Control and Operations for a Long Duration Solar Powered Mars Rover

Zac White[1] / David P. Miller[1] / Matt Roman[1] / Michael Ravine[2] / Daniel Flippo[1] / Brandon Mills[1] / Brian Nixon[2] / Eldar Noe[2] / Michael Malin[2] ®
[1]University of Oklahoma / [2]MSSS Inc.

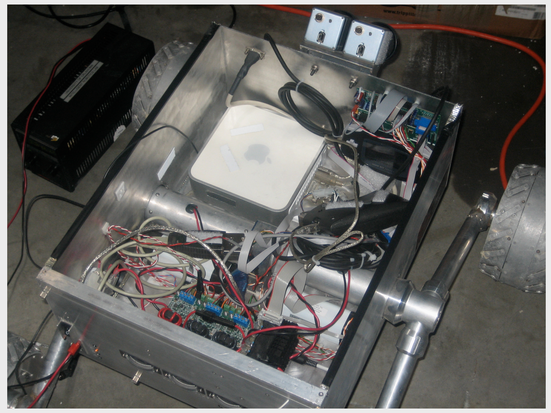
Overview

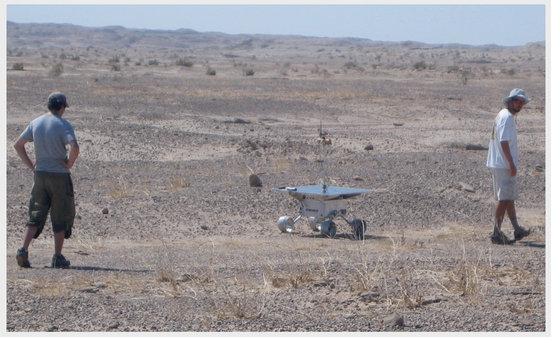
Abstract
SR2, a rover built and designed by the University of Oklahoma and Malin Space Science Systems Inc, completed an autonomous 5km traverse in June 2006. This traverse was made in a Mars-like section of the Anzo Borrego desert in Southern California and it was funded through NASA's Mars Technology Program. The rover's avionics were designed around a Mac mini which runs OS X and the operations software was written in Objective-C with Cocoa.

Objectives
Some of the objectives we were trying to achieve with this field test were:

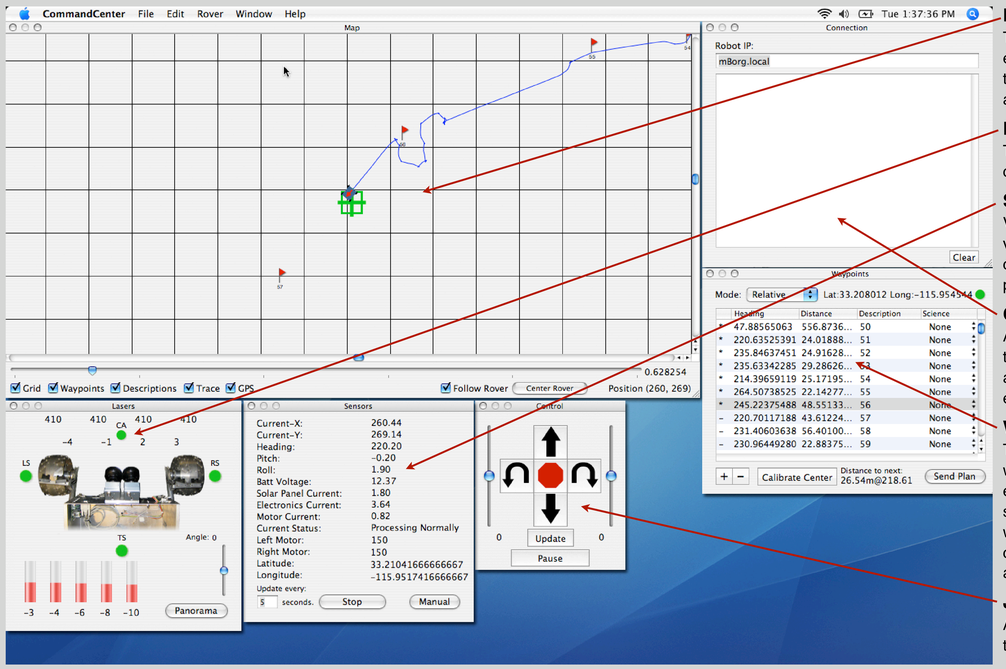
- ➡ Validate reactive navigation as feasible for long distances in rough terrain.
- ➡ Characterize dead-reckoning errors.
- ➡ Validate mobility system.
- ➡ Characterize power usage profile.
- ➡ Test new wheel design.
- ➡ Validate and test operations interface.

OS X's Role
The Cocoa frameworks were instrumental in creating a program that could convey the data returned from the rover in an organized manner. The Apple supplied developer tools allowed us to add features literally in the car on the way to the test site. Things such as OS X's built in speech synthesis allowed for easy debugging of the rover because it would actually tell us what was wrong.





Methods



Map View
This view showed us exactly where the rover thinks it is versus where it actually is.

Laser Data
This shows data from the onboard laser scanners.

Sensor Data
Vital statistics like battery voltage, electronics current, direction, and gps position are shown here.

Connection
Allows for the address of the rover to be entered and also shows a log of errors and events.

Waypoints
This window shows all the waypoints the rover is expected to reach. It shows completed points with a star. New waypoints can be added and sent at any time.

Joystick
Allows for tele-opration of the rover at any time.

The Rover
The SR2 rover is a four-wheel drive rover that uses skid steer to change direction. The motors and drive train run through a hollow tubular suspension. The main chassis is constructed of aluminum honeycomb to reduce weight while maintaining rigidity. At the center of the chassis is a geared differential that increases stability for the sensors and assures that all four wheels maintain ground contact with equal force. The mobility system can deliver 50 watts of power to the wheels allowing the rover to climb over many obstacles larger than a wheel diameter. On level ground only 2 or 3 watts are needed to have the rover cruise along at 15cm/second. The mechanics of the rover are described in detail in (Miller, et.al., 2003) and (Roman, 2003).

The onboard avionics plan the robot's path between waypoints, interpret the real-time sensors and perform real-time control of the actuators. The DC motors are driven by a PWM signal generated on an Xport Botball Controller (XBC) (LeGrand, et.al., 2005). The PID loop is closed using back-EMF sensing for the motor position, in-lieu of optical encoders. The XBC also controls relays and reads the analog sensors for the rover's proprioceptive systems.

The XBC communicates via serial to a PowerPC based Mac Mini running OS X 10.4. The Mini also takes serial input (via USB) from the rover's GPS, compass and laser scanners. The mini also collects images from the pancam imaging system and stereo navigation cameras via USB and firewire respectively. Because the mini is DC powered, can accept a wide range of voltages, uses only as much power as needed for the processing at hand, and can handle high vibration and temperature ranges, it is an ideal rover processor. The only modification made for use on SR2 was to connect an external antenna to the Mini's airport card. The avionics used between 17 and 60 watts depending on sensor and computational loads.

Software
CommandCenter is a Cocoa program made entirely with Objective-C. The program was written using Xcode. CommandCenter was created to mostly observe the rover as it completed it autonomous traverse, but it can also be used to take panoramas or put the rover to sleep.

CommandCenter's role:
- ➡ Show where the rover believes it is.
- ➡ Show where the rover is via GPS.
- ➡ Show waypoints in relation to the rover's position.
- ➡ Add waypoints to a plan.
- ➡ Show sensor values.
- ➡ Remotely tele-operate rover if it needs help.
- ➡ Take panoramic images.
- ➡ Put rover to sleep.

The rover's software is mostly written in LISP. A library for talking to our motor controller was written in C and we use LISP's foreign language interface to call those functions. LISPWorks Objective-C bridge was also utilized in order to communicate with CommandCenter.

The avionics' role:
- ➡ Avoid obstacles using the rover's sensors.
- ➡ Keep track of position and direction using wheel odometry and an electronic compass.
- ➡ Make intelligent decisions about the best path to a waypoint.
- ➡ Monitor solar panel and battery levels and schedule recharge pauses if needed.
- ➡ Respond and relay motor commands if the rover is in tele-operation mode.

Results

Field Test
The results of the field test were very positive. In the desert of Anzo Borrego the rover was able to traverse over 5km along the desired path in 8 days with several days of almost 1km travel per day. On average fewer than three hours per day were spent actually traversing (more was technically possible, but it was just too hot!).

The traverse was completed by letting the rover autonomously get from pre-planned waypoint to waypoint. If it ever felt it was in trouble or not making enough progress, the rover would 'call home'. This would be seen on CommandCenter and it is up to the operator on how to proceed.This type of situation happened very few times and was quickly resolved through a tele-operation of a couple meters to move the rover through a particularly rough spot.

The power system on the rover was another aspect that was being tested in this field test. More work is needed on the power regulation system. The current system only allowed the rover to operate about three hours before needing a recharge sleep.

Google Earth for OS X was instrumental in planning and analyzing the path taken by the rover. The day before the field test, the entire traverse was planned out in Google Earth and then imported by CommandCenter. These points were then converted to relative positions from the start in meters and sent to the rover as a plan. During the entire field test, the rover was constantly recording sensor values and GPS positions into a log file for later analysis.

The path of the rover in Google Earth. Each section is one day's travel.

Converting these GPS points into KML (Google Maps XML format for GPS points) gave us a very accurate track of the rover for all 8 days of travel. This data allowed us to test the accuracy of the dead-reckoning system of the rover. Plotting the track over the original plan, as seen in the image above, showed that the system was very accurate.

Conclusion
These tests indicate that a rover of this size and physical ability could traverse long distances on Mars with minimal outside (ground) instruction or direction. Single communication cycles could result in traverses of a kilometer, rather than the few meters currently being done on Mars. The test also showed that OS X is a viable platform for control and operation of an autonomous rover.

References
Richard LeGrand, Kyle Machulis, David P. Miller, Randy Sargent & Anne Wright. The XBC: a Modern Low-Cost Mobile Robot Controller. Proceedings of IROS 2005. Edmonton, Canada. August 2005.

D. P. Miller, T. Hunt, M. Roman, S. Swindell, L. Tan, and A. Winterholler. Experiments with a long-range planetary rover. In Proceedings of the The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Nara, Japan, May 2003.

Matthew Roman, Design and Analysis of a Four Wheeled Planetary Rover. MS Thesis, University of Oklahoma. 2005.

Acknowledgments
This project has been funded in part by NASA's Mars Technology program and MSSS Inc.