4/13/2016

NASA Swarmathon

Morehouse College

Terrell Glenn, Steven Ragland, Alexander Meyer,
Joshua Pulliam, Ernest Holmes, and Nicholas Riley
DR. DWAYNE JOSEPH & DR. AMOS JOHNSON

NASA Swarmathon: Morehouse College

Terrell Glenn[1], Steven Ragland[1], Alexander Meyer[1], Joshua Pulliam[1], Ernest Holmes[2], Nicholas Riley[1], Dwayne Joseph[1], and Amos Johnson[1]
Department of Physics and Dual Degree Engineering Program[1]
Department of Computer Science[2]
Morehouse College
Atlanta, GA 30314
Email: {Terrell.Glenn, Steven.Ragland, Alexander.Meyer, Joshua.Pulliam, Ernest.Holmes, Nicholas.Riley}
@morehouse.edu

***Abstract* — As the world becomes more technologically advanced, research in implementing robots and robotic systems into society has drastically increased. In a specific field such as swarm robotics, robots are coded to have a "swarm" mentality that insects use to survive in life. By researching animals such as bees and ants, algorithms were replicated onto robotic hardware. NASA, in an effort to conduct a space mission to Mars, has enlisted the efforts of several minority serving institutions, including Morehouse College, to conduct their own individual research, implement their changes to the source code provided to them, perform various simulations, and finally test the physical rovers for real-world debugging. This report provides an in-depth analysis of the steps that the Morehouse students took in order to succeed in the NASA Swarmathon competition.**

Keywords – Swarm Robotics, Ubuntu, Robot Operating System

I. INTRODUCTION

The NASA Swarmathon competition is a technical competition hosted and funded by NASA's Minority University Research & Education Program (MUREP), and is in cooperation with the University of New Mexico. The competition is open to students at Minority Serving Institutions (MSI) in order to provide those students with interactive methods of conducting research, and to encourage them to learn more about robotics, computer programming, and engineering in general. The competition is focused on developing search algorithms for robots that operate with "swarming" patterns and cooperation, and was born out of the need for more efficient exploration methods from the NASA space exploration mission to Mars and the Moon.

The Swarmathon competition is a unique opportunity for minority students to learn about the many challenges that affect the computer scientists and engineers at NASA. Other programs put more emphasis on the mechanics of motion, electrical components, or the coding than any other component of the competition without much room for elaboration in the other aspects. The Swarmathon, however, puts particular emphasis on learning how to create the algorithms using computer programming languages such as C++ and Python, but also provides the students with physical rovers (swarmies) on which to perform real-world tests. This allows students to bring all of their knowledge about mechanics, electricity, and circuits, and apply it to a real-world situation to be solved using computer programming algorithms. Thus, the Swarmathon competition incorporates aspects of all robotics competitions for a better-rounded student experience.

Figure 1: Swarmie Robot from NASA

II. Related Work

A. The Beauty of Swarm Robotics

Swarm robotics is truly a remarkable idea that builds upon the notion that primitive species can work together to accomplish some large task. Most swarm intelligence research is inspired from how naturally occurring swarms (e.g. insects, fish, mammals, etc.) interact with each other within the swarm [1]. According to Dr. Ying Tan, a professor at Peking University of Beijing, China, swarm robotics is a fairly new approach at coordinating multi-robot systems to cooperate and achieve a common goal, and consists of simple physical robots [2]. This implies that there is not a need for overly complicated robot systems; the actions of the individual robots adds to the complexity of the overall system. But, before we can delve further with understanding swarm robotics, we must first understand the nature of swarms and the swarm mentality.

B. Ant Swarms

When the term "swarm" comes to mind, one of the first ideas that come to mind is "ants participate in swarms." Initially, we gathered materials for insight on how ants behave and how we could properly mimic said behavior. Deborah M. Gordon, who is a biologist at Stanford University stated in an interview with the National Geographic magazine, "If you watch an ant try to accomplish something, you'll be impressed by how inept it is. Ants aren't smart, ant colonies are. A colony can solve problems unthinkable for individual ants, such as finding the shortest path to the best food source, allocating workers to different tasks, or defending a territory from neighbors. As individuals, ants might be tiny dummies, but as colonies, they respond quickly and effectively to their environment," [3]. The next question that was raised was, "how do these 'dummy' actions lead to such complex and intelligent solutions?"

The main sources of communication among ants are pheromone usage, sound, and even touch [4]. Ants sniff each other with their antennae to determine what type of ant is returning to the nest, where it has been, and whether it should join or choose a different job to perform. But the most important thing to know is that no one particular ant is telling any other ant where to go or what to do. There are a set of instructions that the ant will naturally come to recognize and understand from these signs. That's exactly how swarm intelligence works; simple creatures following simple instructions stimulated by local information [3]. Doing so simplifies the actions performed by each individual ant, and it is obvious to see that, without the group, an individual ant would never be able to accomplish such a large task by itself.

C. Connection to Swarmathon Competition

Initially, our team thought that it would be best to model the behavior of our rovers using a leadership structure where, at the beginning of the competition, each rover was designated a position, and a set of instructions to carry out individually. We quickly realized that we needed to model our approach to the Swarmathon competition after this basic ant model, where there is no clear leadership structure, but a set of basic instructions that each rover followed given certain cases. This approach, which other researchers have deemed credible, yields adequate results to be discussed later on in this report.

III. Methods

A. Starting Out

At the beginning of the semester, students participating in the NASA Swarmathon Competition were required to download a myriad of software before delving into the body of the competition. To begin, many users had to download the operating system, Ubuntu, version 14.04 or later. Once that was downloaded, Robot Operating System (ROS) Indigo could be installed using certain command line prompts to access the Swarmathon GitHub repository. These operating systems were imperative for running Swarmathon-ROS, as it was designed and tested exclusively for these systems. Afterwards, all the other plug-ins were added, including the Gazebo simulator.

More preparation work was needed to overcome the learning curve of modifying a program with thousands of lines of code. One thing to note is that many of the students had little knowledge of the C++ coding language, and some had no programming experience whatsoever. A remedy for this dilemma included a number of

crash course sessions in basic programming principles and common terms and functions. Additionally, we viewed a few of the software tutorial videos available on the NASASwarmathon.com website. Once everyone was sufficiently informed about coding basics, a couple of days were spent engaging the competition rules and Q&A Forums to make certain the entire team had a firm understanding of the objective. From there, we proceeded to maneuver through the different /src directories held within "rover_workspace" directory.

B. Developing a Game Plan & Pseudocode

As we began to brainstorm techniques for the challenge, we tried to approach the problem from all different angles. We asked questions: how do the Swarmies communicate with one another, how do they use the different `StateMachines` to move, what is the best pattern for searching. At first we wanted to allot a portion (approx. two-thirds) of the competition time purely to detecting April Tags and storing their locations. With the remaining time, the Swarmies would start retrieving tags that had been detected, starting with those closest to the center. With this game strategy, it became crucial for us to devise an efficient method for searching the playing field for tags.

Some of the search patterns that we devised were fairly simple. One pattern tactic involved the Swarmies tracing the barrier walls in a rectangle, working their way towards the center, as shown in Figure 2. A second plan involved making each Swarmie responsible for a 120° section of the field. Then each would make a series of runs from the center to the barrier and back along a different path, incrementally covering about 6° with each run, pictured in Figure 3. Yet another method to detect tags was to have the Swarmies wander until they encounter a tag. Upon finding this tag, it would rotate 30° to the right and then 60° to the left. This back-and-forth sweeping action would serve to visually detect any adjacent tags and is diagramed in Figure 4.

The team dealt with a myriad of issues attempting to implement the above methods, ranging from inconsistency of performance to impractical time-consumption. However, the dilemma that most limited progress was how difficult it was to translate these verbal algorithms into a functioning code. Figuring out how to utilize

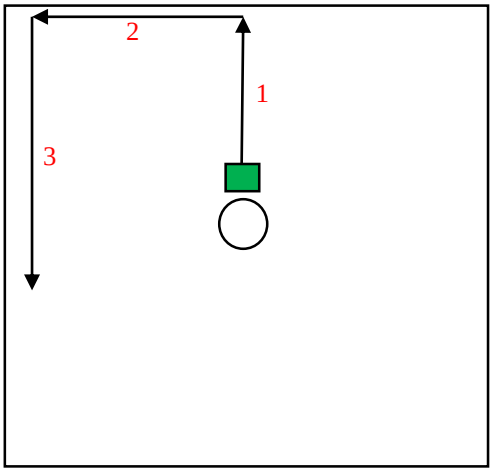

Figure 2: Swarmie Tracing Barrier Wall

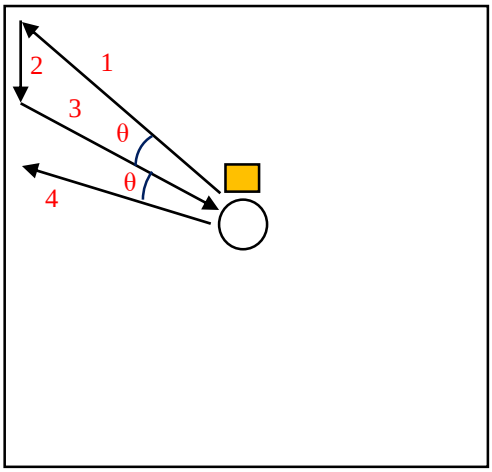

Figure 3: Swarmie Flower Petal Pattern

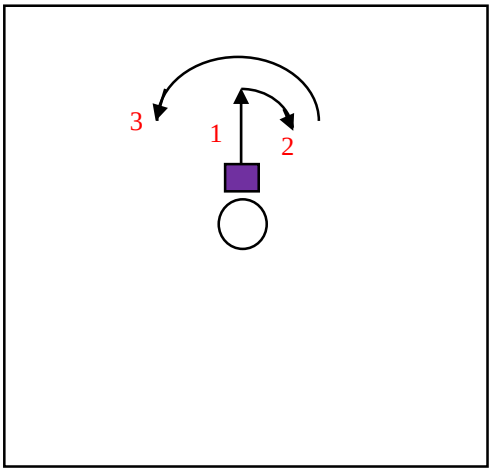

Figure 4: Swarmie Sweeping Pattern

the State Machine took hours, and even then, we still were not able to achieve conclusive results. In the face of this obstacle, the team completely shifted its focus to consistent retrieval of the tags. The term "consistent retrieval" is used in reference to our technique of returning to the location of the last April Tag collected. The method is simple. We were to write a program that will

1. Encounter an April Tag
2. Determine whether the tag has been detected before and publish
3. Set return location and travel to center to score tag
4. Return to previous location to check for another tag .
5. Repeat steps until no more tags are found.

While this tactic is simple, its validity is supported by substantial evidence. Under Power Law Distribution, there is a 24.7% (Equation 1) chance that when a Swarmie detects its first tag, there will be another tag in the same clustered location. And under the Cluster Distribution, there is a 100% chance that there will be multiple tags at that particular location. Although the distribution of tags is unknown before the start of the match, returning to the location of the first detected tag will yield multiple opportunities to score. Even if there is not a cluster, the Swarmie will be able to search for the next tag and perform the same routine.

$$\begin{array}{r} 64\ clusters\ of\ 1\ tag \\ 16\ clusters\ of\ 4\ tags \\ 4\ clusters\ of\ 16\ tags \\ +1\ cluster\ of\ 64\ tags \\ \hline 85\ total\ clusters \end{array}$$

Note, that there are 21 clusters that contain more than 1 April Tag; thus, the probability of finding another April Tag is simply

$$\frac{21\ clusters}{85\ clusters} = 24.7\%$$

After some deliberation, we were able to write a properly working program, and test it out for the first time.

IV. Experiments

A. Swarmathon-ROS Simulations

In Swarmathon-ROS, there are several options that were at the disposal of the team. First, we had the option of choosing the surface upon which our rovers would be moving (Kennedy Space Center concrete, Parking Lot concrete, and Gravel). The next was the distribution of tags around the field (uniform, clustered, Power Law). Finally, we could choose between the preliminary setting (3 rovers) and the finals setting (6 rovers). We kept the KSC concrete as a control variable, and ran multiple simulations while changing the distribution of tags and round type. Our results are shown in Table 1.

The "targets missed" column denotes the number of times we received the error message "`rover_name attempted a drop off but was not carrying a target,`" which is being shown because a robot is attempting to publish an image of the collection zone tag on the `/targetDropOffImage` topic without first identifying a tag and publishing its image on the `/targetPickUpImage` topic. Updates to the Swarmathon-ROS base code eliminated a lot of these errors, but not all. After our simulations were complete, it was time to move on to the physical rover testing.

B. Swarmathon-ROS Physical Test

Although off to a slow start, we were able to run some physical tests using the rovers. We tested our rovers inside a room with linoleum tile on the floors, fluorescent lighting above, and a moderately weak Wi-Fi connection. Despite this, the rovers performed well and we were able to gather some data and produce the results found in Table 2.

V. Results

Table 1					
Round Type	**Tag Dist**	**Sim. Time**	**Targets Collected**	**Targets Missed**	**Total Tags**
P	U	30m 34s	22	15	37
P	C	31m 4s	25	20	45
P	PL	32m 34s	57	60	117
P	PL	30m 10s	60	76	136
F	U	60m 12s	44	34	78
F	C	60m 22s	50	35	85
F	PL	60m 54	128	155	283

As shown in the table, when using the uniform and clustered tag distributions, it is obvious to see that our rovers collected fewer tags, and received the error message fewer times. However, when utilizing the Power Law distribution, we collected a considerable amount of tags, but also received the error message more times.

Table 2					
Round Type	**Tag Dist**	**Sim. Time**	**Targets Collected**	**Targets Missed**	**Total Tags**
P	U	30m 0s	18	11	29
P	C	30m 0s	20	10	30

We were not able to run complete tests on all of our rovers, but our results are fairly consistent. We believe the issues to be caused by bad Inertial Measurement Unit (IMU) calibration, the difference in friction between our floor surface and the wheels of the rovers, as well as difference in lighting caused by our indoor lighting fixtures. In the future, we hope to be able to run more simulations, and more physical testing in order to receive a more thorough investigation.

VI. Conclusion

All in all, our experiments were successful in creating an environment in which we could come implement some basic principles from previous swarm robotics work. Through experimentation, we have determined that the "no-leader" approach is useful in these types of multi-robot systems because it allows for simplicity when writing algorithms, and proves to be a more powerful method when compared to others. Due to the lack of experience with ROS, linux-based operating systems, and programming in general, it will be useful for us to gather a more experienced team that may not have the same pitfalls that we encountered. In the future, it is the hope to implement some of the aforementioned algorithms in order to see which of those approaches might prove more fruitful than the approach we ultimately chose.

VII. References

[1] E. Bonabeau, M. Dorigo, G. Theraulaz. "From Natural to Artificial Swarm Intelligence." Oxford: Oxford University. 1999. Press.

[2] Y. Tan, Z.Y. Zheng, "Research Advance in Swarm Robotics." Science Direct, vol. 9. Issue 1. Beijing. pp. 18 – 39.

[3] P. Miller, "The Genius of Swarms." National Geographic Magazine. 2007. pp. 1-7.

[4] D. Jackson, F. Ratnieks. "Communication in ants." Curr Biol 2006; 16(15): 570-4.