The Best of Both Worlds: Integrating Textual and Visual Command Interfaces for Mars Rover Operations

Scott A. Maxwell
Mars Exploration Rover Project
Jet Propulsion Laboratory
M/S 126-114 4800 Oak Grove Dr.
Pasadena, CA, USA 91109
marsroverdriver@yahoo.com

Brian Cooper, Frank Hartman, John Wright, and Jeng Yen
Mars Exploration Rover Project
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109

Abstract - *A Mars rover is a complex system, and driving one is a complex endeavor. Rover drivers must be intimately familiar with the hardware and software of the mobility system and of the robotic arm. They must rapidly assess threats in the terrain, then creatively combine their knowledge of the vehicle and its environment to achieve each day's science and engineering objectives.*

To help the Mars Exploration Rover project's rover drivers meet their goals, the software we developed to drive the rovers -- the Rover Sequencing and Visualization Program, or RSVP -- combines two representations of command sequences, one textual and one using three-dimensional graphics. Changes to one representation are instantly reflected in the other, and the combination's advantages exceed the mere sum of the parts: the different representations offer different levels of abstraction, engage different areas of the driver's brain, and complement each other's strengths. This combination plays a crucial role in simplifying a complex feat of interplanetary exploration.

Keywords: Mars Exploration Rovers, RSVP

1 Introduction

The Mars Exploration Rover (MER) project landed twin six-wheeled robotic geologists on Mars in January 2004. Since then, it has been the responsibility of the Earth-based rover drivers -- a group of about ten members of the engineering team -- to write, test, and validate the command sequences that drive each vehicle to new locations and make use of its robotic arm. Because of the lengthy light-time delays between the two planets, it's infeasible to command the rover in real time. Instead, rover drivers must prepare the commands for an entire sol (Martian day), review them, and then uplink them to the rover, which executes them over the course of the sol.

These command sequences are complex. On a typical day, the rover drivers prepare more than 100 commands -- sometimes up to 500 commands -- for each rover. Think of it as writing hundreds of lines of assembler code every day, in an environment where a single bug could cost up to four hundred million dollars (the approximate total cost of one rover). Add in conditionality (the rovers have "IF" commands) and the inherent uncertainty of operating a remote vehicle in an unknown environment -- a literally *alien* environment -- and you begin to appreciate the scope of the task [1, 2].

We recognized early in the project's life cycle that the rover drivers would need fast, accurate, and powerful tools. Indeed, this was an important lesson learned from the only previous Mars rover mission, JPL's 1997 Mars Pathfinder mission. To meet the rover drivers' needs, we developed state-of-the art rover-driving software: RSVP, the Rover Sequencing and Visualization Program (see Figure 1). A crucial feature of RSVP's design was the inclusion of interlinked text-based and graphics-based views of the command sequences.

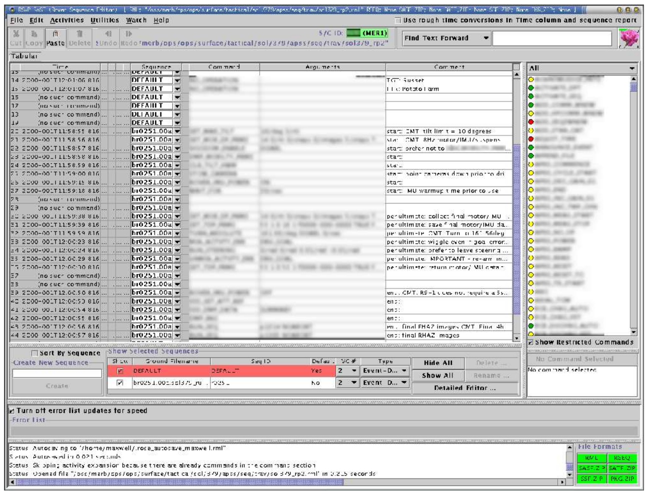
Figure 1. Rover drivers Frank Hartman (foreground) and Brian Cooper, hard at work using RSVP

2 Text-based Commanding

The text-based component of RSVP is called RoSE, the Rover Sequence Editor. RoSE's primary role is to provide a tabular view of the entire command sequence (or group of sequences). RoSE does not attempt to abstract the view of the command sequence in any meaningful way -- indeed, that's the point. In RoSE, every command, and every command argument, is visible to and editable by the user -- from low-level heater-control commands, to high-level "drive autonomously to this position" commands.

Figure 2 shows a view of RoSE editing a recent command sequence. Certain information is blurred due to legal restrictions, but its basic layout should be apparent: it is dominated by a table of commands, arguments, and comments on the commands. Within each sequence, the commands are listed in timeline order, according to the most recent modeling results.

In brief, RoSE is a left-brain tool, focused on logic and detail. In RoSE, sequences are composed of words and numbers.



RSVP RoSE (Rover Sequence Editor)

File Edit Activities Utilities Watch Help

Use rough time conversions in Time column and sequence report

Cut Copy Paste Delete Undo Redo "merb/ops/ops/surface/tactical/sol/379/apss/seq/trav/sol379_rp2"

S/C ID: (MER1)

Find Text Forward

Tabular

	Time	Sequence	Command	Arguments	Comment
14	2000-001T12:01:06.816	DEFAULT	[illegible]		TGT: Russet
15	2000-001T12:01:07.816	DEFAULT	[illegible]		FTC: Potato Farm
16	(no such command)	DEFAULT			
17	(no such command)	DEFAULT			
18	(no such command)	DEFAULT			
19	(no such command)	DEFAULT			
20	2000-001T11:58:55.816	br0251.00a	[illegible]	[illegible]	start: CMT tilt limit = 10 degrees
21	2000-001T11:58:56.816	br0251.00a	[illegible]	[illegible]	start: CMT RHZ motor/IMU/S spens
22	2000-001T11:58:57.816	br0251.00a	[illegible]	[illegible]	start: prefer not to [illegible]
23	2000-001T11:58:58.816	br0251.00a	[illegible]		start:
24	2000-001T11:58:59.816	br0251.00a	[illegible]		start:
25	2000-001T11:59:00.816	br0251.00a	[illegible]		start: point cameras down prior to dri
26	2000-001T11:59:15.816	br0251.00a	[illegible]	[illegible]	start:
27	2000-001T11:59:18.816	br0251.00a	[illegible]	[illegible]	start: IMU warmup time prior to use
28	(no such command)	br0251.00a			
29	(no such command)	br0251.00a			
30	2000-001T11:59:38.816	br0251.00a	[illegible]	[illegible]	penultimate: collect final motor/IMU
31	2000-001T11:59:39.816	br0251.00a	[illegible]	[illegible]	penultimate: save final motor/IMU da...
32	2000-001T11:59:59.816	br0251.00a	[illegible]	[illegible]	penultimate: CMT Turn to 161.56deg
33	2000-001T12:00:23.816	br0251.00a	[illegible]	[illegible]	penultimate: wiggle even if goal error...
34	2000-001T12:00:24.816	br0251.00a	[illegible]	[illegible]	penultimate: prefer to leave steering ...
35	2000-001T12:00:29.816	br0251.00a	[illegible]	[illegible]	penultimate: IMPORTANT - re-arm in...
36	2000-001T12:00:30.816	br0251.00a	[illegible]	[illegible]	penultimate: return motor/IMU data
37	(no such command)	br0251.00a			
38	(no such command)	br0251.00a			
39	2000-001T12:00:50.816	br0251.00a	[illegible]	[illegible]	end: CMT. RS-1 does not require a 5s...
40	2000-001T12:00:53.816	br0251.00a	[illegible]		end:
41	2000-001T12:00:54.816	br0251.00a	[illegible]	[illegible]	end:
42	2000-001T12:00:55.816	br0251.00a	[illegible]		end:
43	2000-001T12:00:56.816	br0251.00a	[illegible]	[illegible]	end: final FHAZ images CMT. Final 4h
44	2000-001T12:00:57.816	br0251.00a	[illegible]	[illegible]	end: final RHAZ images

All

Show Restricted Commands

Sort By Sequence

Create New Sequence

Create

Show Selected Sequences

Show	Ground Filename	Seq ID	Default	VC #	Type
✓	DEFAULT	DEFAULT	Yes	2	Event-D...
✓	br0251.00a.sol379_rp...	r025...	No	2	Event-D...

Hide All Delete ... Show All Rename ... Detailed Editor ...

No Command Selected

No command selected

Turn off error list updates for speed

Error List

```
Status  Autosaving to "/home/maxwell/.rose_autosave_maxwell.rml"
Status  Autosaved in 0.021 seconds
Status  Skipping activity expansion because there are already commands in the command section
Status  Opened file "/oss/merb/ops/ops/surface/tactical/sol/379/apss/seq/trav/sol379_rp2.rml" in 0.215 seconds
```

File Formats

RML RSEQ SASF.ZIP SATF.ZIP SSF.ZIP PKG.ZIP

Figure 2. The Rover Sequence Editor (RoSE); parts of image are blurred due to legal restrictions

3 Graphical Commanding

By contrast to RoSE, in HyperDrive -- the 3-D graphics-based component of RSVP -- sequences are composed of pictures and motion. In a sense, HyperDrive shows what the command sequence *means*, by displaying an animated 3-D model of what the rover will do in response to the commands we intend to send it. Indeed, part of the uplink team's daily process is to review such an animation, generated in real time by HyperDrive. Only later does the team review the low-level details.

Figure 3 shows a basic view of HyperDrive, with the model of the rover placed into a 3-D environment constructed from images sent to Earth from the real rover. In this view, the rover's arm, which normally is tucked safely under the vehicle's main body, has been extended to place the Rock Abrasion Tool on a target of interest to the science team.

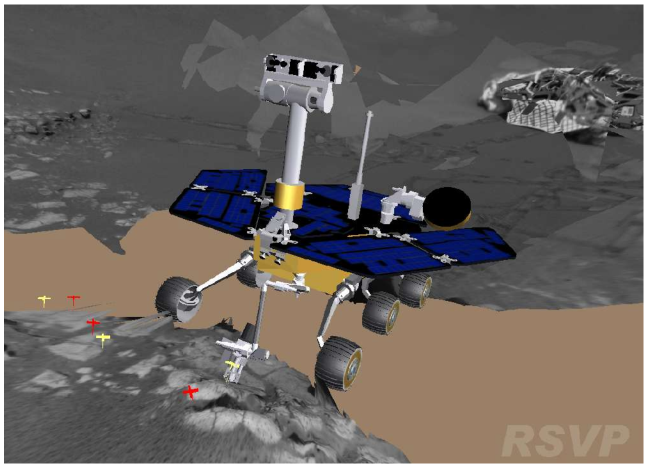


Figure 3. The graphics-oriented sequence viewer (HyperDrive)

If RoSE is a left-brain tool, HyperDrive is a right-brain tool, emphasizing vision and spatial relationships. And unlike RoSE, HyperDrive deliberately omits many commands from its view of the sequence. HyperDrive shows only those commands with a sensible visual representation -- mainly driving, arm motion, and imaging commands. Such commands are represented by 3-D icons inserted into the computer-generated view of the rover's environment. For instance, a command to turn to a particular heading is represented by a 3-D arrow anchored at the current location (that is, at the location where the rover will execute the turn command) and pointed in the goal direction.

4 The Combined Approach

RoSE and HyperDrive, along with the other RSVP component applications (not presented in this paper), are linked by a message-passing system based on PVM, the Parallel Virtual Machine software from Oak Ridge National Laboratories. Any changes in one RSVP tool are immediately broadcast to the other tools in the form of PVM messages. For example, selecting one or more commands in RoSE causes RoSE to broadcast a message designating those commands as selected; in response, HyperDrive highlights the 3-D icons corresponding to all of the selected commands. Similarly, adding a command in HyperDrive causes the command to show up in RoSE as well, and so on.

One simple but extremely useful way we exploit this linkage is with what we call "attached mode." In attached mode, selecting a command in RoSE causes HyperDrive to display the simulated rover in the state corresponding to the completion of that command. For example, if you select the "unstow IDD" (Instrument Deployment Device, or robotic arm) command in RoSE, the simulated rover instantly snaps its arm out. While one result can be an amusing stop-motion animation effect, it's also an invaluable sequence development aid: it lets you step through the sequence, one command at a time, and see what the rover will do in response to each command.

The reverse linkage also exists. HyperDrive can produce smooth (not stop-motion-style) animations representing the current command sequence(s), with all vehicle articulation fully modeled. As HyperDrive plays the animation, it sends messages that cause RoSE to select the command whose effects are currently being animated.

The two tools share responsibility for error-checking. RoSE knows the rover's entire command dictionary, and checks mainly for syntactic errors and command argument-level errors. For every command, RoSE knows the valid range of that command's arguments. Depending on the argument, this may be a valid set of keywords, a valid set of numeric ranges, a maximum string length, or a pattern a string argument must match (expressed as a Perl-style regular expression). HyperDrive's checking is narrower but deeper: for the subset of commands it understands, it simulates each command, tracking the vehicle's changing state and looking for collisions (e.g., between the robotic arm and the vehicle body), violations of joint limits, and similar mistakes [3]. This same simulation provides the data for HyperDrive's animations.

Although no one has attempted to produce hard numbers showing the value of the combined textual and visual commanding modes, the longer-than-expected mission has yielded a great many anecdotal results. The following sections present a series of examples drawn from real-world experience.

4.1 Using the combined commanding approach to control the robotic arm

In the first case, a text-based arm-motion command in RoSE set the rover arm's five joint angles to [-0.003, -0.217, 1.436, 3.495, 3.230] (all values in radians). The HyperDrive simulation reported a collision error in the following command, which rotated the turret (the collection of tools at the end of the arm). Taken alone, the numbers above don't tell even an experienced user much about what the problem could be. But a single glance at the corresponding HyperDrive view made the problem clear. As shown in Figure 4, when the turret begins to rotate, the Alpha Particle X-ray Spectrometer -- the dark cylindrical tool pointing upward in the figure -- will run into the rover's own forearm. The rover driver needed to sequence a more complex series of moves, first drawing the arm farther away from the terrain and tilting the turret away from the forearm, in order to achieve the desired arm configuration.

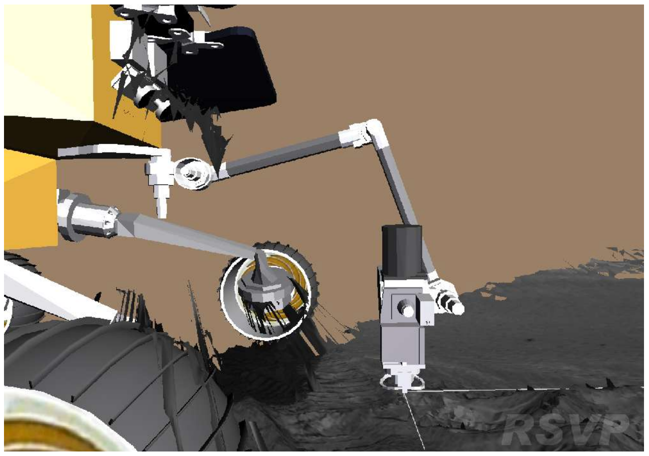


Figure 4. The cause of an error is made visually apparent in HyperDrive

Importantly, HyperDrive serves to abstract certain information about the sequence. There are usually several ways to produce the same arm motion, but when looking at the HyperDrive simulation, the user does not know and does not have to care exactly how each command was specified. In this case, it happens that the arm was moved into its awkward position by specifying joint angles, but it wouldn't have mattered to the analysis if it had been done another way (e.g., by telling the IDD to move the current tool to a chosen 3-D location). Those details are hidden in the HyperDrive view, so the rover driver can concentrate on the effect the command has on the rover. The RoSE window always provides the details.

In another instance, the HyperDrive simulation showed the arm moving at normal speed through most of a sequence, then suddenly slowing to a crawl. While the symptom was visually obvious, the cause was not -- until the rover driver looked at the corresponding textual version of the offending command in RoSE. Each arm motion command includes a "speed ratio" parameter, which says at what fraction of their maximum speed the arm's joints should attempt to move. In this case, the rover driver had accidentally entered 0.1 for this parameter, instead of the intended 1.0 (full speed). As a result, the arm moved at only one-tenth normal speed for that command. The fix was trivial and obvious, and its efficacy was quickly confirmed by HyperDrive.

4.2 Using the combined commanding approach to control rover mobility

Not only is the combined interface useful for controlling the robotic arm, it's also helpful in the other half of the rover drivers' job: driving the rovers.

In many cases, the most natural way to drive the rover is through the visual interface, HyperDrive. Relatively low-level driving commands, such as arcs and point turns, can be added using the HyperDrive interface. More powerfully, HyperDrive also allows the user to interactively move the rover model to the desired location in the terrain model and then add a high-level command that translates to "go here."

This style of driving fits well with the automated terrain analysis provided by RSVP. RSVP includes a copy of the portion of the rover flight software that evaluates terrain for hazards. By running this code on a current terrain model from the rover, RSVP can automatically evaluate hazards in the rover's current surroundings and color-code the model accordingly, simplifying the job of planning a path through treacherous terrain. Figure 5 shows a screen shot of one such evaluation, performed when Spirit was on its way to the so-called "Missoula" Crater. Green areas are very safe, yellow areas are less safe but still traversable, and red areas represent likely mobility hazards. (The light blue zone in the foreground represents an area for which the rover had no data -- its view of that terrain was blocked by its own solar panels.)

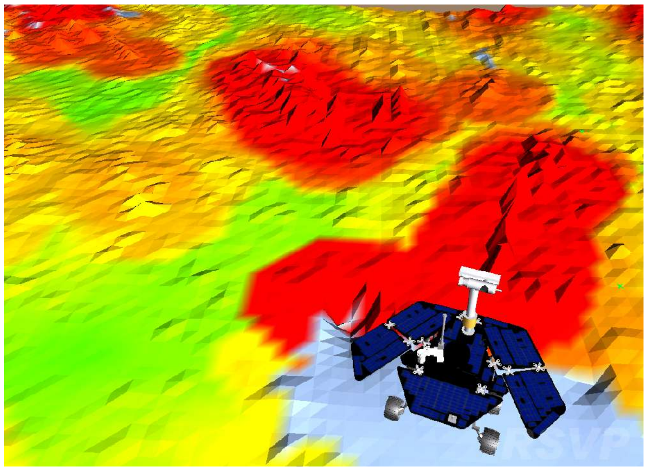


Figure 5. Automated terrain analysis

Yet, in other cases, it's more natural to drive the rover using the text interface. A common example involves imaging a terrain feature with the rover's MTES (Mini-Thermal Emission Spectrometer) instrument, which is mounted on the raised mast. After the rover drills into a rock with the RAT instrument and investigates the subsurface layers, the science team commonly wants the rover to back up 85 cm to reach a good standoff position for MTES imaging of the RAT hole. Since the exact desired behavior is known in advance and there are usually no obstacles to avoid in such a short drive, it's easier and more accurate to enter the backup using the text interface, as a single command that says "drive straight backward 85 cm."

Several rover drivers (such as this writer) prefer a hybrid style of driving, entering most commands in the RoSE interface but constantly keeping an eye on the results in HyperDrive.

4.3 Using the combined commanding approach for imaging

The rover's main picture-taking command requires 47 arguments, with many complex interactions. And the rover often takes not just one image, but a mosaic of overlapping images from the same position, so that they can later be combined to produce the effect of a single larger image. The command parameters for one imaging command may therefore interact not only with each other, but also with the other commands in the mosaic. This is a clear-cut case where the detail present in the textual interface, while unavoidable, can at times make it impossible to see the forest for the trees. If RSVP provided only a textual commanding interface, users often would find it difficult indeed to achieve the correct imaging results.

Since imaging is a primarily visual task, it makes sense that HyperDrive would offer a great deal of assistance in this area. One of HyperDrive's features is a Camera View, which dramatically simplifies the work of image pointing in many cases. Figure 6 shows a recent example from Opportunity. Having finished exploring a trench it had dug, Opportunity was going to back away and turn slightly. The science team wanted an image of the trench from the post-backup position, so the rover driver used HyperDrive's Camera View to generate the required image pointing. He simulated the drive, then opened the Camera View and interactively aimed the navigation cameras in HyperDrive until the camera view showed that they would cover the area of interest. The resulting image pointing was used on the real vehicle, and the results (the lower image in the figure) were practically identical to the prediction (the upper image).

The Camera View is not limited to modeling the navigation cameras. It can also simulate the view from any of the rover's other cameras -- the panoramic cameras (which, like the navigation cameras, are mounted on the mast), the hazard-avoidance cameras (located on the front and rear of the rover body), and the Microscopic Imager (one of the tools on the arm).

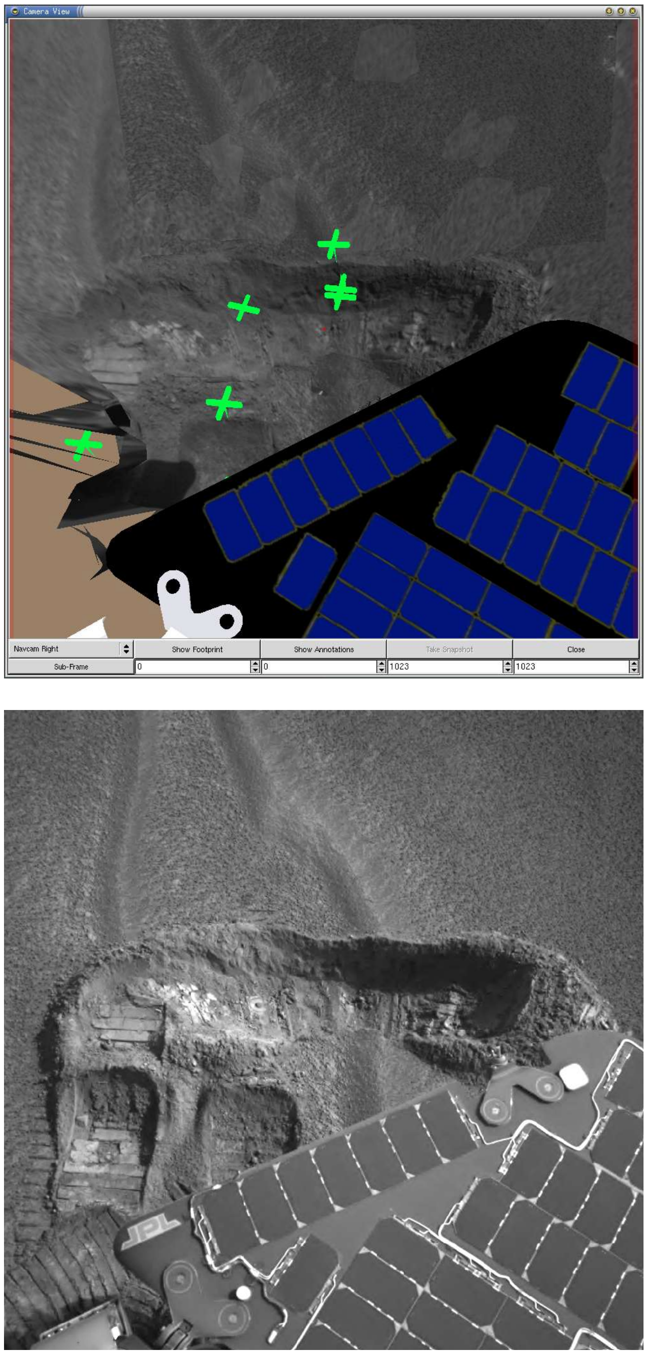


Figure 6. Predicted and actual views from one of the rover's navigation cameras. The green markers in the predicted image show the locations of IDD targets

As usual, the linkage works both ways. Imaging commands entered in RoSE (or in HyperDrive itself) show up in HyperDrive as 3-D icons, showing where the image will be taken and how the cameras will be pointed. In addition, HyperDrive paints a "footprint" on the terrain, showing the approximate area to be covered by each image. The user can edit imaging commands in either tool, and view the results in HyperDrive. While this does not test for all possible interactions between command arguments, it does provide a quick, reasonably accurate visual check for image pointing.

5 Conclusions

The Mars Exploration Rover mission has been, and continues to be, a highly successful story of interplanetary exploration. One of the reasons for this success has been the powerful, flexible tools available to the rover-driving team. Rover drivers must assess a prodigious quantity of information every sol, then act on it rapidly, with little room for error. One of the ways RSVP helps them succeed is by providing multiple viewing and editing modes, so that the user can see the same information in different ways and can enter different commands using the most natural interface for each.

Sometimes a picture is worth a thousand words, and sometimes a word is worth a thousand pictures. By giving rover drivers both textual and visual command interfaces, RSVP ensures that they can continue to make the most of this priceless opportunity to investigate our neighbor planet.

6 References

[1]Chris Leger *et al.*, “Mars Exploration Rover Surface Operations: Driving Spirit at Gusev Crater,” Proc. IEEE-SMC, Waikoloa, HI, October 2005.

[2]Chris Leger *et al.*, “Mars Exploration Rover Surface Operations: Driving Opportunity at Meridiani Planum,” Proc. IEEE-SMC, Waikoloa, HI, October 2005.

[3]Jeng Yen *et al.*, “Sequence Rehearsal and Validation on Surface Operations of the Mars Exploration Rovers,” Proc. IEEE-SMC, Waikoloa, HI, October 2005.