CL# 06-0060

# The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition

Jeffrey J. Biesiadecki and Mark W. Maimone
Jet Propulsion Laboratory
Pasadena, CA USA
jeffrey.j.biesiadecki@jpl.nasa.gov
mark.maimone@jpl.nasa.gov

*Abstract*—

NASA's Mars Exploration Rovers' (MER) onboard Mobility Flight Software was designed to provide robust and flexible operation. The MER vehicles can be commanded directly, or given autonomous control over multiple aspects of mobility: which motions to drive, measurement of actual motion, terrain interpretation, even the selection of targets of interest (although this mode remains largely underused).

Vehicle motion can be commanded using multiple layers of control: Motor Control, Direct Drive operations (Arc, Turn in Place), and Goal-based Driving (Goto Waypoint). Multiple layers of safety checks ensure vehicle performance: Command limits (command timeout, time of day limit, software enable, activity constraints), Reactive checks (e.g., motor current limit, vehicle tilt limit), and Predictive checks (e.g., Step, Tilt, Roughness hazards).

From January 2004 through October 2005, Spirit accumulated over 5000 meters and Opportunity 6000 meters of odometry, often covering more than 100 meters in a single day. In this paper we describe the software that has driven these rovers more than a combined 11,000 meters over the Martian surface, including its design and implementation, and summarize current mobility performance results from Mars.

*Keywords*—MER, robotics, Mars rover, flight software, robot mobility, autonomous navigation, fault protection, visual odometry, egomotion

## TABLE OF CONTENTS

| | | |
|---|---|---|
| 1 | INTRODUCTION | 1 |
| 2 | MER FLIGHT SOFTWARE ARCHITECTURE | 2 |
| 3 | MOBILITY MANAGER SOFTWARE | 3 |
| 4 | LOW-LEVEL DRIVING | 4 |
| 5 | AUTONOMOUS DRIVING | 5 |
| | PRIMARY AUTONOMOUS CAPABILITIES | 5 |
| | TERRAIN ASSESSMENT OVERVIEW | 7 |
| | ROBUST STEREO IMAGE PROCESSING | 8 |
| | TERRAIN ASSESSMENT | 8 |
| | AUTONOMOUS GOAL SELECTION | 10 |
| 6 | DEPLOYMENT RESULTS | 10 |
| | SOFTWARE VERSIONS | 11 |
| 7 | CONCLUSION | 13 |
| 8 | ACKNOWLEDGEMENTS | 13 |

[1] 0-7803-9546-8/06/$20.00©2006 IEEE

## 1. INTRODUCTION

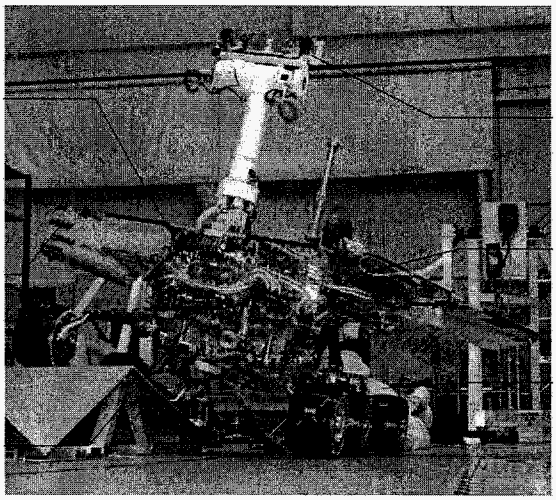


**Figure 1.** Mars Exploration Rover.

NASA successfully landed two mobile robot geologists on the surface of Mars in January 2004: the Spirit and Opportunity Mars Exploration Rovers (MER). Their primary goal is to find evidence of past water at Gusev Crater and Meridiani Planum, two geologically distinct sites on opposite sides of the planet. Although the achievement of their successful landings stands out as a technological tour de force, it is their ability to traverse while on the surface of Mars that has enabled both rovers to succeed in their primary goals.

The MER rovers are typically commanded once per Martian day. A sequence of commands sent in the morning specifies the day's activities: what images and data to collect, how to position the robotic arms, and where to drive. Then at the end of each day, the rovers send back the images and data human operators will use to plan the next day's activities. The next day's mobility commands are selected based on what is known – and what

is unknown – about the terrain ahead.

The rovers are driven using three primary modes: low-level commands that specify exactly how much to turn each wheel and position steering actuators, directed driving primitives for driving along circular arcs (of which straight line driving and turn-in-place are special cases), and autonomous path selection. Low-level commands enable "non-standard" activities such as using the wheels to dig holes in Martian soil, scuff rocks, and perform mechanism health diagnostic tests. Directed drives allow human operators to specify exactly which driving primitives the rover will perform. Autonomous path selection mode allows the rover to take its current state into account when selecting which driving primitives to execute to more effectively reach a Cartesian goal location supplied by human operators.

The mobility system has six 25 centimeter diameter wheels, of which the four corner wheels may be steered – a mechanical configuration derived from the Mars Pathfinder rover Sojourner [17]. The rover body has 30 centimeter ground clearance, and large solar panels on the top of the rover require additional clearance to tall rocks (60 centimeters from ground to solar panel). Wheel baseline is roughly 1 meter side-to-side and 1.25 meters front-to-back. The MER rovers can turn in place about a point between the two middle wheels, drive straight forward or backward, and have at best a one meter turn radius for driving along circular arcs. Straight line driving speed is set to 3.75 centimeters/second, and the rovers turn in place at roughly 2.1 degrees/second. The rovers are statically stable at a tilt of 45 degrees, however driving on more than 30 degree slopes is not recommended due to the possibility of uncontrolled sliding. Rocks taller than a wheel are considered mobility hazards.

Both directed and path selection modes of driving can make use of on-board Stereo Vision processing and Terrain Analysis software to determine whether the rover would encounter any geometric hazards as it drives along its chosen path. In directed driving, the rover can preemptively "veto" a specific mobility command from the ground if it appears too risky. In Autonomous Navigation (Autonav) and other path selection modes, the rover can select its own driving primitives to steer around obstacles and make progress toward its goal. This software provides the unique capability of enabling the vehicle to drive safely even through areas never before seen on Earth: more than 2500 meters of the rovers' combined distance was driven autonomously [12].

The rovers maintain an estimate of their local position and orientation updated at 8 Hz while driving. Position is first estimated based on how much the wheels have turned (wheel odometry). Orientation is estimated using an Inertial Measurement Unit that has 3-axis accelerometers and 3-axis angular rate sensors. In between driving primitives, the rover can make use of camera-based Visual Odometry (autonomous tracking of image features and subsequent rover motion estimation) to correct the errors in the initial wheel odometry-based estimate that occur when the wheels lose traction on large rocks and steep slopes. Visual Odometry software, not originally part of the baseline design but incorporated as "extra credit", has generated over 2500 combined successful position updates on both rovers [6].

## 2. MER Flight Software Architecture

The overall architecture of the MER flight software was based on that used for the Mars Pathfinder lander (MPF), and the command and telemetry infrastructure in particular had significant design and implementation inheritance.

Both MPF lander and MER rovers use the commercial operating system VxWorks from Wind River. This is a pre-emptive multi-tasking OS used for many real-time systems and spacecraft. MER employs the same RAD6K flight computer as used by the MPF lander (although the MER clock runs at 20 MHz, four times faster than MPF), and has a VME interface to several boards that control motors, radios, cameras, and science instruments. Some of the VME boards have serial interfaces to other devices, interfaced to the flight computer by way of FPGAs that connected both the VME bus and the serial busses. There are 128 Mbytes of DRAM and 256 Mbytes of flash memory, and 3 Mbytes of EEPROM. The Mobility task (described below) is given 80 Kbytes of stack, and nominally 13 Megabytes of dedicated heap space (of which at most 7 Mbytes has been used, as of October 2005). The flight software itself requires 9 Mbytes, of which 1.4 Mbytes is used by the Surface Navigation module.

Like MPF, the MER software is divided into high-level objects. Each electronics device is generally represented by an object, while additional software-only objects provide infrastructure services to the rest of the modules (for example, recording telemetry measurements or storing state in nonvolatile memory).

Each software object has its own thread of control, or task. Each object has its own set of state variables, and does not share memory with other objects. Instead of sharing memory, objects communicate with each other via self-contained messages (that do not include pointers to task memory). Objects implement finite state machines, and the messages from other tasks may cause transitions in an object's state machine.

This paradigm greatly simplifies rapid development of robust software (developers do not need to protect memory accesses with semaphores, because only one task ac-

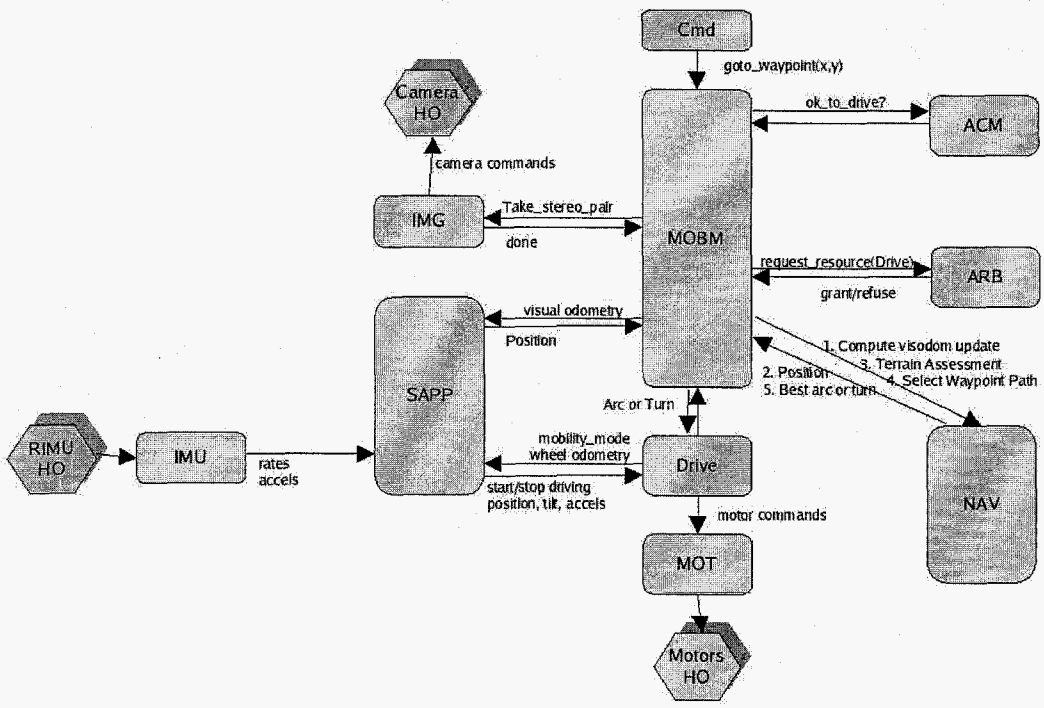



**Figure 2.** Schematic drawing of Mobility and other related software modules, and examples of the types of messages or information that pass between them. **IMG** does Image Acquisition, **Cmd** processes ground commands, **ACM** is the Activity Constraint Manager, **ARB** arbitrates use of resources, **MOBM** is the Mobility Manager, **NAV** does the high level navigation processing, **Drive** executes primitive drive commands, **MOT** controls all motors, **SAPP** manages onboard pose estimation, **IMU** processes IMU data. Hexagons represent Hardware Objects.

cesses any given variable), and ensures real-time deadlines are met (because tasks do not block on semaphores or use task locks, high-priority tasks for control loops and device drivers will pre-empt lower priority tasks that do not have real-time requirements such as image compression).

That said, there are places in the flight software where the above pattern was not followed - images, for instance, contain too much data to be passed as messages, so imaging flight software employs a buffer reservation and release scheme, and pointers to reserved buffers are passed between applications (such as mobility) and the imaging flight software. But there are few such exceptions.

One important area of the flight software to understand is the *uplink* flight software which parses commands sent to the spacecraft and dispatches each command to the part of the software that implements it. Commands can either be executed immediately upon receipt (Real-Time commands) or stored as command sequences to be executed serially. Almost all commands sent to MER rovers are embedded in these event-driven sequences: a command in a sequence is dispatched only after the preceding command completes. The functions called by the command system that implement commands are referred to as *command handlers* - these send messages to the appropriate software object. Eventually, each object receiving a command message must respond back to the command system when the command completes, successfully or not, so the next command in the sequence will execute.

An overview of the MER flight software in general can be found in [19].

## 3. Mobility Manager Software

The Mobility Manager software object (MOBM - the "B" is silent) implements all command handler functions for mobility commands. This includes both low-level drive commands to perform a specific driving primitive such as turn-in-place, as well as high-level driving commands to drive to a specified Cartesian location in small steps avoiding hazards along the way.

The autonomous navigation software, GESTALT, is a generic library employed on several JPL rovers. It does not depend on any particular hardware or software interfaces for cameras and motors. Rather, given a stereo pair of images, it updates an internal map of any nearby hazards seen in the image pair. A different function is provided a Cartesian goal location, and referring to the

| Fault Type | Description |
|---|---|
| *Higher level errors - vehicle not necessarily in a dangerous state* | |
| ARB | resource rescinded, may be due to comm pass |
| ACM | ACM said not-OK-to-drive |
| POT | CAL_STEERING using a bad potentiometer |
| TOD | time-of-day limit reached during command |
| STOPPED | STOP_DRIVING or a shutdown command |
| WAYPT_TIMEOUT | GO_TO_WAYPOINT command timed out |
| NO_PATH | NAV could not find a safe path |
| NO_PROGRESS | insufficient progress, limit cycle or stuck |
| BUSY | sequencing error, mobility already running |
| VISODOM | too many Visodom steps failed to converge |
| *Reactive hazard detection - something bad DID happen with the hardware* | |
| IMG | IMG reported an error grabbing images |
| SAPP | SAPP error, probably a problem with the IMU |
| MOT | MOT reported error other than contact switch |
| IDD | IDD unstowed during drive |
| DRIVE_TIMEOUT | single step during GO_TO_WAYPOINT timed out |
| CSW | motion was stopped due to contact switch |
| TILT | excessive or unknown tilt during drive |
| SUSPENSION | excessive or unknown susp during drive |
| BAD_TABLES | drv tables enabled but corrupted |
| NORTH | insufficient northerly tilt angle |

**Table 1**. Mobility Manager Fault Types. The topmost faults result in a *Goal* error, which indicates the vehicle is still safe but failed to achieve its objective. The lower faults result a in *Motion* error, i.e., the vehicle has already entered into an unexpected state and is precluding any further motion pending Earth-based analysis.

hazard map, returns a recommended drive primitive to safely make progress toward the goal. GESTALT software is part of the NAV module which provides infrastructure services to MOBM.

It is the mobility manager software that interfaces the MER system to GESTALT, and implements the main loop for autonomous navigation. This consists of selecting camera pairs from which images for hazard detection will be acquired, requesting those images from the imaging services flight software, and passing the image data to GESTALT for analysis. If GESTALT sees a sufficiently safe path toward the goal, the mobility manager will instigate the drive primitive by sending a message to the low-level driving software. When the maneuver is complete, the mobility manager determines if the waypoint is reached, responding back to the command system if so. Otherwise, another stereo pair of images is requested, and the cycle (a "step" in the autonav loop) is repeated.

The general framework for the software modules that comprise the mobility software is illustrated in Table 2.

The MER rovers do not have computers fast enough to process hazard avoidance images while the vehicle is driving - they would be continually "driving past their headlights" if they tried to. Acquiring and processing a single image pair may take upward of 3 minutes, seeing hazards a meter away, yet the rovers would have driven more than six meters during this image processing.

So, instead the rovers remain stationary while taking and processing images to determine which direction to move. This removes any real-time requirements from the image processing phase, and the mobility manager and GESTALT software is run in a low-priority task, so as not to slow down high priority tasks and cause them to miss their deadlines.

## 4. Low-Level Driving

Reactive hazard detection – handling those hazards encountered during vehicle motion, such as excessive rover body tilt - requires real-time response. Hence we implemented a separate, high-priority task for performing the actuation motion primitives in the DRIVE software module. Messages indicating which driving primitive to perform (e.g., turn in place, arc) are sent to this module, which then computes steering angles and the expected number of wheel revolutions based on Ackermann steering on flat ground.

The first phase of the turn-in-place and arc driving primitives is to steer the corner wheels to the computed angles. When steering is complete, the wheels are turned. Each wheel speed is scaled according to the Ackermann calculations - wheels on the outside of an arcing turn have a longer path length to cover, and are rotated at higher speed than the inside wheels.

While the motors are being actuated, the DRIVE software is checking for the nominal termination condition (all steering at prescribed angles for a steering-only command, desired heading achieved for turn in place, or at least half the wheels rotated the precomputed amount for arc maneuvers). The drive software is also looking for off-nominal termination conditions, including excessive body tilt, articulation of the rocker-bogie suspension system, too lengthy of an actuation, and others shown in Table 1. Finally, the software is recording system state in an array for subsequent transmission, including steering and wheel motor position, motor current, and vehicle attitude as measured by the IMU.

Incidentally, the DRIVE software is the only code on board that knows the geometry of the mobility system. GESTALT has parameters to indicate what types of maneuvers the vehicle is capable of performing, and parameters that dictate what constitutes a hazard to the vehicle - parameters that make sense even for entirely different mobility platforms. Self-collision detection done by the instrument arm software and wheel-removal filters in GESTALT consider swept-out volumes of the mobility system, for simplicity and conservatism (it is rare that any part of the arm intersects even the swept out volumes during most operations).

Actual interfacing to motor control electronics is done by the motor control device driver software, MOT. Mobility, instrument arm, high gain antennae pointing, and mast camera pointing all use this driver for running motors - they all use brushed motors with quadrature relative encoders operated by identical PID controllers implemented in FPGAs. The MOT software is given a set of motors to run, and a goal position for each motor. FPGAs are programmed appropriately, and all motors in the set are started simultaneously. Maximum motor velocities are scaled so that nominally they all complete their motions simultaneously. Should any motor experience a fault, such as stall or overheat, all motors in the group are automatically shut down. The PID loop for an individual motor is run at 1 KHz in the FPGA. Software monitors progress (position measured by encoders and potentiometers, current draw by the motors) at 8 Hz.

The onboard position is nominally estimated by combining the wheel motions measured by encoders with heading changes measured by the gyros in the IMU, but this initial estimate can be improved by applying Visual Odometry software if resources permit [1].

## 5. Autonomous Driving

One of the great design challenges of the MER mission was meeting the goal that the system should be able to drive itself *safely* through 100 meters of Viking Lander I-like (VL-I) terrain [9] (mostly flat with occasional obstacles every 5 or more meters) in a single day, with no help from Earth once its drive has begun. This requirement, dictated by the scientific goals of the mission, meant that a simple teleoperated mode of driving would provide insufficient capability in all but the most benign terrains. The potential position estimation error due to slip, the inability to see beyond nearby rocks and rolling terrain, the limited number of communication opportunities and limited bandwidth would all restrict the vehicle's maximum achievable distance each day, to varying amounts depending on terrain. Therefore the decision was made to incorporate the capability of Autonomous Driving onboard.

The problem with primitive driving commands is that they cannot adapt to unexpected conditions, e.g., slip that occurs in loose soil or heading changes that occur on slopes. When driving autonomously, the rover makes its own decision about the best way to reach its intended goal location. At each step it reassesses its situation based on the latest information from the environment: rover position estimated by wheel encoders, IMU and/or Visual Odometry; rocker/bogie configuration measured by potentiometers and encoders; the allotment of time remaining for its drive; and a prediction about the shape of its environment computed from recent and current pairs of stereo images. These capabilities allow the rover to drive safely, even in areas for which there is no high resolution imagery available beforehand.

### *Primary Autonomous Capabilities*

MER vehicles provide three orthogonal capabilities for autonomous driving: Terrain Assessment, Path Selection, and Visual Pose Update. All combinations of these modes are available to the human rover drivers planning rover activities; Table 2 names all the combinations of these modes and summarizes their use.

In *Terrain Assessment* (or *Predictive Hazard Detection*) mode, one or more stereo pairs of images are processed into a traversability or *goodness* map, and merged with an existing map. Spirit uses the body-mounted 120 degree FOV HAZCAMs for terrain assessment, but Opportunity uses pointable, mast-mounted 45 degree FOV NAVCAMs because the Meridiani terrain has very fine grained soil that is not resolvable in the HAZCAM images at 256x256 resolution. If path selection is also enabled, candidate motion paths are projected into this

| Driving Mode | Terrain Assessment | Path Selection | Visual Odometry | Spirit | | Opportunity | |
|---|---|---|---|---|---|---|---|
| Directed Driving | no | no | no | 451 m | 9% | 1973 m | 33% |
| Visodom | no | no | YES | 410 m | 8% | 561 m | 9% |
| Blind Goto Waypoint | no | YES | no | 2196 m | 46% | 1911 m | 32% |
| Visodom Goto Waypoint | no | YES | YES | 379 m | 7% | 121 m | 2% |
| Guarded Motion | YES | no | no | 36 m | 1% | 117 m | 1% |
| Guarded Visodom | YES | no | YES | 0 m | 0% | 0 m | 0% |
| Autonav | YES | YES | no | 1315 m | 27% | 1262 m | 21% |
| Autonav with Visodom | YES | YES | YES | 3 m | 0% | 0 m | 0% |
| | | | | 4798 m | 100% | 5947 m | 100% |

**Table 2.** MER driving mode usage as of 15 August 2005, counting 573 sols for Spirit and 555 sols for Opportunity. (Distances are as measured onboard, and can be overestimates of actual distance traveled if wheels slip during non-Visodom drives. Only rover translation distances are reflected here; turn in place and single-wheel trench digging motions are not reported.)

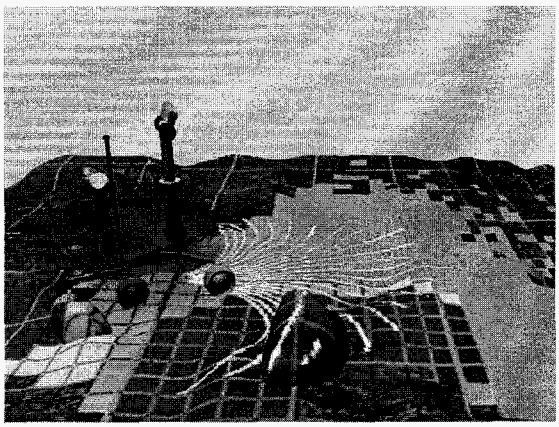


**Figure 3.** Illustration of Terrain Assessment and Path Selection. Red cells indicate unsafe areas around the large rock, yellow cells indicate traversable but rougher areas around the smaller rock, and green cells indicate safe and flat areas.

map (see Figure 3), and a weighted evaluation of the constituent cells is assigned to each path. This results in a set of *Obstacle* path evaluations; low values indicate a less traversable path, high values indicate a more traversable path.

In *Path Selection*, the rover is given autonomous control over its drive direction (in contrast to *Directed Driving*, in which it follows a single pre-commanded arc path). To select among multiple paths, the current X,Y goal (and current rover Z) is reprojected from world frame into rover frame, and votes are assigned to all possible candidate paths according to how effectively each path would drive the rover toward its goal point. The path that would lead directly toward the goal is given the highest evaluation, other paths are assigned lesser values according to a Gaussian distribution. The variance of each distribution is configurable, nominally 3.2 curvature units for arcs, 97 degrees for point turns.

There are also two off-nominal behaviors available: backups and limit cycle turns. Goals that lie inside the tightest turning circle (corresponding to an arc with 1 meter radius) cannot be reached using arcs, so in those situations an arc that backs away from the goal is selected as the preferred arc path. When backing up in this mode, the rover does a "K-turn", driving in its tightest arc away from the goal location. As of February 2005, when a backup behavior is chosen the highest path evaluation is halved, which in practice causes a point turn to be preferred over an arc that backs up. The result of this processing is a set of *Waypoint* evaluations, which are merged with the Obstacle path evaluations and result in *Hazard Avoidance* capability, if Terrain Assessment is also enabled.

The second off-nominal behavior detects when the rover fails to make progress after multiple steps. This will happen if Visual Odometry detects so much slip that little progress is made, or if the path to the goal is blocked by too large an obstacle and the rover keeps trying unsuccessfully to drive around it. In either case, the *Limit Cycle Check* will detect the condition and either terminate the command or force the rover to turn in place some amount to get a new heading, from which it might find a better path around the obstacle.

In *Visual Pose Update* (or *Visual Odometry*), the rover updates its current position and/or attitude by comparing features found in stereo image pairs taken before and after a small motion step. Both MER vehicles use NAVCAMs for this Visual Odometry (or Visodom) processing, since the scale changes induced by even small motions in the wide FOV HAZCAM images make autonomous tracking of features difficult. This processing only converges successfully if the terrain has a sufficient number of features visible in each adjacent image pair, so the human rover driver must ensure that NAVCAMs are pointed toward useful features anytime Visodom is used.

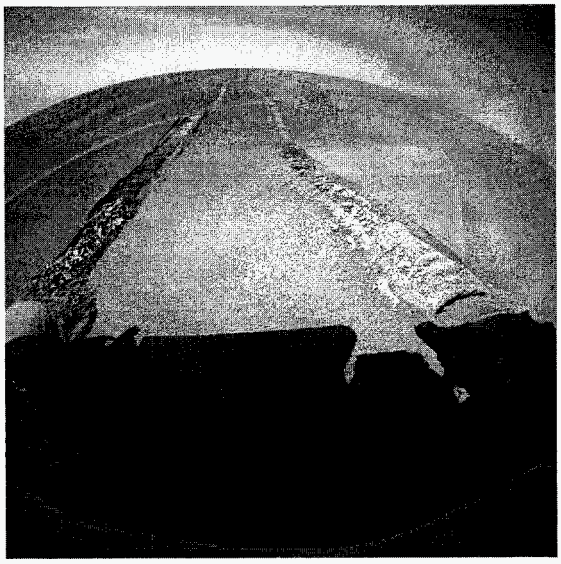


**Figure 4.** Front HAZCAM showing a non-geometric hazard encountered by Opportunity on Sol 446: 50 meters of commanded driving resulted in only 2 meters progress, which you can see in the tracks leading up to Purgatory Ripple, in which Opportunity was mired for over a month. Although this was a "blind" or Directed drive, Opportunity might have gotten stuck even if GESTALT had been turned on because there were no geometric hazards to avoid, just loose terrain. Fortunately, Visual Odometry has been used ever since as a Slip Check to ensure we never command more than 5 meters of driving without a guarantee of motion; this new safety check helped out on Sol 603, when Visodom measured 44% slip while climbing a similar dune, and stopped the drive before getting further bogged down. Opportunity was able to back out easily during its next drive.

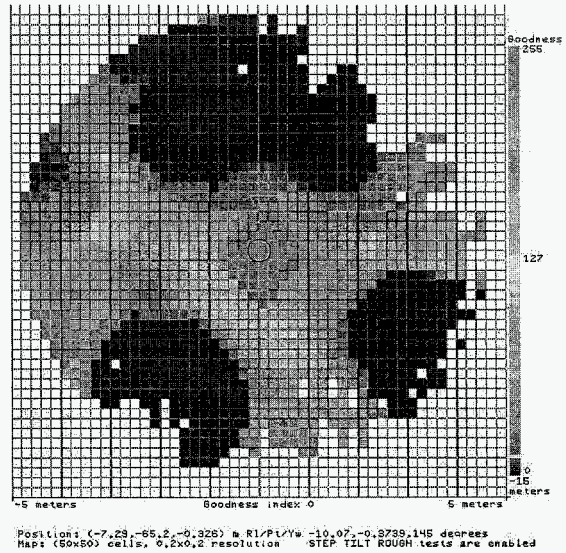



**Figure 5.** Illustration of Goodness Map from Spirit's Sol 107. Red areas indicate obstacles (typically a rock taller than 20cm at the center of the red blob), yellow/orange indicate traversable areas. Only information within a 5-meter radius is maintained in the onboard map.

*Terrain Assessment Overview*

When performing Terrain Assessment, each MER vehicle is able to find geometric hazards in the nearby terrain *before* it ever drives there. Only geometric hazards are found; no assessment of the stability of the terrain is performed onboard (see Figure 4 for an example of why that would be useful). For example, a Step hazard is defined as an object large enough to potentially high-center the vehicle, either on the Warm Electronics Box (WEB) or the rocker/bogie suspension. MER wheels are 25cm in diameter, and on flat ground the belly of the WEB is 29cm high. Considering that the vehicle might tilt as much as 35 degrees, and allowing for measurement error in the stereo data, the parameter describing the maximum height of a still-traversable obstacle was set to 20cm for nominal driving.

The MER vehicles' hazard detection software models the world as a 2D grid. The size of the grid cells is chosen to match the mechanically-determined obstacle size, 20cm. Each cell stores a continuously-varying scalar *goodness* value that represents how safe the rover would be if its center were located on that cell. This results in a *configuration space* world map, because it describes safe vehicle configurations, not precise object boundaries. For example, a lone 20cm rock is not represented as a single "unsafe" grid cell. Instead, all cells in a rover-sized disc centered over the rock will be marked as unsafe (as in Figure 3, because having the rock anywhere under the rover would be unsafe; that is, all vehicle configurations (locations in the world) that include that rock *anywhere* underneath the rover must be declared unsafe. Finding a safe path to drive is thus reduced to finding only a 1-cell-wide path through the goodness map.

Although the software supports orientation-dependent goodness assessment, which might make it possible to navigate gaps narrower than its solar panels, the processing overhead required to support it (a separate goodness map for each orientation) would have slowed down the autonomous driving to an unreasonable rate. Instead, terrain assessment is performing by modeling the rover by its turning circle as a 2.6 meter diameter disc. This design also gives us a useful invariant: *the autonomy system will only drive the rover into areas where it can safely turn in place.*

Our analysis of the terrain comes from local plane fitting of the stereo-computed range data. The rover's assessment of the terrain is based on how much each rover-sized patch of ground differs from a flat plane. To analyze a given rover-sized patch of terrain, all XYZ points falling within that patch are fit to a plane. The parameters of

that plane fit are used to assess how safe the rover would be if it were there, in any orientation. A continuously-varying *goodness* is independently computed in each of several filters: the Step, Tilt, and Roughness filters. The final evaluation of a cell's goodness is the most conservative one found by any of these filters, described below.

No Terrain Assessment can be done unless geometric information about the terrain has been measured. On MER, stereo cameras provide these measurements.

*Robust Stereo Image Processing*

MER rovers use passive stereo image processing to measure geometric information about nearby terrain. Stereo vision is an attractive technology for planetary exploration because it has low power requirements and nominally requires no moving parts. JPL's stereo vision software has a long history [16], [25], especially in real robotic systems (Mars rover research [13], [22], [21], Athena [3], Unmanned Ground Vehicles [14], Pioneer [11], Urbie [15], and Perceptor [2], [18]).

The Sojourner rover demonstrated the first use of autonomous stereo triangulation on a planetary rover [17]. But the Sojourner system relied on active projection of 5 laser stripes, and only found at most 20 XYZ points from each pair of stereo images. In contrast, MER uses passive stereo vision, relying on sunlight to illuminate the terrain. And by virtue of their faster processor (20 MHz compared to 0.1 MHz on Sojourner) and new software, MER vehicles compute many more point measurements: Opportunity has measured an average of 48,000 XYZ points in each of 69 NAVCAM image pairs as of sol 322, and Spirit has measured an average of 15,000 XYZ points in each of 1687 HAZCAM image pairs as of sol 342.

The general algorithm is described in [8], but we summarize the MER implementation here. Images can be acquired using any of the available stereo pairs, typically the front and rear HAZCAMs on Spirit, and the NAVCAMs on Opportunity. The raw data is read at 1024x256 resolution and software-downsampled to 256x256 resolution. Images are then rectified (i.e., resampled) to eliminate any lens distortion using fixed CAHVOR (radial distortion) or CAHVORE (fisheye) camera models [7]. Images are then correlated on a scanline-by-scanline basis using a 7x7 square pixel subwindow and the Sum of Absolute Difference (SAD) metric. SAD scores are generated at all candidate integer disparities at the nominal 256x256 resolution. A variety of consistency checks is applied to the resulting correlation curve, and only those curves that exhibit a unique minimum are accepted as valid. Each valid integer disparity is then mapped to a 3D point using the camera model. The final result of stereo image processing is thus an image whose pixel values are either Unknown, or the 3D location of the feature at that pixel.

Although this algorithm has been used successfully in research settings, several enhancements were made to improve its robustness for Mars operations. In particular, several filters were added to the already existing set of Left/Right Line of Sight (LRLOS), Blob, Border, and Curvature Threshold filters described in [8].

*Gamma Correction* To better enhance the detail of terrain hidden by shadow from the rover and nearby rocks, we apply gamma correction to the raw images.

*Flat Filter* Any curve whose minimum is not unique (i.e. shares the same value with *either* its neighbors) will be rejected.

*Overhang Filter* Purely horizontal features cause problems for scanline-based stereo processing. For example, in our testbed environment, dark horizontal power strips along a white wall appeared to the rover like spears pointing down at the cameras. Since our nominal driving environment has nothing hanging down from overhead, we filter out any range values that appear to loom back toward the rover. Any pixel whose range value is some percentage closer than those below it in the image is eliminated. The filter has proven useful in flight by occasionally eliminating noisy range data beyond the nominal 5 meter Terrain Assessment distance.

*Fixed Vehicle Mask* The 120 degree field of view of the HAZCAM optics is so large that some parts of the vehicle appear in each image. These need to be eliminated before terrain processing can occur; it wouldn't do for the rover to be afraid of its own arm or wheel, for instance. Those vehicle parts that always appear in the same place in the image (e.g., the stowed IDD, cabling mounted near the cameras, the solar panel above the rear HAZCAMs) are eliminated from consideration using a fixed pixel-based mask.

*Moving Vehicle Mask* Unfortunately, the fixed vehicle mask is not enough to eliminate all parts of the rover from the HAZCAM FOV. In particular, the wheels can move freely throughout the field of view, so they are correlated along with the terrain and are filtered out of the *range* image, rather than the original image. A filter comprising a swept cylindrical volume reflecting the range of motion of the rocker-bogie system and steering motor limits is applied to all range data; any points lying in that volume are eliminated.

*Terrain Assessment*

The MER vehicles use the Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) system for terrain assessment [8]. Inspired by Carnegie Mellon's Morphin algorithm [20], [24], this system uses

3D geometric information to estimate how safe the rover would be at each point in the world. Originally demonstrated on the Athena Rover prototype [3], it is now being evaluated on testbed vehicles for potential use in the upcoming MSL mission as well [23].

A central idea in this implementation is that *new data always overrides old data.* In particular, no matter how safe or unsafe a given cell had been considered, whatever the latest range data shows should be taken as true. This is a useful way to compensate for two problems. First, the range data computed for objects far from the rover is less accurate than nearby range data; by overlaying new data we eliminate any misevaluations that occurred because of inaccuracies in the range data. Second, overlaying of new data makes the rover's ability to navigate more robust to any position estimation errors, since it always prefers the newest 3D information over any potentially misregistered earlier data.

The actual assessment is computed from the stereo data associated with each rover-sized patch independently. The parameters of the best-fit plane are found for each rover-sized patch, and are used to compute the following:

*Tilt Filter* The surface normal of each planar patch is compared against the maximum allowed vehicle tilt. Patches with tilt between 0 and some minimum tilt are given a perfect goodness value (255); tilts above the maximum allowed are obstacles (goodness value of 0); those in between are weighted linearly.

*Roughness Filter* The residual from the planar fit in the direction of the surface normal provides an overall measure of how rough or uneven the region is.

*Step Filter* Once the planar parameters for a rover-sized disc are found, the *local elevation* (distance from each constituent cell's mean elevation to that plane) is computed. A filter using a smaller WEB-sized disc is run over the rover patch, computing the difference of the min and max local elevations within each smaller disc yields the largest step found within that smaller disc. Finally, the largest such step found in the larger rover-size disc is compared against the maximum allowed obstacle size. Patches with small max step size are given a perfect goodness value (255); steps above the maximum are obstacles (goodness value of 0); those in between are weighted linearly.

*Robust Terrain Assessment*— Several key ideas have gone into making GESTALT more robust.

*Layered Goodness Maps*

The field of view and range resolution of 256x256 HAZCAM images are just barely large enough to generate one complete rover-sized footprint. Hence all Terrain Assessment performed near the edges of the field of view is *not* based on a complete disc, but rather a partial disc. That means that when an obstacle leaves the field of view, all future terrain analysis would fail to take it into account. This causes the configuration space analysis to break down; placing the center of the rover near the edge of the visible field of view might seem perfectly safe, because the partial 3D data does not include the obstacle. As a result, in early implementations of the software the rover would start driving around an obstacle, then steer back directly into it.

To address this concern, we introduced a multi-scale representation of the world: Layered Goodness Maps. Instead of processing the stereo-derived goodness directly, we first "shrink" obstacles blobs and merge them into each layer separately, then "grow" obstacles from each layer back to their original size, and finally use the most conservative assessment of goodness found in any layer as our estimate of the terrain safely. The goodness map computed from the latest stereo data is filtered into multiple layers using discs varying from a 1 cell radius (layer 6) up through rover-sized discs with a 7 cell radius (layer 0). The net effect is to shrink any obstacle blobs by (6 - layer numbers) cells. Low individual goodness values in layer 0 often represent "point" obstacles like individual large rocks; low values found only in higher layers represent "dispersed" obstacles like a 10cm rock 1 meter away from a 10cm ditch.

*Partial Range Data - Clipping*

The overlaying of new data can cause cells on the edge of the field of view to be misevaluated. If only a small piece of an obstacle is visible, the range data representing it will not be classified as a hazard. The percentage of obstacle range data in the cell at the edge of the FOV drops, and the cell is mistakenly evaluated as safe. We address this by being more conservative in replacing goodness values at the edge of the field of view. Instead of simply overlaying new range data everywhere in the map, around the edges of the field of view we take the most conservative of the current and previous values.

*Pitchers Mounds*

An early implementation of the Step Filter simply compared mean elevations at adjacent cells. Field testing revealed that this approach was not conservative enough, because the prototype rover high-centered on a pyramid-shaped obstacle. The next implementation extended the search for step differences to anywhere within the rover-sized (2.6 meter diameter) patch. But this was found to result in too-conservative assessments of flat/hill transitions, even on low "pitcher's mound" hills. Restricting attention to the smaller WEB-sized disc still preserves

vehicle safety; anything that constitutes a hazard for the solar panels will still be detected. But now the rover can climb pitcher's mounds again.

*Tall Thin Obstacles*

A tall, thin obstacle will not occupy enough of a grid cell to make a meaningful change to the mean elevation at that cell. But it still must be avoided. Having found this problem during field testing, the Step filter was updated to not just the mean elevation within a cell, but the mean plus some number of standard deviations.

*Autonomous Goal Selection*

MER vehicles are nominally commanded toward a precise, metrically specified goal in rover frame (e.g., 5 meters forward) or as X,Y coordinates in world frame (e.g., 2 meters north and 1.1 meters east of the current Site Frame origin). There is some amount of autonomy in deciding when to step (e.g., a radial tolerance in meters around each X,Y goal, use of the onboard IMU during turns in place to determine when the desired heading has been reached), but most commands require human drivers to decide in advance where the rover needs to go.

MER vehicles also provide a limited means of autonomous goal selection using the TURN_TO_A_ROCK command. Unlike the Autonav driving modes which force the rover to drive around obstacles, this command causes the vehicle to turn *toward* the nearest obstacle. This provides a means of servoing toward to a target even if the rover slips along the way. However, there is no way to ensure that this command will lock into any particular rock, although a wedge of yaw values may be used to constrain the area of terrain that will be searched. It will simply find the one nearest the rover; it turns toward *a* rock, not *the* rock. Obstacles are found using certain data computed during the onboard traversability analysis: the goodness map and an "elevation max count" map. This latter map stores the number of times a given cell was found to have the highest mean elevation in all rover-sized patches that contain it.

Originally envisaged as a means of compensating for slip when approaching certain targets, this command has seen little use on MER other than during engineering tests. The MER position estimation system (including Visual Odometry) has performed so well during the primary and extended missions that it was typically not necessary to rely on this capability for re-pointing. Even when it might have been useful, other considerations dictated that precise pre-planned motions be used instead, e.g., when a feature's topology was not representable as an obstacle to the onboard software, the uncertainty in how much time and power would be needed to complete the drive made planning more difficult, a certain orientation was needed to optimize communication, or there was a need to keep the Instrument Deployment Device (IDD, or rover arm) work volume free of obstacles.

## 6. Deployment Results

Both MER vehicles have lasted far beyond their design lifetime of 90 days; so far, each has run for more than 650 days on the surface of Mars in geologically diverse areas. Papers describing Spirit's drive toward the Columbia Hills [10], Opportunity's exploration of Endurance Crater and its multi-kilometer trek southward [4], general tradeoffs considered in choosing between the various driving modes [5], and a systems-level view of the onboard autonomy [12] describe the system's performance in detail.

One of the MER software design principles was to implement only what was necessary for the mission; the more parameters and conditionals, the more code branches would need to be tested. However, uncertainty in what terrain would be encountered necessitated keeping the onboard Terrain Assessment software flexible: for instance, the February 2005 version of software has 292 surface navigation-specific parameters. This flexibility proved crucial for successful operation on Mars. For example, in the original design stereo HAZCAM images were processed at 128x128 pixel resolution. This was sufficient for good results in our indoor testbed environment. However, Earth-based outdoor field tests suggested that 256x256 resolution resulted in better results at the outer range of stereo processing, 4–5 meters. This same behavior was observed in the initial images from Spirit, and as a result Spirit was reconfigured to use higher resolution stereo.

*Spirit Rover: Gusev Crater*

Spirit landed on 3 January 2004 about 2 kilometers west of the Columbia Hills inside Gusev Crater. Unlike Opportunity, which ultimately found evidence for water just 9 meters from its landing site, Spirit had to drive over 3 kilometers before finding convincing evidence starting at the base of the Columbia Hills. Having promised to deliver only 600 meters of traverse, the hills seemed unreachable at first. But the combination of human terrain assessment and onboard navigational autonomy enabled Spirit to reach the base of the hills by June 2004.

Spirit's driving began as a run for the hills (shown in the daily drive distances for sols 90–160 in Figure 6), but was followed by a year of exploration on the slopes of the Columbia Hills (see Figure 7). Driving on the slopes led to new styles of driving, with an emphasis on using Visual Odometry software to detect and/or compensate for high slip, commanded use of explicitly-sequenced keep-out zones to avoid known obstacles, and novel uses of the low-level driving code to compensate for temporary

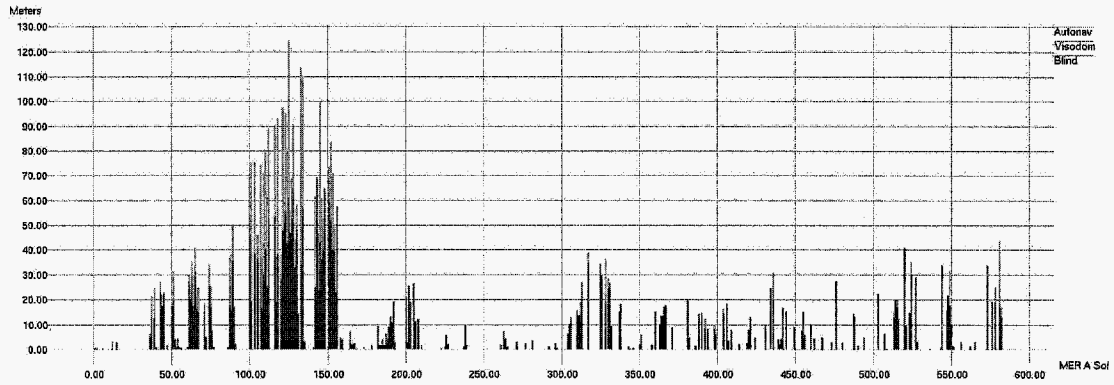



**Figure 6**. Plot of Spirit's complete drive history per sol through Sol 588; the most driving in one sol was 124 meters on Sol 125, the longest contiguous autonomous drive was 78 meters on sol 133. Note that the vertical scale here is stretched compared to Figure 9. Red indicates blind driving, green indicates autonav (rows 5–8 in Table 2), blue indicates Visodom (rows 2, 4 in Table 2).

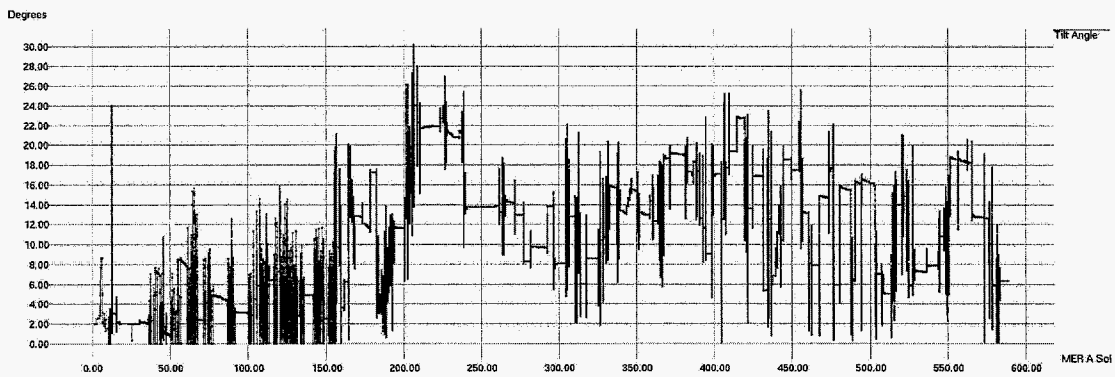



**Figure 7**. Plot of Spirit complete tilt history through Sol 588.

problems (e.g., a rock stuck in a wheel, dragging one wheel to extend drive motor lifetime). The success of its mission owes much to the robustness and flexibility not only of the onboard mobility software, but also of the human ground operations team who kept creating new and better ways to drive as explained in [10].

*Opportunity Rover: Meridiani Planum*

Opportunity landed on 24 January 2004 inside Eagle Crater at Meridiani Planum. Scientists were thrilled when the first images revealed bedrock outcrops a mere 9 meters from the rover, but engineers were horrified to see very little 3D information recovered from the first stereo images.

Flexibility of software design was even more critical for Opportunity. The fine-grained particles covering the majority of Eagle crater were unresolvable by stereo image processing of the 1-bit-per-pixel compressed 1024x1024 45 degree FOV NAVCAM images. Fortunately, images with more bits per pixel were acquired quickly and proved easy to process. But at the nominal size used by onboard autonomy software (256x256 squared pixels), even 8 bits per pixel were insufficient to resolve the soil in 120 degree FOV HAZCAM images. This meant that the autonomy system, tested almost entirely using rigidly-mounted HAZCAMs, had to be reconfigured to use pointable mast-mounted NAVCAMs. Fortunately, we were able both to reconfigure the mobility software to these conditions and also uplink new, more flexible software. Table 3 shows some of the differences between the vehicles that persist across the mission, and others that change daily when the terrain requires.

Although the rovers' design requirements had been for traverse at tilts less than 15 degrees (see the large daily drive distances in Figure 9), Opportunity spent more than six months in Endurance crater with tilt *above* 15 degrees, as shown in Figure 10. Strategies that were adopted for high tilt driving can be found in [4], [5].

*Software Versions*

There have been three versions of the mobility flight software used on Mars as of October 2005.

The version used during the first 90 days of operation on each rover (the Primary Mission) had autonomy software

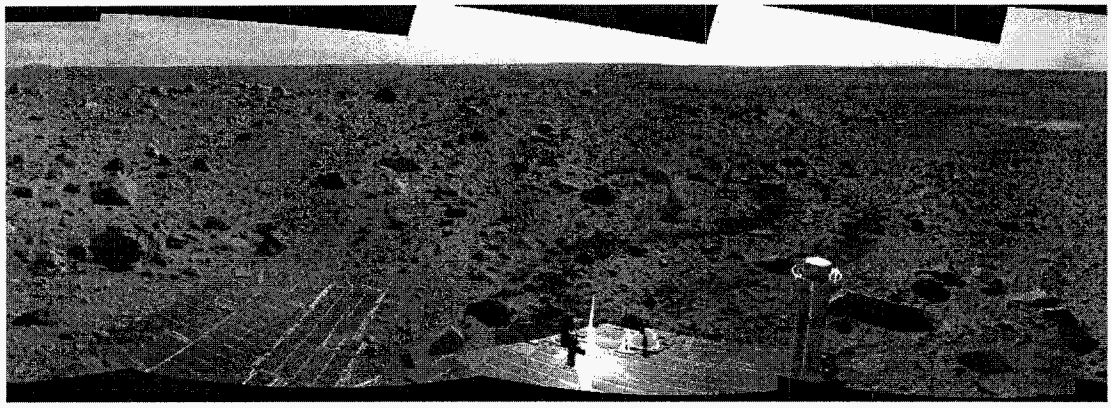


**Figure 8.** Image of Spirit's tracks following an obstacle avoidance maneuver on Sol 107.

| Parameter | Spirit | Opportunity |
|---|---|---|
| Autonav camera pair | HAZCAM (front or rear) | NAVCAM |
| Stereo processing resolution | 256×256 | 256×256 |
| Useful (unmasked) stereo image size | 184×256 (front) 108×216 (rear) | 256×256 |
| Stereo search range | 0.3 - 5.0 $m$ | 1.2 - 10.0 $m$ |
| Grid cell resolution | 0.2×0.2 $m^2$ | 0.2×0.2 $m^2$ |
| Grid size | 10×10 $m^2$ | 12×12 $m^2$ |
| *As needed for daily operations* | | |
| Max distance between image acquisitions | 0.5 – 1.5 $m$ | 0.5 – 2 $m$ |
| Engineering diagnostic verbosity | 0 – 4 | 0 – 4 |
| Max allowed Visodom failures | 0 – 15 | 0 – 20 |
| Number of Point Turns to enable | 0 – 46 | 0 – 46 |

**Table 3.** Some parameter differences between Spirit and Opportunity

that was overly conservative. It required that imaging and terrain processing occur after each autonomously-commanded motion, and also checked for step obstacles everywhere within each rover-sized disc (2.6m diameter). In practice this meant that the rovers were unable to autonomously climb over small "pitcher's mound"-like hills; this occurred several times on Spirit.

In April 2004, a new version of software was uplinked to both rovers, initiating the Extended Mission. This version incorporated several robustness enhancements made to the autonomy software during pre-landing outdoor field tests and some lessons learned during the primary mission. The most dramatic changes included the ability to skip Terrain Assessment when the existing data already ensure that all paths are safe, and the ability to autonomously traverse small hills. This enabled the much longer drives seen during sols 90–160 in Figure 6.

The next software upgrade was made in February 2005 to help streamline commanding and data analysis, improve position estimation in highly sloped terrain, and enable Terrain Assessment using more than only two Navcam images. This upgrade also included the first changes made to the low-level mobility software since 2002: use of look-up tables to generalize the steering angles used by primitive commands, use of look-up tables to estimate slip based on rover attitude, and the ability to alternate between wheel dragging and non-dragging modes in the event of a disabled drive motor. Before this release, rover drivers had to explicitly command each step of a Visodom-enabled drive to ensure that there would be sufficient overlap between images (nominally 60%). But now the Goto Waypoint command could be configured to restrict autonomously-commanded motions to ensure sufficient overlap, assuming the actual heading change is no larger than what was commanded.

A fourth upgrade, planned for mid-2006, might incorporate several technologies now being evaluated. These include autonomous in situ instrument placement following a successful drive (aka Go and Touch), global path planning to enable intelligent backtracking, visual servoing, and autonomous detection of dust devils and clouds in onboard imagery. Other planned enhancements include explicit keepout zones, speed optimizations to Visodom and Autonav, and new commands intended to reduce the complexity of commanded sequences.

## 7. Conclusion

The MER Mobility Software has enabled an unprecedented amount of driving across the surface of another world. The flexibility of its design enabled it to function effectively in two dramatically different terrains.

## 8. Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Thousands of people were involved in the mission at JPL and throughout the world, in government, academia, and industry. We gratefully acknowledge their outstanding work, which enabled us to explore Mars through Spirit and Opportunity. We would like to thank the entire Mars Exploration Rover team, and in particular the mobility working group and test team, including Paolo Bellutta, Diana Darus, Jeff Favretto, David Henriquez, Robert Liebersbach, David Lokshin, Carolina Maldonado, Rich Petras, Aaron Simo, Eddie Tunstel, Hung Vo, Rick Welch, and Reg Willson.

***Jeffrey Biesiadecki*** *has been a software engineer at NASA's Jet Propulsion Laboratory since 1993, after completing his Master's degree in Computer Science at the University of Illinois, Urbana-Champaign. He designed and implemented the core motor control and non-autonomous mobility flight software for the Mars Exploration Rovers, and is also one of the rover drivers for the Mars Exploration Rover "Opportunity", responsible for command sequences that tell the rover where to drive and how to operate its robotic arm on the surface of Mars.*

***Dr. Mark Maimone*** *is a Navigation and Machine Vision researcher at the Jet Propulsion Laboratory. He earned his Ph.D. in Computer Science from the Computer Science Department of Carnegie Mellon University in 1996, and was then a Post-doctoral Research Associate at Carnegie Mellon's Robotics Institute. Since starting at JPL in 1997, he has worked on the several Mars Rover research projects and a vision system for inspection of the Chornobyl reactor. As a member of the 2003 Mars Exploration Rover flight software team, Mark developed the vision and navigation subsystems for the MER vehicles. Mark is now part of the MER ground operations team and is developing the autonomous navigation software for the Mars Science Laboratory rover, NASA's next Mars rover mission.*

## References

[1] Khaled S. Ali, C Anthony Vanelli, Jeffery J. Biesiadecki, Mark W. Maimone, Yang Cheng, Miguel San Martin, and James W. Alexander. Attitude and position esimation on the Mars Exploration Rovers. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[2] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for demo iii. In *Proc. Intelligent Vehicles Symposium*, Dearborn, MI, October 2000.

[3] Jeffrey Biesiadecki, Mark Maimone, and Jack Morrison. The Athena SDM rover: A testbed for Mars rover mobility. In *International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, Montreal, Canada, June 2001. http://robotics.jpl.nasa.gov/people/mwm/sdm-mobility/.

[4] Jeffrey J. Biesiadecki, Eric T. Baumgartner, Robert G. Bonitz, Brian K. Cooper, Frank R. Hartman, P. Christopher Leger, Mark W. Maimone, Scott A. Maxwell, Ashitey Trebi-Ollenu, Edward W. Tunstel, and John R. Wright. Mars Exploration Rover surface operations: Driving opportunity at meridiani planum. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[5] Jeffrey J. Biesiadecki, Chris Leger, and Mark W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. In *International Symposium of Robotics Research*, San Francisco, CA, USA, October 2005.

[6] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the Mars Exploration Rovers. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[7] Donald B. Gennery. *Calibration and Orientation of Cameras in Computer Vision*, chapter Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points, pages 123–136. Springer Verlag (A. Gruen and T. Huang, ed.), 2001.

[8] Steven B. Goldberg, Mark W. Maimone, and Larry Matthies. Stereo vision and rover navigation software for planetary exploration. In *IEEE Aerospace Conference*, volume 5, pages 2025–2036, Big Sky, Montana, USA, March 2002. http://robotics.jpl.nasa.gov/people/mwm/visnavsw/.

[9] 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.

[10] Chris Leger, Ashitey Trebi-Ollenu, John Wright, Scott Maxwell, Bob Bonitz, Jeff Biesiadecki, Frank Hartman,

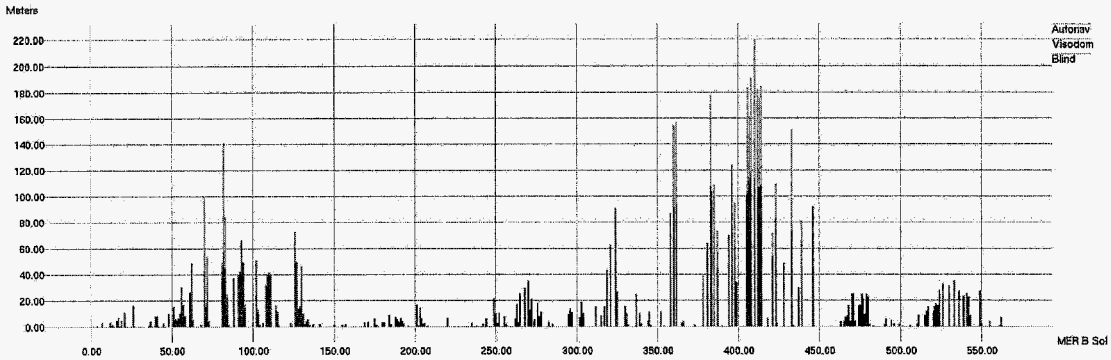



**Figure 9**. Plot of Opportunity's complete drive history per sol through Sol 567; the most driving in one sol was 219 meters on Sol 410, the longest contiguous autonomous drive was 280 meters during sols 383–385. Note that the vertical scale here is more compressed than that in Figure 6. Red indicates blind driving, green indicates autonav (rows 5–8 in Table 2), blue indicates Visodom (rows 2, 4 in Table 2).

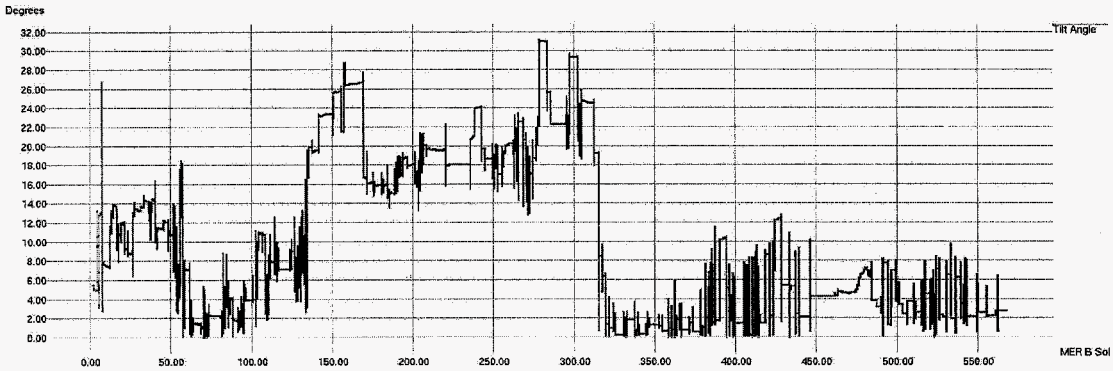



**Figure 10**. Plot of Opportunity's complete tilt history through Sol 567.


Brian Cooper, Eric Baumgartner, and Mark Maimone. Mars Exploration Rover surface operations: Driving spirit at gusev crater. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[11] M. Maimone, L. Matthies, J. Osborn, E. Rollins, J. Teza, and S. Thayer. A photo-realistic 3-D mapping system for extreme nuclear environments: Chornobyl. In *International Robotics and Systems Conference (IROS)*, pages 1521–1527, Victora B.C., Canada, October 1998. http://robotics.jpl.nasa.gov/people/mwm/pioneer/iros98/.

[12] Mark Maimone, Jeffrey Biesiadecki, Edward Tunstel, Yang Cheng, and Chris Leger. *Surface navigation and mobility intelligence on the Mars Exploration Rovers*, chapter 3, pages 45–69. TSI Press, Albuquerque, NM, USA, 2006. http://www.intelligentspacerobotics.com/.

[13] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous Robots Journal, Special Issue on Autonomous Vehicle for Planetary Exploration*, 2(4), 1995.

[14] L. Matthies, A. Kelly, T. Litwin, and G. Tharp. *Obstacle Detection for Unmanned Ground Vehicles: A Progress Report*. Springer-Verlag, 1996.

[15] 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.

[16] L. H. Matthies. Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. *International Journal of Computer Vision*, 8(1):71–91, July 1992.

[17] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, and B. Wilcox. Experiences with operations and autonomy of the mars pathfinder microrover. In *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass at Aspen, Colorado, March 1998. http://robotics.jpl.nasa.gov/people/mishkin/papers/IEEE_aerospace98.pdf.

[18] A. Rankin, C. Bergh, S. Goldberg, and L. Matthies. Passive perception system for day/night autonomous off-road navigation. In *SPIE UGV Symposium*, Orlando, FL, April 2005.

[19] Glenn E. Reeves and Joseph F. Snyder. An overview of the Mars Exploration Rovers flight software. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[20] 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. http://www.cs.cmu.edu/~reids/papers/AIAAobsAvoid.pdf.

[21] 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. http://www.frc.ri.cmu.edu/projects/mars/ publications/global_local_icra2000.ps.gz.

[22] Richard Volpe. Navigation results from desert field tests of the Rocky 7 Mars rover prototype. *International Journal of Robotics Research*, 18(7):669–683, Special Issue on Field and Service Robots, July 1999. http://robotics.jpl.nasa.gov/people/volpe/papers/JnavMay.pdf.

[23] Richard Volpe. Rover functional autonomy development for the Mars mobile science laboratory. In *IEEE Aerospace Conference*, Big Sky, Montana, March 2003.

[24] 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. http://robotics.jpl.nasa.gov/people/mwm/papers/98ras.nomad.pdf.

[25] 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.

06-0060

Submitted to 2006 IEEE Aerospace Conference Proceedings, March 2006, Big Sky, Montana, USA

# The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition

Jeffrey J. Biesiadecki and Mark W. Maimone
Jet Propulsion Laboratory
Pasadena, CA USA
jeffrey.j.biesiadecki@jpl.nasa.gov
mark.maimone@jpl.nasa.gov

*Abstract*—

NASA's Mars Exploration Rovers' (MER) onboard Mobility Flight Software was designed to provide robust and flexible operation. The MER vehicles can be commanded directly, or given autonomous control over multiple aspects of mobility: which motions to drive, measurement of actual motion, terrain interpretation, even the selection of targets of interest (although this latter mode remains largely underused).

Vehicle motion can be commanded using multiple layers of control: Motor Control, Direct drive operations (Arc, Turn in Place), and Goal-based driving (Goto Waypoint). Multiple layers of safety checks ensure vehicle performance: Command limits (command timeout, time of day limit, software enable, activity constraints), Reactive checks (e.g., motor current limit, vehicle tilt limit), and Predictive checks (e.g., Step, Tilt, Roughness hazards).

From January 2004 through October 2005, Spirit accumulated over 5000 meters and Opportunity 6000 meters of odometry, often covering more than 100 meters in a single day. In this paper we describe the software that has driven these rovers more than a combined 11,000 meters over the Martian surface, including its design and implementation, and summarize current mobility performance results from Mars.

*Keywords*— MER, robotics, Mars rover, flight software, robot mobility, autonomous navigation, fault protection, visual odometry, egomotion

## TABLE OF CONTENTS

| | | |
|---|---|---|
| 1 | INTRODUCTION | 1 |
| 2 | MER FLIGHT SOFTWARE ARCHITECTURE | 2 |
| 3 | MOBILITY MANAGER SOFTWARE | 3 |
| 4 | LOW-LEVEL DRIVING | 4 |
| 5 | AUTONOMOUS DRIVING | 5 |
| | PRIMARY AUTONOMOUS CAPABILITIES | 5 |
| | TERRAIN ASSESSMENT | 6 |
| | ROBUST STEREO IMAGE PROCESSING | 8 |
| | TERRAIN ASSESSMENT | 8 |
| | AUTONOMOUS GOAL SELECTION | 10 |
| 6 | DEPLOYMENT RESULTS | 10 |
| | SOFTWARE VERSIONS | 11 |
| 7 | CONCLUSION | 13 |
| 8 | ACKNOWLEDGEMENTS | 13 |

[1]0-7803-9546-8/06/$20.00©2006 IEEE

## 1. INTRODUCTION

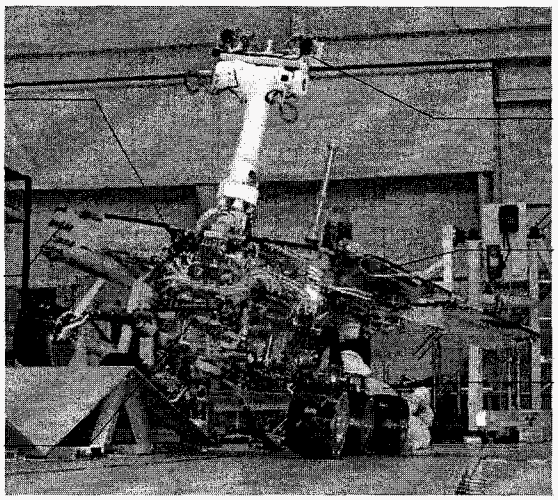


**Figure 1.** Mars Exploration Rover.

NASA successfully landed two mobile robot geologists on the surface of Mars in January 2004: the Spirit and Opportunity Mars Exploration Rovers (MER). Their primary goal was to find evidence of past water at Gusev Crater and Meridiani Planum, two geologically distinct sites on opposite sides of the planet. Although the achievement of their successful landings stands out as a technological tour de force, it was their ability to traverse while on the surface of Mars that enabled both rovers to succeed in their primary goals.

The MER rovers are typically commanded once per Martian day. A sequence of commands sent in the morning specifies the day's activities: what images and data to collect, how to position the robotic arms, and where to drive. Then at the end of each day, the rovers send back the images and data human operators will use to plan the next day's activities. The next day's mobility commands are selected based on what is known – and what

is unknown – about the terrain ahead.

The rovers are driven using three primary modes: low-level commands that specify exactly how much to turn each wheel and position steering actuators, directed driving primitives for driving along circular arcs (of which straight line driving and turn-in-place are special cases), and autonomous path selection. Low-level commands enable "non-standard" activities such as using the wheels to dig holes in Martian soil, scuff rocks, and perform mechanism health diagnostic tests. Directed drives allow human operators to specify exactly which driving primitives the rover will perform. Autonomous path selection mode allows the rover to take its current state into account when selecting which driving primitives to execute to more effectively reach a Cartesian goal location supplied by human operators.

The mobility system has six 25 centimeter diameter wheels, of which the four corner wheels may be steered – a mechanical configuration derived from the Mars Pathfinder rover Sojourner [17]. The rover body has 30 centimeter ground clearance, and large solar panels on the top of the rover require additional clearance to tall rocks (60 centimeters from ground to solar panel). Wheel baseline is roughly 1 meter side-to-side and 1.25 meters front-to-back. The MER rovers can turn-in-place about a point between the two middle wheels, drive straight forward or backward, and have at best a one meter turn radius for driving along circular arcs. Straight line driving speed is set to 3.75 centimeters/second, and the rovers turn in place at roughly 2.1 degrees/second. The rovers are statically stable at a tilt of 45 degrees, however driving on more than 30 degree slopes is not recommended due to the possibility of uncontrolled sliding. Rocks larger than a wheel are considered mobility hazards.

Both directed and path selection modes of driving can make use of on-board Stereo Vision processing and Terrain Analysis software to determine whether the rover would encounter any geometric hazards as it drives along its chosen path. In directed driving, the rover can preemptively "veto" a specific mobility command from the ground if it appears too risky. In Autonomous Navigation (autonav) and other path selection modes, the rover can select its own driving primitives to steer around obstacles and make progress toward its goal. This software provided the unique capability of enabling the vehicle to drive safely even through areas never before seen on Earth: more than 2500 meters of the rovers' combined distance was driven autonomously [12].

The rovers maintain an estimate of their local position and orientation updated at 8 Hz while driving. Position is first estimated based on how much the wheels have turned (wheel odometry). Orientation is estimated using an Inertial Measurement Unit that has 3-axis accelerometers and 3-axis angular rate sensors. In between driving primitives, the rover can make use of camera-based Visual Odometry to correct the errors in the initial wheel odometry-based estimate that occur when the wheels lose traction on large rocks and steep slopes. Visual Odometry software, not originally part of the baseline design but incorporated as "extra credit", has generated over 2500 combined successful position updates on both rovers [6].

## 2. MER Flight Software Architecture

The overall architecture of the MER flight software was based on that used for the Mars Pathfinder lander (MPF), and the command and telemetry infrastructure in particular had significant design and implementation inheritance.

Both MPF lander and MER rovers use the commercial operating system VxWorks from Wind River. This is a pre-emptive multi-tasking OS used for many real-time systems and spacecraft. MER employs the same RAD6K flight computer as used by the MPF lander (although the MER clock runs at 20 MHz, four times faster than MPF), and has a VME interface to several boards that control motors, radios, cameras, and science instruments. Some of the VME boards have serial interfaces to other devices, interfaced to the flight computer by way of FPGAs that connected both the VME bus and the serial busses. There are 128 Mbytes of DRAM and 256 Mbytes of flash memory, and 3 Mbytes of EEPROM. The Mobility task (described below) is given 80 Kbytes of stack, and nominally 13 Megabytes of dedicated heap space (of which at most 7 Mbytes has been used, as of October 2005). The flight software itself requires 9 Mbytes, of which 1.4 Mbytes is used by the Surface Navigation module.

Like MPF, the MER software is divided into high-level objects. Each electronics device was generally represented by an object, while additional software-only objects provide infrastructure services to the rest of the modules (for example, recording telemetry measurements or storing state in nonvolatile memory).

Each software object has its own thread of control, or task. Each object has its own set of state variables, and does not share memory with other objects. Instead of sharing memory, objects communicate with each other via self-contained messages (that do not include pointers to task memory). Objects implement finite state machines, and the messages from other tasks may cause transitions in an object's state machine.

This paradigm greatly simplifies rapid development of robust software (developers do not need to protect memory accesses with semaphores, because only one task accesses any given variable), and ensures real-time dead-

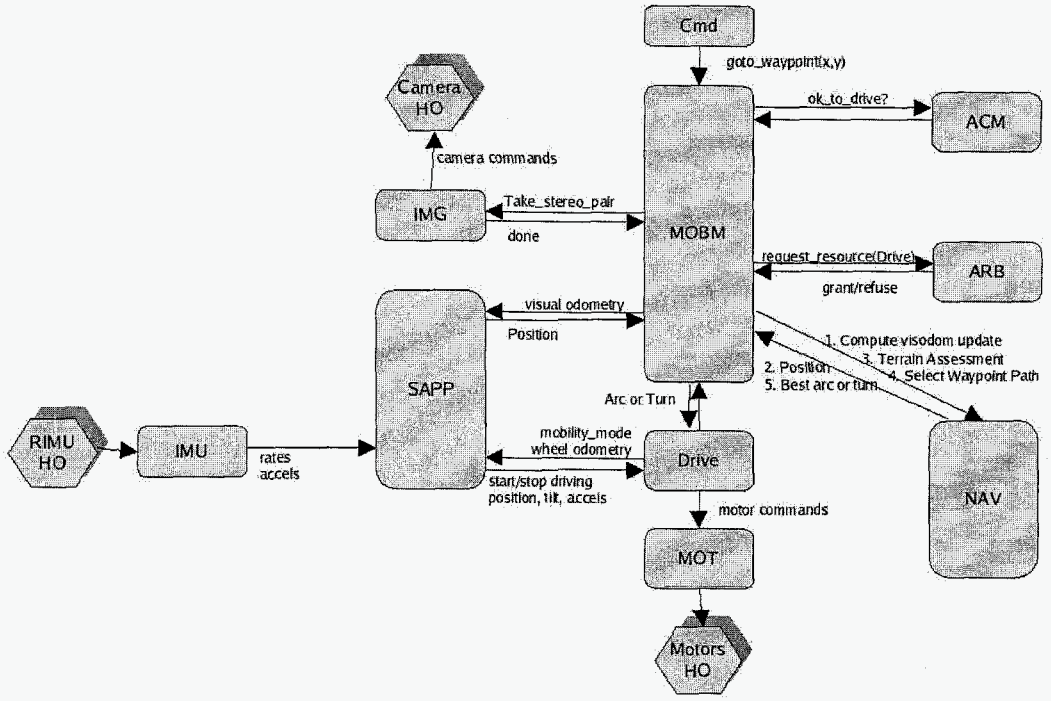



**Figure 2.** Schematic drawing of Mobility and other related software modules, and examples of the types of messages or information that pass between them. **IMG** does Image Acquisition, **Cmd** processes ground commands, **ACM** is the Activity Constraint Manager, **ARB** arbitrates use of resources, **MOBM** is the Mobility Manager, **NAV** does the high level navigation processing, **Drive** executes primitive drive commands, **MOT** controls all motors, **SAPP** manages onboard pose estimation, **IMU** processes IMU data. Hexagons represent Hardware Objects.

lines are met (because tasks do not block on semaphores or use task locks, high-priority tasks for control loops and device drivers will pre-empt lower priority tasks that do not have real-time requirements such as image compression).

That said, there were places in the flight software where the above pattern was not followed - images, for instance, contain too much data to be passed as messages, so imaging flight software employs a buffer reservation and release scheme, and pointers to reserved buffers are passed between applications (such as mobility) and the imaging flight software. But there are few such exceptions.

One important area of the flight software to understand is the *uplink* flight software which parses commands sent to the spacecraft and dispatches them to the other parts of the flight software that implement the command. Commands can either be executed immediately upon receipt ("real-time" commands) or stored as command sequences to be executed serially. Almost all commands sent to MER rovers are embedded in these event-driven sequences: a command in a sequence is dispatched only after the preceding command completes. The function called by the command system that implements a command is referred to as a *command handler* - these send a message to the appropriate software object. Eventually, the object receiving the command message must respond back to the command system when the command is completed, successful or not, so the next command in the sequence will be executed.

An overview of the MER flight software in general can be found in [19].

## 3. Mobility Manager Software

The "mobility manager" software object (MOBM - the "B" is silent) implements all command handler functions for mobility commands. This includes both low-level drive commands to perform a specific driving primitive such as turn-in-place, as well as high-level driving commands to drive to a specified Cartesian location in small steps avoiding hazards along the way.

The autonomous navigation software, GESTALT, is a generic library employed on several JPL rovers. It does not depend on any particular hardware or software interfaces for cameras and motors. Rather, given a stereo pair of images, it updates an internal map of any nearby hazards seen in the image pair. A different function is provided a Cartesian goal location, and referring to the hazard map, returns a recommended drive primitive to

| Fault Type | Description |
|---|---|
| | |
| *Higher level errors - vehicle not necessarily in a dangerous state* | |
| ARB | resource rescinded, may be due to comm pass |
| ACM | ACM said not-OK-to-drive |
| POT | CAL_STEERING using a bad potentiometer |
| TOD | time-of-day limit reached during command |
| STOPPED | STOP_DRIVING or a shutdown command |
| WAYPT_TIMEOUT | GO_TO_WAYPOINT command timed out |
| NO_PATH | NAV could not find a safe path |
| NO_PROGRESS | insufficient progress, limit cycle or stuck |
| BUSY | sequencing error, mobility already running |
| VISODOM | too many Visodom steps failed to converge |
| *Reactive hazard detection - something bad DID happen with the hardware* | |
| IMG | IMG reported an error grabbing images |
| SAPP | SAPP error, probably a problem with the IMU |
| MOT | MOT reported error other than contact switch |
| IDD | IDD unstowed during drive |
| DRIVE_TIMEOUT | single step during GO_TO_WAYPOINT timed out |
| CSW | motion was stopped due to contact switch |
| TILT | excessive or unknown tilt during drive |
| SUSPENSION | excessive or unknown susp during drive |
| BAD_TABLES | drv tables enabled but corrupted |
| NORTH | insufficient northerly tilt angle |

**Table 1**. Mobility Manager Fault Types. The topmost faults result in a "Goal" error, which indicates the vehicle is still safe but failed to achieve its objective. The lower faults result a in "Motion" error, i.e., the vehicle has already entered into an unexpected state and is precluding any further motion pending Earth-based analysis.

safely make progress toward the goal. GESTALT software is part of the NAV module which provides infrastructure services to MOBM.

It is the mobility manager software that interfaces the MER system to GESTALT, and implements the "main loop" for autonomous navigation. This consists of selecting camera pairs from which images for hazard detection will be acquired, requesting those images from the imaging services flight software, and passing the image data to GESTALT for analysis. If GESTALT sees a sufficiently safe path toward the goal, the mobility manager will instigate the drive primitive by sending a message to the low-level driving software. When the maneuver is complete, the mobility manager determines if the waypoint is reached, responding back to the command system if so. Otherwise, another stereo pair of images is requested, and the cycle (a "step" in the autonav loop) is repeated.

The general framework for the software modules that comprise the mobility software is illustrated in Table 2.

The MER rovers do not have computers fast enough to process hazard avoidance images while the vehicle is driving - they would be continually "driving past their headlights" if they tried to. Acquiring and processing a single image pair may take upward of 3 minutes, seeing hazards a meter away, yet the rovers would have driven more than six meters during this image processing.

So instead the rovers image while stationary, and process those images to determine which direction to move while stationary. This removes any real-time requirements from the image processing phase, and the mobility manager and GESTALT software is run in a low-priority task, so as not to slow down high priority tasks and cause them to miss their deadlines.

## 4. Low-level Driving

Reactive hazard detection - those hazards found while the vehicle is moving, such as excessive rover body tilt - must be responded to in a timely fashion. Hence we implemented a separate, high-priority, task for performing the actuation motion primitives. This was referred to as the DRIVE software.

The DRIVE module is sent messages indicating which driving primitive to perform. It then computes steering angles and expected number of wheel revolutions based on Ackerman steering on flat ground.

The first phase of the turn-in-place and arc driving primitives is to steer the corner wheels to the computed angles. When steering is complete, the wheels are turned. Each wheel speed is scaled according to the Ackerman calculations - wheels on the outside of an arcing turn have a longer path length to cover, and are rotated at higher speed than the inside wheels.

While the motors are being actuated, the DRIVE software is checking for the nominal termination condition (all steering at prescribed angles, or for turn-in-place, desired heading achieved, or for arc maneuvers, at least half the wheels rotated the precomputed amount). The drive software is also looking for off-nominal termination conditions, including excessive body tilt, articulation of the rocker-bogie suspension system, too lengthy of an actuation, and others shown in Table 1. Finally, the software is recording system state in an array for subsequent telemetering, including steering and wheel motor position, motor current, and vehicle attitude as measured by the IMU.

Incidentally, the DRIVE software is the only code on board that knows the geometry of the mobility system. GESTALT has parameters to indicate what types of maneuvers the vehicle is capable of performing, and parameters that dictate what constitutes a hazard to the vehicle - parameters that make sense even for entirely different mobility platforms. Self-collision detection done by the instrument arm software and wheel-removal filters in GESTALT consider swept-out volumes of the mobility system, for simplicity and conservatism (it is rare that any part of the arm intersects even the swept out volumes during most operations).

Actual interfacing to motor control electronics is done by the motor control device driver software, MOT. Mobility, instrument arm, high gain antennae pointing, and mast camera pointing all use this driver for running motors - they all use brushed motors with quadrature relative encoders operated by identical PID controllers implemented in FPGAs. The MOT software is given a set of motors to run, and a goal position for each motor. FPGAs are programmed appropriately, and all motors in the set are started simultaneously. Maximum motor velocities are scaled so that nominally they all complete their motions simultaneously. Should any motor experience a fault, such as stall or overheat, all motors in the group are automatically shut down. The PID loop for an individual motor is run at 1 KHz in the FPGA. Software monitors progress (position measured by encoders and potentiometers, current draw by the motors) at 8 Hz.

The onboard position is nominally estimated by combining the wheel motions measured by encoders with heading changes measured by the gyros in the IMU, but this initial estimate can be improved by applying Visual Odometry software if resources permit [1].

## 5. Autonomous Driving

One of the great design challenges of the MER mission was meeting the goal that the system should be able to drive itself *safely* through 100 meters of Viking Lander I-like (VL-I) terrain [9] (mostly flat occasional obstacles every 5 or more meters) in a single day, with no help from Earth once its drive has begun. This requirement, dictated by the scientific goals of the mission, meant that a simple teleoperated mode of driving would provide insufficient capability in all but the most benign terrains. The potential position estimation error due to slip, the inability to see beyond nearby rocks and rolling terrain, the limited number of communication opportunities and limited bandwidth would all restrict the vehicle's maximum achievable distance each day, to varying amounts depending on terrain. Therefore the decision was made to incorporate the capability of Autonomous Driving onboard.

The problem with primitive driving commands is that they cannot adapt to unexpected conditions, e.g., slip that occurs in loose soil or heading changes that occur on slopes. When driving autonomously, the rover makes its own decision about the best way to reach its intended goal location. At each step it reassesses its situation based on the latest information from the environment: rover position estimated by wheel encoders, IMU and/or Visual Odometry; rocker/bogie configuration measured by potentiometers and encoders; the allotment of time remaining for its drive; and a prediction about the shape of its environment computed from recent and current pairs of stereo images. These capabilities allow the rover to drive safely, even in areas for which there is no high resolution imagery available beforehand.

### *Primary Autonomous Capabilities*

MER vehicles provide three orthogonal capabilities for autonomous driving: Terrain Assessment, Path Selection, and Pose Update. All combinations of these modes are available to the human rover drivers planning rover activities; Table 3 names all the combinations of these modes and summarizes their use.

In *Terrain Assessment* (or *Predictive Hazard Detection*) mode, one or more stereo pairs of images are processed into a traversability or *goodness* map, and merged with an existing map. Spirit uses the body-mounted 120 degree FOV HAZCAMs for terrain assessment, but Opportunity uses mast-mounted 45 degree FOV NAVCAMs because the Meridiani terrain has very fine grained soil that is not resolvable in the HAZCAM images at 256x256 resolution. If path selection is also enabled, candidate motion paths are projected into this map (see Figure 4),

| Driving Mode | Terrain Assessment | Path Selection | Visual Odometry | Spirit | | Opportunity | |
|---|---|---|---|---|---|---|---|
| Directed Driving | no | no | no | 451 m | 9% | 1973 m | 33% |
| Visodom | no | no | YES | 410 m | 8% | 561 m | 9% |
| Blind Goto Waypoint | no | YES | no | 2196 m | 46% | 1911 m | 32% |
| Visodom Goto Waypoint | no | YES | YES | 379 m | 7% | 121 m | 2% |
| Guarded Motion | YES | no | no | 36 m | 1% | 117 m | 1% |
| Guarded Visodom | YES | no | YES | 0 m | 0% | 0 m | 0% |
| Autonav | YES | YES | no | 1315 m | 27% | 1262 m | 21% |
| Autonav with Visodom | YES | YES | YES | 3 m | 0% | 0 m | 0% |
| | | | | 4798 m | 100% | 5947 m | 100% |

**Figure 3.** MER driving mode usage as of 15 August 2005, counting 573 sols for Spirit and 555 sols for Opportunity. (Distances are as measured onboard, and can be overestimates of actual distance traveled if wheels slip during non-Visodom drives. Only rover translation distances are reflected here; turn-in-place and single-wheel trench digging motions are not reported.)

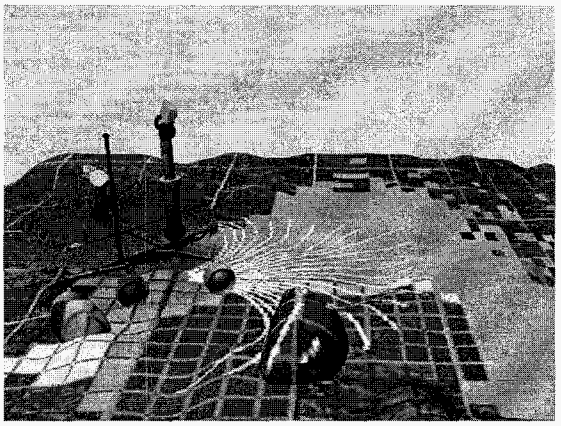


**Figure 4.** Illustration of Terrain Assessment and Path Selection. Red cells indicate unsafe areas around the large rock, yellow cells indicate traversable but rougher areas around the smaller rock, and green cells indicate safe and flat areas.

and a weighted evaluation of the constituent cells is assigned to each path. This results in a set of *Obstacle* path evaluations; low values indicate a less traversable path, high values indicate a more traversable path.

In *Path Selection*, the rover is given autonomous control over its drive direction (in contrast to *Directed Driving*, in which it follows a single pre-commanded arc path). To select between multiple paths, the current X,Y goal (and current rover Z) is reprojected from world frame into rover frame, and votes are assigned to all possible candidate paths according to how effectively each path would drive the rover toward its goal point. The path that would lead directly toward the goal is given the highest evaluation, other paths are assigned lesser values according to a Gaussian distribution. The variance of each distribution is configurable, nominally 3.2 curvature units for arcs, 97 degrees for point turns.

There are also two off-nominal behaviors available: backups and limit cycle turns. Goals that lie inside the tightest turning circle (corresponding to an arc with 1 meter radius) cannot be reached using arcs, so in those situations a backup arc is selected as the preferred arc path. When backing up, the rover does a "K-turn", driving in its tightest arc away from the goal location. As of February 2005, when a backup behavior is chosen the highest path evaluation is halved, which in practice causes a point turn to be selected instead of an arc. The result of this processing is a set of *Waypoint* evaluations, which are merged with the Obstacle path evaluations and result in *Hazard Avoidance* capability, if Terrain Assessment is also enabled.

The second off-nominal behavior detects when the rover fails to make progress after multiple steps. This will happen if Visodom detects so much slip that little progress is made, or if the path to the goal is blocked by too large an obstacle and the rover keeps trying unsuccessfully to drive around it. In either case, the *Limit Cycle Check* will detect the condition and either terminate the command or force the rover to turn in place some amount to get a new heading, from which it might find a better path around the obstacle.

In *Pose Update* (or *Visual Odometry*), the rover updates its current position and/or attitude by comparing features found in stereo image pairs taken before and after a small motion step. Both MER vehicles use NAVCAMs for this Visual Odometry (or Visodom) processing, since the scale changes induced by even small motions in the wide FOV HAZCAM images make autonomous tracking of features difficult. This processing only converges successfully if the imaged terrain has a sufficient number of features, so the human rover driver must ensure that NAVCAMs are pointed toward useful features anytime Visodom is used.

*Terrain Assessment*

When performing Terrain Assessment, each MER vehicle is able to find geometric hazards in the nearby terrain

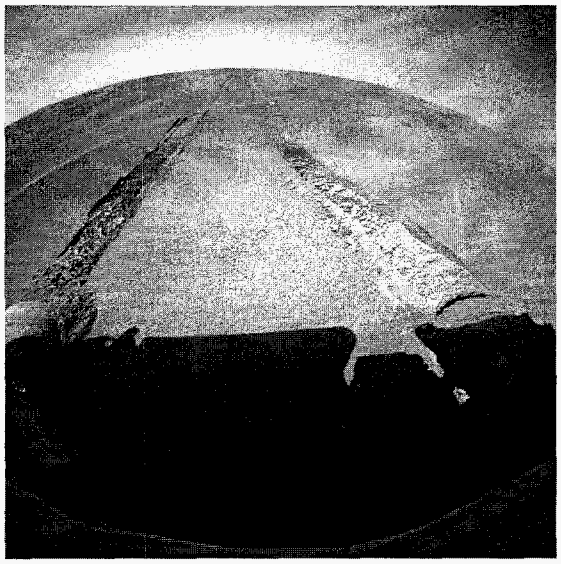


**Figure 5.** Front HAZCAM showing a non-geometric hazard encountered by Opportunity on Sol 446: 50 meters of commanded driving resulted in only 2 meters progress, which you can see in the tracks leading up to Purgatory Ripple, in which Opportunity was mired for over a month. Although this was a "blind" or Directed drive, Opportunity might have gotten stuck even if GESTALT had been turned on because there were no geometric hazards to avoid, just loose terrain. Fortunately, Visual Odometry has been used ever since as a Slip Check to ensure we never command more than 5 meters of driving without a guarantee of motion; this new safety check helped out on Sol 603, when Visodom measured 44% slip while climbing a similar dune, and stopped the drive before getting further bogged down. Opportunity was able to back out easily during its next drive.

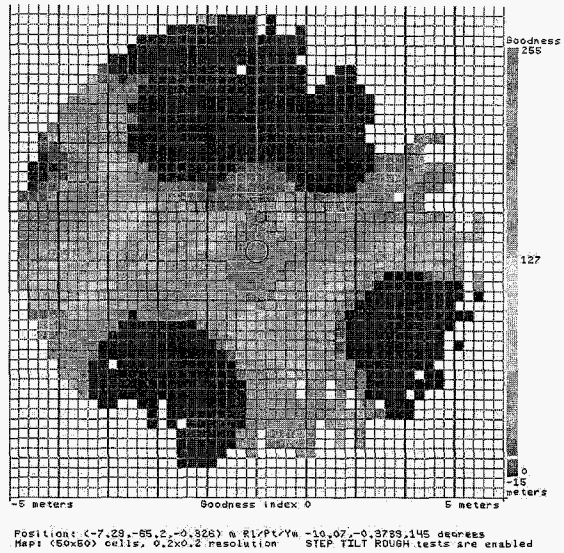



**Figure 6.** Illustration of Goodness Map from Spirit's Sol 107. Red areas indicate obstacles (typically a rock taller than 20cm at the center of the red blob), yellow/orange indicate traversable areas. Only information within a 5-meter radius is maintained in the onboard map.

*before* it ever drives there. Only geometric hazards are found; no assessment of the stability of the terrain is performed onboard (see Figure 5 for an example of why that would be useful). A hazard is defined as an object large enough to potentially high-center the vehicle, either on the WEB or the rocker/bogie suspension. MER wheels are 25cm in diameter, and on flat ground the belly of the WEB is 29cm high. Considering that the vehicle might tilt as much as 35 degrees, and allowing for measurement error in the stereo data, the parameter describing the maximum height of a still-traversable obstacle was set to 20cm for nominal driving.

The MER vehicles' hazard detection software models the world as a 2D grid. The size of the grid cells is chosen to match the mechanically-determined obstacle size, 20cm. Each cell stores a continuously-varying scalar "goodness" value that represents how safe the rover would be if its center were located on that cell. This results in a *configuration space* world map, because it describes safe vehicle configurations, not precise object boundaries. For example, a lone 20cm rock is not represented as a single "unsafe" grid cell. Instead, all cells in a rover-sized disc centered over the rock will be marked as unsafe (as in Figure 4, because having the rock anywhere under the rover would be unsafe; that is, all vehicle configurations (locations in the world) that include that rock *anywhere* underneath the rover must be declared unsafe. Finding a safe path to drive is thus reduced to finding only a 1-cell-wide path through the goodness map.

Although the software supports orientation-dependent goodness assessment, which might make it possible to navigate gaps narrower than its solar panels, the processing overhead required to support it (a separate goodness map for each orientation) would have slowed down the autonomous driving to an unreasonable rate. Instead, terrain assessment is performing by modeling the rover by its turning circle as a 2.6 meter diameter disc. This design also gives us a useful invariant: *the autonomy system will only drive the rover into areas where it can safely turn in place.*

Our analysis of the terrain comes from local plane fitting of the stereo-computed range data. The rover's assessment of the terrain is based on how much each rover-sized patch of ground differs from a flat plane. To analyze a given rover-sized patch of terrain, all XYZ points falling within that patch are fit to a plane. The parameters of that plane fit are used to assess how safe the rover would be if it were there, in any orientation. A continuously-varying "goodness" is independently computed in each of several filters: the Step, Tilt, and Roughness filters. The final evaluation of a cell's goodness is the most con-

servative one found by any filter. Descriptions of these filters are given below.

No terrain assessment can be done unless geometric information about the terrain has been measured. On MER, stereo cameras provide these measurements.

*Robust Stereo Image Processing*

MER rovers use passive stereo image processing to measure geometric information about nearby terrain. Stereo vision is an attractive technology for planetary exploration because it has low power requirements and nominally requires no moving parts. JPL's stereo vision software has a long history [16], [25], especially in real robotic systems (Mars rover research [13], [22], [21], Athena [3], Unmanned Ground Vehicles [14], Pioneer [11], Urbie [15], and Perceptor [2], [18]).

The Sojourner rover demonstrated the first use of autonomous stereo triangulation on a planetary rover [17]. But the Sojourner system relied on active projection of 5 laser stripes, and only found at most 20 XYZ points from each pair of stereo images. In contrast, MER uses passive stereo vision, relying on sunlight to illuminate the terrain. And by virtue of their faster processor (20 MHz compared to 0.1 MHz on Sojourner) and new software, MER vehicles compute many more point measurements: Opportunity has measured an average of 48,000 XYZ points in each of 69 NAVCAM image pairs as of sol 322, and Spirit has measured an average of 15,000 XYZ points in each of 1687 HAZCAM image pairs as of sol 342.

The general algorithm is described in [8], but we summarize the MER implementation here. Images can be acquired using any of the available stereo pairs, typically the front and rear HAZCAMs on Spirit, and the NAVCAMs on Opportunity. The raw data is read at 1024x256 resolution and software-downsampled to 256x256 resolution. Images are then rectified (i.e., resampled) to eliminate any lens distortion using fixed CAHVOR (radial distortion) or CAHVORE (fisheye) camera models [7]. Images are then correlated on a scanline-by-scanline basis using a 7x7 square pixel subwindow and the Sum of Absolute Difference (SAD) metric. SAD scores are generated at all candidate integer disparities at the nominal 256x256 resolution. A variety of consistency checks is applied to the resulting correlation curve, and only those curves that exhibit a unique minimum are accepted as valid. Each valid integer disparity is then mapped to a 3D point using the camera model. The final result of stereo image processing is thus an image whose pixel values are either Unknown, or the 3D location of the feature at that pixel.

Although this algorithm has been used successfully in research settings, several enhancements were made to improve its robustness for Mars operations. In particular, several filters were added to the already existing set of Left/Right Line of Sight (LRLOS), Blob, Border, and Curvature Threshold filters described in [8].

*Gamma Correction* To better enhance the detail of terrain hidden by shadow from the rover and nearby rocks, we apply gamma correction to the raw images.

*Flat Filter* Any curve whose minimum is not unique (i.e. shares the same value with *either* its neighbors) will be rejected.

*Overhang Filter* Purely horizontal features cause problems for scanline-based stereo processing. For example, in our testbed environment, dark horizontal power strips along a white wall appeared to the rover like spears pointing down at the cameras. Since our nominal driving environment has nothing hanging down from overhead, we filter out any range values that appear to loom back toward the rover. Any pixel whose range value is some percentage closer than those below it in the image is eliminated. The filter has proven useful in flight by occasionally eliminating noisy range data beyond the nominal 5 meter terrain assessment distance.

*Fixed Vehicle Mask* The 120 degree field of view of the HAZCAM optics is so large that some parts of the vehicle appear in each image. These need to be eliminated before terrain processing can occur; it wouldn't do for the rover to be afraid of its own arm or wheel, for instance. Those vehicle parts that always appear in the same place in the image (e.g., the stowed IDD, cabling mounted near the cameras, the solar panel above the rear HAZCAMs) are eliminated from consideration using a fixed pixel-based mask.

*Moving Vehicle Mask* Unfortunately, the fixed vehicle mask is not enough to eliminate all parts of the rover from the HAZCAM FOV. In particular, the wheels can move freely throughout the field of view, so they are correlated along with the terrain and are filtered out of the *range* image, rather than the original image. A filter comprising a swept cylindrical volume reflecting the range of motion of the rocker-bogie system and steering motor limits is applied to all range data; any points lying in that volume are eliminated.

*Terrain Assessment*

The MER vehicles use the Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) system for terrain assessment [8]. Inspired by Carnegie Mellon's Morphin algorithm [20], [24], this system uses 3D geometric information to estimate how safe the rover would be at each point in the world. Originally demonstrated on the Athena Rover prototype [3], it is now being evaluated on testbed vehicles for potential use in

the upcoming MSL mission as well [23].

A central idea in this implementation is that *new data always overrides old data.* In particular, no matter how safe or unsafe a given cell had been considered, whatever the latest range data shows should be taken as true. This is a useful way to compensate for two problems. First, the range data computed for objects far from the rover is less accurate than nearby range data; by overlaying new data we eliminate any misevaluations that occurred because of inaccuracies in the range data. Second, overlaying of new data makes the rover's ability to navigate more robust to any position estimation errors, since it always prefers the newest 3D information over any potentially misregistered earlier data.

The actual assessment is computed from the stereo data associated with each rover-sized patch independently. The parameters of the best-fit plane are found for each rover-sized patch, and are used to compute the following filters.

*Tilt Filter* The surface normal of each planar patch is compared against the maximum allowed vehicle tilt. Patches with tilt between 0 and some minimum tilt are given a perfect goodness value (255); tilts above the maximum allowed are obstacles (goodness value of 0); those in between are weighted linearly.

*Roughness Filter* The residual from the planar fit in the direction of the surface normal provides a measure of how "rough" or uneven the region is.

*Step Filter* Once the planar parameters for a rover-sized disc have been found, the "local elevation" (distance from each constituent cell's mean elevation to that plane) is computed. A filter using a smaller WEB-sized disc is run over the rover patch, computing the difference of the min and max local elevations within each smaller disc; this is the largest step found within that smaller disc. Finally, the largest step found anywhere in the larger rover-size disc is compared against the maximum allowed obstacle size. Patches with small max step size are given a perfect goodness value (255); steps above the maximum allowed are obstacles (goodness value of 0); those in between are weighted linearly.

*Robust Terrain Assessment*— Several key ideas have gone into making GESTALT more robust.

*Layered Goodness Maps*

The field of view and range resolution of 256x256 HAZCAM images are just barely large enough to generate one complete rover-sized footprint. Hence all terrain assessment performed near the edges of the field of view is *not* based on a complete disc, but rather a partial disc. That means that when an obstacle leaves the field of view, all future terrain analysis would fail to take it into account. This causes the configuration space analysis to break down; placing the center of the rover near the edge of the visible field of view might seem perfectly safe, because the partial 3D data does not include the obstacle. As a result, in early implementations of the software the rover would start driving around an obstacle, then steer back directly into it.

To address this concern, we introduced a multi-scale representation of the world: Layered Goodness Maps. Instead of processing the stereo-derived goodness directly, we first "shrink" obstacles blobs and merge them into each layer separately, then "grow" obstacles from each layer back to their original size, and finally use the most conservative assessment of goodness found in any layer as our estimate of the terrain safely. The goodness map computed from the latest stereo data is filtered into multiple layers using discs varying from a 1 cell radius (layer 6) up through rover-sized discs with a 7 cell radius (layer 0). The net effect is to shrink any obstacle blobs by (6 - layer numbers) cells. Low individual goodness values in layer 0 often represent "point" obstacles like individual large rocks; low values found only in higher layers represent "dispersed" obstacles like a 10cm rock 1 meter away from a 10cm ditch.

*Partial Range Data - Clipping*

The overlaying of new data can cause cells on the edge of the field of view to be misevaluated. If only a small piece of an obstacle is visible, the range data representing it will not be classified as a hazard. The percentage of obstacle range data in the cell at the edge of the FOV drops, and the cell is mistakenly evaluated as safe. We address this by being more conservative in replacing goodness values at the edge of the field of view. Instead of simply overlaying new range data everywhere in the map, around the edges of the field of view we take the most conservative of the current and previous values.

*Pitchers Mounds*

An early implementation of the Step Filter simply compared mean elevations at adjacent cells. Field testing revealed that this approach was not conservative enough, because the prototype rover high-centered on a pyramid-shaped obstacle. The next implementation extended the search for step differences to anywhere within the rover-sized (2.6 meter diameter) patch. But this was found to result in too-conservative assessments of flat/hill transitions, even on low "pitcher's mound" hills. Restricting attention to the smaller WEB-sized disc still preserves vehicle safety; anything that constitutes a hazard for the solar panels will still be detected. But now the rover can climb pitcher's mounds again.

*Tall Thin Obstacles*

A tall, thin obstacle will not occupy enough of a grid cell to make a meaningful change to the mean elevation at that cell. But it still must be avoided. Having found this problem during field testing, the Step filter was updated to not just the mean elevation within a cell, but the mean plus some number of standard deviations.

*Autonomous Goal Selection*

MER vehicles are nominally commanded toward a precise, metrically specified goal in rover frame (e.g., 5 meters forward) or as X,Y coordinates in world frame (e.g., 2 meters north and 1.1 meters east of the current Site Frame origin). There is some amount of autonomy in deciding when to step (e.g., a radial tolerance in meters around each X,Y goal, use of the onboard IMU during turns in place to determine when the desired heading has been reached), but most commands require human drivers to decide in advance where the rover needs to go.

MER vehicles also provide a limited means of autonomous goal selection using the TURN_TO_A_ROCK command. Unlike the Autonav driving modes which force the rover to drive around obstacles, this command causes the vehicle to turn *toward* the nearest obstacle. Although a wedge of yaw values may be used to constrain the area of terrain that will be searched, there is no way to ensure that this command will lock into any particular rock, it will simply find the one nearest the rover; it turns toward *a* rock, not *the* rock. Obstacles are found using certain data computed during the onboard traversability analysis: the goodness map and an "elevation max count" map. This latter map stores the number of times a given cell was found to have the highest mean elevation in all rover-sized patches that contain it.

Originally envisaged as a means of compensating for slip by continually servoing on a feature of the terrain, this command has seen little use on MER other than during engineering tests. The MER position estimation system (including Visual Odometry) performed so well during the primary and extended missions that it was typically not necessary to rely on this capability for re-pointing. Even when it might have been useful, other considerations dictated that precise pre-planned motions be used instead, e.g., feature topology not representable as an obstacle, the uncertainty in how much time and power would be needed to complete the drive, the vehicle orientation needed to optimize communication, the need to keep the Instrument Deployment Device (IDD, or rover arm) work volume free of obstacles.

## 6. Deployment Results

Both MER vehicles have lasted far beyond their design lifetime of 90 days; as of this writing, each has run for more than 650 days on the surface of Mars in geologically diverse areas. Papers describing Spirit's drive toward the Columbia Hills [10], Opportunity's exploration of Endurance Crater and its multi-kilometer trek southward [4], general tradeoffs considered in choosing between the various driving modes [5], and a systems-level view of the onboard autonomy [12] describe the system's performance in detail.

One of the MER software design principles was to implement only what was necessary for the mission; the more parameters and conditionals, the more code branches would need to be tested. However, uncertainty in what terrain would be encountered necessitated keeping the onboard terrain assessment software flexible: for instance, the February 2005 version of software has 292 surface navigation-specific parameters. This flexibility proved crucial for successful operation on Mars. For example, in the original design stereo HAZCAM images were processed at 128x128 pixel resolution. This was sufficient for good results in our indoor testbed environment. However, Earth-based outdoor field tests suggested that 256x256 resolution resulted in better results at the outer range of stereo processing, 4–5 meters. This same behavior was observed in the initial images from Spirit, and as a result Spirit was reconfigured to use higher resolution stereo.

*Spirit Rover: Gusev Crater*

Spirit landed on 3 January 2004 about 2 kilometers west of the Columbia Hills inside Gusev Crater. Unlike Opportunity, which ultimately found evidence for water just 9 meters from its landing site, Spirit had to drive over 3 kilometers before finding convincing evidence starting at the base of the Columbia Hills. Having promised to deliver only 600 meters of traverse, the hills seemed unreachable at first. But the combination of human terrain assessment and onboard navigational autonomy enabled Spirit to reach the base of the hills by June 2004.

Spirit's driving began as a run for the hills (shown in the daily drive distances for sols 90–160 in Figure 7), but was followed by a year of exploration on the slopes of the Columbia Hills (see Figure 8). Driving on the slopes led to new styles of driving, with an emphasis on using Visual Odometry software to detect and/or compensate for high slip, commanded use of explicitly-sequenced keep-out zones to avoid known obstacles, and novel uses of the low-level driving code to compensate for temporary problems (e.g., a rock stuck in a wheel, dragging one wheel to extend drive motor lifetime). The success of its mission owes much to the robustness and flexibility not

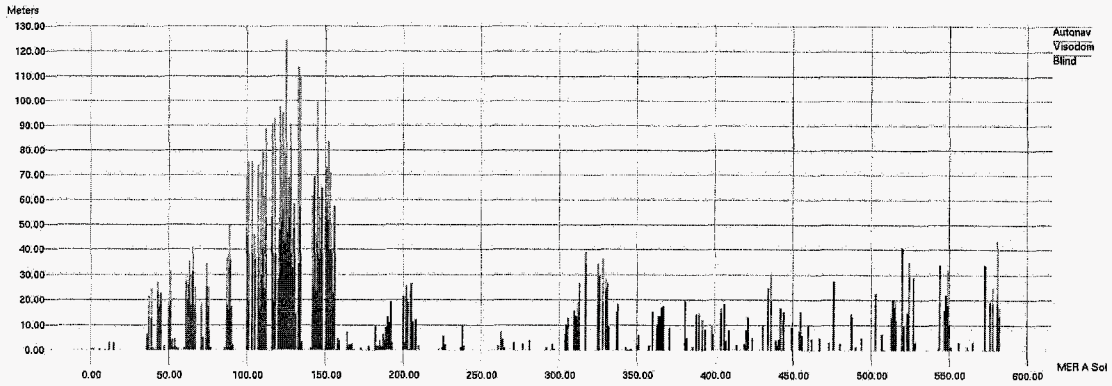



**Figure 7**. Plot of Spirit's complete drive history per sol through Sol 588; the most driving in one sol was 124 meters on Sol 125, the longest contiguous autonomous drive was 78 meters on sol 133. Note that the vertical scale here is stretched compared to Figure 10. Red indicates blind driving, green indicates autonav (rows 5–8 in Table 3), blue indicates Visodom (rows 2, 4 in Table 3).

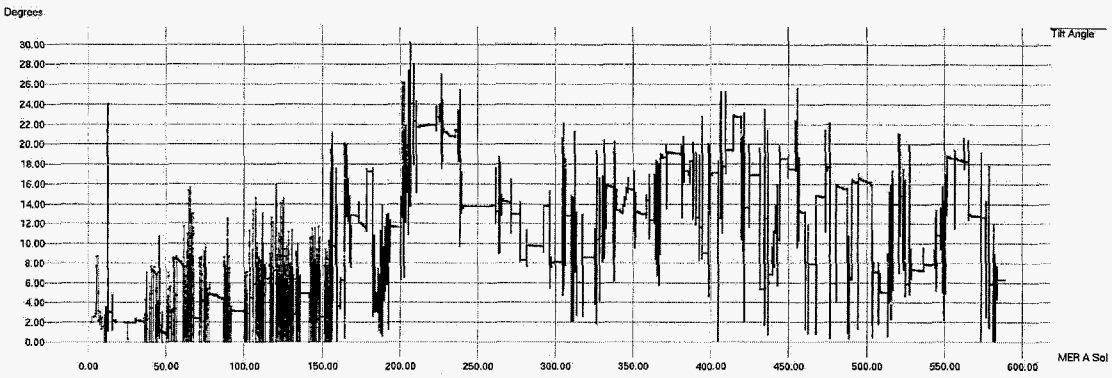



**Figure 8**. Plot of Spirit complete tilt history through Sol 588.

only of the onboard mobility software, but also of the human ground operations team who kept creating new and better ways to drive as explained in [10].

*Opportunity Rover: Meridiani Planum*

Opportunity landed on 24 January 2004 inside Eagle Crater at Meridiani Planum. Scientists were thrilled when the first images revealed bedrock outcrops a mere 9 meters from the rover, but engineers were horrified to see very little 3D information recovered from the first stereo images

Flexibility of software design was even more critical for Opportunity. The fine-grained particles covering the majority of Eagle crater were unresolvable by stereo image processing of the 1-bit-per-pixel compressed 1024x1024 45 degree FOV NAVCAM images. Fortunately, NAVCAM images with more bits per pixel were acquired quickly and proved easy to process. But at the nominal size used by onboard autonomy software (256x256 squared pixels), even 8 bits per pixel were insufficient to resolve the soil in 120 degree FOV HAZCAM images. This meant that the autonomy system, tested almost entirely using rigidly-mounted HAZCAMs, had to be reconfigured to use steerable mast-mounted NAVCAMs. Fortunately, we were able to reconfigure the mobility software to adapt to these conditions, and the uplink of new software versions made them even more capable and more easily commanded. Table 2 shows some of the differences between the two vehicles that have persisted across the mission, and that can be changed daily as the terrain requires.

Although the rovers' design requirements had been for traverse at tilts less than 15 degrees (see the large daily drive distances in Figure 10), Opportunity spent more than six months in Endurance crater with tilt *above* 15 degrees, as shown in Figure 11.

*Software Versions*

There have been three versions of the mobility flight software used on Mars as of October 2005.

The version used during the first 90 days of operation on each rover (the Primary Mission) had autonomy software that was overly conservative. It required that imaging and terrain processing occur after each autonomously-

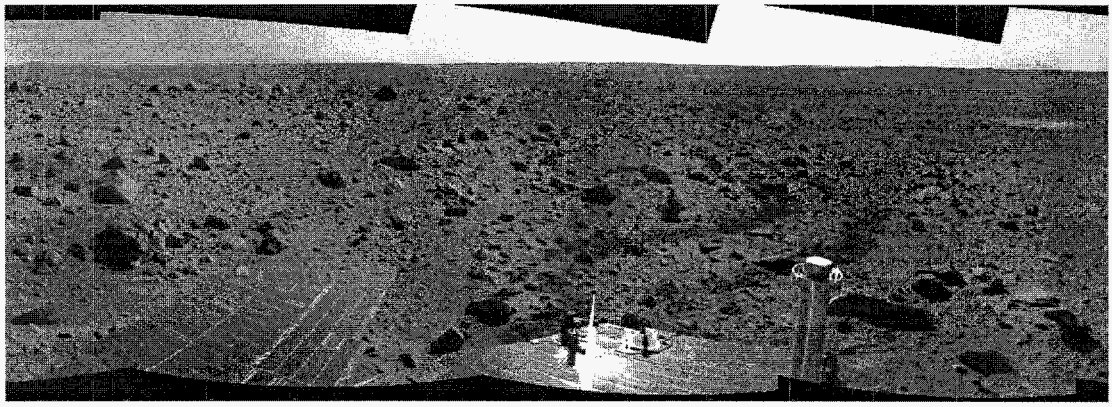


**Figure 9**. Image of Spirit's tracks following an obstacle avoidance maneuver on Sol 107.

| Parameter | Spirit | Opportunity |
|---|---|---|
| Autonav camera pair | HAZCAM (front or rear) | NAVCAM |
| Stereo processing resolution | 256×256 | 256×256 |
| Useful (unmasked) stereo image size | 184×256 (front) 108×216 (rear) | 256×256 |
| Stereo search range | 0.3 - 5.0 $m$ | 1.2 - 10.0 $m$ |
| Grid cell resolution | 0.2×0.2 $m^2$ | 0.2×0.2 $m^2$ |
| Grid size | 10×10 $m^2$ | 12×12 $m^2$ |
| *As needed for daily operations* | | |
| Max distance between image acquisitions | 0.5 – 1.5 $m$ | 0.5 – 2 $m$ |
| Engineering diagnostic verbosity | 0 – 4 | 0 – 4 |
| Max allowed Visodom failures | 0 – 15 | 0 – 20 |
| Number of Point Turns to enable | 0 – 46 | 0 – 46 |

**Table 2**. Some parameter differences between Spirit and Opportunity

commanded motion, and also checked for step obstacles everywhere within each rover-sized disc (2.6m diameter). In practice this meant that the rovers were unable to autonomously climb over small "pitcher's mound"-like hills; this occurred several times on Spirit.

In early April 2004, a new version of software was uplinked to both rovers, initiating the Extended Mission. This version incorporated several robustness enhancements made to the autonomy software during prelanding outdoor field tests, as well as some lessons learned during the primary mission. The most dramatic changes included the ability to skip terrain assessment when the existing data were sufficient to ensure that all paths are safe, and the ability to autonomously traverse small hills. This enabled the much longer drives seen during sols 90–160 in Figure 7.

The next software upgrade was made in February 2005 to help streamline commanding and data analysis, improve position estimation in highly sloped terrain, and enable terrain assessment using more than only two mast-mounted Navcam images. This upgrade also included the first changes made to the low-level mobility software since 2002: use of look-up tables to generalize the steering angles used by primitive commands, use of look-up tables to estimate slip based on rover attitude, and the ability to alternate between wheel dragging and non-dragging modes in the event of a disabled drive motor. Prior to this release, rover drivers had to explicitly command each step of a Visodom-enabled drive to ensure that there would be sufficient overlap between images (nominally 60%). But now the Goto Waypoint command could be configured to restrict autonomously-commanded motions to ensure sufficient overlap, assuming the actual heading change is no larger than what was commanded.

A fourth upgrade, planned for mid-2006, might incorporate several technologies now being evaluated. These include autonomous in situ instrument placement following a successful drive (aka Go and Touch), global path planning to enable intelligent backtracking, visual servoing, and autonomous detection of dust devils and clouds in onboard imagery. Other planned enhancements include explicit keepout zones, speed optimizations to Visodom and Autonav, and new commands intended to reduce the complexity of commanded sequences.

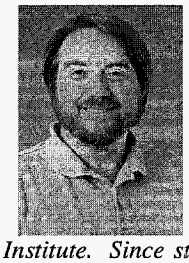
## 7. CONCLUSION

The MER Mobility Software has enabled an unprecedented amount of driving across the surface of another world. The flexibility of its design enabled it to function effectively in two dramatically different terrains.

## 8. ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Thousands of people were involved in the mission at JPL and throughout the world, in government, academia, and industry. We gratefully acknowledge their outstanding work, which enabled us to explore Mars through Spirit and Opportunity. We would like to thank the entire Mars Exploration Rover team, and in particular the mobility working group and test team, including Paolo Bellutta, Diana Darus, Jeff Favretto, David Henriquez, Robert Liebersbach, David Lokshin, Carolina Maldonado, Rich Petras, Aaron Simo, Eddie Tunstel, Hung Vo, Rick Welch, and Reg Willson.

***Jeffrey Biesiadecki*** *has been a software engineer at NASA's Jet Propulsion Laboratory since 1993, after completing his Master's degree in Computer Science at the University of Illinois, Urbana-Champaign. He designed and implemented the core motor control and non-autonomous mobility flight software for the Mars Exploration Rovers, and is also one of the rover drivers for the Mars Exploration Rover "Opportunity", responsible for command sequences that tell the rover where to drive and how to operate its robotic arm on the surface of Mars.*



***Dr. Mark Maimone*** *is a Navigation and Machine Vision researcher at the Jet Propulsion Laboratory. He earned his Ph.D. in Computer Science from the Computer Science Department of Carnegie Mellon University in 1996, and was then a Post-doctoral Research Associate at Carnegie Mellon's Robotics Institute. Since starting at JPL in 1997, he has worked on the several Mars Rover research projects and a vision system for inspection of the Chornobyl reactor. As a member of the 2003 Mars Exploration Rover flight software team, Mark developed the vision and navigation subsystems for the MER vehicles. Mark is now part of the MER ground operations team and is developing the autonomous navigation software for the Mars Science Laboratory rover, NASA's next Mars rover mission.*

## REFERENCES

[1] Khaled S. Ali, C Anthony Vanelli, Jeffery J. Biesiadecki, Mark W. Maimone, Yang Cheng, Miguel San Martin, and James W. Alexander. Attitude and position esimation on the Mars Exploration Rovers. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[2] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for demo iii. In *Proc. Intelligent Vehicles Symposium*, Dearborn, MI, October 2000.

[3] Jeffrey Biesiadecki, Mark Maimone, and Jack Morrison. The Athena SDM rover: A testbed for Mars rover mobility. In *International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, Montreal, Canada, June 2001. http://robotics.jpl.nasa.gov/people/mwm/sdm-mobility/.

[4] Jeffrey J. Biesiadecki, Eric T. Baumgartner, Robert G. Bonitz, Brian K. Cooper, Frank R. Hartman, P. Christopher Leger, Mark W. Maimone, Scott A. Maxwell, Ashitey Trebi-Ollenu, Edward W. Tunstel, and John R. Wright. Mars Exploration Rover surface operations: Driving opportunity at meridiani planum. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[5] Jeffrey J. Biesiadecki, Chris Leger, and Mark W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. In *International Symposium of Robotics Research*, San Francisco, CA, USA, October 2005.

[6] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the Mars Exploration Rovers. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[7] Donald B. Gennery. *Calibration and Orientation of Cameras in Computer Vision*, chapter Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points, pages 123–136. Springer Verlag (A. Gruen and T. Huang, ed.), 2001.

[8] Steven B. Goldberg, Mark W. Maimone, and Larry Matthies. Stereo vision and rover navigation software for planetary exploration. In *IEEE Aerospace Conference*, volume 5, pages 2025–2036, Big Sky, Montana, USA, March 2002. http://robotics.jpl.nasa.gov/people/mwm/visnavsw/.

[9] 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.

[10] Chris Leger, Ashitey Trebi-Ollenu, John Wright, Scott Maxwell, Bob Bonitz, Jeff Biesiadecki, Frank Hartman,

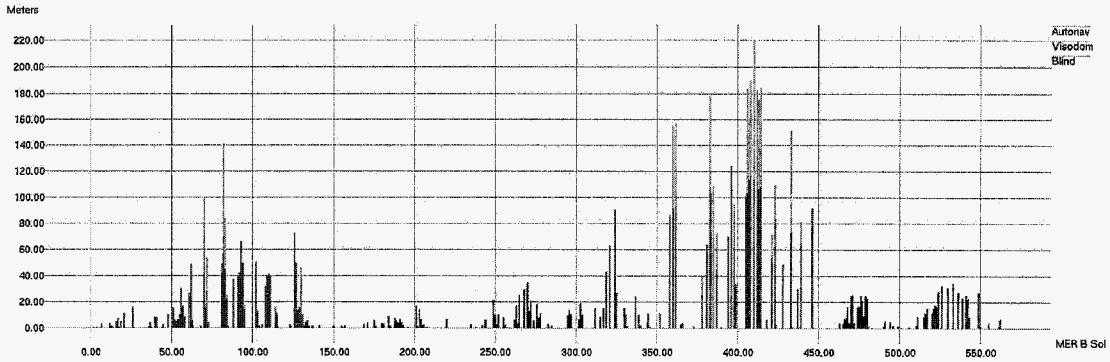



**Figure 10**. Plot of Opportunity's complete drive history per sol through Sol 567; the most driving in one sol was 219 meters on Sol 410, the longest contiguous autonomous drive was 280 meters during sols 383–385. Note that the vertical scale here is more compressed than that in Figure 7. Red indicates blind driving, green indicates autonav (rows 5–8 in Table 3), blue indicates Visodom (rows 2, 4 in Table 3).

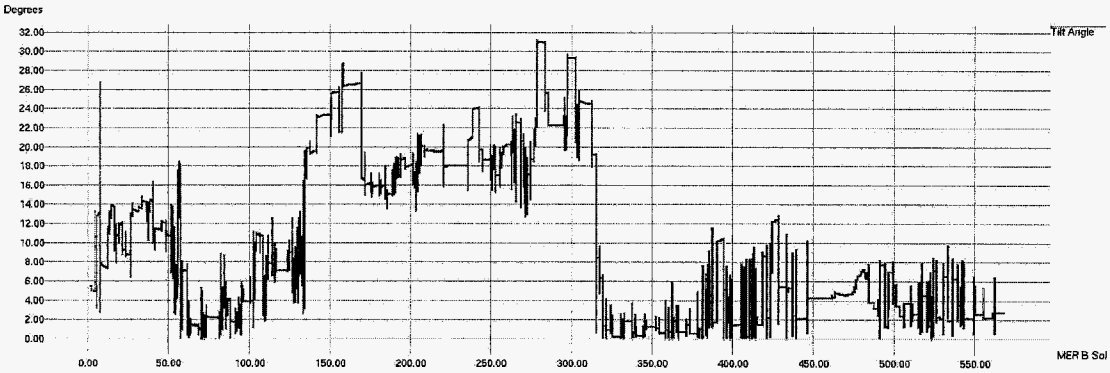



**Figure 11**. Plot of Opportunity's complete tilt history through Sol 567.


Brian Cooper, Eric Baumgartner, and Mark Maimone. Mars Exploration Rover surface operations: Driving spirit at gusev crater. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[11] M. Maimone, L. Matthies, J. Osborn, E. Rollins, J. Teza, and S. Thayer. A photo-realistic 3-D mapping system for extreme nuclear environments: Chornobyl. In *International Robotics and Systems Conference (IROS)*, pages 1521–1527, Victora B.C., Canada, October 1998. http://robotics.jpl.nasa.gov/people/mwm/pioneer/iros98/.

[12] Mark Maimone, Jeffrey Biesiadecki, Edward Tunstel, Yang Cheng, and Chris Leger. *Surface navigation and mobility intelligenece on the Mars Exploration Rovers*, chapter 3. TSI Press, Albuquerque, NM, USA, 2006. http://www.intelligentspacerobotics.com/.

[13] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous Robots Journal, Special Issue on Autonomous Vehicle for Planetary Exploration*, 2(4), 1995.

[14] L. Matthies, A. Kelly, T. Litwin, and G. Tharp. *Obstacle Detection for Unmanned Ground Vehicles: A Progress Report*. Springer-Verlag, 1996.

[15] 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.

[16] L. H. Matthies. Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. *International Journal of Computer Vision*, 8(1):71–91, July 1992.

[17] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, and B. Wilcox. Experiences with operations and autonomy of the mars pathfinder microrover. In *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass at Aspen, Colorado, March 1998. http://robotics.jpl.nasa.gov/people/mishkin/papers/IEEE_aerospace98.pdf.

[18] A. Rankin, C. Bergh, S. Goldberg, and L. Matthies. Passive perception system for day/night autonomous off-road navigation. In *SPIE UGV Symposium*, Orlando, FL, April 2005.

[19] Glenn E. Reeves and Joseph F. Snyder. An overview of the Mars Exploration Rovers flight software. In *IEEE Conference on Systems, Man and Cybernetics*, The Big Island, Hawaii, USA, October 2005.

[20] 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. http://www.cs.cmu.edu/~reids/papers/AIAAobsAvoid.pdf.

[21] 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. http://www.frc.ri.cmu.edu/projects/mars/ publications/global_local_icra2000.ps.gz.

[22] Richard Volpe. Navigation results from desert field tests of the Rocky 7 Mars rover prototype. *International Journal of Robotics Research*, 18(7):669–683, Special Issue on Field and Service Robots, July 1999. http://robotics.jpl.nasa.gov/people/volpe/papers/JnavMay.pdf.

[23] Richard Volpe. Rover functional autonomy development for the Mars mobile science laboratory. In *IEEE Aerospace Conference*, Big Sky, Montana, March 2003.

[24] 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. http://robotics.jpl.nasa.gov/people/mwm/papers/98ras.nomad.pdf.

[25] 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.