International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 7, Issue 1, 2020, PP 22-28
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0701003
www.arcjournals.org

Square Spiral Search (SSS) Algorithm for Cooperative Robots: Mars Exploration

Tariq H. Tashtoush*, Carlos Ruiz, Tiffany Estevis, Esteban Herrera, Raul Bernal, Rolando Martinez, Luis Reyna

Texas A&M International University, Laredo, Texas, USA

Corresponding Author:** ***Tariq H. Tashtoush, *Texas A&M International University, Laredo, Texas, USA*

Abstract: *The purpose of this project is to develop and implement an optimal search algorithm into multiple rovers, a.k.a. Swarmies. Swarmies are compact rovers, designed by NASA; these rovers are aimed to mimic Ants behavior for searching simulated Mars objects. TAMIU DustySWARM3.0, compared numerous search algorithm designs, such as the team previous DustySWARM2.0's Epicycloidal spiral wave, the Fibonacci, and snake path. After multiple trial sutilizing simulation and real-life experimentation, a square-spiral path was developed to be implemented for the 2018 NASA Swarmathon Physical Competition. This paper covers a detailed overview of DustySWARM3.0'ssystems engineering process and code development utilizing computer science techniques.*

Keywords: *Swarm Robotics, Searching Algorithm, Autonomous, Robot Swarm. Robot Operating System (ROS), NASA Space Exploration, Mars Mining, Simulation, Autonomous Robot Swarm; Collaborative robots, Autonomous Behavior, Cooperative Robotics, Swarmies, Square Spiral Search*

Public Interest Statement: *NASA had been investing to develop new technologies that will allow human being to explore the space and discover new resources out of our earth sphere. They had been collaborating with different teams form US universities and colleges to encourage the students develop their own methods and out of the box ideas. Mimicking insects behavior mainly Ants, to find food and avoided any danger, our team aimed to develop a program that can be implemented in multiple robots to be able to collect different recourses without damaging themselves in the space.*

1. INTRODUCTION

National Aeronautics and Space Administration (NASA) ignited a movement for the exploration of space, in collaboration with the University of New Mexico (UNM). In 2016, NASA Swarmathon was established, granting opportunities for students to test their programming proficiency competitively against colleges across the nation. The main objective of this competition is for multiple rovers to search, retrieve, and deliver resources to a designated home base autonomously simulating ant behavior.

TAMIU DustySWARM had been invited to participate again in the physical competition for the second year in a row. DustySWARM 1.0 competed in the virtual competition in 2016, placing third. The team used various techniques in the field of computer science to achieve their goal algorithm referred to as the "Reverse-Twister" [1]. The concepts that the team created for their project followed towards the next iteration of the team, DustySWARM 2.0. DustySWARM 2.0 improved on the design and implementations of their "Spiral Epicycloidal Wave (SEW)" algorithm. With the search pattern, the team successfully placed sixth in their first physical competition [2]. For the latest iteration of DustySWARM, DustySWARM 3.0, the same concepts utilized in the past were reviewed and improved upon with developing the "Squared Spiral Path (SSS)" algorithm. The results of this project will reflect on the efficiency of the square spiral trials that will be tested virtually and physically. Additionally, this paper will explain and reveal the results of DustySWARM 3.0 progress in the NASA Swarmathon competition.

The paper is organized as follows: Section 2 is a background and literature review, Section 3 describes the model and algorithm development, Section 4 deals with the analysis and development of the experimental simulation runs; Section 5 shows the results of the proposed code compared to the original and alternatives, and Section 6 concludes the paper and describe the team future plan.

2. LITERATURE REVIEW

2.1. Original SwarmBaseCode-ROS:

The original code provided by the University of New Mexico (UNM) was the foreground for our search algorithm. By understanding the variables, functions, and code structure, the team reached the threshold of knowledge, which allowed them to invent a search algorithm. However, the provided code dictates the rover to perform a random search. The functions within the searchcontroller.cpp were heavily modified to create a uniform and homogenous search pattern [1].

2.2. DustySWARM 2.0 Spiral Epicycloidal Wave (SEW):

Throughout multiple endeavors, the team decided on an original path, different from the previous team, DustySWARM 1.0. The path the Swarmies would follow is a Spiral Epicycloidal wave (SEW), which is a continuous spiral wave, closely related to a spring formation. Their path would allow maximum coverage, but due to the competition field being a square, this caused for some limitations. DustySWARM 2.0 was aware of the corner issue, so it was written off as a constraint/tradeoff to their path. They proceeded with the path since it would be easy to implement for three to six rovers with few complications [2].

2.3. A Practical Coverage Algorithm for Intelligent Robots with Deadline Situations:

The algorithm is intended for maximum coverage utilizing the available rovers. The article provided the team with an understanding to compile a code to maximize coverage within a certain time frame. The competition is timed, so the rovers must retrieve, collect, and deliver subjects (AprilTags) quickly and efficiently. The algorithm is best suited for intelligent robots unaware of their surrounding environment. The most powerful coverage algorithms rely heavily on having a complete grid map of the environment. For this reason, the authors utilize Simultaneous Localization and Mapping (SLAM) algorithms to help their robot operate efficiently in an unknown environment. Being in an unfamiliar area requires the robots to be able to handle dynamic obstacles and moving objects. Thus, the new proposed algorithm DMax Coverage is made with these things in mind. However, the competition involves static obstacles and resources to retrieve, so if SLAM were used, adjustments would be necessary [4].

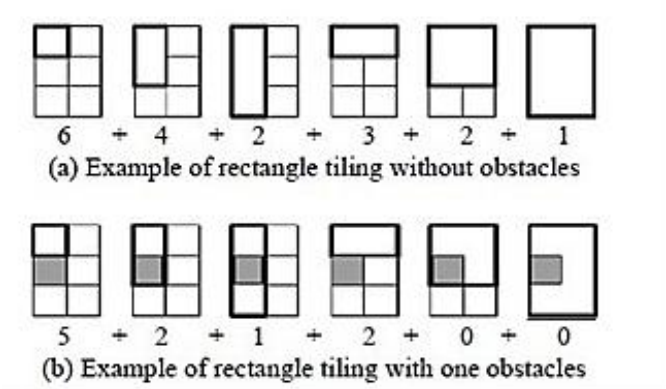

Figure1. *Example of SLAM Algorithm [4]*

The DMax algorithm works by first using a SLAM algorithm to find out the boundaries of a workspace and to find the position and orientation of the robot within this unknown environment. After the area is mapped, the DMax algorithm computes a minimum bounding rectangle (MBR). An MBR is a rectangle that includes all free areas and areas with obstacles. This rectangle is then simplified into smaller rectangles that do not contain obstacles. The Rectangle Tiling Scheme, a common mathematical algorithm, deconstructs the rectangles into smaller ones, Figure (1) gives a visual example of how SLAM works.

3. METHODOLOGY & ALGORITHM DEVELOPMENT

The DustySWARM 3.0 team members studied various methods to come up with an improved search algorithm from prior iterations of the competition. This was achieved by following the systems engineering concepts that have learned throughout their academic career and by performing a multitude of research collaboratively in conjunction with the assistance of university resources.

3.1. Epicycloidal:

The Epicycloidal search pattern, programmed by DustySWARM 2.0, instructs the rover to perform a circular search wave for multiple iterations. This was done by setting multiple equally distanced points around the arena's home base. Those points then act as origins of a circle that the rovers revolve around. After completion of the first circle, each robot moves to its next point in a clockwise direction. This process was repeated by all three rovers until a complete revolution had been made around the home base as shown in Figure 2 [1]. The code could be changed by making different sizes circles as well as adjusting the number of points and their distances from each other. The drawback to this search pattern is the amount of coverage it would miss due to its circular nature. Since the arena is a square, the circular pattern would have trouble cover corners, with the issue that rovers would want to search beyond the border.

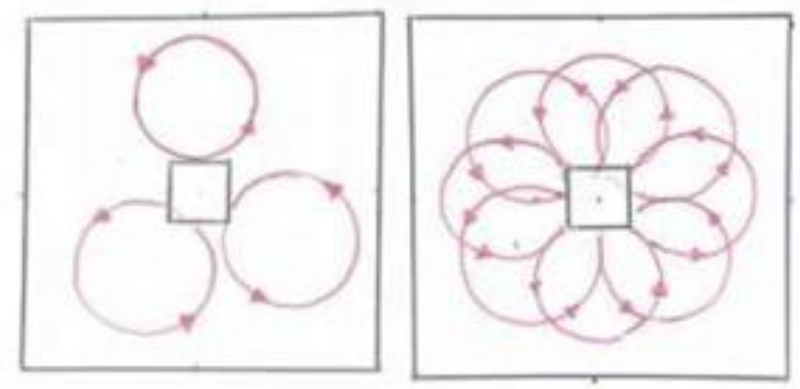

Figure2. *The Epicycloidal Arrangement*

3.2. Fibonacci:

The Fibonacci is a pattern of integers that follows an arrangement in which the next integer is equal to the sum of the previous two integers. This pattern can be denoted by the expression:

$$Fn = (Fn - 1) + (Fn - 2)$$

When squares have an area equal to these values that are arranged next to each other, they create a spiral. This golden ratio spiral is the proposed shape that the rovers follow in this particular search algorithm. However, the drawback of the Fibonacci is that after the seventh iteration the numbers substantially increase. This, in turn, led to realize that this search algorithm exponentially reduced the coverage of the field and the corners of the arena would remain undetected as illustrated in Figure (3). Therefore, this sequence was deemed inoperable as the ultimate search algorithm for DustySWARM 3.

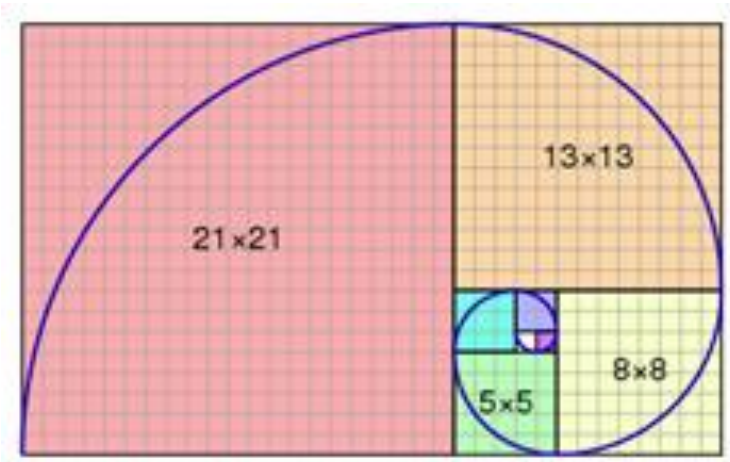

Figure3. *The Fibonacci sequence*

3.3. Square-Spiral Search (SSS):

Upon consideration of different algorithms and their associated drawbacks during the initial processes of this project, the research led the team to develop the Square-Spiral Search (SSS) algorithm. In the circular or round-shaped search patterns, negligible errors in calibration would result in inaccuracy

that exponentially expands making it arduous to use a singular reference point for the three rovers. Propositions for each rover to search the dimensions of a quadrant were considered in order to maximize the coverage area. There are numerous advantages of using a square-spiral pattern. The primary advantage would be to survey the entirety of the arena, searching corners will no longer be a complication and its simplicity would allow the team to make adjustments to maximize the pattern. Additionally, the rovers will each survey their dedicated area rather than searching collectively, leading the rovers to cover a higher percentage of the arena in a fraction of the time. Consequently, this searching algorithm contained a flaw, with only three rovers on the field for the preliminary rounds; one quadrant remained unexamined as shown in Figure (4). Utilizing this search pattern would require the need to move to the missing quadrant floor to gather the AprilTags there, therefore, a combination with the Spiral Epicycloidal Wave would be utilized to complete the search pattern.

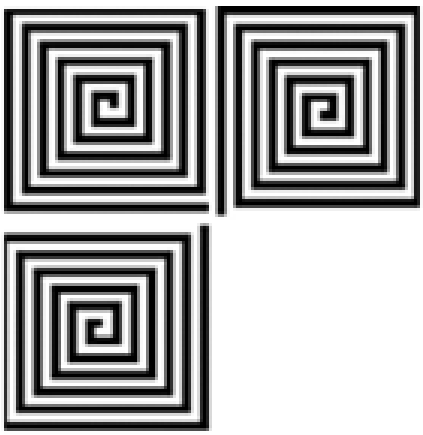

Figure4. *The Snake Path aka Square Spiral Pattern*

4. EXPERIMENTS

4.1. Simulation Runs:

The experiments conducted involved simulation and physical trials. The Dell computer used to run the Gazebo simulations was equipped with an Intel Core i5 processor with 8GB of DDR3 RAM and an integrated graphics chip. Due to the specification constraints, the team was limited to do simulations without obstacles and AprilTags for three Swarmies. Also, the team members have a limited number of computers with Ubuntu that our team members can use; this slowed our programing process and limited our simulation time. With more analysis and understanding of the provided base code, our team was able to correlate the different variables and functions to enhance our algorithm performance by modifying the search pattern following the results of our simulation run. The team then created a negligible difference between simulation and physical trials.

4.2. Physical Runs:

Once the team finalized a working code that endured simulation tests, physical trials then began. The makeshift arena placement and setup would change depending on the time of the day. During the school semester, the team was limited to two quadrants of a complete arena. Before running physical trials, the rovers were tested for their overall performance and components endurance, which resulted in few discoveries such as damaged ultrasound sensors by an unforeseen event, and two of the rovers having faulty Inertial Measurement Units (IMU). Most of these issues were resolved after multiple troubleshooting and the physical one-on-one sessions with the UNM team, where new IMUs replaced the faulty units. The rover's endurance is limited; during our rigorous trials, the rover's motor shaft would occasionally malfunction. This would prompt the team to address this issue imperatively.

As previously mentioned, during school hours, our test space was severely limited. However, during these trials, the team members discovered some faults; one of the discovered faults was the GPS and Odometer not being synchronized. The differential tolerance, also known as the do-work tolerance, is the difference between the GPS and Odometer. Drift tolerance causes the center to be updated less frequently.

Physical trials became more elaborate during the campus Spring Break week as full experiments with the AprilTags on the arena were conducted. These complete runs, allowed the team to find all the undesired and mismatched behavior between the near-perfect simulated code and physical trials.

Some of these undesired behaviors include rovers that did not share any resemblance to the square-spiral path, Graphical User Interface (GUI) errors, and GPS program crashes. The team noticed that the rovers were overheating due to intense Texas heat as they were in direct sunlight for hours at a time. It was assumed that it was the cause behind the GUI and GPS crashes.

Figure5. *Physical arena*

Figure (5) shows one rover running on an arena (15meter × 15meter), which was the main venue to determine the optimum drift-tolerance values for the code.

5. RESULTS

After much trials and research on the Q&A forums, the team learned which files were the most important to learn about and alter. These files included obstacle controller, dropoff controller, and the most important file being search controller. The obstacle controller file controlled the autonomous behavior that the rovers use for avoiding obstacles. The dropoffcontroller file affected the behavior exhibited after picking up and when dropping off cubes, while the searchcontroller files used were integrated into the base code.

After minimal changes of other files within the src folder, the team began constructing the square spiral algorithm. Inside the searchcontroller.cpp file, more discoveries had been found; the most important one was that changing the values in searchLocation sections after the initial first_waypoint, which would allow the team to designate movement and direction. First_waypoint can be interpreted as a pinned location on a map; each waypoint is a place within the arena. The team's goal was to create a uniform path; this was done by removing the random base search and applying constant values.

In the searchController.h file, first_waypoint is a Boolean statement declared as true. It is then initialized as false in the searchcontroller.cpp to call/activate the if statement. Once the parameters have been met, it will proceed to the first loop, which constitutes its location within the field, and moves towards the wall. The initial 5.15 value is the maximum distance in meters the rover would travel without running into a wall. After the rover reaches the target distance, it turns 90 degrees to the right, moves linearly along the y-axis, and continues to decrement the size of the square until it reaches the center of the quadrant. Subsequently, the rover would continue its path by shifting its location to the next adjacent quadrant to maximize the arena coverage. Since the code instructs the rovers to move along the x-axis, the team used the searchLocation.x function only. The rover will move approximately 8.0 meters along the x-axis before commencing a reverse spiral. The reverse square-spiral is nearly identical to the square-spiral, per contra, M_PI/2 must be subtracted as it is an increasing square-spiral.

Finally, after this phase is completed, the rover will commence its larger coverage of the field to compensate for either the shifting of quadrants or the size of the arena. The parameters within this loop are similar; the only change is in searchLocation.x, where the values are multiplied by 8 because of the larger field. The largest loop is an exact template of the square spiral, only the variables change. This behavior pattern is shown in Figure (6 and 7).

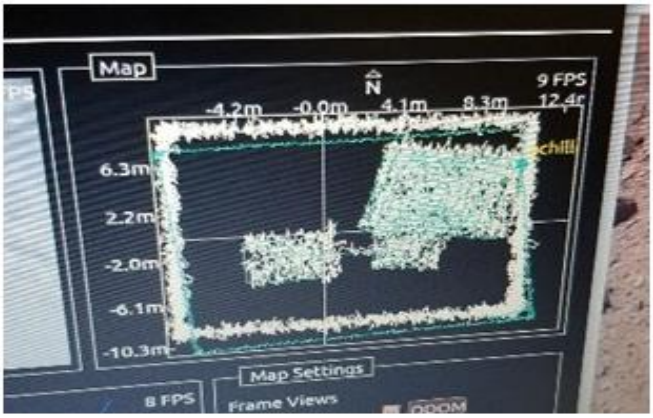

Figure6. *Single Rover Coverage Using Gazebo*

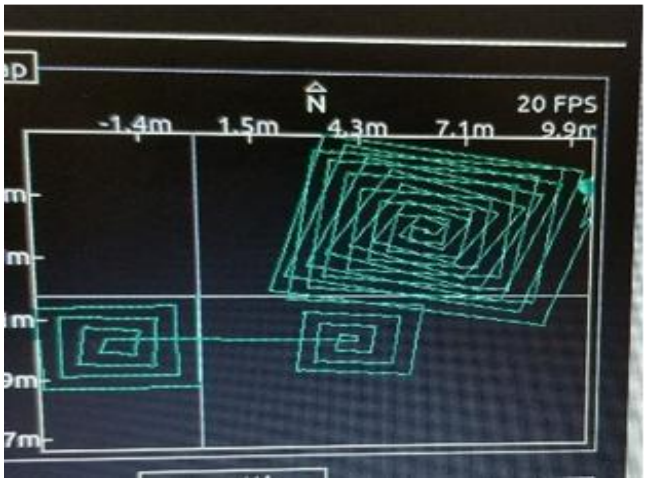

Figure7. *Odometer Depiction of Single Rover Coverage*

Lastly, the else statement will make the first_waypoint true, which would restart the code. Initially, the code was several "else if" statements to activate the loops, but when conducting physical trials, the code failed to follow the programmed path. The team placed the 'cout' commands (cout << searchLocation.x and cout <<searchLocation.y) in the code of the first two loops to show the execution of the code. Results from the log folder, showed whenever running the code in the simulation, the x and y coordinates for the first block of code would appear but not for the second. Meaning after the first line of commands, the code would fail to continue to the next. To explore the origin of the complication, the "else if" was changed to an "if" statement to see if the values would change while maintaining a working algorithm, and it did. Not only that, but the team was able to read the x and y parameters for the second block of code. This breakthrough prompted the team to address the "else if" statements, and that was a sign that the code was finally adjusted and completed. Despite all the challenges the team faced, the code was successfully completed within the allocated time. It can be concluded that the Gazebo simulations were the lifeline of the project, due to the numerous endeavors faced with the code and physical trials. Through the team's insistence, determination, and perseverance, physical trials were conducted and marked as a triumph, concluding the project.

6. CONCLUSION

The construction of an autonomous, synonymous algorithm has strengthened the team members' programming assets. Cohesively, the team created an algorithm, communicated effectively, and critically solved issues that would occur during simulations or physical trials.

As with any code, there is room for improvement. A note that can be passed to future DustySWARM teams is to make sure they have a proper arena with sufficient shading or run physical trials in intervals to prevent overheating. Other notable improvements can be placed in the rovers' behavior when picking-up and dropping-off AprilTags along with the rovers' ability to communicate amongst each other. Other areas of interest can also be in programming the rovers with the ability to stack AprilTags when dropping-off at the home base and also the ability to push AprilTags, the ones close to home, into the home base for a sure point.

DustySWARM team 3.0 with their Spiral Epicycloidal Wave (SEW) search algorithm placed the fourth (semi-finalist) in the physical competition and first place in the outreach paper competition among twenty-four (24) participating teams from all over the USA.

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.

REFERENCES

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

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

[3] Kantrasiri, Supichai, Pramote Jirakanjana, and On- Uma Kheowan. "Dynamics of rigidly rotating spirals under periodic modulation of excitability." Chemical physics letters 416.4 (2005): 364-369.

[4] Jeon, Noh, Oh, Park, and Park. "A Practical Coverage Algorithm for Intelligent Robots with Deadline Situations". Hongik University. Department of Computer Engineering (2010).

[5] W. E. Shots, "The Linux Command Line", 2nd ed., San Francisco, CA: Creative Commons, 2013.

[6] J. M. O'Kane, "A Gentle Introduction to ROS", Columbia, SC: University of South Carolina, 2014.

[7] http://nasaswarmathon.com/

AUTHOR'S BIOGRAPHY

Dr. T. Tashtoush is an Assistant Professor at the School of Engineering in Texas A&M International University (TAMIU), Laredo, TX. He got his Ph.D. and M.S. degrees in Systems and Industrial Engineering from State University of New York (SUNY) at Binghamton on 2013 and 2009, respectively and his B.S. in Mechatronics - Mechanical Engineering from Jordan University of Science and Technology (JUST), Irbid, Jordan on 2005. He is a multi-discipline engineer, who has industrial and academic experience in the field of Systems Simulation and Design, Production Quality and Management, Lean Manufacturing, Six-Sigma Processes, Industrial Robotics and Automation, Exploration and Autonomous Robotics, Artificial Intelligent (AI), Computer Integrated Manufacturing, 3D Printing Processes, Engineering Statistical Analysis, Project Management, Optimization, Instruments and Electrical Devices, Reliability, Healthcare Systems, Nano-Technology and Energy Harvesting, and Human Factors/Egomaniacs Studies.

Citation: *Tariq H. Tashtoush et.al., (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.22-28. DOI:http://dx.doi.org/10.20431/2349-4859.0701003*