EZ-RASSOR 2.0

Gold Team - Group 13

2019 - 2020 Senior Design Project

Gold Team Members

Autumn Esponda

Martin Power

Daniel Silva

Daniel Simoes

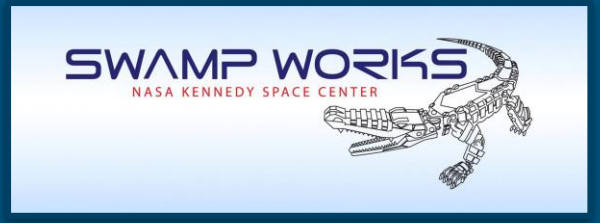
Chin Winn

Sponsor - Mike Conroy and the Florida Space Institute

Background

- The Regolith Advanced Surface Systems Operations Robot (RASSOR) Excavator
- Developed by NASA to mine regolith and perform additional tasks on other planets
- Controllable via teleoperations
- Equipped with basic sensors and stereo cameras for situational awareness

Introduction



- A combined effort between UCF and NASA SwampWorks
- Main purpose is to recreate a modular version of the RASSOR using only open source technologies
- Immediately utilized by Kennedy Space Center as a demo unit
- Scoped as a multi-year project by the Florida Space Institute
- Designed for long-term prototyping and fine-tuning of computational requirements by other universities and organizations

Impacts

- Creating an open sourced EZ-RASSOR system will grant universities and other research institutes access to this software, and essentially open up this problem to the best minds in robotics and AI around the globe

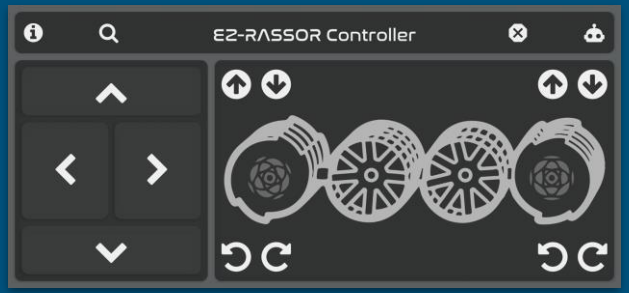
- Advancements made with the EZ-RASSOR software could potentially impact NASA's full scale RASSOR rover, and by extension, our work here could aid in the advancement of robotic space exploration and colonization

EZ-RASSOR 1.0 Functionality

- The functionality developed by the original EZ-RASSOR team is as follows:
  - Roving across slightly treacherous terrain
  - Mining, storing, and dumping regolith with rotating drums
  - Autonomously detecting and avoiding obstacles
  - Includes a simulation which emulates the EZ-RASSOR

The Problem

- While the original system supported basic functionality, it did not allow groups of rovers to collaborate efficiently or intelligently
  - Rovers had to be manually controlled, or will behave autonomously and completely independently
  - Did not account for rover battery constraints
  - Lacked support for scheduling rovers between multiple dig sites
  - Didn’t have efficient or robust path planning

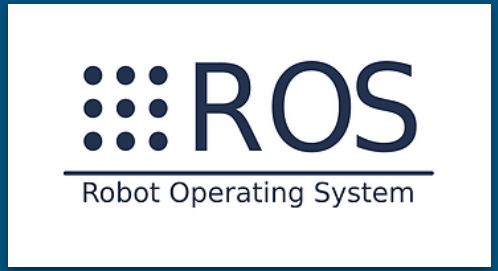


The Solution

The EZ-RASSOR 2.0 solves this problem by adding true **autonomy** and **swarm control** functionality to the EZ-RASSOR platform.

Technologies

- ROS - Robotic Operating System
  - Serves as a middleware system for interfacing with EZ-RASSOR hardware
- Gazebo - 3D robotics simulator
  - Integrates with ROS to emulate the rover's functionality
- Python



Requirements

- Implement a swarm management system into the EZ-RASSOR platform
  - Enable a swarm of rovers to efficiently collaborate in mining regolith on the lunar surface
  - Delivered as a ROS package which functions with the open source EZ-RASSOR system

- Given the coordinates of the charging station(s) and dig site(s), the swarm AI should generate efficient schedules and paths for each rover
  - Support for up to 10 EZ-RASSOR rovers
  - Paths should be returned in the form of waypoints for each rover to navigate towards

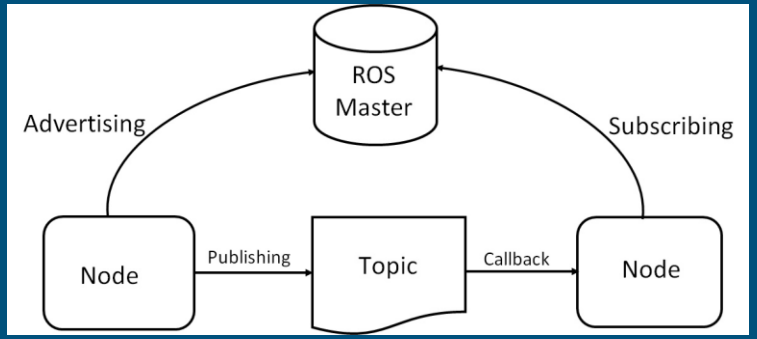
- Update the existing Gazebo simulation to support swarm functionality
- The EZ-RASSOR platform shall remain open source, well documented, and suitable for similar robotic systems

Stretch Goals

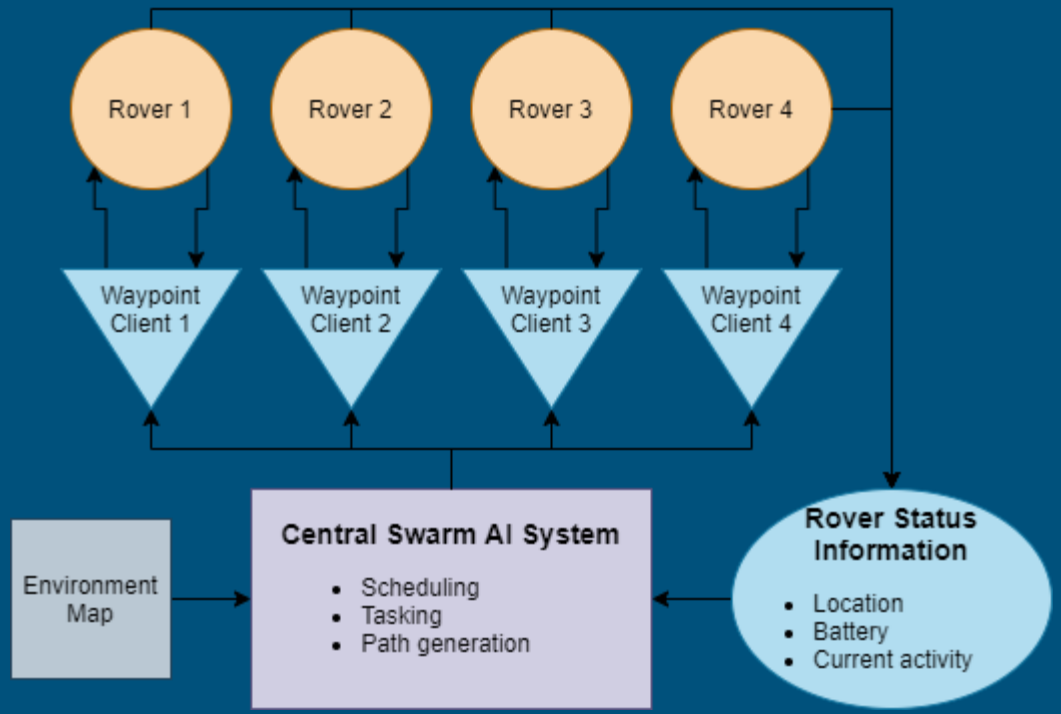
- Enable individual rovers to communicate with the central swarm system to send additional metrics while roving
  - Unknown obstacles, lost/stuck rovers, wheel slip rate, etc. at specific locations
- Update the lunar map as rovers explore and learn about the environment
  - Allows the map to become increasingly more accurate over time
  - Track locations which have been historically more dangerous
- Allow the system to consider the physical effects rovers will have on their environment
  - Deterioration of paths due to overuse, etc.

Introduction to ROS

- Flow of Information
  - Pathfinding & Scheduling algorithms -> ROS -> Rover
- Each system there is a publisher, subscriber, or both, and they all continuously rely on one another while the swarm simulation is running



System Overview



Design Approach - Scheduling

- Why do we need to think about scheduling?
  - Each rover has basic behaviors such as digging, traveling, and charging.
  - Each rover has a finite amount of time to do work before the battery runs out and the rover goes back to the charging station.
  - Rovers also need a certain amount of battery left to make sure they can travel back to the lander to charge up.
  - Charging takes time.
  - There is a finite amount of charging stations available, often much less than the amount of rovers. If the charging stations are full, rovers need to wait in line until it is their turn to use the station.

Design Approach - Scheduling

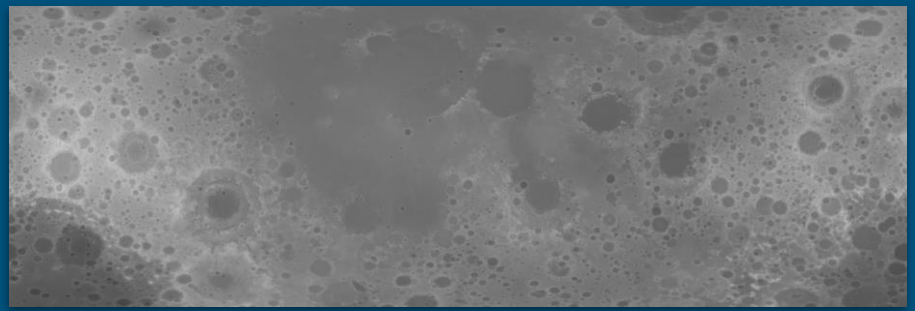
- Simulation and data analytics
  - Before we put everything into motion with ROS and Gazebo, we built a program to simulate the scheduling algorithm for rovers in order to observe how they behave and get some data out of it for analytics
    - What is the distance (away from the lander) that it is no longer worth it for the rovers to dig (for example, the battery and time spent on traveling to and from the lander is more than the battery and time spent on actual digging)
    - How much stuff each rover, which digs at different locations, can dig up after N amount of time has passed
  - We combined the scheduling algorithm with the path planning algorithm to simulate the behaviors of the rovers around different areas of the lunar surface

Implementation - Scheduling

- The scheduler will check on each rover and decide if that rover's activity should be updated, based on the rover's battery level
    - If a rover's battery level is insufficient for the task that it's currently executing, the scheduler will recall it
    - If the rover has been charging and it has reached an adequate battery level, the scheduler will allow it to begin working again

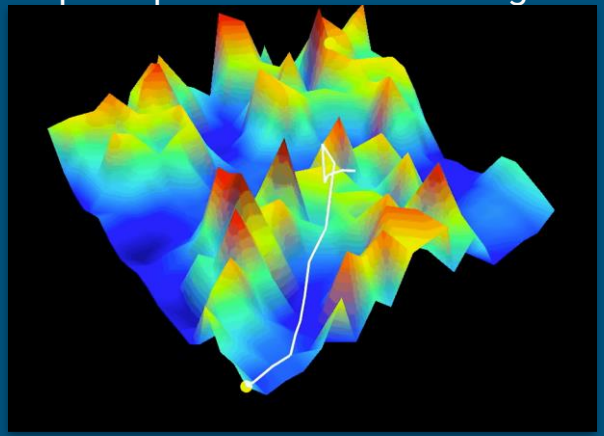
Utilizing NASA’s Lunar Data

- We leveraged NASA's Lunar Surface model to instantiate the system's environment map
  - Data was converted into an elevation map to be utilized by the system's path planner
- NASA’s elevation data available in two formats
  - .JP2 and .IMG
  - Our team converted this to an image where each pixel’s value represents it’s elevation



Design Approach - Path Planning

- Path planning subsystem called by the scheduler when a rover must move between the lander and a dig site
  - Returns a set of waypoints which are published to a rover to autonomously navigate along
  - Allows us to predict the energy required for some action and perhaps alter the scheduling decision in response

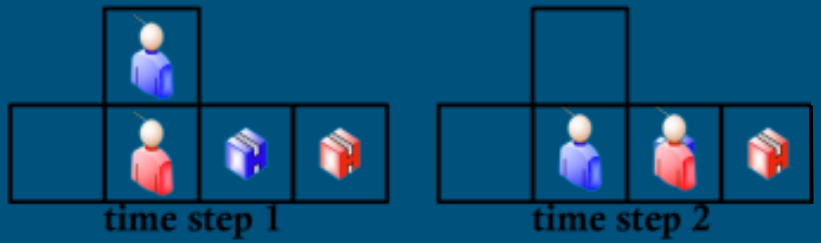


- Relies on 3D A* pathfinding algorithm
  - Essentially an informed or *best* first search
  - Generates the shortest distance path while avoiding obstacles and rough terrain

- But how do we avoid collisions with other rovers?

Design Approach - Path Planning

- The optimal approach is a multi-agent cooperative A* search algorithm
  - These algorithms leverage 3D time-space reservation tables to plan agent movement
  - Introduces a wait step as a possible movement
  - Once a path is found, each step along the path is marked as occupied at the appropriate time step in the reservation table
    - Represents the system's global knowledge of each agent's position over time
  - Unfortunately, planning for each time step is quite costly and rovers would likely be unable follow paths to 100% accuracy



Design Approach - Path Planning

- An alternative is Local Repair A*
  - Naively plans paths, ignoring other agents, and handle collisions on the fly
  - Navigation -> collision detection -> avoidance -> localization -> replan path and continue
  - Much less robust and relies on accurate obstacle detection, avoidance, and localization

- We thought the best solution would fall between the two approaches
- Windowed Hierarchical Cooperative A*
  - Abstracts the map into higher level 'quadrants', rather than precise coordinates
  - Limits the search depth to a dynamic time window, spreading the path planning over the duration of the route and reducing the chance of wasted computation
  - Still requires an active Local Repair system to deal with unforeseen obstacles

Implementation - Path Planning

- Ultimately, Local Repair A* was chosen as the path finding approach to be used by our system

- Reasoning
  - We found that the lunar terrain was quite treacherous, frequently forcing rovers to avoid small obstacles while causing wheel slip and rover capsizing
  - This local replanning approach enables the system adapt to these situations on the fly
  - By ignoring other agents, this approach does not require planning in the time dimension, reducing runtime by a large margin when compared to other techniques
    - Great benefit given the extremely large environment this system functions in

Rover Status Tracking

- Ensures the central swarm AI system is able to monitor metrics such as current position and battery life for each rover
  - System is able to replan paths for rovers that have gone too far off course
  - Helps scheduler decide what each rover should be tasked with
  - Also tracks what task each rover is currently executing

- These data points are critical for the system to make informed decisions based on the status of rovers and their surrounding environment

Updating the Gazebo Simulation

- The Gazebo Simulation has been updated to support the new functionalities that our team implemented and to aid future EZ-RASSOR projects

- We updated the simulation environment to reflect a section of the moon's surface using the lunar elevation data
  - This allows us to test our path planning and scheduling algorithms in a simulation which reflects the data we tested on

Division of Tasks

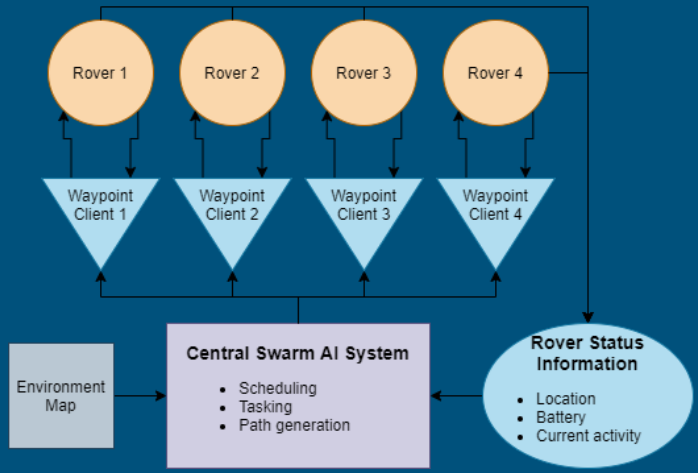
Member	Central Swarm AI System	Pathfinding System	Rover Scheduling	Environment Grid System	Gazebo Update
Danny	X	X	X		
Autumn	X				X
Chin	X	X	X		
Daniel	X			X	X
Marty	X			X	X

Budget

- Software: $0
  - All of the software we use are free and open-source
    - Python
    - ROS
    - Gazebo
- Hardware: $0
  - No need for actual hardware since we are developing for a simulation

Summary

- Path Planner
  - Local Repair A* algorithm used to generate paths for rovers
- Scheduler
  - Scheduler decides what task a rover should execute based on battery levels and position
- Gazebo Simulation
  - GUI for rover data has been created and lunar elevation map imported into simulation



Demo