Search and Site Fidelity for Multi-robot Target Collection
Swarmathon Technical Report 2017

Submitted by:
Casey Lorenzen, Joshua Cohenour, Daphne Faircloth, Catherine Spooner, Brittany Jennette, Donny Lopera, Ashlee Gaskins

Faculty Advisor: Dr. Sambit Bhattacharya

Department of Mathematics and Computer Science
Fayetteville State University
Fayetteville, North Carolina 28301

Dr Bhattacharya has read and authorized the submission of this paper.

Search and Site Fidelity for Multi-robot Target Collection
Swarmathon Technical Report 2017

Casey Lorenzen, Joshua Cohenour, Daphne Faircloth, Catherine Spooner, Brittany Jennette
Donny Lopera, Ashlee Gaskins

Advisor: Dr. Sambit Bhattacharya
Department of Mathematics and Computer Science
Fayetteville State University
Fayetteville, North Carolina 28301

Abstract- The Swarmathon project is a collaborative effort by the NASA Minority University Research and Education Program (MUREP), the University of New Mexico, and Swamp Works at Kennedy Space Center. It was designed to not only encourage the development of effective search algorithms by students for possible use in robotic space exploration, but also to engage them in new research and innovative technology. The Fayetteville State Team embarked upon this project by exploring potential improvements to search area coverage, including use of beacons or the implementation of rivaling force. Challenges posed by localization, newly introduced Swarmathon 2017 code and hardware requirements, as well as unexpected test results prompted a narrowed focus on formulating an effective search strategy incorporating site fidelity and an explore-and-exploit algorithm.

Keywords: - algorithm; robotic; search; swarm.

I. INTRODUCTION

When one thinks of robotics, often the images that come to mind are the automations introduced into our industries or the usually frightening villains of science fiction films. Even for most scientists, it is still a relatively new field of study and application.

One such application is being pursued by the National Aeronautics and Space Administration (NASA) and has led to a collaboration with the Moses Biological Computation Lab at the University of New Mexico (BCLab-UNM). The result of this collaboration is the NASA Swarmathon, a competition for undergraduate students at Minority Serving Universities and Colleges intended to improve coding and research skills, encourage participation in research and technology, and hopefully gain new search algorithms for use in Swarm Robotics and space exploration.

Swarm robotics involves the deployment of multiple simple robots to perform a variety of tasks rather than utilizing a single agent. As citizens of Earth, we have seen the considerable investment of time and resources the Curiosity Project [1] required. Imagine, then, the Mars rover experiencing a mechanical issue that renders it inert. Nearly 3 billion dollars has just been reduced to a useless pile of electronics. Imagine instead that the same price tag yielded not a single complex agent, but a hundred smaller, simpler agents. The loss of one would not be a catastrophic event, and even a 50% loss would still leave fifty rovers on Mars, exploring, recording, transmitting, and collecting samples.

II. RELATED WORK

A. Swarmathon repository

The GitHub Swarmathon repository [2] created by BCLab-UNM provided the base code for the competition. Our first task was to read this repository and gain an understanding of what the variables represented, what each function did, and how the rover processed and shared data.

For most of the team, this was also an introduction to C++, which posed its own challenges and rewards.

B. OpenCV and the AprilTag Library

Before our team even officially entered the competition, we were working on ways to improve target detection and collection. Swarmathon organizers announced that cubes with identical AprilTags printed on all sides would be collected in the next competition instead of flat AprilTags being digitally returned.

Using the OpenCV library [3] and a demo developed at MIT available from the AprilTags C++ Library [4], the team worked on creating a training bank of positive and negative annotations as a machine learning exercise. We used a training bank of 834 images for our detector and gathered additional images to use as a test bank. Of the 255 cubes in the test images, our detector correctly identified 101 while the MIT demo found 181. However, we also had 75 reported as false positives and MIT had none as illustrated in Fig. 1.

We determined we needed to increase our training annotations to at least 1000 annotations and perhaps as many as 3000. However, it was during this time that Swarmathon announced an AprilTags Library object detector would be incorporated in the 2017 base code. The team set aside this project as something to be pursued later for other applications.

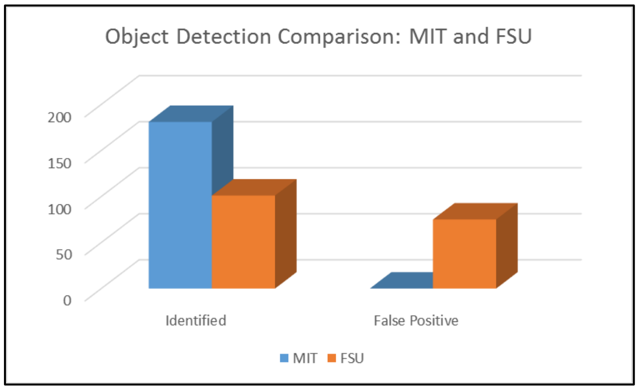

Figure 1. Early testing of Object detector

C. Search area coverage and path planning.

In the early days of our research we wanted to determine if we could improve not just *how* we searched, but *where* we were searching. We hoped to avoid having multiple rovers searching the same area while other areas were unexploited. Informing other robots of a target rich environment was also desirable.

The dissertation by Hoff [5] seemed promising as it discussed foraging, returning targets to a home, and communicating with other searchers. Unfortunately, his method requires some robots that can be dedicated to being beacons that can communicate localization information with the remaining walkers. The number of robots we will be able to deploy is too small to allow us to replicate his method. In addition, given the necessary downward camera focus for cube identification, beacons would not be visualized by walkers but instead avoided as obstacles when detected by sonar.

The strategy described by Mirzaei [6] for including separate search and coverage agents was something we thought we could incorporate, with modifications. As with the beacon issue previously mentioned, we have a limited number of robots to work with and assigning dedicated roles would be imprudent. However, changing their focus at different times during the competition is something we decided we would pursue. This is discussed more under Experiments within *Explore and Exploit.*

In addition, we reviewed the findings of Kolushev and Bogdanov [7] which explored methods that would improve collision avoidance by extending it to include other robots as well. This led us to the concept of rivaling force [8] as a potential means of limiting overlapping search by rovers in proximity.

The concept of rivaling force allows path selection based on a cost function as seen in Fig 2. In our case, we wanted to make an overlap less desirable. At time k, rivaling force $F_{ij}(k)$ would be non-zero if two conditions were met. Condition 1 requires that RoverA$_2$ be within a maximum distance μ and a maximum angle $\pm\Theta$ from Rover A$_1$. Condition 2 stipulates the difference in their headings is within

$$(180° - \bar{\chi}\,,\, 180° + \bar{\chi}\;)$$

with $\bar{\chi}$ representing the maximum allowed difference in heading angle.

Thus, the rivaling force formula (1) will return 0 if Conditions 1 and 2 are not met. K_1 and α are positive constants to be tuned by the investigator, ρ_{ij} is the shortest distance between RoverA$_1$ and the path of RoverA$_2$, and $\boldsymbol{\rho}_{ij}$ is a unit vector of the associated normalized partial derivative.

$$F_{ij}(k) = \begin{cases} k_1 e^{-\alpha \rho_{ij}} \vec{\rho}_{ij} \\ 0 \end{cases} \tag{1}$$

III. METHODS

A. Hardware

Sonar was tested to confirm that it was operational for obstacle avoidance. We noticed that the rover would turn left whenever an obstacle was encountered even if that obstacle was located to the left of the rover. A quick code search in mobility revealed the function obstacleHandler had a sign error on the constant used to turn away from the current heading. Changing the sign to a negative in the else block allowed our rovers to turn right when encountering an obstacle on their left.

Incorrect wiring of one wheel was discovered and corrected during the hardware check. What impact it may have had on testing prior to the current project is unknown.

Installation and testing of the newly introduced gripper is addressed later in this paper.

B. Software

Most team members opted to dual-boot Ubuntu alongside the native OS on their personal devices. We encountered a wide variety of errors that could be attributed to mistakes made due to unfamiliarity with Linux systems. Repeated reports of necessary reinstallations of Ubuntu and redownloading of Swarmathon code resulted in everyone agreeing that making any suggested updates to Ubuntu or other installed applications would be unwise.

C. Localization

We had been warned the GPS would not be precise enough to be useful for pinpointing the location of a rover or a target. Testing confirmed this as shown in Fig.3, with significant drift present and results often several meters off target.

Likewise, the wheel encoders had limited usefulness for tracking rover location. Over time, even minute differences in wheel motion, terrain type, and traction not only caused the rover to be inaccurate in predicting its location on the x-y plane, but in reaching a goal location. On a long drive, it would drift from its course, no doubt thinking it was moving in a straight line, but in reality it would eventually produce a curving path. Periodic returns to home to reset x and y was vital.

The inertial measurement unit (IMU) was reinitialized for each rover both before testing and again closer to the deadline when localization issues during physical testing became increasingly apparent.

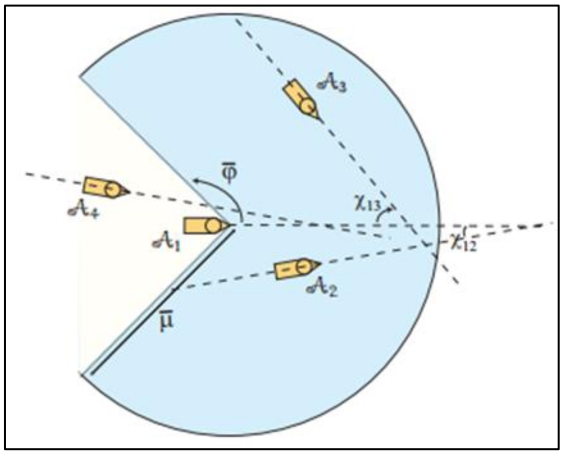

Figure 2. Rivaling forces between agents [8]

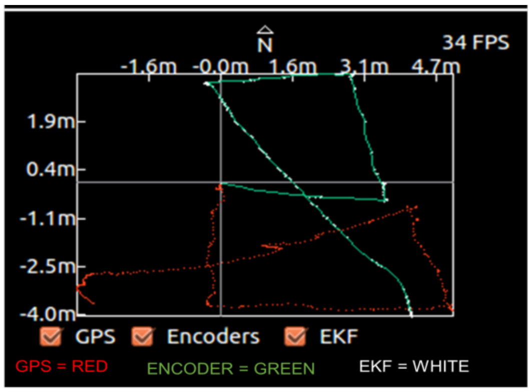

Figure 3. Early localization testing

D. Rivaling Force

Our simulation results testing rivaling force were promising. When rovers were within the set minimum range of 3 meters and had headings that would result in search area overlap, the rover subjected to the rivaling force would change to a new heading consistently. Physical testing was not implemented for reasons addressed in our Conclusion.

E. Explore and Exploit

1) Explore: Phase one of our search and collect algorithm takes place in the first 60% of the time available for the given round. The duration will be determined from the number of rovers deployed, which will also reveal the size of the arena. This information is determined during the start of the competition when the rovers communicate with the host device. From the starting position, rovers will pick a random heading, move 50cm away, and then begin a spiraling search for targets.

2) Exploit: Phase two is scheduled for the final 40% of time remaining of the given round. If a cluster has been identified and published in a global array, and the number of cubes remaining is at least three, any rover still searching at this elapsed time marker will move toward the cluster location. Given localization issues, it may arrive at the coordinates and not see the cluster, in which case it will begin a spiral search in the new area. Ideally, the cluster will be located and the remaining cubes returned. The intent is to leave fruitless searches as the deadline draws near and instead assist in returning targets that have already been identified but not yet collected.

F. Final Search Algorithm

After examining various other search methods, we determined that a spiral-shaped search would be best for the purposes of the competition. It seemed ideal because it would cover a large amount of ground while allowing each rover to easily cover its own separate region. We implemented this by telling the rovers to generally set their goal location to a constant distance to the side, while then adding a variable offset to the distance they travel before that turn. This offset increases over time and resets when it reaches a certain point. Attempting to determine the most effective offset was a major point of focus and made up a large fraction of our testing efforts.

We combined our spiral search with a site fidelity system that allows the rover to remember cube-rich areas and return to them. This was implemented by giving each rover a queue, which was initially empty. The queue would be populated with a location when the rover saw more than three tags, indicating multiple cubes near the rover. The rover will then return to areas stored in the queue, popping them off the queue as it begins its search.

Adding time-based triggers and sharing a global queue were not pursued as originally intended.

IV. EXPERIMENTS

A. Gripper Testing

One of our team members with an Engineering focus took responsibility for attaching and wiring the grippers once they were delivered. On one rover the gripper did not deploy properly until pressure was applied at the pin insertion point. All pins were checked for breakage and appeared intact. The problem remains intermittent.

B. Simulation

We decided to test our code in simulation before running physical tests to identify some of the more obvious problems we might encounter. We quickly determined simulation was of limited use for fine-tuning our code and was best employed to test changes in coarse maneuvering.

a) Gripper holds cube in low position: We noted this and considered it a graphical issue, given that the simulations are GPU intensive. It may be worth addressing at a later date for Virtual teams because we did notice in simulation a cube-gripping rover continuing to engage in search behavior.

b) Repeated failure to collect cube: Multiple rovers experienced problematic cube collection. From observations and testing it was suspected to be due to a communcation delay, wherein the rover decided the attempt was a failure before the blockBlock was registered.

C. Physical Testing

Initial tests were conducted indoors to determine if code changes were being implemented as expected in a general manner. We noticed some of the same behaviors previously noted in our simulations. One rover would select a heading away from home, then begin a search, find a cube, collect it, and turn to face the general direction of Center, and return the cube as expected. A fellow rover would move away from the start point and begin a search, but after collecting a cube an inappropriate heading home is selected. Indeed, in several instances the rover would select a heading directly away from Center.

Another undesired behavior occurred numerous times wherein a rover would return a cube to Center, then cross into Center again after reversing out. It would then find itself trapped and simply spin in place until physically relocated.

A frequent problem involved the rover running over cubes while attempting to collect a cube or while returning one. This was addressed by Swarmathon officials in March by the addition of 'cow-catchers' to the front and rear panels of the rovers.

V. Results

Our code was effective in simulated testing with both clustered cubes and power law distribution cubes. In the clustered setting, we found that it often took a long time for our robots to discover the large clusters, but once found, they consistently returned to those clusters repeatedly, resulting in efficient cube retrieval. Power law cube distributions resulted in an even more effective search, with our robots both finding smaller clusters of cubes quickly and remembering the locations of the larger clusters. Our results were least effective when there were no clusters of cubes. Without multiple cubes in a tight cluster, our site fidelity was useless and our spiral search had difficulty locating some of the cubes.

Physical testing closely mirrored simulated results, with some continued issues related to fine localization.

VI. Conclusion

As we had been informed by the previous Swarmathon team, localization was by far the biggest challenge to address. Understandably, we do not have the satellite coverage and GPS precision that robot swarms on Mars will theoretically benefit from. If we were able to pinpoint rover and target location within centimeters or millimeters, problems introduced by the wheel encoders and rover-specific x-y-planes could be avoided.

In the end, we did not implement the rivaling force function or introduce the timing triggers of explore/exploit because we were focused on the localization issue and getting our search algorithm to work properly. We hope to include them in the future when our search algorithms perform as desired.

We have concerns that the code submitted will perform in the expected manner at Kennedy Space Center given the age of our rovers, the unreliability of our simulations, and the inconsistent physical testing results we recorded.

Acknowledgments

The team would like to extend our appreciation to Dr. Sambit Bhattacharya for his patience as he guided a group of novices through this research project. We stand in greater awe now of his many achievements. We also wish to thank Dr. Denny Czjedo for his aid in securing replacement parts for our aging rovers and his unfailingly positive encouragement.

References

[1] NASA, Curiosity Rover
[2] https://github.com/BCLab-UNM/Swarmathon-ROS
[3] Kaehler, Adrian and Bradski, Gary. "Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library." O'Reilly Media, Inc., Sebastopol, CA. September 2015.
[4] AprilTags C++ Library
[5] Hoff, Nicholas Roosevelt, III. "Multi-Robot foraging for swarms of simple robots." Harvard University, ProQuest Dissertations Publishing, 2011.
[6] Mirzaei, Mostafa. "Cooperative multi agent search and coverage in uncertain environments." Ph.d Thesis. Concordia University, April 2015.
[7] Kolushev F.A., Bogdanov A.A. (2000) Multi-agent optimal path planning for mobile robots in environment with obstacles. In: Bjøner D., Broy M., Zamulin A.V. (eds) Perspectives of System Informatics. PSI 1999. Lecture Notes in Computer Science, vol 1755. Springer, Berlin, Heidelberg.
[8] M.M. Polycarpou, Yanli, Yang, and K.M. Passino. "A cooperative search framework for distributed agents." IEEE Intelligent Control, (ISIC '01). pp. 1-6. Sept 2001.