TAMIU DustySWARM 2.0

Technical Paper

NASA Swarmathon 2017
Physical Competition

Team Members:

Abraham Pena Juan A. Medina Oscar Gutierrez Esteban Herrera Edgar Varela

University Name:

Texas A&M International University (TAMIU)

Faculty Advisor Name:

Dr. Tariq Tashtoush

Tariq Tashtoush

This document has been reviewed by the faculty advisor prior to its submission to NASA and verified that the document reflects the design used for the 2017 NASA Swarmathon Physical Competition.

Design of a Swarm Search Algorithm: DustySWARM Spiral Epicycloidal Wave (SEW) Code for NASA Swarmathon

Oscar Gutierrez, Esteban Herrera, Juan Medina, Abraham Peña, Edgar Varela, Roger Hernandez and Tariq Tashtoush
School of Engineering
Texas A&M International University, Laredo, Texas 78041
Email:Tariq.Tashtoush@tamiu.edu,
{oscar.gutierrez, estebanherrera, juan.medina, abraham_pena, edgar_varela}@dusty.tamiu.edu

Abstract: This paper focuses on the design and implementation of a search algorithm that DustySWARM team will use in the NASA Swarmathon 2017 physical competition. The rovers, or "swarmies", provided by NASA must be able to communicate between each other while searching autonomously for resources in a simulated environment. Some of the methods implemented during this project were the square spiral path, spiral path, and the epicycloidal wave path. The results indicated that the most efficient method was the epicycloidal wave path as it collected more resources within the allowed time limit. This paper will summarize TAMIU DustySWARM 2.0's design and code development process.

Keywords: *Swarm Robotics; Search Algorithm; Autonomous Robot Swarm; Collaborative robots*

I. *Introduction:*

NASA is a governmental agency dedicated to space exploration, they started the Swarmathon Competition in 2016 where they invited different universities from across the country to work on creating a robotics system that mimic ant's behavior. Each university in the physical competition will be provided with three small robots called "swarmies" and they should search, find, and gather resources from a sounding simulated environment while communicating among each other and working in autonomous configuration, similar to ants.

TAMIU DustySWARM team is participating for the second year of this competition. In 2016, DustySWARM 1.0 placed 3rd in the virtual competition. For the 2017 NASA Swarmathon, TAMIU was invited to participate in physical competitions. DustySWARM 2.0 is in charge of developing a code that will be used to control the swarmies and achieve the goal of collecting the most targets within the time limit. The Spiral Epicycloidal Wave (SEW) path illustrated in this paper was developed by DustySWARM 2.0 team members. The methods and designs followed to reach such search algorithm are described in detail in this paper. The results demonstrated the efficiency of the selected method and its score toward the final goal.

The winner of this competition will be selected according to the number of resources gathered. NASA will consider all universities participating in this event to gather ideas to implement when sending robots to the Martian planet. Overall, the NASA Swarmathon Competition is a great opportunity to demonstrate teamwork, outreach, planning, and coding skills.

II. *Related Work:*

a. *Original iAnt Code and Return Logic*

The original iAnt Code developed by the University of New Mexico (UNM) will be used as the base for the rovers search and collect approach. DustySWARM 1.0 modified the original iAnt Code by adding a "return logic", that allowed the robots to return to the position where they had found a resource after dropping it off into the collection zone. By implementing the "return logic" code, the team noticed an increase in the amount of tags collected [1].

Based on DustySWARM 1.0 code changes, our current team decided to modify different sections of the code. Some changes include SearchController.cpp and random generator factors, which resulted in collecting more resources.

b. *Dynamics of rigidly rotating spirals under periodic modulation of excitability*

DustySWARM 2.0 builds their code on the design of the epicycloidal wave seen in figure 1 (d). This wave had been designed and mathematically

modeled based on Belousov-Zhabotinsky (BZ) reactor. Kantrasiri et. al. conducted two experiments: an experimental observation and a simulation. Experimental observations consisted of exposing the BZ reactor to various light intensities and pulsatory modulations (short light pulses), while simulation was performed using numerous equations within the Oregonator model to output the desired results [2]. The equations involved within the simulation aided DustySWARM team in the development of their Spiral Epicycloidal Wave (SEW) Code.

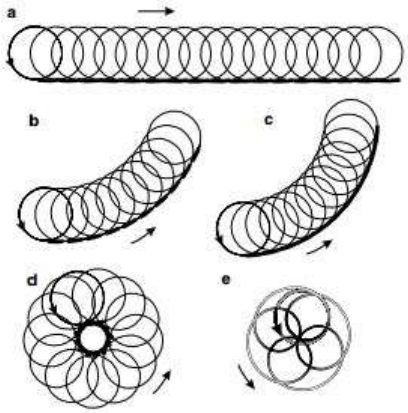

Figure 1: (a) – (d) are the results of the experimental observations using pulsatory modulations. (e) is the result of using sinusoidal modulations within the simulation experiment. The spiral waves resemble hypocycloidal and epicycloidal figures. [2]

III. Methods:

DustySWARM 2.0 team spent time to get familiarized with both the UNM code and DustySWARM 1.0 code. Because the objective of the swarmies is to collect as many resources as possible in an efficient manner, the team researched multiple possible patterns that would satisfy that goal. By the end of the research, the team decided to experiment in two possible paths, namely square spiral and circular spiral, which eventually led to a new path pattern that is currently being used and modified.

a. *Square Spiral Path*

The first path algorithm was developed due to the idea that increasing the search area of the swarmies exponentially would increase the amount of objectives recovered at the end of the simulation. The decision of making a square approach of the spiral was made by the belief of making a simpler code and, at the beginning of the project; dealing with cosine and sine functions (covered in a traditional spiral) seemed too complicated. To implement this path, we needed a dependent variable 'X' that would increase each time a task was finished. The task was separated into two sub-task that consisted on a displacement of 'X' on the x-axis followed by a 90 degrees turn, and a displacement of 'X' on the y-axis 'followed by a 90 degrees turn. After a task was finished, we had a variable 'P' (with an initial value of -1) that would change its sign in order to determine the polarity of the displacement of both axes.

Example: we want a spiral that has an initial negative displacement on both axes of 10cm each and after its first turns it should follow a uniform path, this means that after two turns, the 'X' variable should increase its value by 10cm and change its initial polarity to positive, and so on following the same pattern. Technically making a generalized function of:

a) X = X + 0.1m (after each task)
b) P = P * (-1) (after each task)

A task is represented as followed:

c) Displacement in x-axis= current_position.x + X * (P)
   Turn = 1.57 radians
d) Displacement in y-axis= current_position.y + X * (P)
   Turn = 1.57 radians

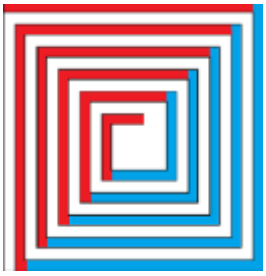

Figure 2: Represents the previous example with a negative displacement in red and the positive displacements in blue. Eventually implementing these changes into the code provided by NASA was harder than it seemed and the square spiral path was dropped

b. *Spiral Path*

After further analyzing the code, and playing with different values, the team managed to understand the necessary steps needed to create the spiral pattern. Changing the goal objective functions in the Searchcontroller.cpp allowed us to aim toward the completion of the spiral path in a different approach. The way in which the spiral was meant to be achieved was by imitating its behavior rather

than replicating it. This means that instead of making a perfect spiral across the field, the swarmies would replicate a spiral based on translations and angle changes between each goal objective. This would allow a small translation of 'X' followed by a change in angle of 'A' represent the Spiral Path gradually as shown in Figure 3.

Figure 3: Illustration of the square spiral path as the search algorithm for DustySWARM 2.0

Nonetheless, by continuing to analyze the spiral method, the team concluded that eventually one of the goal locations would lay outside the perimeter of the field (as seen in Figure 4) making the rovers get inside a loop of following an unreachable goal blocked by an obstacle (in this case the walls).
While looking for a solution to this problem, one of the changes inside the values of the SearchController.cpp completely changed how the pattern of the spiral behaved, and to the team, this new pattern worked for the best of the project.

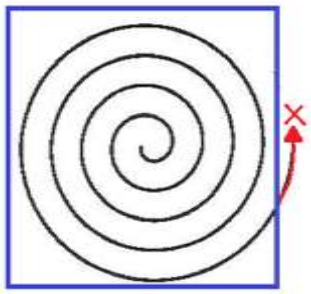

Figure 4: Limits of the spiral path

c. Epicycloidal/ Spring Wave Path

As mentioned in the previous section, this path was achieved by updating the SearchController.cpp by changing the goalLocation.theta to be equal to the currentLocation.theta plus a random Gaussian distribution of 9 to 12 degrees. By utilizing the theta into the provided values of goalLocation.x and goalLocation.y, we managed to develop a loop path that origins from the center of the map towards the boundaries. One of the positive aspects of this path is that after a swarmie is presented with an obstacle or an objective, the center of that loop will shift its center allowing the swarmie to cover more ground

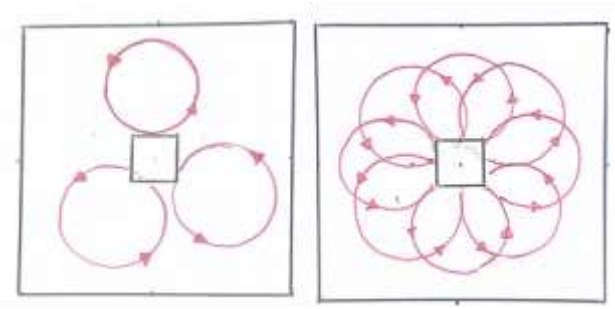

Figure 5: Illustration of the Epicyclodial/Spring Wave path as the search algorithm for DustySWARM 2.0

in the field. Figure 5 shows the origin loop for the three swarmies, followed by a representation of a loop with a shifting center.
The Spiral Epicyclodial/Spring Wave Path allows an uninterrupted search as none of the goal locations will end outside the field, but with the sacrifice of leaving possible objectives left behind in the corners. The size of the loops can be altered by modifying the angle and distance traveled after each goal location is achieved. After weighing all pros and cons, this path was chosen by the team as it will provide the safest path while collecting the most resources.

IV. Experiments:

The experiments were conducted using the simulation provided by NASA. For the team to decide if the Spiral Epicycloidal Wave path was going to be used for the competition, a number of simulation runs where made to determine its functionality. The first initial runs of the simulation with the Spiral Epicycloidal Wave path demonstrated fewer simulation crashes and malfunction of the swarmies, making the decision of taking this path as our main development easier. While conducting the simulations, we stumbled upon several problems that caused variations in our data. These problems included: inaccessible goal locations, rovers going inside the center, drop-offs made outside the center, objectives getting stuck within the swarmie's claw, and others. All simulation runs had at least one or two of the mentioned problems preventing the affected swarmie from collecting targets, figure 6 illustrates one of the problems mentioned before.

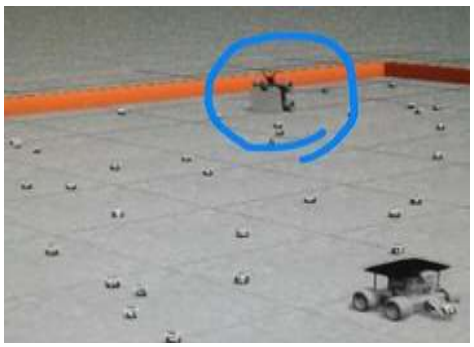

Figure 6: Issues happened while running the developed code in the simulation system

Nonetheless, we ran the simulations with the disabled swarmies and still managed to collect a total of 19 – 30 objectives of 100 within the 30 minutes. The purpose of running simulations of 100 objectives was to let the swarmies collect objectives and still let us view if the path taken was functioning the way it should.

Once the decision of taking the Spiral Epicycloidal Wave path was set in stone, the team worked on fixing the issues that prevented the rovers from functioning the way they should. These changes where mainly addressed by changing DropOffController.cpp, PickUpController.cpp and Mobility.cpp files. Most of the problems occurred when a swarmie is going to drop an objective at the center of the map, or trying to collect the objective. Each change was analyzed in the simulation to determine whether if it benefited the code or not. This trial and error approach allowed us to get more familiarized with the code and make further changes.

V. Results:

The first change to solve an issue with the rovers missing the targets when attempting a pickup. By increasing the value of the result.cmdVel from 0.18m to 0.2m when a target was still not locked, the rovers managed to get the extra push needed to successfully pickup target 90% of the time. The team also deleted the negative the movement of backing up after completing a successful pickup. Although the previous change benefited the code, they did not solve the problem entirely. In Figure 7 and 8 we can see two cases representing two possibilities of a swarmie coming in to the center attempting a drop-off. It is important to take in consideration how the code behaves in order to detect the center as this part of the code is critical for the swarmies's functionality. There are three statements that affect how the swarmies move: right, left, and seenEnoughCenterTags. There is also an extra variable by the name of TurnDirection that is dependent of seenEnoughCenterTags being true, if that is the case, the value will change from a positive one to a negative one. If seenEnoughCenterTags is false, variable right will execute a negative displacement of 10cm and an angle turning of 0.2 radians allowing the swarmie to position itself towards the center. Same goes for variable left but with a negative angle. If the statement of seenEnoughCenterTags is true, the negative 1 will affect both the direction and the angle of the right and left variables, changing right to a positive displacement turning left, and vice versa. It is also worth mentioning that seenEnoughCenterTags will allow the program to determine if there are too many center tags in the left or right. If the value of right minus five is greater than left, the value of left will be false making allowing a right turn, and the reverse scenario happens if there are too many in the right. Figure 7 illustrates a successful drop-off once it sees enough tags in the left (1) then enough tags on the right (2) followed by going inside to drop-off (3) and back up (4). Yet if the angle in which the swarmie is coming in is too acute, the rover will execute only one of the turns making a drop-off outside the center and eventually getting inside the center due to the reversing; leaving the swarmie inside and pushing all collected targets outside the center as seen in Figure 8.

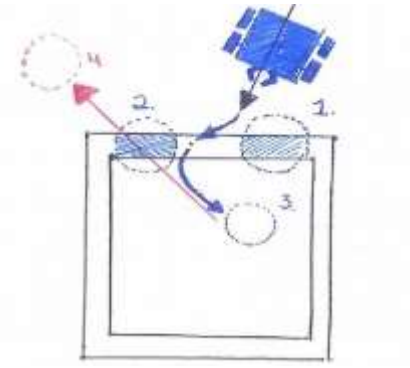

Figure 7: Successful target drop-off.

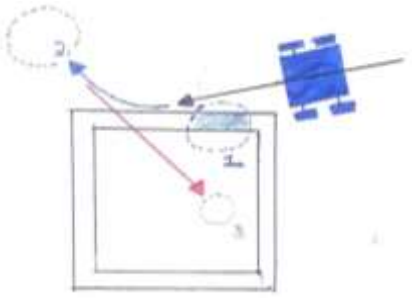

Figure 8: Error when target dropping-off.

Further modification included changes in the DropOffController.cpp by adding 0.1 radians more to the angle turned when seeing a tag in the left or right. A variable by the name of angleDiv was created in order to decrease the value of the previous turn by 0.2 radians. In the beginning of the Epicycloidal/Spiral wave pattern we have a constant increase of 10 degrees in theta per each goal location completed. The following figures are the result of how the swarmies behave when there are no targets in the field. Figure 9 illustrates the path that one swarmie will follow when there are no targets, coalition; with other swarmies existence, the center of the loop will be shifted which will make the swarmies cover more terrain. Figure 10 represents the simulation run for 30 min with targets in the field, we can observe that the path changes drastically as the center of the loop is shifted for each objective and obstacle encountered. At the end of the simulation the rovers collected 24 targets, and decided that adding the Gaussian distribution of 9-12 degrees in theta will prevent the rovers from staying in a perfect circle in case of no targets encountered.

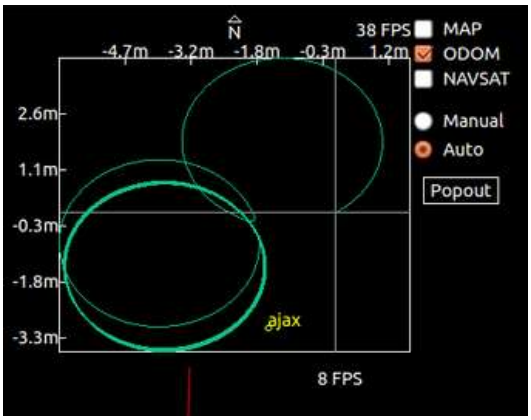

Figure 9: Terrain explored by swarmies when obstacle avoidance

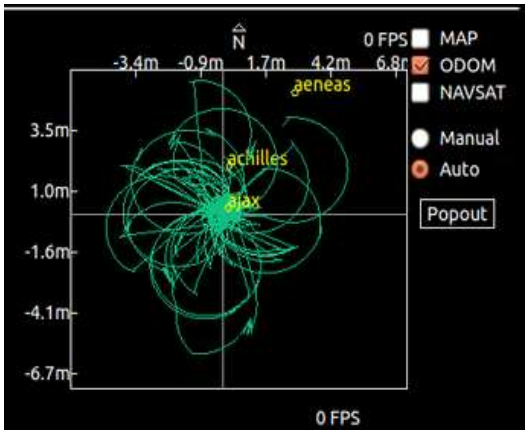

Figure 10: Terrain explored by swarmies during simulation

VI. Conclusion:

In the end, the development of a searching algorithm allows the team to understand the range of possibilities in which the swarmies can behave. Because there is still a significant amount of randomness within the simulation as of now, the team is still working on optimizing the code in order to increase efficiency. When it comes to understanding the code, there has not been a more reliable method than simple trial and error, as it takes independent changes from independent members in order to understand how that modification altered the code. Eventually, if a change is noticeable, the team learns about it and can refine it in the future if needed.

References

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

[2] 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.