Developing a Radiating L-shaped Search Algorithm for NASA Swarm Robots

Tariq Tashtoush*[1], Jalil Ahmed[2], Valeria Arce[3], Heriberto Dominguez[4], Kevin Estrada[5], William Montes[6], Ashley Paredez[7], Pedro Salce[8], Alexia Serna[9], Mireya Zarazua[10]

School of Engineering
Texas A&M International University
Laredo, TX, 78041 USA

***Abstract*—This paper focuses on designing a search algorithm that the DustySWARM team used in the 2019 NASA Swarmathon competition. The developed search algorithm will be implemented and tested on multiple rovers, a.k.a. Swarmies or Swarm Robots. Swarmies are compact rovers, designed by NASA to mimic Ants behavior and perform an autonomous search for simulated Mars resources. This effort aimed to assist NASA's mission to explore the space and discover new resources on the Moon and Mars. NASA's going-on project has the goal to send robots that explore and collect resources for analysis before sending Astronauts, as the swarm option is safer and more affordable. All rovers must utilize the exact algorithm and collaborate and cooperate to find all available resources in their search path and retrieve them to the space station location. Additionally, swarmies will autonomously search while avoiding obstacles and mapping the surrounding environment for future missions. This algorithm allows a swarm of six robots to search an unknown area for simulated resources called AprilTags (cubes with QR codes). The code was developed using C/C++, GitHub, and Robotics Operation Systems (ROS) and tested by utilizing the Gazebo Simulation environment and by running physical trials on the swarmies. The team analyzed a few algorithms from previous years and other researchers then developed the Radiating L-Shape Search (RLS) Algorithm. This paper will summarize the algorithm design, code development, and trial results that were provided to the NASA Space Exploration Engineering team.**

Keywords*—NASA Swarmathon competition; swarm robotics; search algorithm; autonomous; Robot Operating System (ROS); NASA space exploration; simulation; autonomous robot swarm; collaborative robots; autonomous behavior; cooperative robots; swarmies; L-Shaped search; GitHub

I. INTRODUCTION

National Aeronautics and Space Administration (NASA) has been leading the space exploration since before humans even landed on the moon. The first probes and satellites are designed to be unmanned. Nowadays, NASA is looking to utilize robots to their full potential, because there are many habitats in which humans will not be able to explore without costly equipment to protect them from the planets hazardous ecosystem [1-3]. This goal can be achieved by using small and inexpensive rovers to explore new planets surfaces while maintaining low costs. Not only costs would be kept low, but safeguarding our astronauts from potential dangers until planets have been analyzed enough to ensure their safety. Robots help prevent putting humans in hazardous situations and they will reduce the cost related to transporting humans to space. For instance, humans would require a constant supply of water, food, and oxygen. All this would prove to be far too costly to be feasible whereas robots can be crammed anywhere on a spaceship without the need for food or water just energy. This energy could be harnessed from solar panels set up on the planet by the robots themselves.

Not just any robot can be designed and sent into space, for example, if an expensive rover is sent into outer space and was damaged then it would be almost impossible or too costly to send either another robot or a human to do the required repair. With this in mind, NASA decided that the best course of action would be to send inexpensive rovers, which are mostly 3D printed and running on budget-friendly components [2-6]. To make this more effective, NASA is analyzing the possibility of having multiple small robots "swarmies" that are programmed with an ant-like behavior. These rovers traverse exploring multiple areas, collect resources, and communicate with each other, which will allow searching a wider area at a time, a swarmie robot is shown in Fig. 1.

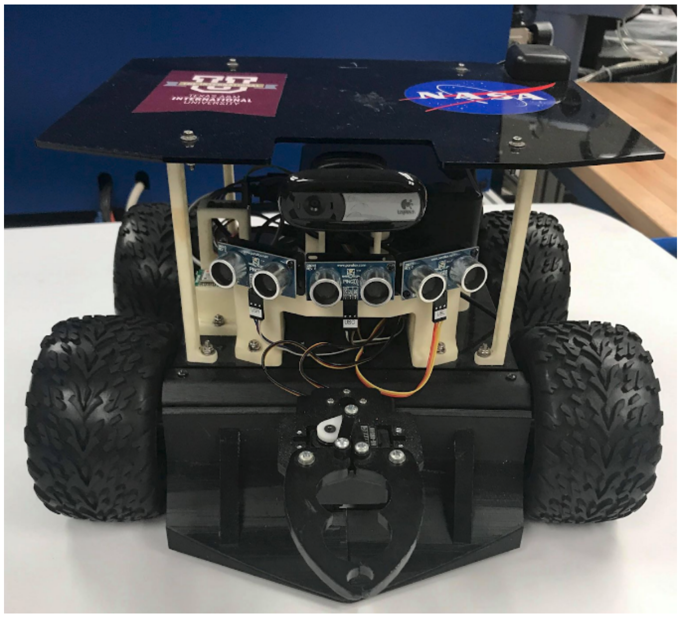


Fig. 1. DustySWARM Swarmie Robot

The University of New Mexico (UNM) were the founders of Swarmathon competition, Dr. Melanie Moses and Joshua

Hecker constructed the SwarmBaseCode-ROS to have a foundation to start developing new search algorithms [2-10]. After founding the competition with the support of Theresa Martinez, NASA manager of Minority University Research and Education Program STEM Management, it was turned into a national level competition. The competition has historically been composed of three parts: Physical Competition, Virtual Competition, and High School Competition.

For the 2019 competition, NASA has decided to have a Virtual competition only. The focus was to develop and simulate a code that will be used on multiple rovers at the same time. One of the most significant changes to this year's challenge was the use of six rovers instead of three. Each rover has around 16 main controller components that control the swarmie's behavior. The most fundamental controllers are the Search Controller, the Pickup Controller, the Drop off Controller, and the Obstacle Avoidance. All coding was conducted using Ubuntu-based C programming and GitHub was utilized to share codes and monitor updates and modifications between the swarmies, team members, and NASA, which was a very valuable tool to implement all Software Engineering principles.

Throughout the literature review, the team was able to analyze the code provided by UNM, Durham Technical College team, Florida International University, and the Japanese Rovers on the asteroid. The team planned to modify the search controller code as it is a major component in a successful search algorithm. The change from the previous DustySWARM team's Square Spiral Search (SSS) into an L -shaped search will allow the robots to move from the square field center to its outside border. Swarmie team consist of six rovers, where a rover will be located at the middle of each side of the squared home base and two extra rovers that each will be placed randomly at the home base corner. This means two extra rovers would be covering same areas, as all rovers must run the same code. Therefore, to solve this overlapping of resources, DustySWARM team decided to set up these two extra rovers with an alternate search route. Essentially, these two extra rovers will go out beyond the scope of the side rovers (using the L-Shape) and cover two quadrants each.

The paper is organized as follows: Section 2 is a background and literature review, Section 3 describes the problem statement and challenge provided by NASA, Sections 4 deals the methodology, while Section 5 illustrates the proposed solutions and implementations, Section 6 summarises the results of both simulations physical runs, and Section 7 concludes the paper and describe the team future plan.

II. Literature Review

The purpose of the project is to create inexpensive improved rover systems that can perform multiple functions, such as image capturing, rock mining, and data collection. There have already been many instances where rovers, as such, are utilized [2-5].

One concurrent example is the Japan Aerospace Exploration Agency (JAXA) small rover that was sent to analyze the surface of an asteroid that is about 280 million kilometers away from Earth. This rover has a designed movement that allow a jumping action, which allowed capturing images of the asteroid and transmit them back to researchers on Earth [11-12]. This is one of many examples in the breakthrough of space exploration using inexpensive space rovers, which lays the foundation in the lore of how the project began its life.

To achieve the project goal, the team was engaged in the engineering process formulating using past resources. Every year, each new DustySWARM team may have the decision to either revise and improve previous search patterns or create a completely new algorithm that will be implemented from scratch. DustySWARM teams developed a couple of search algorithms for NASA Swarmathon challenge and competed against multiple schools across the nation. They dedicate time to obtain the most optimal path for rovers and alias swarmies to work autonomously and in synchrony. The swarmies must search, retrieve, and deliver AprilTags to the home base within a designated arena and given time limit.

DustySWARM 1.0 team developed a reverse twister search algorithm that depends on Archimedean Spiral using parametric equations, where rovers followed a counter-clockwise rotation spiral shown in Fig. 2 [13-14].

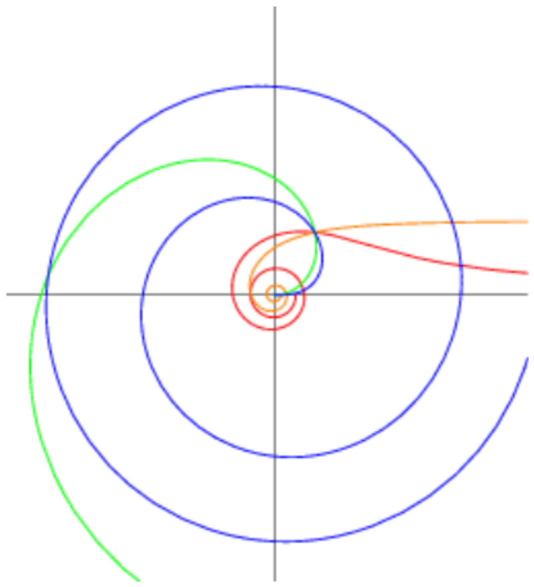


Fig. 2. Archimedean Spiral

DustySWARM 2.0 team analyzed multiple other possible search patterns before they selected their search pattern; Spiral Search was listed as one of their alternatives. Throughout multiple endeavors, DustySWARM team decided on an original path of a spiral path; however, this proved to be inefficient as rovers were not able to explore corners as the team wished. As a result, the team design evolved towards a Spiral Epicycloidal Wave (SEW) path [15-16]. Fig. 3 shows a visual representation of this path. In non-technical terms, the path is essentially a continuous spiral, what would allow maximum coverage. However, there are limitations with circular search patterns, such as the competition field has squared corners. However, the team was well aware of the corner situation and it was decided as a constraint/trade-off to the selected path. They proceeded with the path since it would be easier to implement within three to six rovers with minimum complications.

The DustySWARM 2.0 team divided the effort into three

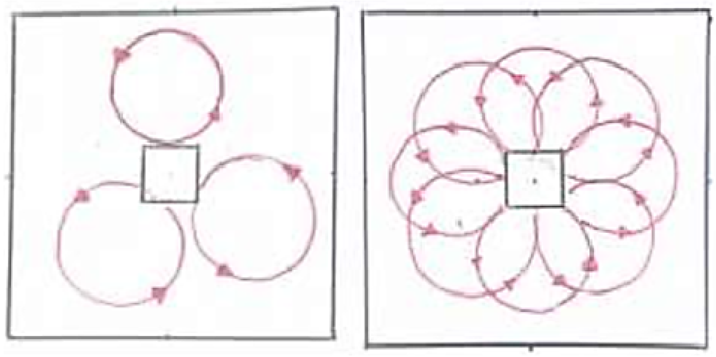


Fig. 3. DustySWARM Spiral Epicycloidal Search Pattern

main concepts, namely, rover task division, square spiral pattern, and spiral motion. The purpose of the rover task division was to dedicate two rovers searching the middle-section of the arena and one of the rovers searching the boundaries. However, this design was discarded since some regulation had been enforced that rovers cannot be individually programmed; meaning that there is only one search algorithm for all rovers. The second concept was the square spiral pattern, where the rovers will scan the area to perform a square spiral path. However, whenever a rover encountered a resource, the rover would return to the home original position and forget it last AprilTags pick-up position it was in. The last design was spiral motion following an Archimedean Spiral through the implementation of parametric equations.

DustySWARM utilized two separate processes in the experimental observations period: a sinusoidal modulation and pulsatory modulation. The sinusoidal modulation consists of exposing the reactor with a rotation period of 30.31 seconds and a wavelength of 1.71 millimeters. The obtained results indicated the formation of hypocycloids and epicycloids. For the pulsatory modulation, the reactor was exposed to the same light luminosity, but with short light pulses, which resulted in a different formation, it was a pronounced linear drift. The shape of the spiral waves resembled a similar result as the sinusoidal modulation: a hypocycloidal and an epicycloid trajectory. DustySWARM 2.0 built the design based on the Spiral Epicycloidal Wave (SEW), and the equations provided the necessary shape after converting them into C++ language [4-6].

DustySWARM 3.0 improved the SEW and developed the Square-Spiral Search (SSS) path shown in Fig. 4. There are numerous advantages of using a square-spiral pattern, it will allow rovers to survey the entirety of the arena including corners and its simplicity will allow future adjustments to maximize the pattern. Additionally, the rovers will be allocated to survey a dedicated area rather than searching collectively, which will increase the coverage percentage as shown in Fig. 5 [17-18].

Florida International University (FIU) Swarmathon team enhanced the rovers' performances through code optimization, by utilizing the sensor data, and compiling it to a single location. This methodology developed a predetermined search pattern in which the rovers would follow. For example, rovers would move a one-meter increment in the X-direction, and then another one-meter in the Y-direction [19-20]. Then the rover's coordinates were stored and a 360-degree radial turn will be initiated to scan for any AprilTags. If none were found, the rover will continue its stair-step motion, but if AprilTags were detected within the rover's proximity, the rover would proceed to retrieve and return the tags to the home base in one motion as shown in Fig. 6. The rover returns to home-based by recording the summation of all X and Y displacements that were traveled [7-8].

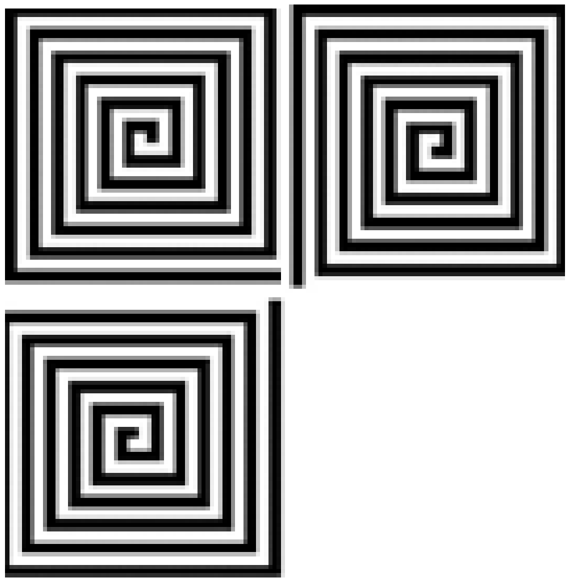


Fig. 4. The Snake Path a.k.a Square Spiral Pattern

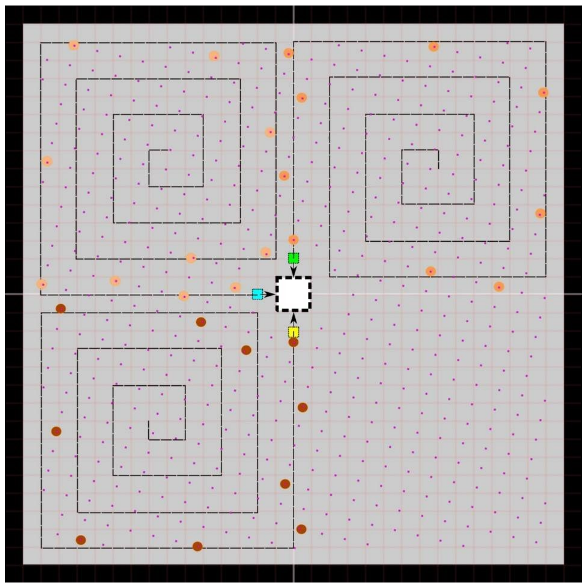


Fig. 5. DustySWARM 3.0 Square Searching Algorithm Path

Additionally, FIU team increased the turning increment for the Object Detection to 1.57 radians (90 degrees) from the original value of 0.2 radians (11.45 degree) which was their solution to avoid the frequent rover collisions.

The FIU team simplified the rover collected sensor data,

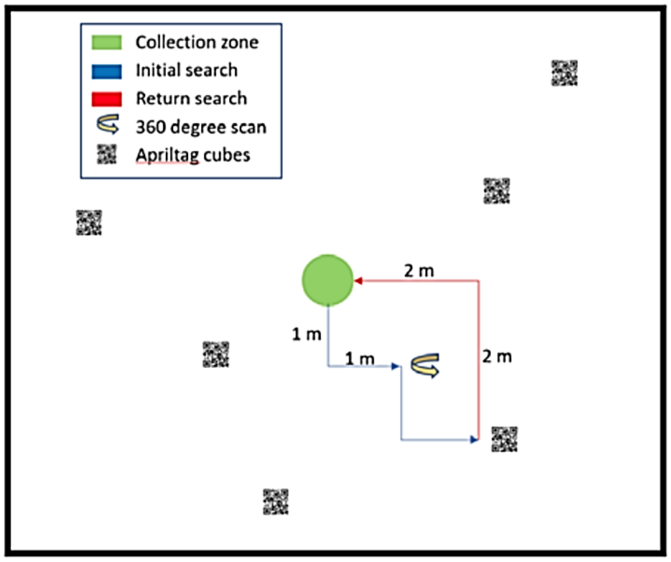


Fig. 6. FIU Rovers Path

which helped in implementing a viable accurate code that will find AprilTags. The definitions of sensor data as stored and generated is shown in Table I. Since the Swarmathon competition prohibits the extraction of the rover's IP addresses, the FIU team assigned random numbers to each rover that was saved in a local variable (R), where the rovers would compare the individual saved variables to each other. The difference in data would determine their rankings. The largest random number created by the rover would be assigned rank one, and subsequently, the rest would follow a numerical order. This approached was discussed in other research and illustrate its success [21-25].

TABLE I. SENSOR DATA

Symbol	Variable	Variable Name	Variable Location
R	Robot#		
T	Time		
CC	Cube Count	Detections[i].pose	Mobility.cpp
X	x-position	currentLocation.x	Mobility.cpp
Y	y-position	currentLocation.y	Mobility.cpp
θ	angular-position	currentLocation.theta	Mobility.cpp
S1	Left Sensor	sonarLeft-rang	Obstacle.cpp
S2	Center Sensor	sonarCenter-rang	Obstacle.cpp
S3	Right Sensor	sonarRight-rang	Obstacle.cpp

Park, et al., reworked the Coverage algorithms to account for time constraints, which is a significant factor as the Swarmathon competition is timed, so rovers must find, collect, and deliver AprilTags quickly and efficiently [26]. The Coverage algorithm was intended and best suited for intelligent robots that are unaware of the surrounding environment, that must be covered by the rovers within a certain period. The most powerful coverage algorithms rely heavily on having a complete grid map of the environment. For this reason, the authors utilized Simultaneous Localization and Mapping (SLAM) algorithms to help their robots to operate efficiently in an unknown environment. In addition to being stationed within an unfamiliar location, the rover robots must be able to adapt to dynamic or static obstacles within the environment. Therefore, a new proposed algorithm by the name of DMax Coverage was taken into consideration for rover use. However, our competition involves static obstacles and resources to retrieve, so if SLAM were used, adjustments would be necessary [27-30].

DMax algorithm works by first using the SLAM algorithm to find out the boundaries of the unknown environment workspace and the robot's position and orientation. Then, the area will be mapped where the DMax algorithm computes a minimum bounding rectangle (MBR) [27-30]. An MBR is a rectangle that includes all free areas and areas with obstacles. Then this rectangle is simplified into smaller rectangles with no obstacles found, this approach is a common mathematical algorithm and called Rectangle Tiling Scheme, shown in Fig. 7.

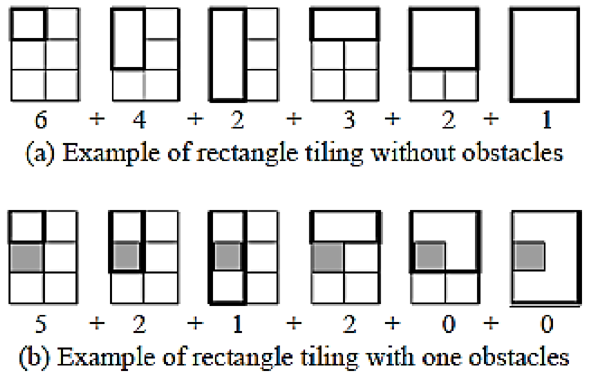

6 + 4 + 2 + 3 + 2 + 1
(a) Example of rectangle tiling without obstacles
5 + 2 + 1 + 2 + 0 + 0
(b) Example of rectangle tiling with one obstacles

Fig. 7. Examples of Tiling Rectangle Filling

Rectangle Tiling Scheme aims to see how many times the rectangle in the bold outline can fit within the space without being rotated or overlapping an obstacle. After separating the rectangles into sub-spaces, the algorithm must then decide the sequence in which to visit them. The size of the rectangle estimates coverage time, number of turns the robot has to make, number of obstacles, and estimates time to reach the rectangle from its current location while taking into account the maximum coverage [27-30]. While the robot is moving from one point to another, it is continuously mapping the environment and looking for changes that would affect its next path.

Comparing the DMax algorithm to a Random and Boustrophedon algorithms in both simple and complex map environment, showed its superiority by 5% as revealed in Table II. The longer the given deadline, the more drastic the improvement will appear. Although the test conducted is not catered for object retrieval, this algorithm has some useful mapping applications.

TABLE II. DMAX COVERAGE AREA (%) WITH VARIOUS DEADLINES

Deadline (Sec.)	Covered Area (%) without Deadline	Covered Area (%) with Deadline
700	35.28	37.14
800	37.95	38.10
900	39.12	42.61
1000	44.12	48.312

III. Problem Statement

NASA Swarmathon 2019 challenge aimed to to develop cooperative robotics to revolutionize space exploration and further advance technology for future NASA space exploration missions. This competition started with new and unexpected rules and regulations. The first major change was making previous competitions' codes available to all teams, which allowed every team to compare and enhance their code. The second major change was six rovers per team would be used bigger field, which posed the need of creating a search algorithm that allows all six rovers to search equally and efficiently and change parameters to adjust for a larger field. The team goal is to design and test a new algorithm based on all available information that can be used to control 6 swarmies to collect and deliver the max number of simulated resources within the allotted time.

IV. Methodology

Through rigorous planning and analysis, DustySWARM 4.0 found that the L-shaped pattern might be more viable to use in covering the most area searching for AprilTags. The Gazebo simulator showed that the L-shaped search pattern could run better with few modifications. Although the simulation went well, there were complications during the physical trials. This comparison helped the team to identify the complication sources, namely, the Drop-off controller has the rover driving through the home base, thus knocking out some of the collected AprilTags. Another issue found was rover needs code adjustment to include error measurements to follow the desired search pattern.

The team compared their code with other teams' previous codes. An important observation was that Durham Tech code was based on several switch cases and multiple libraries, thus making their code shorter in lines, which can essentially help the rovers to "think" quicker as it executes code. DustySWARM team decided to build their code utilizing all such libraries' knowledge gained from previous codes.

V. Proposed Solutions and Implementation

The team planned to develop codes and libraries for every rover motion such as the gripper, the movement, the communication, mapping, and so on. The team focused on five main processes to achieve their successful search algorithm;
A) Basic search and return algorithms,
B) Inter-connectivity and intermediate search algorithm,
C) Mapping,
D) Advanced algorithms using Mapping Techniques and
E) Drop-Off Controller.

All these processes are discussed in the following subsections. Fig. 8 shows the differences between DustySWARM developed SSS and the Radiant L-Shaped search Algorithms.

A. Basic Search and Return Algorithms

Based on the new base code, the team decided to optimize it by addressing two key issues: the search algorithm and the return algorithm. Improvements are suggested based on two criteria: expected distance traveled between the resources' pick

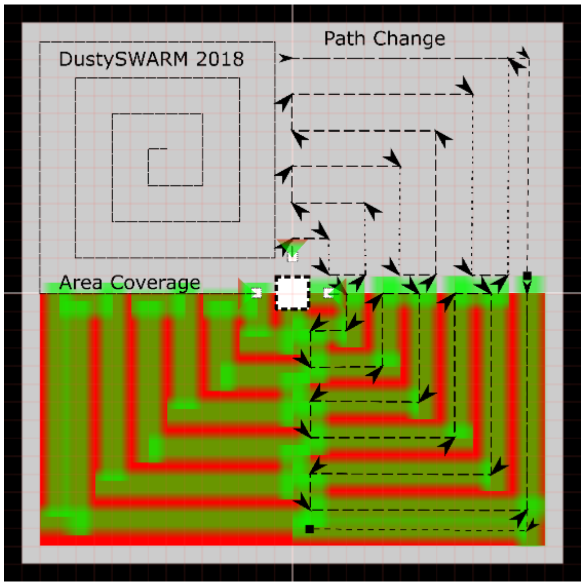


Fig. 8. DustySWARM SSS Vs. Radiant L-Shaped Search Algorithms

up location and the drop off location and the area in which the robot travels.

The square spiral pattern is an efficient method for searching a quadrant, but the implementation of the reverse pattern prioritizes the resources furthest away from the delivery location, which resulted in significant time and distance consumption compared to picking up the closest resources to the delivery location. Additionally, this algorithm allows the understanding of what areas had been covered and potential obstacles had been reduced. DustySWARM 2.0 return algorithm relies on Global Positioning System (GPS) to guide the rover to return to its initial starting spot for drop-off, which was a challenge as the rover small scale resulted in inaccurate GPS readings while determining the correct position, therefore the rovers will drift significantly as shown in Fig. 9.

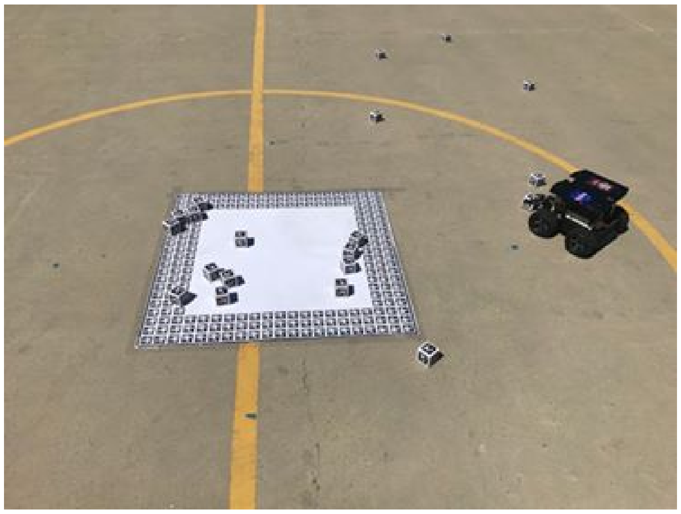


Fig. 9. Swarmie Rover with AprilTags and Home Base

Using the reverse square algorithm resulted in a direct line of travel between the pickup and the delivery locations, which exposed a problem with resource picked up and ready for delivery but blocking the sensor and front camera. This is a major reduction in the rovers' ability to detect and navigate obstacles in unmapped terrains.

DustySWARM 4.0 search algorithm utilized the L-shaped pattern to search a quadrant in a radiating fashion, which resulted in collecting the closest resources and minimizing the time needed for this task as illustrated in Fig. 10.

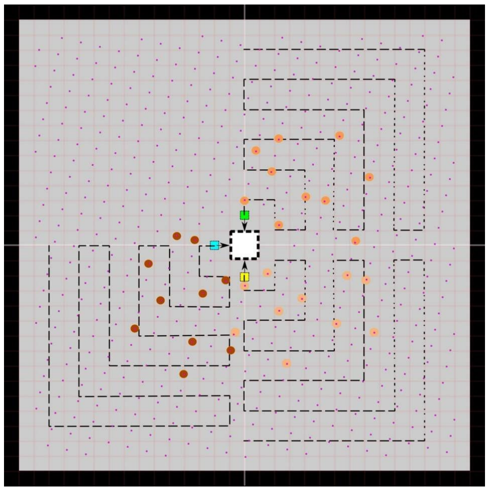


Fig. 10. Three Rovers following DustySWARM Radiating L-Shaped Search Algorithm Path

This method made it easier to potentially map out the covered area and provide a minimal obstacle for deliveries of further resources. The return algorithm had been updated to rely more heavily on the odometery instead of the GPS, as the robot encoders have a more accurate calibration with smaller drift values. In this algorithm, the two corner (extra) rovers will be assigned to search an area outside the highest parameter of the side rovers. This assignment was done by developing a function called (IAmCorner), where the rover will turn around and detect its location based on detecting the surrounding rovers and home base QR codes. Therefore, the side rovers will cover the area between the two corners (0, 0) and (5, 5), while the corner rovers will search the area beyond (6, 0) and explore from there out following a mirrored version of the L-Shape pattern as shown in Fig. 11.

To conduct an accurate test, search pattern for corner rovers was hard coded with IAmCorner value set to true. This IAmCorner value is equal to the return of a function, which at the start of the competition will see how many rovers it has to its left and right. If the corner rovers are placed at opposite corners then the side rovers will only see a maximum of one (1) rover next to them. The corner rovers, on the other hand, will always have two (2) rovers next to them. This function simply returns either true or false and assigns it to IAmCorner, which is shown in Fig. 12. If IAmCorner is false then the rover will run the L-Shape pattern cases, if it is true then the rover will use the extended mirrored L-Shape pattern. This method has proven to work roughly around 80% of the time. There are times when the northeast (north being up on the map) corner rover follows the same pattern as the east side rover.

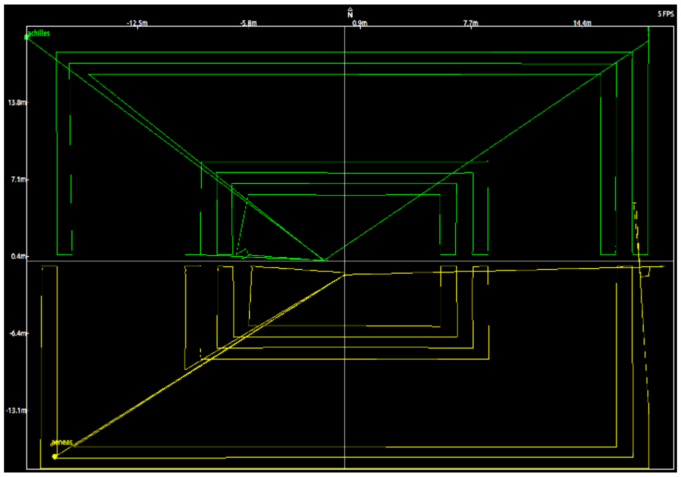
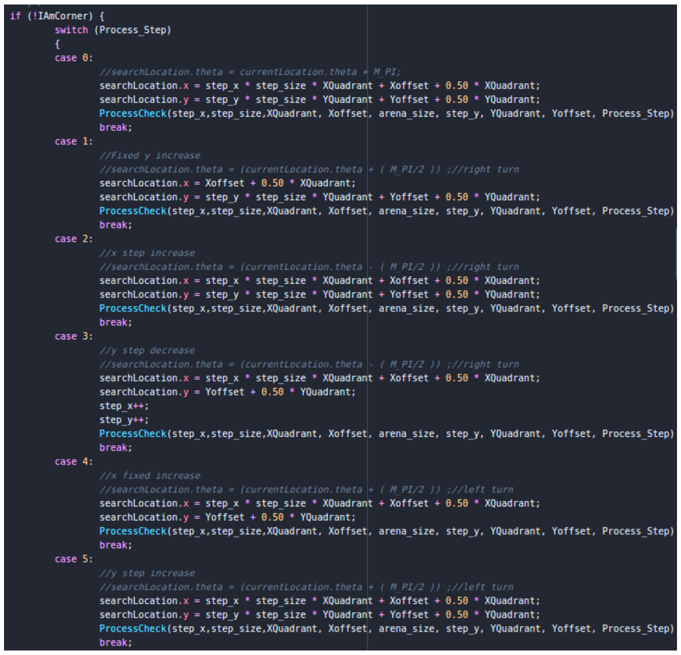


Fig. 11. Corner Rovers Searching Zones


```
if (!IAmCorner) {
        switch (Process_Step)
        {
        case 0:
                //searchLocation.theta = currentLocation.theta + M_PI;
                searchLocation.x = step_x * step_size * XQuadrant + Xoffset + 0.50 * XQuadrant;
                searchLocation.y = step_y * step_size * YQuadrant + Yoffset + 0.50 * YQuadrant;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
        case 1:
                //Fixed y increase
                //searchLocation.theta = (currentLocation.theta + ( M_PI/2 )) ;//right turn
                searchLocation.x = Xoffset + 0.50 * XQuadrant;
                searchLocation.y = step_y * step_size * YQuadrant + Yoffset + 0.50 * YQuadrant;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
        case 2:
                //x step increase
                //searchLocation.theta = (currentLocation.theta - ( M_PI/2 )) ;//right turn
                searchLocation.x = step_x * step_size * XQuadrant + Xoffset + 0.50 * XQuadrant;
                searchLocation.y = step_y * step_size * YQuadrant + Yoffset + 0.50 * YQuadrant;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
        case 3:
                //y step decrease
                //searchLocation.theta = (currentLocation.theta - ( M_PI/2 )) ;//right turn
                searchLocation.x = step_x * step_size * XQuadrant + Xoffset + 0.50 * XQuadrant;
                searchLocation.y = Yoffset + 0.50 * YQuadrant;
                step_x++;
                step_y++;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
        case 4:
                //x fixed increase
                //searchLocation.theta = (currentLocation.theta + ( M_PI/2 )) ;//left turn
                searchLocation.x = step_x * step_size * XQuadrant + Xoffset + 0.50 * XQuadrant;
                searchLocation.y = Yoffset + 0.50 * YQuadrant;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
        case 5:
                //y step increase
                //searchLocation.theta = (currentLocation.theta + ( M_PI/2 )) ;//left turn
                searchLocation.x = step_x * step_size * XQuadrant + Xoffset + 0.50 * XQuadrant;
                searchLocation.y = step_y * step_size * YQuadrant + Yoffset + 0.50 * YQuadrant;
                ProcessCheck(step_x,step_size,XQuadrant, Xoffset, arena_size, step_y, YQuadrant, Yoffset, Process_Step)
                break;
```
Fig. 12. IAmCorner Function Process to Assign the L-Shape Path

B. *Inter-connectivity and Intermediate Search Algorithm*

Further improvement was done by implementing Robot Operating System (ROS) to develop actual communication channels between all rovers [30-33]. These communication channels were used to send the position, sensor data, and visible tag information of each rover. The rules for the NASA Swarmathon allowed understanding the initial positioning and orientation of the rovers based on the data provided by the rover's Inertial Measurement Unit (IMU). However, these data did not take into account the orientation of the field. Correction of the field orientation was done by analyzing the delivery location AprilTags and drawing a straight line or corner to establish the delivery zone by comparing the data provided by other rovers.

The search algorithm depended on the ROS communications information to direct the rovers to the suitable behavior. For example, in a three-rover system, if every rover had been searching a quadrant and one of the rovers finished the area, then that rover will be direct to move to the fourth quadrant to prevent any duplication of the search effort. An active tally of the collected resources allowed the team to change the behavior of the search algorithm by increasing the size of the steps to cover more area and find the resources that exist away from the delivery zone.

C. Mapping

The mapping controller was developed to include two different arrays that are updated continuously. These two arrays represent the Drop-off controller and the Search controller, they are composed of 1's, 5's, and 9's and each number represents a different field representation and condition. Number (1) represents the status of a rover being next to the home base when it is going for a drop-off. Number (5) represents the areas where it has to go; and there are AprilTags and the rovers should head over in that direction, while Number (9) means an area, where the rover cannot go because it represents the home base or obstacles.

Each array had an additional "if/else" and "for" loop statements, to let the rovers know what functions to perform autonomously. In essence, the numbers served as the size of the search nodes, as well as how far the search radius is needed for the rovers to head in those directions to collect AprilTags. The arrays are part of an algorithm that allows the rovers to compute the lowest possible cost to search for and collect AprilTags. Mapping using one rover with few obstacles and AprilTags is illustrated in Fig. 13. Fig. 14 and 15 show the mapping output of one/four rovers exploring the complete field.

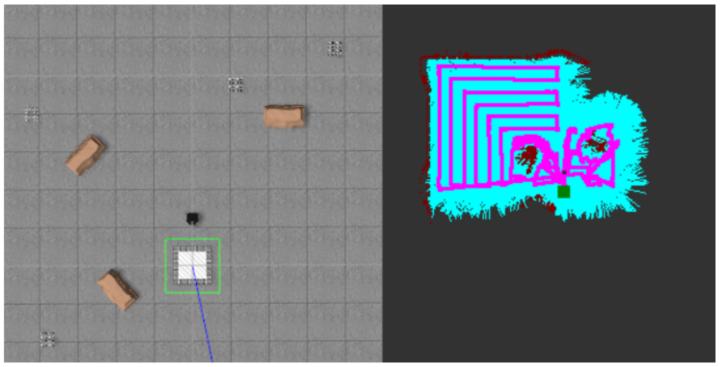


Fig. 13. Single Rover Mapping Simple Field with Obstacles and AprilTags

D. Integrating Mapping Techniques within Advanced Search Algorithms

In this stage, the automated processes of rovers driving to possible resource zones were developed. These zones will be established by using ultrasound detection, mapping techniques to prioritizing areas with probable resources over those that lack resources. All rovers will be programmed to communicate and share the location of any resources or obstacles they discover and any probable resource locations. Based on the percentage of collected resources and/or area covered, this mapping will go into full effect and override the base search sequence.

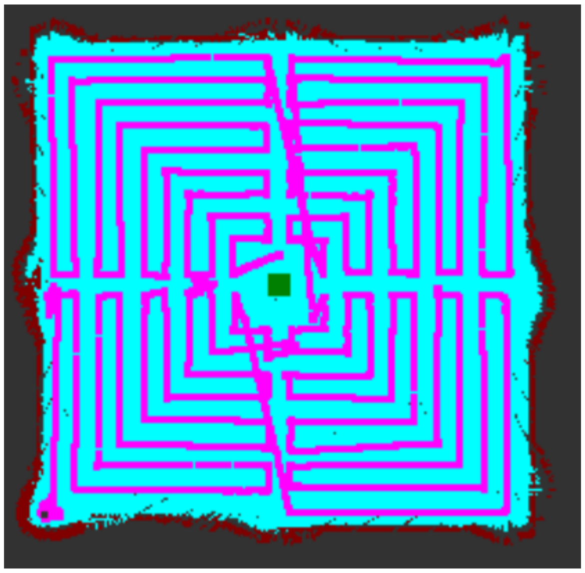


Fig. 14. One Rover Mapping a Complete Field

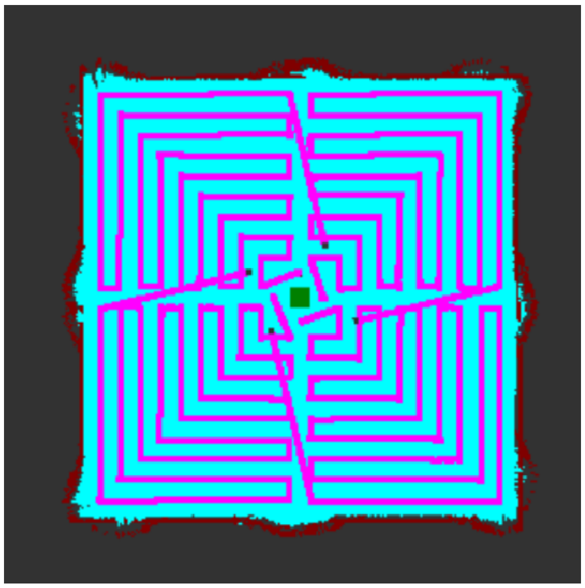


Fig. 15. Four Rovers Mapping a Complete Field

Depending on trials, the execution conditions of map integration might change to default in the cases of ROS Communication failure, searching in the immediate vicinity of the delivery zone, and valid detection from the ultrasound sensor. The automated process will map out the destination location by prioritizing areas, which have been previously transverse and potentially avoid obstacles. This system can be improved by tracking the time required to move to the location in question, which will allow the prediction of rover motion and paths crossing. This will reduce the collisions chance between rovers. An additional technique that can be implemented before the search period ends is to establish a routine for the rovers with no assigned task to move around the delivery zone and push back resources that have fallen out.

E. Drop-off Controllers

One of the challenges was the inconsistency of the Drop-off controller performance. In a few instances, once the rover completed the search pattern, and proceeding with the retrieval algorithm, it failed in dropping the AprilTags successfully, where AprilTags will be dropped outside the home base. This situation occurred when the rover would enter the home base at an angle, which would initiate the drop off algorithm by searching left and right for home base QR codes and following only the edge of the base and then proceeded to leave the AprilTag outside of the desired location as shown in Fig. 16 and 17.

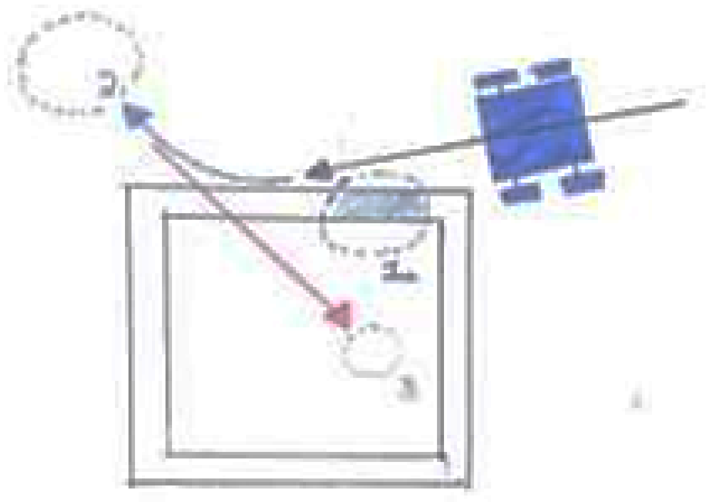


Fig. 16. Illustration of Rover Dropping off the AprilTag Outside Home Base

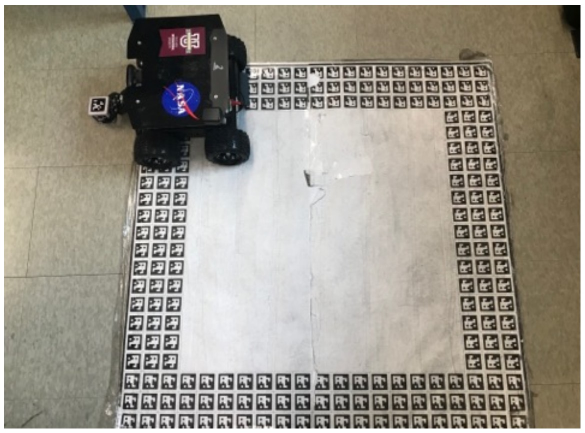
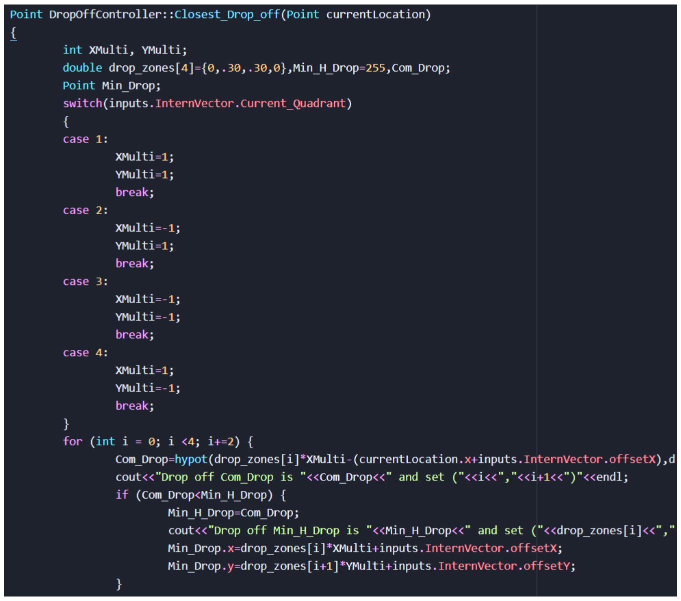


Fig. 17. Rover Dropping off the AprilTag Outside Home Base

The first proposed solution was to assign a quadrant of the home base to every rover, by assigning the quadrant x and y coordinates to each rover. The rover would proceed to go to the nearest cardinal axis of the grid and enter the home base perpendicularly without any angular miscalculation. This idea evolved into another approach that uses waypoints, Fig. 18 shows a segment of the improved drop-off code. The waypoints concept is a type of direction algorithm. It stores the desired location and current location, which will be used later in matching these two locations coordinates. The code would calculate the nearest cardinal axis relative to the rover current location and then proceed to the axis intercept to enter the base perpendicularly for a drop-off.


```
Point DropOffController::Closest_Drop_off(Point currentLocation)
{
        int XMulti, YMulti;
        double drop_zones[4]={0,.30,.30,0},Min_H_Drop=255,Com_Drop;
        Point Min_Drop;
        switch(inputs.InternVector.Current_Quadrant)
        {
        case 1:
                XMulti=1;
                YMulti=1;
                break;
        case 2:
                XMulti=-1;
                YMulti=1;
                break;
        case 3:
                XMulti=-1;
                YMulti=-1;
                break;
        case 4:
                XMulti=1;
                YMulti=-1;
                break;
        }
        for (int i = 0; i <4; i+=2) {
                Com_Drop=hypot(drop_zones[i]*XMulti-(currentLocation.x+inputs.InternVector.offsetX),d
                cout<<"Drop off Com_Drop is "<<Com_Drop<<" and set ("<<i<<","<<i+1<<")"<<endl;
                if (Com_Drop<Min_H_Drop) {
                        Min_H_Drop=Com_Drop;
                        cout<<"Drop off Min_H_Drop is "<<Min_H_Drop<<" and set ("<<drop_zones[i]<<","
                        Min_Drop.x=drop_zones[i]*XMulti+inputs.InternVector.offsetX;
                        Min_Drop.y=drop_zones[i+1]*YMulti+inputs.InternVector.offsetY;
                }
```
Fig. 18. Waypoints Drop-off Code

VI. Results Summary

After several virtual simulation and physical trials, the team was able to develop a new search algorithm that can be implemented in NASA Swarmathon rovers, by programming the swarmies to search a predefined quadrant in an L-shaped radiating fashion. Additionally, These robots had been coded to determined their proximate location based on the number of surrounding robots through the IAmCorner function.

This Algorithm allowed the robots to collect the closest AprilTags/resources while minimizing the search time needed and without interfering or interrupting each other. Utilizing ROS communication, the rovers were able to share the collected AprilTags count that was used for changing the search steps to cover more area and find the resources that exist away from the delivery zone and if rovers finish searching the assigned quadrant, they will be able to select the next search area. Some of the results are shown in Table III.

TABLE III. L-Shaped Radiating Search Algorithm Results

Target Distribution	Simulation Runs Results	Physical Trials Results
Clustered	32	27
Power Law	118	110
Uniform	72	63

Moreover, the team developed and integrated a mapping technique, that used the ultrasound sensors, GPS and IMU, where rovers will create their own map as an array by identifying and prioritizing the home base, the obstacles and the safe search areas. This map will be used as a fail-safe measurement in case of ROS or WiFi connection failure. Additionally, the drop-off controller had been improved by including the start location of the rovers to determine the best home base entering point. Fig. 19 shows the simulation run with four rovers to search the predefined field.

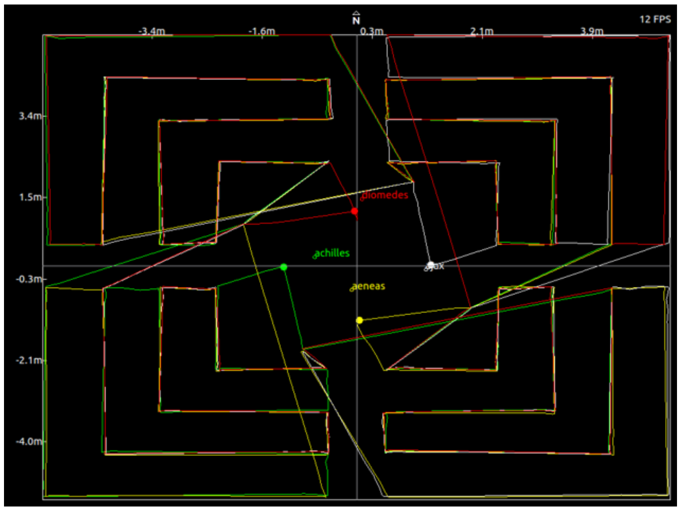


Fig. 19. L-Shaped Radiating Search Gazebo Simulation Results

VII. Conclusion

In conclusion, the team developed a searching algorithm that was implemented to control NASA swarmies to search unknown terrain for simulated resources and retrieve them back to the home base. The current code had multiple improvements and both Gazebo simulation and real-life runs showed an improved consistency of the rovers' performance.

This opportunity was valuable for the team as they practiced the systems engineering and management principals in a real-life project and expanded their computer science techniques to contribute to NASA's mission of exploring space.

Currently NASA Swarmathon had been paused but with any code, there is room for improvement. The team will continue to further improve the algorithm code especially the rovers' behavior when picking-up and dropping-off AprilTags and enhance the rovers' ability to communicate amongst each other. Other areas of future improvements are to programming the rovers with the ability to stack and push AprilTags when dropping-off when swarmies are close to home base.

Acknowledgment

Thanks to NASA and the University of New Mexico (UNM) teams for all the help and the opportunity to participate in such great competition. Thanks to all our sponsors from Texas A&M International University (TAMIU) and Laredo, TX.

References

[1] Voosen, Paul. "Mars rover steps up hunt for molecular signs of life." (2017). Science 03 Feb 2017, Vol. 355, Issue 6324, pp. 444-445, DOI: 10.1126/science.355.6324.444

[2] Secor, P. (2016). "NASA Swarmathon".

[3] Glenn, T., Ragland, S., Meyer, A., Pulliam, J., Holmes, E., and Riley, N. NASA Swarmathon.

[4] Hecker, J. (2015). "Swarmie User Manual. Quick Start Guide for Physical Robots". University of New Mexico. www.Github.com.

[5] Montague, G. (2014). "Swarmie User Manual: A Rover Used for Multi-agent Swarm Research". www.ntrs.nasa.gov

[6] NASA Swarmathon Home Page. (2015). Retrieved from http://nasaswarmathon.com/

[7] Ackerman, Sarah M., G. Matthew Fricke, Joshua P. Hecker, Kastro M. Hamed, Samantha R. Fowler, Antonio D. Griego, Jarett C. Jones, J. Jake Nichol, Kurt W. Leucht, and Melanie E. Moses. (2018). "The Swarmathon: An autonomous swarm robotics competition". arXiv preprint arXiv:1805.08320.

[8] BCLab-UNM. (2015). BCLab-UNM/SwarmBaseCode-ROS.

[9] Bhattacharya, S., and Agrawal, R. (2017, March). "Development of robot swarm algorithms on an extensible framework". In SoutheastCon 2017 (pp. 1-6). IEEE.

[10] Koris, Daniel R., and Jason Isaacs. (2017) "A Formal Approach to Extended State Machines for Multi-Objective Robots Operating in Dynamic Environments." Proceedings of the 2017 Midstates Conference on Undergraduate Research in Computer Science and Mathematics

[11] Kubota, Takashi, and Tetsuo Yoshimitsu. "Intelligent rover with hopping mechanism for asteroid exploration." In 2013 6th International Conference on Recent Advances in Space Technologies (RAST), pp. 979-984. IEEE, 2013.

[12] Ulamec, Stephan, Patrick Michel, Matthias Grott, Ute Böttger, Heinz-Wilhelm Hübers, Naomi Murdoch, Pierre Vernazza et al. "A rover for the JAXA MMX Mission to Phobos." In 70th InternationalAstronautical Congress, pp. IAC-19. International Astronautical Federation, 2019.

[13] Tashtoush, T., Hernandez, R., Yanez, R., Gonzalez, J., Moreno, H., and Escobar, V. (2020). "Reverse-Twister Swarm Search Algorithm Design: NASA Swarmathon Competition", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 7(1), pp.13-20.

[14] Hernandez, R., Yanez, R., Gonzalez, J., Moreno, H., Escobar, V., and Tashtoush. T., (2016) "Design of a Swarm Search Algorithm: DustySWARM Reverse-Twister Code for NASA Swarmathon." Texas A&M International University, School of Engineering.

[15] Tashtoush, T., Gutierrez, O., Herrera, E., Medina, J., Peña, A., Varela, E., and Hernandez, R. (2020). "Design of a Swarm Search Algorithm: DustySWARM Spiral Epicycloidal Wave (SEW) Code for NASA Swarmathon", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 7(1), pp.28-36.

[16] Gutierrez, O., Herrera, E., Medina, J., Peña, A., Varela, E., Hernandez, R., and Tashtoush. T. (2017) "Design of a Swarm Search Algorithm: DustySWARM Spiral Epicycloidal Wave (SEW) Code for NASA Swarmathon". Texas A&M International University, School of Engineering.

[17] Tashtoush, T., Ruiz, C., Estevis, T., Herrera, E., Bernal, R., Martinez, R., and Reyna, L. (2020). "Square Spiral Search (SSS) Algorithm for Cooperative Robots: Mars Exploration", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 7(1), pp.21-27.

[18] Ruiz, C., Estevis, T., Herrera, E., Bernal, R., Martinez, R., Reyna, L., and Tashtoush, T., (2018) "Design of a Swarm Search Algorithm: Square Spiral Search (SSS) Algorithm for NASA Swarmathon". Texas A&M International University, School of Engineering.

[19] FIU Panther Swarm Team (2016). NASA SWARMATHON 2016: FIU Panther Swarm Team Technical Report.

[20] Jagolinzer, S., Larrarte, J., Guerrero, R., and Tosunoglu, S. (2016, May). Development of Swarm Algorithms for Space Exploration. In Proceedings of the 29th Florida Conference on Recent Advances in Robotics, FCRAR 2016.

[21] Richard, W. K., and Majercik, S. M. (2012, July). Swarm-based path creation in dynamic environments for search and rescue. In Proceedings of the 14th annual conference companion on Genetic and evolutionary computation (pp. 1401-1402).

[22] Aguilar, J., Blanchard, A., Sibiski, A., Soto, J., Jagolinzer, S., and Tosunoglu, S.(2017) Performance Optimization of Swarm Algorithm and Sensor Data for NASA Swarmathon Competition.

[23] Miller, P. (2007). The genius of swarms. National Geographic, 212(1), 126-147.

[24] Lim, S. S., and Bouffanais, R. (2019). From Senseless Swarms to Smart Mobs: Tuning Networks for Prosocial Behaviour. arXiv preprint arXiv:1910.01303.

[25] Park, J. K., Jeon, H. S., Noh, S. H., Park, J. H., and Oh, R., (2010, October) "A practical coverage algorithm for intelligent robots with deadline situations" In ICCAS 2010 (pp. 299-303) IEEE.

[26] Kantrasiri, S., Jirakanjana, P., and Kheowan, O. U., (2005) "Dynamics of rigidly rotating spirals under periodic modulation of excitability" Chemical physics letters, 416(4-6), 364-369.

[27] Ramalingam, B., Veerajagadheswar, P., Ilyas, M., Elara, M. R., and Manimuthu, A. (2018). Vision-Based Dirt Detection and Adaptive Tiling Scheme for Selective Area Coverage. Journal of Sensors, 2018.

[28] Monaci, M., and dos Santos, A. G. (2018). Minimum tiling of a rectangle by squares. Annals of Operations Research, 271(2), 831-851.

[29] Idri, A., Oukarfi, M., Boulmakoul, A., and Zeitouni, K. (2017). Design and Implementation Issues of a Time-dependent Shortest Path Algorithm for Multimodal Transportation Network. In TD-LSG@ PKDD/ECML (pp. 32-43).

[30] Martinez, A., and Fernández, E., (2013) "Learning ROS for robotics programming" Packt Publishing Ltd.

[31] Shotts Jr, W. E. (2012) "The Linux command line: a complete introduction" No Starch Press.

[32] J O'Kane, J. M. (2014) "A gentle introduction to ROS". www.cs.rpi.edu

[33] Quigley, M., Gerkey, B., and Smart, W. D. (2015). Programming Robots with ROS: a practical introduction to the Robot Operating System. " O'Reilly Media, Inc.".