Virtual Competition Technical Report
EPCC Swarmies
El Paso Community College
El Paso, Texas

Team Members:
Adrian Barbee
Pedro Barragan
Sergio Galindo
Jason Ivey
James Jodoin
Alan Melendez
Carolina Rico
Christopher Tarango

Reviewed report by Faculty Advisor:
Erik Valdes

Abstract:
Development of a hybrid collection algorithm for virtual robots tasked with the detection of resources required modification and creation of code in C++ and ROS. Efforts were focused in four main areas.

1. Uniform search behavior, to distribute the search tasks into 'weighted' near and remote regions, where the weight depends on the remoteness and region size.
2. Clustered search behavior, the rover's ability to return to a location containing multiple resources. Cluster memory, the robot's ability to communicate location and resources to other robots.
3. Pick-up controller, the actions required of the robot during acquisition of resources was streamlined to increase speed reduce false-positives.
4. Drop-off controller, custom made algorithm to reduce bottleneck traffic close to the center resources storage and reduce the chance of accidentally ejecting its resources.

I. INTRODUCTION

Robots are constructed to accomplish tasks in conditions that humans could not. Our long term goal was to contribute to the body-of-knowledge of robots teams (swarmies) to help in space exploration of inhabitable environments such as asteroids and planets. Our short term goal was to collect and retain the maximum number of "cube resources" in the least amount of time by utilizing an organized swarm behavior. For this competition the NASA Swarmathon organizers provided El Paso Community College (EPCC) with a version of the robot workspace source code. EPCC developed our own strategies and tests for the virtual robots in the Gazebo simulator. After consulting the available literature [1] we concluded that:

1) Previous simulations showed robots undergoing numerous collisions.
2) The stochastic noise in the sensors prevented precise trajectories for search, collection, and retrieval or communication of site information between robots.

Our goals became to increase the efficiency of all the controllers and develop at least two search methods: one specialized for uniform resource distribution and the other specialized for clustered and semi-clustered distributions.

II. RELATED WORK

Our team chose to start the optimization of the search parameters by creating flowcharts of the controllers we wanted to modify. This included the pickup, drop-off, search controller and mobility. See Figure 1

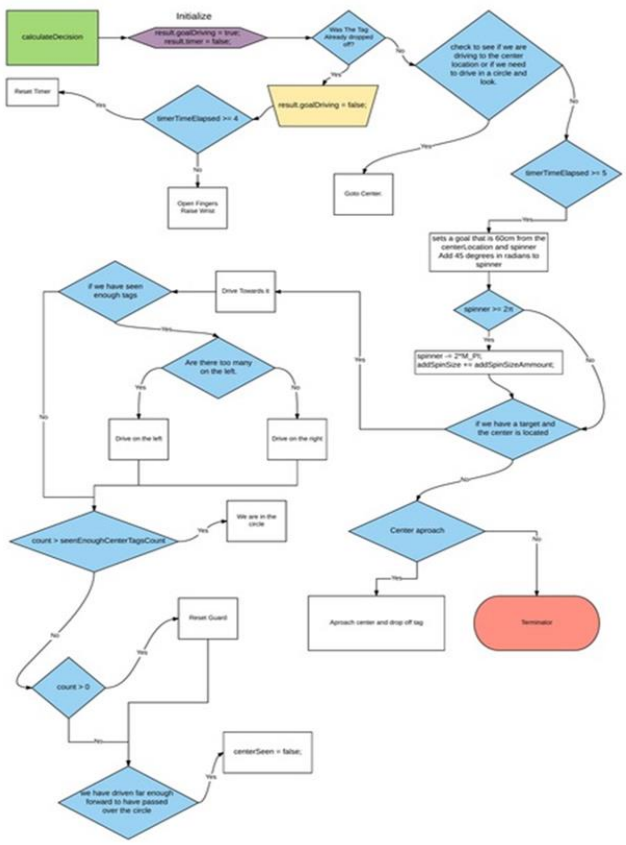

Figure 1 - Drop Off Controller Flowchart Diagram

The flowcharts provided a visual representation of the different sections of the code and helped to isolate those areas where modifications would be most beneficial. After reviewing the 2016 Technical Reports, the Cabrillo Robotics Club, [3], strategy was effective and we used it as a starting point. We tasked those who were weaker at coding with testing modifications to code as well as performing Netlogo/Outreach. The University of Houston Technical Report, [2] informed our teams efforts to have robots not only count the amount of resources found but to publish the locations to other robots for more efficient retrieval.

III. METHODS

A. Uniform Search

Taking into account that there are 256 cubes in each round and the preliminary stage is a square arena of 53.29m^2 the average resource density is:

$$\sigma = \frac{N}{A} \approx 4.8\ resources/m^2$$

Nonetheless, cubes located further away take longer to be found and returned to storage (which consist of a 1m^2 center area demarked by april-tags with a value of zero).

With the robots moving at a maximum speed of 1m/s, it would take approximately 3.75 seconds to reach a hypothetical resource located at a far boundary starting from the storage location. In order to maximize robot usage, we needed to assign smaller areas to robots working further from the storage location. Two proximity zones where created: equatorial and polar.

Equatorial regions where adjacent to the center thus their area was larger, while polar regions were further away and had smaller area. See figure 2.

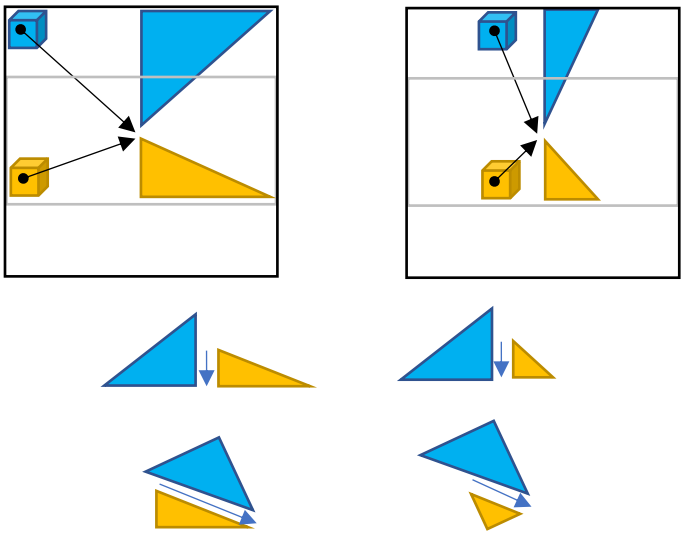

Figure 2. Comparison of time required to transport a resource from different polar and tropical zones. Upper right: location of resource in the far west meridian but in different latitude. Upper left: Location of resource in the center meridian and different latitude. Bottom left and right: The equation to weigh the density was empirically computed taking into account transport trajectories larger than or smaller than the tropic: "R" and "r" respectively.

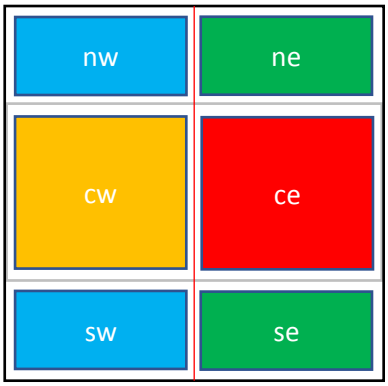

Figure 4- Uniform Search Strategy

With rovers in dedicated areas, would either search randomly or utilize another search strategy (Site Fidelity) for the area. The rovers in the outer regions would bring the resources to either the goal or the inner areas. The inner rovers would then pick up the resources and carry them to the goal area. The outer rovers would decide to drop off the resource in the inner quadrant or proceed to goal area by checking on how busy the inner rovers were and by determining distance to the goal and choosing the most efficient option. Leaving the resource in the inner area would give more time to the outer rovers to search for more resources.

B. Site Fidelity

Site Fidelity is the robot's ability to recognize a grouping of resources, and return to the site where those resources were found. Site fidelity was activated when the robot recognized more than 3 april cube tags. Once it was determined that there were multiple cubes in an area, the position of the robot was added to a collective memory. Once a successful drop off of a cube was completed, the robot would search the memory for known resource positions. If positions were present, the distance was calculated to the position closest to the storage location, and that position became the goal location for the robot. The robot then returned to the last known position of a cluster of cubes. Once arrived the robot would do a spin search of the area to identify the location of other cubes. If the search did not identify other cubes, the position was removed from the memory and either another position was set as the goal location, or the random search pattern was resumed.
Other iterations of the code involved publishing the memory to a topic to make the positions available to all the robots so that their search could be interrupted to retrieve cubes from an identified cluster.

C. Cluster Memory

A difficulty with the base code's search functionality emerged early on - when simulations were run with cluster or power-law cube distribution, the robots would spend too much time re-exploring empty area. This poor performance, combined with a desire to have the robots work together to solve problems, was the motivation for implementing some form of priority-location-memory. The process involved in implementing this new memory functionality is detailed chronologically below:

1. Capture current location of a robot.
2. Store the captured location in a data structure.
3. Communicate captured locations to all robots.
4. Create a condition for when to consider a location high priority.
5. Have robots process all captured locations.

Through testing of this functionality, limitations were discovered:

1. The robot's sensor suite can only reliably detect if there is more than one cube or less than four cubes in front of it.

2. The amount of error involved in driving to a high priority location is proportional to the distance between that location and the storage location.
3. Having multiple robots in the same area increases obstacle avoidance calls dramatically.

Once these limitations were discovered, modification were made to mitigate them. The final logic considered a high priority location to have more than one cube, implemented a circular search pattern upon robot arrival to a high priority location, and limited one robot per location at any given time.

With a memory of clusters, the robots performed significantly better when faced with cluster or power-law distribution types.

D. Drop-off Method

The drop-off method determines the best and fastest route to the storage location, and placing the cube in storage as efficiently as possible. Part of the code modification done included determining a precise storage location. As the rover arrives to storage it tries to determine if it is inside the area by reading the April tags and determining if it is heading into the location or not. It will then orient itself to move inside the storage location. The result is to drop the resource as deep as possible inside the storage location.

E. Pick-up Method

Small modifications were made to the pick-up controller. Speed was increased in the intermediate portion of the pickup. This is when the resource had been identified but not locked on to. Changing the motion direction of the robot, from backing up slowly during the process of closing the gripper to moving forward slowly during pick increased the efficiency as well. Other issues addressed included altering the select target method to show a preference for those targets which were closer to the center of the camera view than those that might be physically closer but off to the periphery of direction of robot motion. Resources in the periphery were more likely to trigger inconsistently, where the rover would approach a cube, then a resource in the periphery would be recognized, and because it physically closer to the location of the rover, the rover would stop its approach and try to focus on the newly recognized resource, but in readjusting would lose the peripheral target, and thus go back to approaching the original cube, ad infinitum.

V. EXPERIMENTS/PROJECT DEVELOPMENT

Experiments were made on these three methods to verify their efficiency and effectiveness. We were able to fully test the site fidelity method (assignment of sectors). All trail runs indicated that the assignment of sectors worked well. See figures 7 and 8

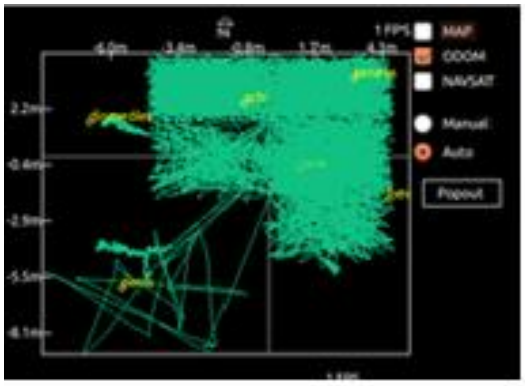

Figure 7 - Swarmies staying within their sector, final round.

After refining the Site Fidelity Method we were able to see better results in making the rovers stay within a particular area, see Figure 8.

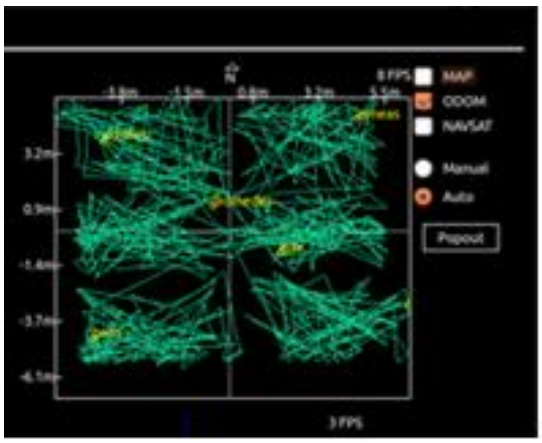

Figure 8 - Swarmies staying within their sector (refined), final round.

IV. RESULTS

We found a significant improvement in the efficiency of each of our individual methods compared with the vanilla version of the source code.

We have made great strides in the all of our methods. Where we shined was in the site fidelity method. We have shown and proved the rovers can be sectored to a specific area and can search within that area only. As we modified the code more we found different issues arising with the rovers. These issues taught us this is not a deterministic but a stochastic problem. There is an "inherent randomness" due to limitations on GPS and IMU limitations. We've have to be able to compensate for these limitations since overtime they can become an issues. Our experience with C++ played a crucial role in the development of our code and needs to be improved to have a better chance next year.

VI. ONGOING WORK

There are plans to keep ongoing work in order to be more ready and successful for next year's Swarmathon Competition. One such talk is to create a Robotics club and to offer a programming class through Zybooks to get students ready and up to par, in their C++ skills. We will be running workshops to familiarize ourselves and future students to Linux, ROS, Slack, GitHub, Gazebo, and other associated applications. We would like to engage more students in the interest of Robotics, Engineering, Science, and Computer Science, and the NASA Swarmathon competition has done a very good job in merging all these fields together, and having them work with each other.

VII. CONCLUSION

Each of our strategies proved to be efficient when tested independently, nonetheless the final integration was not performing as expected due to compatibility issues and time constraints. Nonetheless it is our desire share videos of our final integrations in our dedicated channel [7].

REFERENCES

[1] Warter-Perez, N. (2016) *Cal State LA Swarm Robotics Technical Report 2016*. Retrieved from NASA Swarmathon website: http://nasaswarmathon.com/wp-content/uploads/2016/05/CSULA-Tech-Report.pdf.

[2] Matera, M (2016) *Autonomous Robot Retrieval*. Retrieved from NASA Swarmathon website: http://nasaswarmathon.com/wp-content/uploads/2016/05/Cabrillo-Tech-Report.pdf

[3] Banas, D. (2014, 11 20). *C++ Programming*. Retrieved from YouTube: https://www.youtube.com/watch?v=Rub-JsjMhWY&t=2701s

[4] Burbage, M., Huang, L., Lin, L., & Becker, A. (2016). *Swarmathon Technical Report.* Houston: University of Houston. Retrieved from: http://nasaswarmathon.com/wp-content/uploads/2016/05/UH-Tech-Report.pdf

[5] Club, C. R. (n.d.). *Autonomous Resource Retrieval.* Aptos: Cabrillo College. Retrieved from: http://nasaswarmathon.com/wp-content/uploads/2016/05/Cabrillo-Tech-Report.pdf

[6] Gaddis, T., Walters, J., & Muganda, G. (210). *Starting Out With C++ Early Objects*.

[7] Barbee, Barragan et al. *Swarmies Movies*. Retrieved from: https://www.youtube.com/channel/UCMSerIu1dq5IoV_Z4SCxv8A