THE ARLISS PROJECT

What it is and how we did it

The Goal

Working alongside Electrical and Mechanical Engineering capstone teams, our task was to design and build a small rover that fits inside a standard 12 oz. soda can. This rover will be dropped from a rocket at 12,000' AGL, land safely on the ground and drive itself to a predetermined set of GPS coordinates.

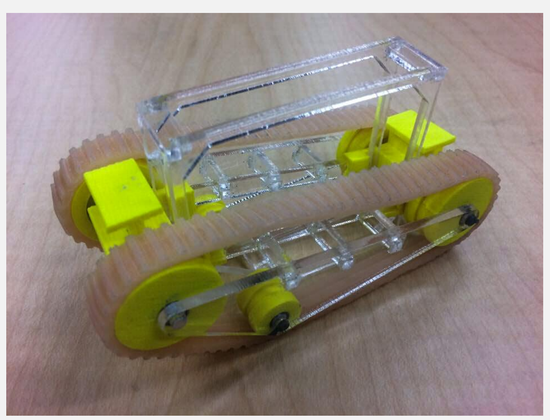
As the Computer Science team, our job was to develop the software package for the rover. This included being able to lock on to and drive towards the GPS coordinates, as well as, an obstacle avoidance system that uses a camera to detect and avoid rough terrain or objects such as rocks that are in the path of our rover.

Implementation

We implemented our software by splitting it up into separate tasks. The tasks we came up with were: parachute deployment, GPS navigation, obstacle avoidance, getting unstuck from obstacles, and hitting the finish pole. In addition, many tasks were implemented utilizing the onboard camera, which we had to create obstacle detection software for.

Parachute Deployment

Once the rover is dropped from the rocket, we had to determine at what point to deploy the parachute. We accomplished this by tracking the GPS coordinates height, and deploying the parachute once we dropped below a certain altitude. This way, the wind won't carry us as far as if we deployed the parachute immediately.



GPS Navigation

For the rover's GPS Navigation functions, we are using an algorithm that determines the shortest path between two given GPS coordinates. The GPS will also keep updating the new best route per request from the obstacle avoidance and unstuck from obstacles modules. This means that the GPS function has to work flawlessly with both of these modules to ensure the rover's safety and efficiency. How the rover behaves during it's driving is also critical, so the GPS function will check if the rover is off-course every few seconds and give route compensation if needed.

Obstacle Avoidance

The obstacle avoidance system ensures that our rover is not impeded on its way to the destination. Taking in filtered images from the obstacle detection software, this system does edge detection on the image to find objects in the rovers path, and then decides how to best get around the object. This is done by treating the filtered black and white image as a matrix of pixels, and summing the number of edges to the left, right or in front of the rover and adjusting the direction of the rover to travel where the fewest edges are found.

Getting Unstuck from Obstacles

In case the obstacle avoidance fails, and we end up hitting an obstacle, we've developed an algorithm to help us get unstuck from what we hit. It works by first attempting to back up the rover, and if the rover doesn't move, back up in different directions until it does move. It detects if the rover has moved by checking the GPS coordinates. This algorithm works best if just the rover's path forward is blocked, but it can still easily move backward.

Find and Touch the Finish Pole

Once the rover get's within the GPS' error range of the finish coordinates, we have to search for the finish pole. This algorithm works by first searching for the finish pole by rotating in place and taking pictures. These pictures are used to detect a traffic cone by our imaging system. Once the cone is detected, the rover is oriented in the direction of the cone, and moves forward, making periodic course corrections along the way.

Computer Vision

Computer vision is used throughout the duration of the rover's expedition, but there are two separate and distinct functions which are being performed. During the stage of the expedition where the rover relies on GPS to direct it towards the target set of coordinates, computer vision will be utilized in the obstacle avoidance system. When the rover is approximately 8 meters away from the target, and the objective of the rover has switched from being directed by GPS to searching for the pole, computer vision will be used to locate the traffic cone at the base of the pole.

To detect obstacles in the path of the rover, each frame is run through a series of filters which are primarily intended to eliminate noise and to detect the edges of the obstacles.

1. Convert Original image to Grayscale
2. Smooth (e.g. blur) Grayscale image
3. Morphological Opening of image
4. Canny Edge Detection

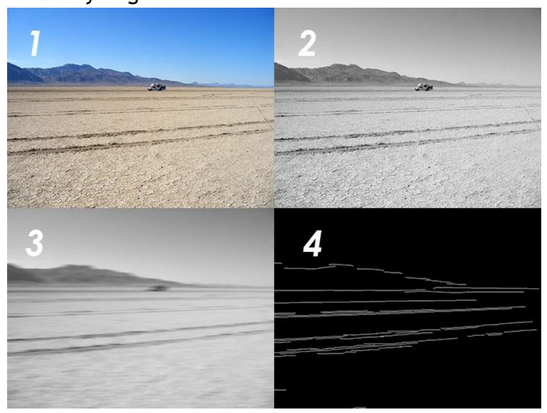


Image 1 shows the original image. Image 2 shows the image after it has been converted to grayscale. Image 3 shows the image after it has been smoothed. Image 4 shows the image after the Canny filter has been applied.

To detect the pole at the end of the rover's expedition, our team has developed a complex system which utilizes machine learning and image processing. For the machine learning, our team trained a Haar Classifier which can be used to detect a traffic cone based on its features and geometry. Though effective at only detecting a traffic cone the majority of the time, machine learning classifiers are prone to returning false positives. The way we have avoided this is by verifying our results by checking the color.

Several points from the result, i.e. pixels, are converted to their HSV (Hue, Saturation & Value) equivalent, and are compared to values which would be typically observed on an orange traffic cone. Without the HSV values being confirmed, the image is rejected. This results in a higher rate of false negatives, but ensures accurate results.

Meet the Team

Team Members

- Zachary DeVita devitaz@oregonstate.edu
- Zhaolong Wu wuzha@oregonstate.edu
- Paul Minner minnerp@oregonstate.edu
- Steven Silvers silverss@oregonstate.edu

Our Client

Dr. Nancy Squires

Senior Instructor of Mechanical Engineering at Oregon State University

squires@engr.orst.edu

Sponsorship

This project was made possible through funding provided by Oregon State University AIAA. To find out more about AIAA, follow the QR code or visit http://groups.engr.oregonstate.edu/aiaa/home

Oregon State UNIVERSITY