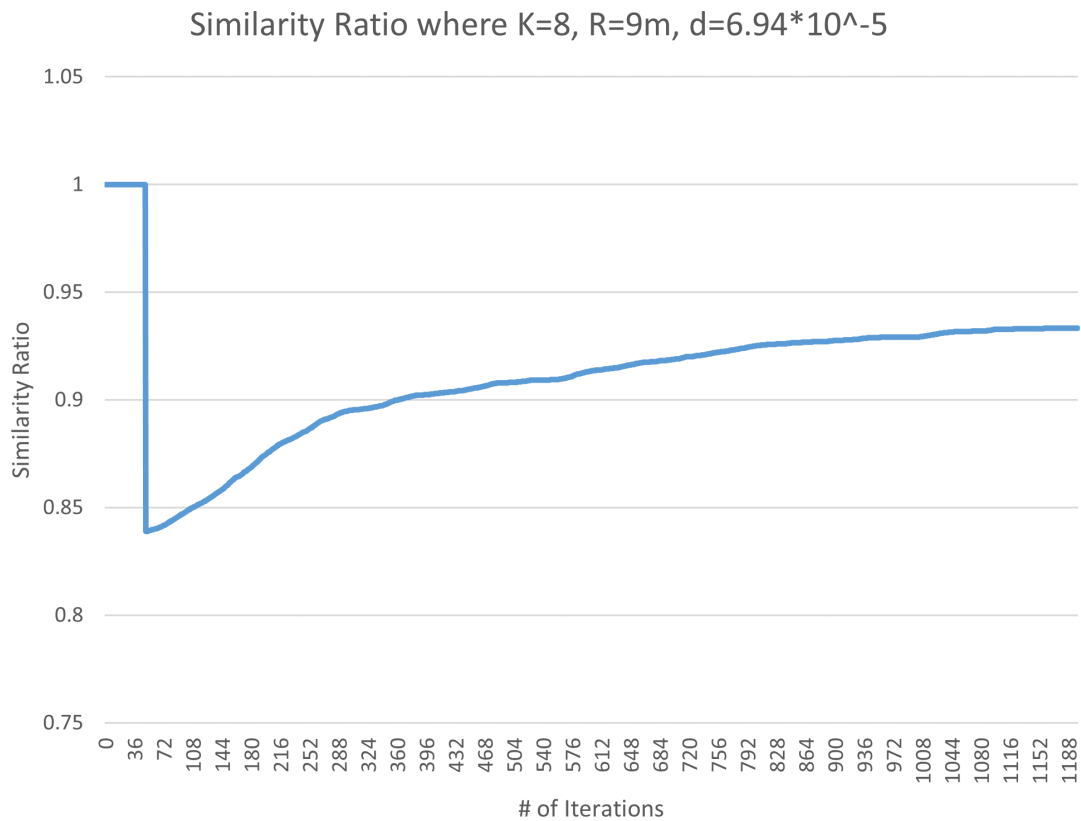


Figure 21: Line graph of simulation results using the new similarity ratio response metric

The true map and the collective map are identical until iteration 50, and then a sharp decline occurs in the mapping accuracy. This occurs because of the dynamic change of the placement of pillars which was shown in Figure 3. Immediately, the swarm begins to recover and maps the environment aggressively. The mapping does not seem to take place linearly, and the swarm slows down its progress and eventually stops being able to continue exploring the environment. It is likely that the swarm would benefit from increasing its explorative behaviour as the mapping begins to wane towards the 1000 iteration mark. So far, every run has had the K and repulsion remain constant throughout the entirety of the simulation, however this might indicate that those variables should change as the performance starts to wane.

4.5 Comparing Response Metrics

The dataset used for these tests will follow the central composite method once again, and the density will remain constant at $6.94 \times 10^{-5} \text{ agents}/m^2$.

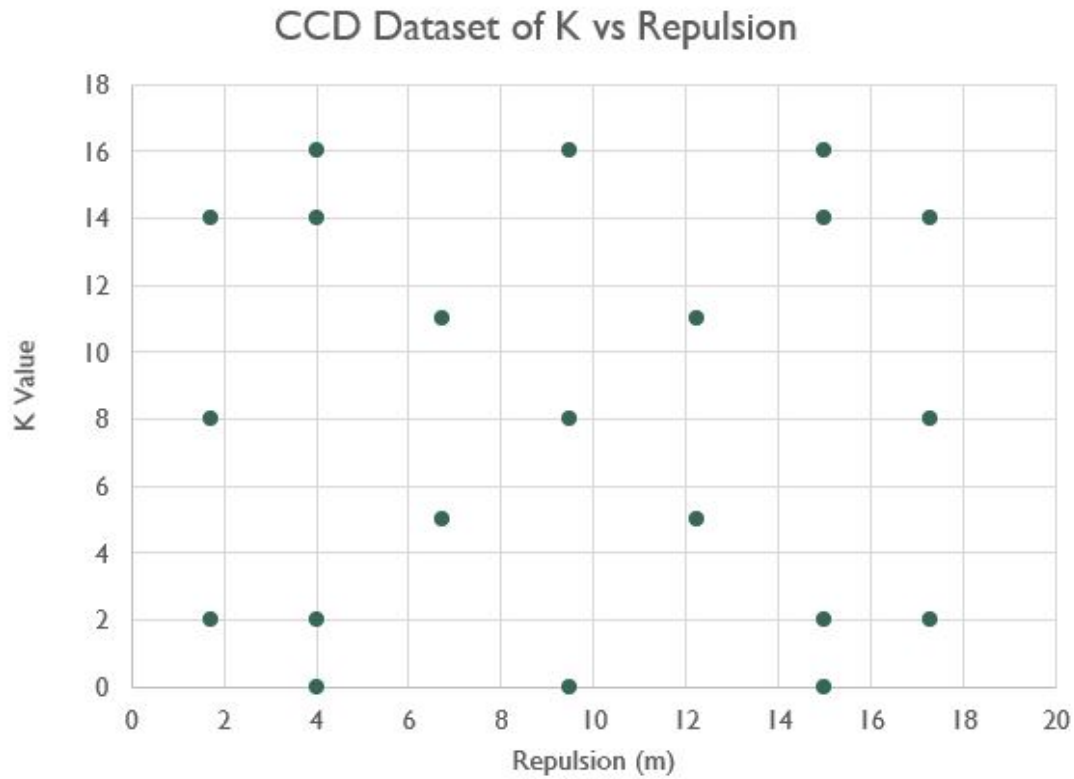


Figure 22: Central composite dataset with a constant density

Simulations will then be executed for the corresponding dataset for a total of 44 runs. Both response metrics, the new similarity ratio and the count of discovered cells, will be tracked for the duration of these tests. The two response metrics will be compared, and it will be useful to see which one provides the most consistent results. The following figure shows the results for the similarity ratio response.

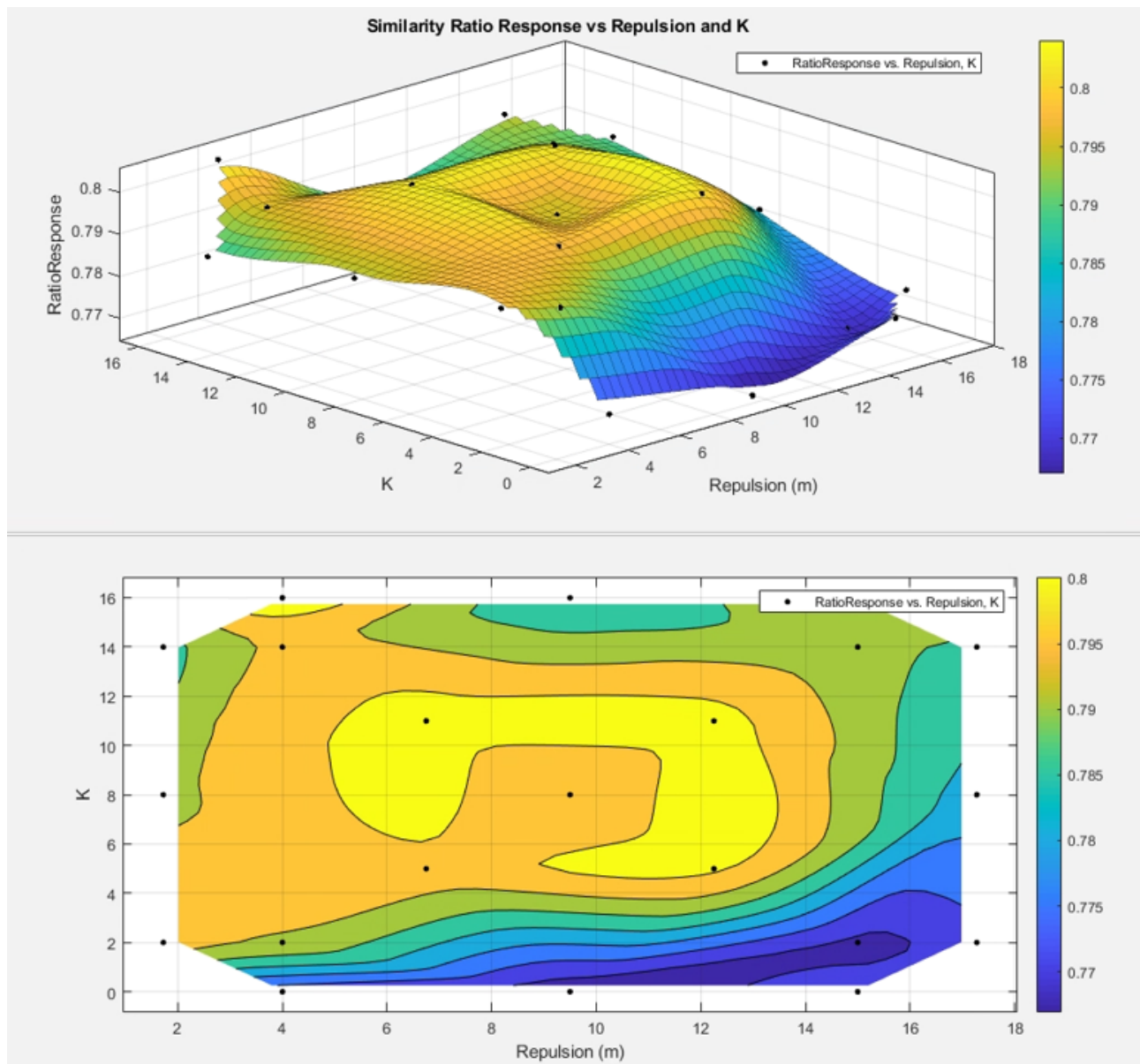


Figure 23: 3D response surface and contour plot where the similarity ratio response is being tracked

The peak performance of the swarm occurs when K is equal to 11 and the repulsion is approximately 12m, according to the similarity ratio response. Next, the following figure displays the results for the count of cells discovered metric.

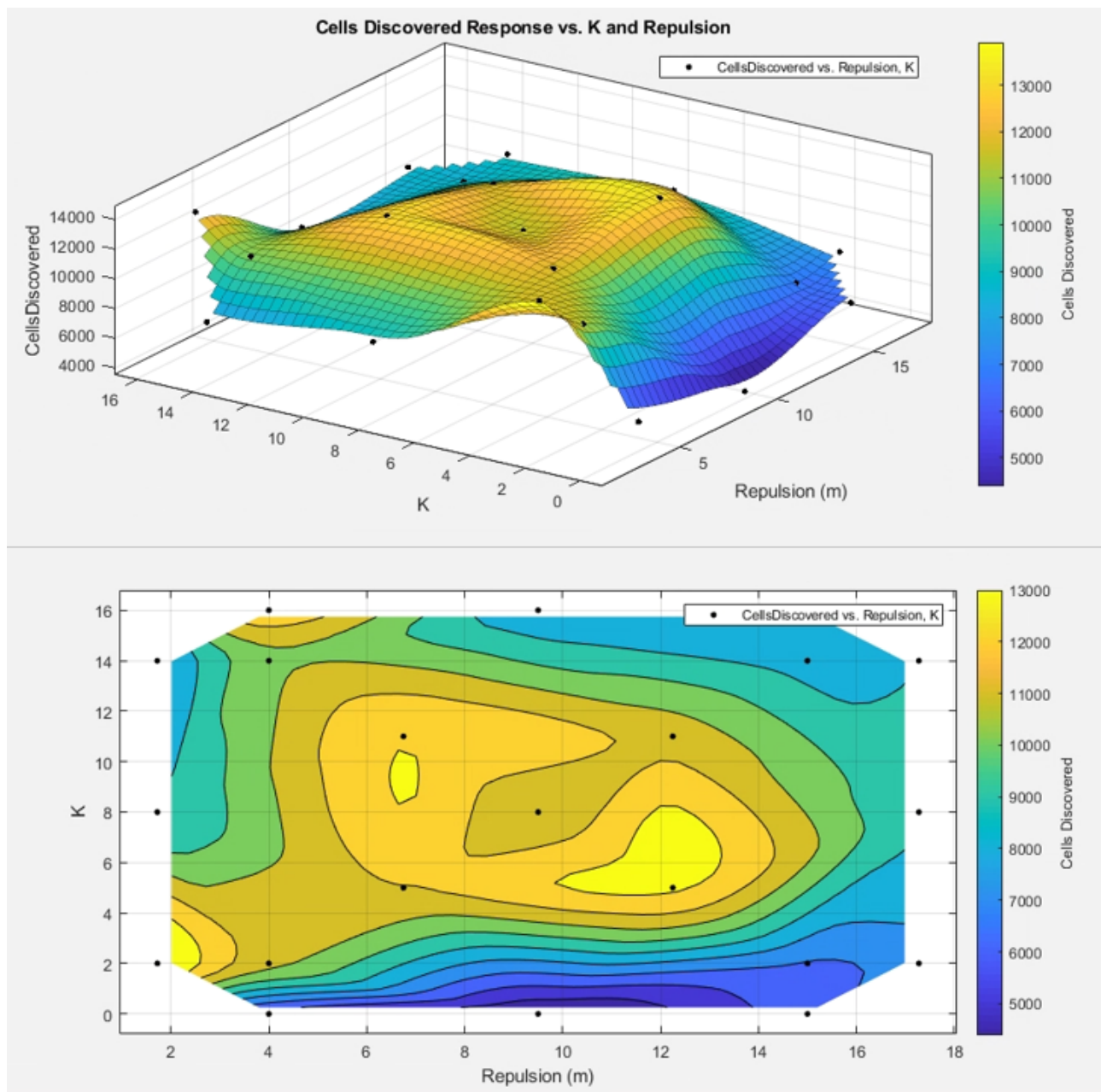


Figure 24: 3D response surface and contour plot where the cells discovered response is being tracked

Looking at the 3D response surfaces of both response metrics, they are both extremely similar in shape. This verifies that the cells discovered response does an accurate job of tracking how the swarm is behaving during its mapping task. The similarity ratio response does a better job of actually tracking how well the swarm is mapping at any given moment, and can give a numerical value which represents the mapping progress at a given iteration. The count of discovered cells itself

is quite meaningless and gives no indication to the mapping progress at a given moment, other than when it is being given in reference to another test.

5 Future Work

Due to time restraints and a lack of resources, it is not always possible to accomplish every idea that comes to mind. The world has been in a pandemic for the last year which forced this project to be completed remotely, so there was never a possibility to carry out swarm mapping experiments with real robots. In the future, real-life robotics experiments would be extremely useful because it usually allows researchers to gain real-world data instead of only simulation data. There are always external variables that play a factor in the real world that might not be apparent in simulations, hence the reason to create real robots and test their swarm capabilities in future works.

Machine learning is a concept that could be heavily implemented in these swarm mapping experiments to enhance the swarm intelligence. Machine learning is specifically useful in experiments where a large amount of trials are carried out to solve an optimization problem, which usually involves data fitting [13]. The main factor which limits the use of machine learning algorithms in this project is the length of time required to complete a simulation experiment. For a machine learning algorithm to be of any use, it usually requires very large datasets which provide much more response data. In the future, it would be prudent to allocate resources and time towards reducing the time length of each simulation run. This would allow much more options to analyze the data since the small datasets used in this experiment proved to provide inconsistent, and sometimes unreliable, results. There is some further work that can be done for the analysis of the similarity ratio to build upon the findings of this paper. Figure 21 showed that the similarity ratio provides real-time information about the mapping progress of the swarm. Once the dynamic change has occurred, the

swarm begins mapping at a pace which can be quantified by the slope of the similarity ratio over the course a few iterations. The non-linear slope noticeably levels out as the iterations continue. Since the swarm maps at a non-linear pace, it would indicate the swarm variables such as repulsion and K value should not remain constant. Logically, if the slope begins to become less steep for an extended amount of iterations, there is most likely a lack of exploration within the swarm. The swarm variables should then adjust to air on the exploration side of the spectrum, by increasing the repulsion and the K value. When the slope begins to steepen again, the swarm should focus on exploiting its new-found information, and adjust its swarm variables accordingly. An algorithm could be created to implement this idea, and machine learning could also be a useful tool when carrying out an experiment such as this (provided that simulation time length can be reduced).

6 Conclusion

Going back to the initial goals of this research project, there was a desire to better understand the quantification of the swarm response, as well as to gain an understanding of how to optimize its performance. The two major findings of this thesis project involved the testing of density in the swarm environment and the new response metric, the similarity ratio.

Through extensive simulation experiments that were carried out in section 4, it is shown that a continuous increase in density starts to work against the system at a certain point, and the effectiveness of the mapping decreases. It is apparent that there is an optimal density range for swarm dynamic mapping. There are further tests which must be carried out to find the optimal density for each situation, however this new information is critical for deciding which automated tests should be carried out in future research.

There has been new progress made in terms of better quantifying the performance of the swarm.

While tracking the count of discovered cells remains a useful response metric, there was value in creating the new similarity ratio metric for tracking the real-time mapping progress of the system. Using a ratio value between the range of 0 and 1 becomes a more relatable response metric for those who wish to analyze the mapping progress of the swarm. In the future works, the creation of the similarity ratio can be built upon by using the slope of the ratio to adjust the swarm variables such as repulsion and K throughout each simulation run, instead of keeping constant swarm variable throughout the entirety of a simulation. It becomes quite evident that to obtain the optimal swarm response, the complex system must consistently adjust itself on the spectrum of exploration-exploitation based on the real-time response data that it is receiving. It is important to note that the similarity ratio does not necessarily maintain a decentralized system, since it would be relying on a central controller to keep track of a true map and collective map. In the long term research studies for this project, there must exist an improved way to keep track of the response without becoming a centralized complex system. However, for the short term future, it is still appropriate to use the count of discovered cells and the similarity ratio to analyze simulation response data, so as to improve the understanding of how the swarm performs.

Bibliography

- [1] Develve, “Central composite design,” Available at <https://develve.net/Central%20Composite%20design.html> (2020).
- [2] J. L. Kit, “Towards a swarm intelligence benchmark framework,” Ph.D. dissertation, Singapore University of Technology and Design, Singapore, 2022.
- [3] M. Technologies, “Robot uses for tracex,” Available at <https://morphtec.com/products/tracex/robot-uses/> (2022).
- [4] Weibull, “Introduction to design of experiments (doe),” Available at <https://www.weibull.com/hotwire/issue84/hottopics84.htm> (2008).
- [5] Merriam-Webster, “Intelligence,” Available at <https://www.merriam-webster.com/dictionary/intelligence> (2018).
- [6] H. L. Kwa, J. L. Kit, and R. Bouffanais, “Balancing collective exploration and exploitation in multi-agent and multi-robot systems: A review,” *Frontiers in Robotics and AI*, vol. 8, 2021.
- [7] H. L. Kwa, J. L. Kit, and R. Bouffanais, “Optimal swarm strategy for dynamic target search and tracking,” in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 672–680.
- [8] E. S. Handbook, “Comparisons of response surface designs,” Available at <https://www.itl.nist.gov/div898/handbook/pri/section3/pri3363.htm> (1987).
- [9] X. Huang, F. Arvin, C. West, S. Watson, and B. Lennox, “Exploration in extreme

- environments with swarm robotic system,” in *2019 IEEE international conference on mechatronics (ICM)*, vol. 1. IEEE, 2019, pp. 193–198.
- [10] M. Kegeleirs, D. Garzón Ramos, and M. Birattari, “Random walk exploration for swarm mapping,” in *Annual conference towards autonomous robotic systems*. Springer, 2019, pp. 211–222.
- [11] G. Li, D. Zhang, and Y. Shi, “An unknown environment exploration strategy for swarm robotics based on brain storm optimization algorithm,” in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 1044–1051.
- [12] D. Friedman, “When to use box plots,” Available at <https://dfrieds.com/data-visualizations/when-use-box-plots.html> (2018).
- [13] H. Bostani, “Machine learning algorithms for optimization problems?” Available at https://www.researchgate.net/post/machine_learning_algorithms_for_optimization_problems (2018).