# The Athena SDM Rover: a Testbed for Mars Rover Mobility

Jeffrey Biesiadecki, Mark W. Maimone
Jet Propulsion Laboratory
Pasadena, CA 91109
Phone: +1 (818) 354 - 0592 Fax: +1 (818) 393 - 2346
Email: mark.maimone@jpl.nasa.gov
http://robotics.jpl.nasa.gov/people/mwm/

Jack Morrison
Logicon Federal Data
Pasadena, CA 91107

**Keywords:** Athena SDM, Mars rover, surface navigation and mobility, rover architecture, system design

## Abstract

The Athena Software Development Model (SDM) rover is an engineering testbed for Earth-based testing of Mars rover capabilities on a realistic platform. Originally developed as a prototype for a Mars Sample Return mission, featuring flight-like electronics, it now serves as a testbed for the 2003 Mars Exploration Rover surface navigation and mobility software.

In this paper we describe the software aspects of the overall Athena SDM rover mobility system in three parts: the control electronics, the software architecture and development environment, and surface navigation software. The Athena SDM architecture has been shown to be capable of meeting mission navigation requirements by being able to safely drive 100 meters using allowable resources within three hours.

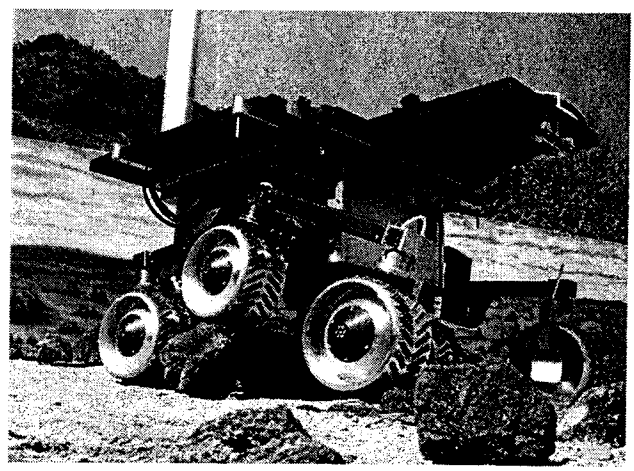

Figure 1: The Athena SDM Rover

## 1 Background

NASA will send a series of missions to Mars during the next decade. Intended to carry the Athena science payload, the Athena rover was designed at JPL to perform a surface exploration and Sample Return mission, but as of August 2000 has been superseded by a new mission plan; two Mars Exploration Rovers (MERs) will explore the Red Planet in early 2004. This vehicle now serves as a research platform on which MER Surface Navigation Software is being refined and tested, and is pictured in Figure 1.

The Athena SDM is approximately 100 x 75 x 45 $cm^3$, with a six-wheel rocker bogie suspension on which four wheels can be steered, weighing approximately 60 kilograms. Its R3000 CPU runs at 12 MHz, a hundred times improvement over the Sojourner rover CPU. [Sto96] FPGAs handle low-level motor control during driving or steering operations. The faster CPU, the additional controllers, a large 32 MByte DRAM memory and 64 MByte Flash memory have allowed the implementation of much more intelligent capabilities for surface navigation than were possible on Sojourner.

The original flight (i.e., spaceflight) Athena rover was designed to provide the capabilities demanded by a Mars Sample Return mission, yet meet the environmental constraints imposed on it by operating in extra-terrestrial environments (including cruise and Martian surface environments). While this paper does not attempt to detail the original flight rover design, many decisions that went into the design of the Athena SDM were motivated by very real flight constraints of the planned Athena Rover.

Several important capabilities were required for the now-defunct Mars Sample Return mission. The rover had to be capable of traversing through terrain similar to that seen by Viking Lander I (VL/I); generally

clear terrain with scattered insurmountable obstacles. [GR97] It had to be able to cover enough ground so that geologically interesting and diverse samples could be found; this was eventually expressed as a requirement that the rover be able to safely traverse up to 100 meters in one day of Mars operations.

The design had to satisfy several environmental constraints. The electronics had to be designed to survive seven months of radiation exposure during cruise to Mars, and to function in the increased radiation environment of the Martian surface. A mission duration of at least 90 days on the surface meant that a renewable power source would be necessary, and the performance of solar cell technology dictated that the rover only be in motion when the Sun was high enough overhead; just over four Earth hours per Martian day. Finally, the flight electronics had to be robust to radiation, able to continue safely after having a bit in memory change due to a radiation hit.

In this paper we describe the overall Athena SDM rover mobility system in three parts: the control electronics, the software architecture and development environment, and surface navigation software. Combined with the mechanical components of the mobility system (described in [LRV99]), the Athena SDM architecture has been shown to be capable of achieving primary navigation mission requirements (while meeting spaceflight constraints): it has enough system resources to drive 100 meter traverses safely in less than four hours.

# 2 Electronics Overview

The electronics in the Athena SDM are flight-like prototypes. While some rearrangement and repackaging was expected in the next round of hardware design, these electronics do incorporate the same CPU, memory architecture, motor controllers, and Inertial Measurement Unit (IMU) that were planned for the flight rover.

The prototype electronics consist of three boards that communicate through the CPU's address/data bus, six Remote Engineering Units (REUs) that have an $I^2C$ interface, and an IMU that also has an $I^2C$ interface. A block diagram showing these boards appears in Figure 2.

## 2.1 CPU Board

Very few boards have been qualified for spaceflight. Of the available choices, two MIPS architecture machines currently lead the high end of processor capabilities: the 20 MHz Rad6000 requiring 14 Watts, and the 12 MHz R3000 requiring 2-3 Watts. Both compare favorably with Sojourner's 100 KIP 8085 (requiring 1 Watt), but the severe power contstraints led to the selection of the R3000 processor.

The Athena SDM CPU board is built around a Synova$^{TM}$ Mongoose-V processor, which is a 32-bit processor with a MIPS R3000 RISC architecture. On-chip it has a floating point unit, 2 kilobyte data cache, and 4 kilobyte instruction cache. The processor is designed for space applications, and has on-chip support for error detection and correction (EDAC) memory which can correct single bit errors and detect double bit errors in memory. Radiation hardened flight-grade units are available; the Athena SDM has an Engineering Model (EM-grade) unit. Since EDAC is a required capability for the final system, having it built into the CPU board reduces the software team's burden.

The CPU board also contains

- 32 megabytes of dynamic RAM
- 4 megabytes of EEPROM
- 64 megabytes of flash memory
- 128 kilobytes of boot PROMs

All of these have EDAC memory associated with them, and the flash memory may be turned off to conserve power.

This board has an $I^2C$ serial interface, and is the sole $I^2C$ master on this bus. It can read and write messages at 400 kbps to the $I^2C$ slave boards, each of which has a unique 7-bit address.

For development purposes, this board has two standard serial ports and a removable ethernet daughtercard that plugs directly onto the CPU board. We use one of the serial ports for a console, and the ethernet card for communication with VxWorks' Tornado tools and command/telemetry tools running on a Sun workstation. Currently the flight software is loaded over the network and executed out of RAM; during a mission the software would execute directly out of EEPROM.

## 2.2 Power Switching Board

The number of rover subsystems that can be powered concurrently is limited by the amount of power generated onboard. So a power switching board was designed to keep overall consumption low. It contains DC-to-DC converters and switches that control power to cameras, remote engineering units, the IMU board, and core power for motors.

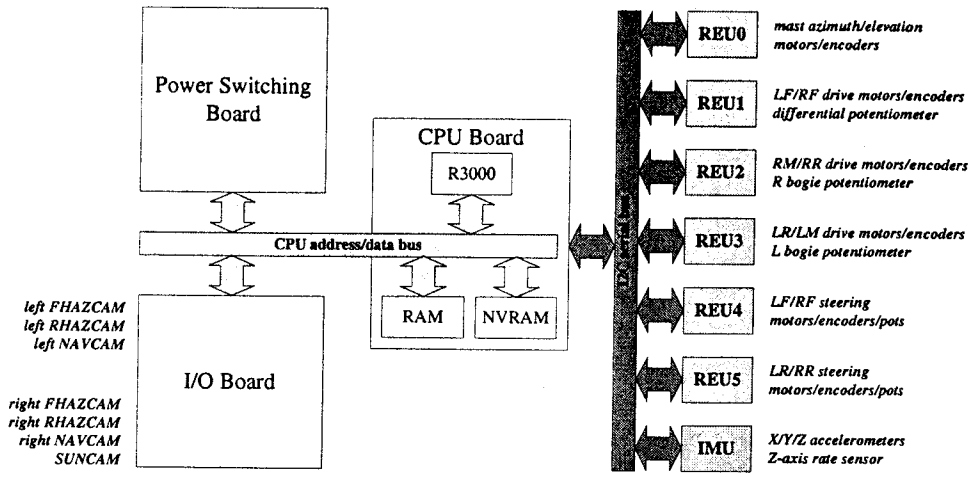


Figure 2: Block diagram of Athena SDM prototype electronics

This board has a current limiter that will clamp the current delivered to the switched peripherals unless explicitly bypassed. If an overcurrent condition persists for more than a few seconds, it will automatically turn all switches off. The CPU will remain powered on, and receives interrupts when limiting becomes active and when shutdown occurs.

Athena SDM power comes either through a tether with a single 15-18V supply, or from a set of batteries in the rover. Approximately 3.5 amps are required to drive on a level surface.

## 2.3 I/O Board

The I/O board also communicates with the CPU board through the processor's bus. It provides an interface to cameras that can acquire images from a pair of cameras simultaneously, and an A/D converter. The A/D converter scans up to 64 channels at either 1 Hz, 10 Hz, or 50 Hz. These channels are used for monitoring voltages and temperatures on the Athena SDM.

## 2.4 Remote Engineering Units

Each Remote Engineering Unit (REU) has:

- an FPGA-based motor controller that can control two brushed motors
- an A/D converter for reading two potentiometers, two temperature sensors, current draw from two motors, and voltages
- interfaces for up to seven contact switches
- switches for heaters and pyro releases

The CPU board communicates to the REUs through the $I^2C$ interface.

The motor controller is a PID controller running at 1 kHz, reading quadrature encoders and outputting the direction and pulse width modulated duty cycle to drive the motor. The controller can be commanded to servo to a particular encoder count, or to maintain a constant rate. It has a profiler that enforces a trapezoidal velocity profile, so that the motor will ramp up smoothly to a maximum rate for the servo, and then will ramp back down smoothly to reach its commanded position. It can detect both motor stalls and deviations from the commanded profile - and optionally shuts down on either of these conditions. The goal position/rate, PID gains, maximum velocity and acceleration limits are all programmable parameters to the controller, sent by software through the $I^2C$ interface. The controllers can also be bypassed, and motors driven at a software-specified duty cycle open-loop.

High resolution encoders are needed to implement fine positioning for sample acquitision. Each channel of an Athena SDM encoder produces 16 pulses per motor revolution, and the controller FPGA counts the edges of both channels, resulting in 64 encoder counts per motor revolution. The gear ratios for steering and drive actuators are 1620:1.

The Athena SDM currently has six REUs that control four steering actuators, six wheel motors, and two mast motors for NAVCAM pan/tilt. There are potentiometers on the steering actuators, on each bogie, and on the differential.

## 2.5 Inertial Measurement Unit

The inertial measurement unit consists of QA-3000 linear accelerometers in three axes, a QRS11 rotational rate sensor about the rover Z-axis, and an A/D converter for digitizing accelerometer and rate sensor outputs, voltages, and temperatures. The CPU board interfaces to the IMU through the $I^2C$ bus. When the rover is known to be motionless, a ten second calibration can be initiated by software to calculate a fixed bias offset. The IMU compensates for this bias when it integrates rate sensor readings while the rover is in motion.

## 2.6 Cameras

The cameras on the Athena SDM are based on a CMOS imager developed at JPL known as the Digital Integrated Camera Experiment 512A Active Pixel Sensor, or DICE APS.

Each chip contains a 512x512 pixel array, an A/D converter for each column of the array (for faster readout rates) yielding 8 or 10 bits per pixel, digital logic which implements a serial command interface, either serial or parallel output, and features such as windowing and pixel subsampling. Exposure time is specified through the command interface; the camera electronics do not do any automatic exposure control. The Athena SDM uses serial output at 10 megabits per second. Each camera draws about 0.4W of power.

# 3 Software Architecture

## 3.1 Design Goals

The design of the Athena software has several drivers, based on the nature of the mission and past experience:

- reliability - robust, valid behavior
- flexibility - adaptable to unexpected and changing conditions
- simplicity - avoiding unnecessary complications to minimize development time and provide predictable operation
- ease of operations - a command interface supporting the needs and convenience of engineers and scientists
- visibility - monitoring and reporting internal and environmental conditions to speed development and provide a clear picture during remote operations
- maximum results per uplink command cycle - autonomy and programmability to limit the number of operator interventions needed to complete a task.

These same characteristics are helpful in a prototype testing environment as well.

## 3.2 Development Environment

In developing software for a flight mission, it is imperative to provide a flexible structure that allows multiple developers to modify shared code, track changes, and be informed quickly of potential errors.

The software development environment is built around a UNIX (Solaris) host platform, and the Wind River VxWorks real-time operating system for the onboard target computer. Onboard software is written in C++, but primarily designed in a functional organization rather than object-oriented. Features of C++ that tend to add complexity, memory, and execution time overhead (such as templates, exceptions, virtual functions, and dynamic object construction and destruction) are avoided. Standard UNIX and open-source tools are used heavily to assist the development process, including "CVS" (Concurrent Version System) version control, "Make" program rebuilding, and the Perl, Tcl, and shell scripting languages.

For reliability's sake, automatic recompiles are performed daily, with all compiler warnings enabled; any warnings detected will abort the build and require correction. Unresolved references (which, in C++, can be due to mismatched declarations) are caught at build time by an extra absolute link step, in addition to the relocatable linking normally used by the VxWorks dynamic loader.

Software testing on the target system is facilitated by a non-flight ethernet interface for downloading code and controlling operation. The communications software that would use a radio in flight can transfer uplink commands and downlink telemetry across this interface, transparently to the rest of the rover. Simplified stand-in versions of ground-based control systems were built to generate commands and display telemetry, allowing end-to-end rover testing using flight commands without dependence on flight-level ground support software.

To support checkout and diagnostics of the prototype electronics, a standalone test package, separate from the flight software, was developed, and is updated incrementally. This program provides a more convenient environment for testing, with direct user control over the rover's hardware components. It can

**System mode control** handles system startup, and shutdown.

**Command sequencing** handles scheduling and execution of uplinked commands.

**Communications** handles uplink and downlink

**Navigation** performs operator commands for driving, with or without stereo image-based hazard avoidance. The basic driving cycle includes calculation of steering angles and drive distance for each of the six wheels to achieve a vehicle path along an arbitrary arc, smooth synchronized wheel motion using a trapezoidal velocity profile, and dead reckoning based on wheel encoders and the IMU's gyroscopic turn rate sensor.

**Science instrument control** performs operator commands for science data collection and data reduction

**Imaging** operates the many cameras on the rover, including image capture, wavelet compression, stereo range mapping, and feature extraction

**Health/status monitoring** performs periodic sensor input, filtering, and alarm monitoring.

**Device control** provides an interface layer to the hardware systems.

**Motion control** operates the motors, providing position and velocity servoing services to the other components.

**Error handling/recovery** provides a uniform error reporting mechanism.

**Support library** provides assorted basic services, including time, math, and specialized memory allocation.

**Operating system** includes the VxWorks executive and system-specific configuration

Table 1: Software Functional Components

be also compiled under Solaris, to allow benchtop testing of REUs and the IMU through a serial port on the host using a commercial I2C serial adaptor.

An extensive internal web site was built so that design documentation, notes, and development status are readily available to the development and test teams, kept up-to-date, and easily searchable. Scripts reduce the effort needed to maintain the documents, by auto-generating flight source code, ground-based source code, and developer documentation from a single description file. Inter-dependencies among these documents are thus automatically kept in sync.

Primary documentation maintained in this way includes the:

**Software Design Document,** which reflects the structure of the implementation by extracting documentation and inter-dependencies directly from the source code

**Command dictionary,** which covers the format and meaning of uplink commands and configuration data supported by the rover

**Telemetry dictionary,** which explains the format and interpretation of downlink data produced by the rover

**Rover Programming Manual,**
which describes the onboard software infrastructure and services. This is written by and for the rover development team.

### 3.3 Software Structure

The Athena SDM flight software executes one command sequence at a time. The ground operations team sends lists of conditional sequences to the rover, which acts on each command in turn. Power, device, and operations constraints override the usefulness of performing multiple operator commands simultaneously.

The onboard software consists of six scheduler tasks, shown in Figure 3, the key one being a single-threaded uplink command sequencer. The remaining tasks perform periodic support duties, including monitoring of sensors and controlling motors. Telemetry data that result from commands and periodic monitoring are buffered for prioritized transmission.

The major data structures in the rover's software architecture are:

- Uplink command sequence data
- Downlink telemetry buffers
- Configuration tables
- Persistent global system state and parameters
- Volatile global system state

The code is partitioned into the functional components given in Table 1.

## 4 Surface Navigation

The Athena SDM relies on three pairs of cameras to provide input for onboard stereo vision range computations. One stereo camera pair is mounted on a mast approximately 140 cm above the ground, the others are rigidly mounted to the body at 48 cm

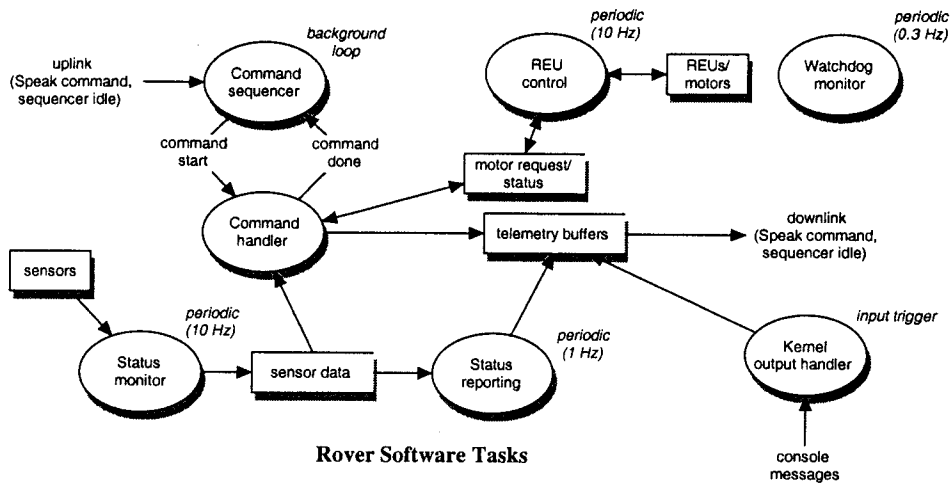


Figure 3: Software Tasks: Each oval represents a distinct VxWorks task.

above ground. JPL Stereo Vision software [XM97] provides the raw range data used to autonomously understand the local terrain elevations, and a variation of Carnegie Mellon's Morphin planner [SHCW96] processes this range data and evaluates a variety of possible paths, choosing the safest route that lets the rover make progress toward its goal.

The Athena surface navigation software is based on the GESTALT package for onboard hazard detection and avoidance. GESTALT (Grid-based Estimation of Surface Traversability Applied to Local Terrain) is a C++ software library developed at JPL that processes environmental data collected by a robot's sensors (e.g., stereo cameras, laser scanner) to determine how to drive safely.

The world is represented as a grid of rectangular cells; imagine melting checkerboard-style kitchen tiles over the surface of Mars, as in the upper right of Figure 4. Nothing need be known about its world in advance, the software only requires that the rover provide new 3D sensory data as it moves throughout the environment (data are expected to be somewhat noisy). These data are collected into grid cells and are evaluated as separate rover-sized planar patches. In this evaluation potential traversability hazards such as step hazards, slope hazards, and rough terrain are found and marked.

Finally, straight and curved paths from the rover's current position are evaluated. Other modules (e.g., a visual servoing module) are expected to evaluate the same fixed set of paths, so that their votes can be easily merged with those generated by the hazard avoidance module. The safest path that helps the rover move toward its goal will be selected. Once a safe steering direction has been chosen, the rover takes a small (35cm) step along the arc in that direction. The small step size is needed both to bound the error in its position knowledge, and to ensure that sufficiently interesting paths can be taken (else the rover would only move in broad arc sweeps).

This software is a fresh implementation and extension of the Morphin algorithm developed by Carnegie Mellon University's Robotics Institute. Although lacking the continuous driving and multithreaded features of CMU's implementation [WBMT99], it improves upon the original algorithm in several ways. Traversability can be evaluated in a direction-specific manner; this software recognizes that a hill which is unclimbable head-on might be climbable on a sideways approach. Also, navigation parameters can be reset at any time, more detailed debugging outputs are available, and multiple rover configurations can be evaluated in a single map. For instance, one map evaluation might consider the wheel-based rover footprint with small obstacles, while a second evaluation would assume a wider rover (with solar panel footprint) yet allow much taller obstacles.

Figure 4 illustrates one of GESTALT's diagnostic images. It has the following features:

- The largest area is an overhead view of the world model, 10 m × 10 m in 20 cm increments. (upper left)
  - Red cells (those outside the circular path) indicate that no information is available. Greyscale cells indicate the goodness value of placing the rover there; white means

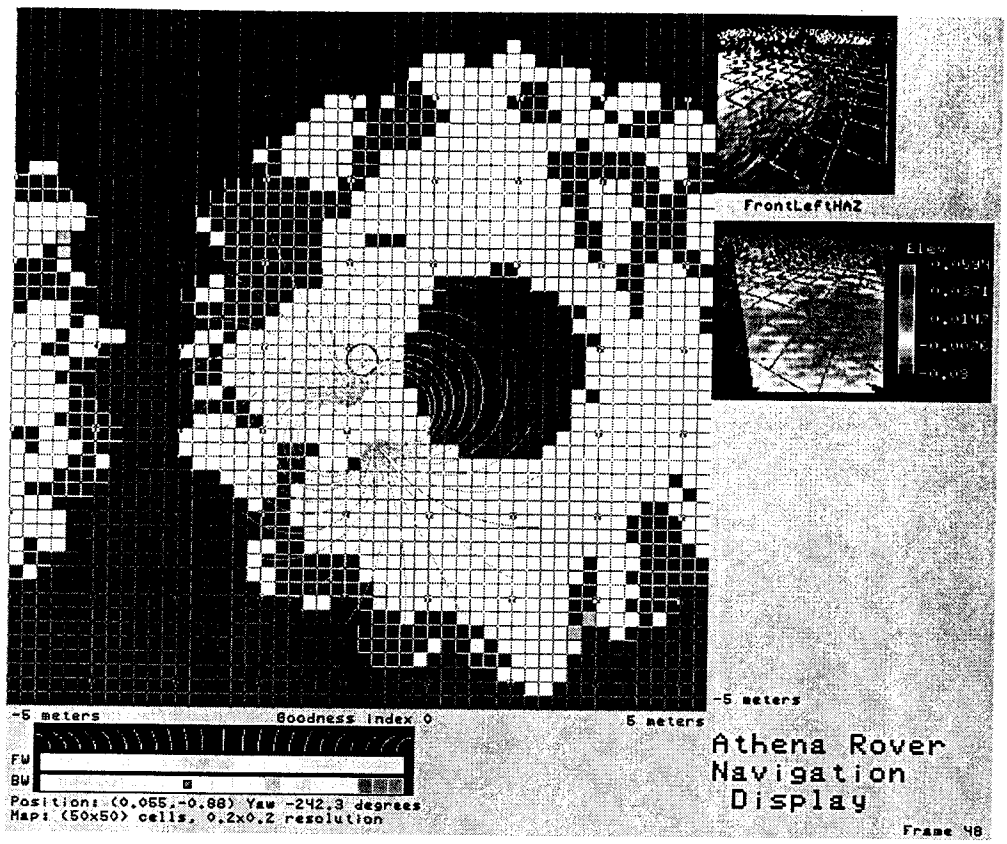


Figure 4: GESTALT Diagnostic Image. Here the rover has been commanded to drive in a circle, and is nearing the end of its traverse.

clear, black means an obstacle, grey indicates a slight roughness to the area but not enough to force it to be avoided.

  - Range data stays fixed in the Grid as the rover moves; our world model wraps around, so when the rover moves off to the right it reappears on the left.

- Left camera rectified image (upper right)
  - Range data availability is color-coded; greyscale pixels have valid range data, tinted pixels have no range data available.
  - Grid cells are marked off not only in the overhead view, but also in this image view. Cell corners are projected onto the virtual ground plane, no correction based on the local topography is performed.
- Subpixel Elevation Image (far right, middle)
  - Elevation of pixels in the input range image (assuming (0,0,0) is on the ground plane) is plotted and color-coded.
  - Grid cells are plotted in the virtual ground plane.
- Arc votes display (lower left)
  - Forward and backward arc evaluations are intensity-coded.
  - The single arc ultimately selected for driving is indicated by a box in the lower left display, and by a different color in the overhead world view display.

# 5 Timing Results

The Morphin algorithm has been demonstrated successfully on several earlier robot systems, e.g. [KHH+99, WBMT99, MXH+00, SSS+00]. Experiments are in progress to validate each component of the present implementation of the navigation system: cell-based traversability analysis, path-based trajectory evaluation, and overall success at navigation toward a goal through several varieties of terrain. Un-

| | |
|---|---|
| Image Acquisition and Downsample | 10 sec |
| Stereo (128x128 images) | 6 sec |
| Nav (unoptimized) | 11 sec |
| Actual Motion | 10 sec |
| TOTAL | 37 sec |

Table 2: Nav algorithm timings on 12MHz R3000 CPU

fortunately those characterizations were not available at time of publication.

Although we have yet to demonstrate a complete 100 meter traverse, qualitative results of shorter traverse tests and timing results of the currently implemented navigation system convince us that this architecture can meet the mission requirement of safely traversing 100 m per day. Mobile operations on Mars will nominally be allowed from 10:00am to 2:00pm Mars Local Time (just over 4 Earth hours). The time required for each step of navigation processing on the 12MHz R3000 CPU is 37 seconds, reported in Table 2 from actual test runs. Since the nominal rover motion is 35 cm along an arc at each step, this results in an average speed of about 1 cm/sec, or 36 m/hr. Hence the rover is capable of driving over 100 m in 3 hours, demonstrating that 100 m/day is realizable.

# 6 Acknowledgements

We are grateful to Athena SDM development team, especially Marcel Schoppers, Todd Litwin, John Waters, Richard Welch, Avionics lead Henry Stone, and the 2003 MER project personnel for their efforts in support of this work.

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract to the National Aeronautics and Space Administration.

# References

[GR97] M. Golombek and D. Rapp. Size-frequency distributions of rocks on mars and earth analog sites: Implications for future landed missions. *Journal of Geophysical Research - Planets*, 102(E2):4117–4129, February 1997.

[KHH+99] Eric Krotkov, Martial Hebert, Lars Henriksen, Paul Levin, Mark Maimone, Reid Simmons, and James Teza and. Evolution of a prototype lunar rover: Addition of laser-based hazard detection, and results from field trials in lunar analogue terrain. *Autonomous Robots*, 7(2), September 1999.

[LRV99] Randel Lindemann, Lisa Reid, and Chris Voorhees. Mobility sub-system for the exploration technology rover. In *33rd Aerospace Mechanisms Symposium*, May 1999.

[MXH+00] L. Matthies, Y. Xiong, R. Hogg, D. Zhu, A. Rankin, B. Kennedy, M. Hebert, R. Maclachlan, C. Won, T. Frost, G. Sukhatme, M. McHenry, and S. Goldberg. A portable, autonomous urban reconnaissance robot. In *Intelligent Autonomous Systems*, Venice, Italy, July 2000. `http://robotics.jpl.nasa.gov/tasks/tmr/papers/UrbanRobotPaper0700.pdf`.

[SHCW96] Reid Simmons, Lars Henriksen, Lonnie Chrisman, and Greg Whelan. Obstacle avoidance and safeguarding for a lunar rover. In *AIAA Forum on Advanced Developments in Space robotics*, Madison, WI, August 1996.

[SSS+00] Sanjiv Singh, Kurt Schwehr, Reid Simmons, Trey Smith, Anthony Stentz, Vandi Verma, and Alex Yahja. Recent progress in local and global traversability for planetary rovers. In *International Conference on Robotics and Automation*, 2000.

[Sto96] Henry W. Stone. Mars pathfinder microrover a low-cost, low-power spacecraft. In *AIAA Forum on Advanced Developments in Space Robotics*, August 1996.

[WBMT99] David Wettergreen, Deepak Bapna, Mark Maimone, and Geb Thomas. Developing nomad for robotic exploration of the atacama desert. *Robotics and Autonomous Systems*, 26(2–3):127–148, February 1999.

[XM97] Yalin Xiong and Larry Matthies. Error analysis of a real-time stereo system. In *Computer Vision and Pattern Recognition*, pages 1087–1093, 1997. http://www.cs.cmu.edu/~yx/ papers/StereoError97.pdf.