Experience with Rover Navigation for Lunar-Like Terrains

Reid Simmons, Eric Krotkov, Lalitesh Katragadda, and Martial Hebert
Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
reids@cs.cmu.edu

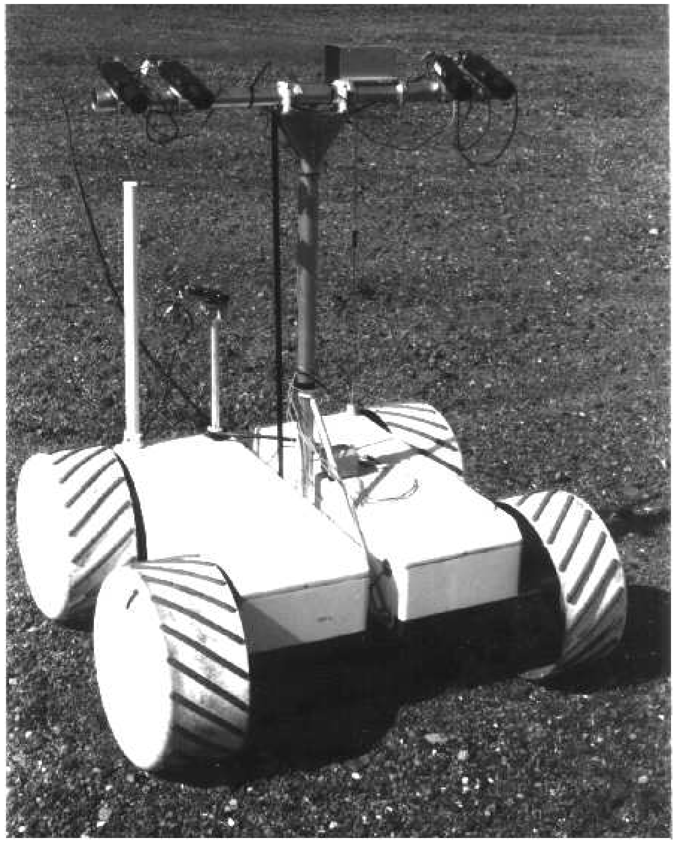
Introduction

The lure of the Moon is strong — and humans are once again responding to the challenge. One promising, near-term scenario is to land a pair of rovers on the Moon, and to engage in a multi-year, 1000 kilometer traverse of historic sights, including Apollo 11, Surveyor 5, Ranger 8, Apollo 17 and Lunokhod 2 [2]. In this scenario, one of the rovers would be driven, returning continuous live video, while the other rover remains stationary, to provide high-resolution imagery of the surroundings and to serve as a communications link with Earth for both rovers. The rovers would periodically switch roles, continuing this way to explore the Moon's surface.

While the hardware aspects of such a mission are daunting — communications, power, thermal, rover reliability and integrity, etc. — the software control aspects are equally challenging. In particular, capabilities are needed to enable driving the rover over varied terrain and to safeguard its operation. Previous experience with planetary robots (in particular, Lunokhod 2 and the arm on Viking) illustrated how laborious and unpredictable time-delayed teleoperation is for remote operators. A better mode of operation is supervised teleoperation, or even autonomous operation, in which the rover itself is responsible for making many of the decisions necessary to maintain progress and safety.

We have begun a program to develop and demonstrate technologies to enable remote, safeguarded teleoperation and autonomous driving in lunar-like environments. The aim is to provide both the techniques and evaluations of their effectiveness and reliability, in order to enable mission planners to make informed cost/benefit tradeoffs in deciding how to control the lunar rovers. To date, we have concentrated on autonomous operation, and have demonstrated a system that uses stereo vision to drive a prototype lunar rover over a kilometer of outdoor, natural terrain. To our knowledge, this is a record distance for autonomous cross-country driving of a vehicle using stereo and only general-purpose processors (in our case, SPARC 10's at 11 MFLOPS).

Besides autonomous driving, we are investigating issues of mixed-mode operation, where a human operator and an autonomous system each provide "advice" on how to drive, with the recommendations arbitrated to produce the actual steering commands to the rover. The idea is to provide a more flexible mode of user interaction, one that reduces the probability of operator fatigue and mistakes that could be damaging to the rover. We have also focused on the problem of estimating the rover's position [5]. This is important in any mission scenario, but particularly one, such as the one described above, that involves long-distance navigation to find particular sites. While the position estimation problem is basically solved for Earth-based rovers, due to the



The Ratler Rover

availability of GPS (Global Positioning System), GPS is probably not a feasible technology for the Moon. Our research on position estimation techniques have concentrated on understanding and filtering sensors such as gyros and inclinometers, using sensor fusion to reduce uncertainty, and developing some novel techniques that involve skyline navigation and Sun tracking [1].

The lessons learned to date are informing our next round of development and experimentation. We plan to demonstrate safeguarded teleoperation of up to 10 km this year, while increasing the complexity of the terrain traversed and the amount of time delay introduced in operating the rover.

The next section describes the rover that is currently being used for our experiments. We then describe the software system developed to drive the rover, and our experimental results. Finally, we address work that is still needed to fulfill the promise to return to the Moon in this millennium.

The Ratler

While we are currently designing a new lunar rover [3], we are using a vehicle designed and built by Sandia National Laboratories [6] as a testbed to develop the remote driving techniques needed for a lunar mission. The Ratler (Robotic All-Terrain Lunar Exploration Rover) is a battery-powered, four-wheeled, skid-steered vehicle, about 1.2 meters long and wide, with 50 cm diameter wheels (Figure 1). The Ratler is articulated, with a passive axle between the left and right body

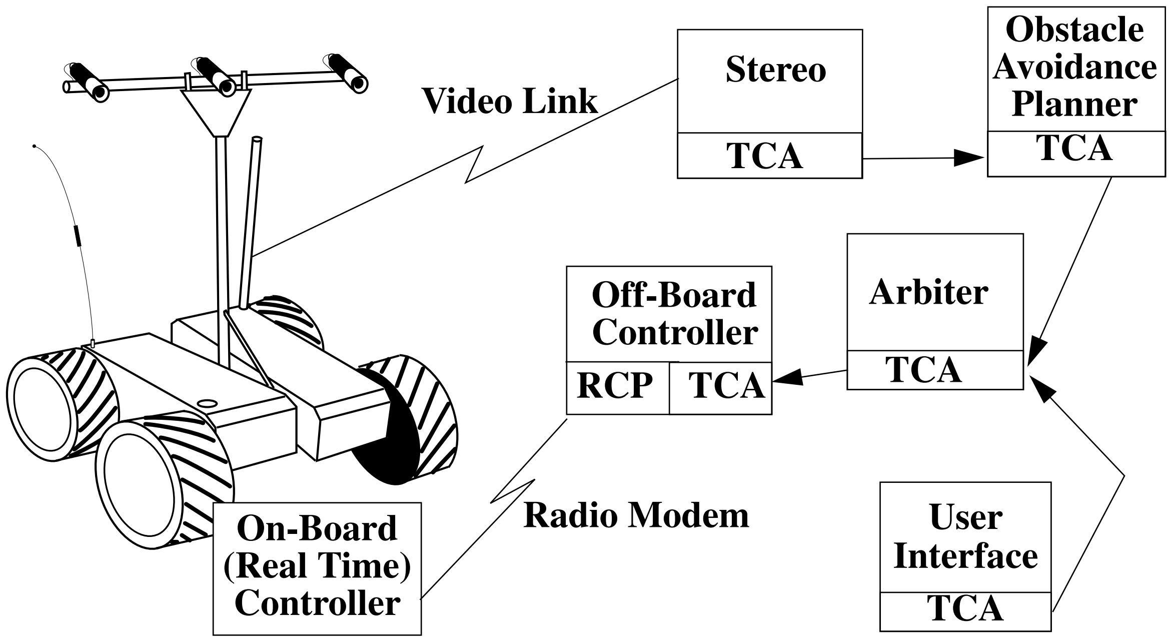


Figure 2. The Navigation System

segments. This articulation enables all four wheels to maintain ground contact, even when crossing uneven terrain, which increases the Ratler's ability to surmount terrain obstacles. The body and wheels are made of a composite material that provides a good strength-to-weight ratio.

Sensors on the Ratler include wheel encoders, turn-rate gyro, a compass, a roll inclinometer, and two pitch inclinometers (one for each body segment). There is one color teleoperation camera, and we have added a camera mast and four black and white cameras for stereo vision (only two of which are currently being used). On-board computation is provided by a 286 and a 486 CPU board, connected by an STD bus, which also contains A/D boards and digitizer boards for the stereo cameras. Currently, the on-board computers handle servo control of the motors, sensor data acquisition, and communication with the off-board computers. Communication is through a 4800 baud data link and a 2.3 GHz microwave video link. Off-board computers, which run the stereo, planning, and user interface modules, consist of several Sun workstations connected via Ethernet.

The Navigation System

Figure 2 presents a block diagram of the overall navigation software system. As described above, it is divided into on-board and off-board components, communicating via two radio links for video and data (eventually, many of the off-board processes will be moved on board the new lunar rover). The basic data flow is that the stereo process produces terrain elevation maps which are passed to the obstacle avoidance planner which uses them to evaluate the efficacy of traveling along different paths. These recommendations are merged with the desires of a human operator to choose the best path to traverse (in autonomous mode, there is typically no operator input). The command arbiter then forwards steering and velocity commands to the off-board controller, which packages them up and ships them, using the RCP protocol developed at Sandia, to the on-board controller which then executes the commands and returns status and sensor information. The off-board controller also performs much of the position estimation task and provides some safety mechanisms, such as stopping the rover if roll or pitch inclinometers exceed certain thresholds.

Stereo

The stereo system used by Ratler takes its input from two black-and-white CCD cameras with auto-iris, 8 mm lenses, mounted on a motion-averaging mast. The output is sets of (x,y,z) triples, given in the camera coordinate frame, along with the pose of the robot at the time the images were acquired. Using the pose, the (x,y,z) values can be transformed into world coordinates to form a (non-uniformly distributed) terrain elevation map. The stereo system can be requested to process only part of the image, and that at reduced resolution (skipping rows and columns in the image), to speed up the overall system cycle time.

The stereo images are first rectified to ensure that the scan lines of the image are the epipolar lines [7]. The best disparity match, within a given window, is then computed using a normalized correlation. Disparity resolution is increased by interpolating the correlation values of the two closest disparities. The normalized correlation method is relatively robust with respect to differences in exposure between the two images, and can be used to produce confidence measures in the disparity values.

Care must be taken to ensure that outlier values (caused by false stereo matches) are minimized. Several methods are used to achieve the level of reliability required for navigation. One method is to have lower bounds on the acceptable correlation values and variance in pixel intensity, to eliminate low-textured areas. Another method eliminates ambiguous matches (caused by occlusion boundaries or repetitive patterns) by rejecting matches that are not significantly better than other potential matches. Finally, the values can be smoothed to reduce the effect of noise. All these methods help to produce elevation maps that accurately reflect the actual surrounding terrain, with only a few centimeters of error.

Obstacle Avoidance Planner

To decide where it is safe to drive, we have adapted techniques developed in ARPA's Unmanned Ground Vehicle (UGV) program for cross-country navigation [4]. The basic idea is to evaluate the hazards along a discrete number of paths (corresponding to a set of steering commands) that the rover could possibly follow in the next few seconds of travel. The evaluation produces a set of "votes" for each path/steering angle, including "vetoes" for paths that are deemed too hazardous to traverse. In this way, the rover steers itself away from obstacles, such as craters or mounds, that it cannot cross or surmount.

The obstacle avoidance planner first merges individual elevation maps produced by the stereo system to produce a more complete terrain map of the area up to seven meters in front of the rover. Map merging is necessary because the limited fields of view of the cameras do not allow a single image to view sufficient terrain; map merging is desirable because it provides the robot with the most up-to-date information with which to make planning decisions. To make the stereo computation tractable, a small segment of the stereo image is requested, at reduced resolution, (typically only about 2% of the total available image). The planner dynamically chooses which portion of the image that the stereo system should process, based on the current vehicle speed, stopping distance, and expected cycle time of the perception/planning/control loop.

To evaluate the potential steering commands, the planner uses a detailed model of the vehicle's kinematics and dynamics to project forward in time the expected path of the rover on the terrain. This produces a set of paths, one for each potential steering direction. The planner then evaluates, at each point along the path, the robot's roll and the pitch of each body segment. If the roll or pitches exceed an allowable threshold, or if the path crosses a significant area that has not yet been imaged by the cameras, then the path is vetoed. Otherwise, the value assigned to that path depends on how level is the overall traverse, and how known is the underlying terrain. The path evaluations are then sent to the arbiter module, along with the pose of the robot at the time of the evaluation. Currently, on a SPARC 10, the obstacle avoidance planner takes about 500 msec per cycle (map merging and path evaluation).

Arbiter

The arbiter module accepts path evaluations from various processes and chooses the best steering angle based on those evaluations. This provides a straightforward way to incorporate information from various sources (such as obstacle avoidance planners, user interface, route planner) in a modular and asynchronous fashion.

Each path evaluation consists of a steering angle, value, and speed. If the value is “veto” then that steering angle is eliminated from consideration. Otherwise, recommendations for that steering angle from all sources are combined using a weighted sum. The arbiter then finds the largest contiguous set of steering angles whose values are all within 90% of the maximum value, and chooses the midpoint of that set as the commanded steering angle to be sent to the controller module (the speed chosen is the minimum speed recommended by all the sources). The idea is to prefer wide, easily traversable areas over directions that might be a bit more traversable, but have less leeway for error if the rover fails to track the path precisely. We have found this to be very important in practice, as the robot's dead reckoning and path tracking ability are only fair, at best.

The path evaluations are also tagged with a robot pose. If the tagged pose differs significantly from the rover's current pose (as determined by the off-board controller), then those path evaluations are ignored. If the evaluations from all the processes are invalidated in this way, then the arbiter issues a command to halt the rover. In this way, the arbiter safeguards against other modules crashing, or otherwise failing to provide timely inputs. Similarly, we guard against a crash of the arbiter module by having it send commands of the form “steer in this direction for X meters.” If the controller does not receive a new command before the Ratler has traversed that distance, then it stops the robot.

When operating in autonomous mode, the obstacle avoidance planner occasionally cannot find any acceptable path. This often occurs when the stereo data has been noisy for a while, or when the robot turns and finds itself facing another, unexpected obstacle. To handle such situations, if the arbiter receives several consecutive path evaluations that are all “vetoed,” then it will command the rover to turn in place by fifteen degrees. This behavior continues until the planner starts sending valid path evaluations again.

User Interface

We have focused our user interface work on facilitating mixed-mode operation, where the human and rover share responsibility for controlling the robot. The current graphical user interface consists of an "electronic joystick," which utilizes the computer mouse to command the robot's direction and speed, and a number of indicators, both textual and graphical, that indicate pertinent information such as commanded and instantaneous robot speeds, roll and pitches, position, and status. Visualization of the terrain is provide by a color camera mounted toward the rear of the Ratler, which is transmitted to a monitor over the microwave radio link.

The user interface supports several driving modes. In the direct teleoperation mode, the human has full control over the rover — almost all safeguarding is turned off. Direct teleoperation is necessary when the rover gets into situations where the software would otherwise prevent motion. For instance, there may be occasions where the pitch limits must temporarily be exceeded to drive the rover out of a crater. This mode is reserved for experienced drivers in exceptional situations.

On the other side of the spectrum, in the autonomous mode the software system has complete control over the robot, choosing the direction to travel based on the stereo input. While we have done some experiments in which we add input from a goal-directed planner, to bias the robot in particular directions, the majority of our experiments have consisted in letting the robot "wander" autonomously, while avoiding obstacles.

The third mode, safeguarded teleoperation, is seen as the standard way in which the lunar rover will be operated. In this mode, input from the human and the obstacle avoidance planner are combined: the user presents a desired direction to travel, and the obstacle avoidance planner can veto it, causing the robot to refuse to travel in that direction. The idea is that the software safeguards should prevent the user from damaging the rover, but not otherwise interfere with the control. Basically, this mode is implemented by having the arbiter combine input from the direct teleoperation mode and the autonomous mode. In addition, if the human chooses not to provide input, the rover will navigate autonomously. In this way, operator fatigue can be reduced by letting the robot operate on its own when it is in benign terrain, while still enabling the user to take over control at any moment.

System Integration

The perception, planning, arbiter, user interface, and off-board controller processes all run on two Sun workstations, connected via Ethernet. Interprocess communication and task sequencing is performed by the Task Control Architecture (TCA). TCA is a general-purpose architecture for mobile robots that provides support for distributed communication via sockets, task decomposition and scheduling, resource management, execution monitoring, and error recovery [7]. TCA-based systems consist of a number of distributed, concurrent processes that communicate via coarse-grained message passing. TCA connects processes, routes messages, and coordinates overall control and data flow. It also provides capabilities to log and analyze message traffic, which has

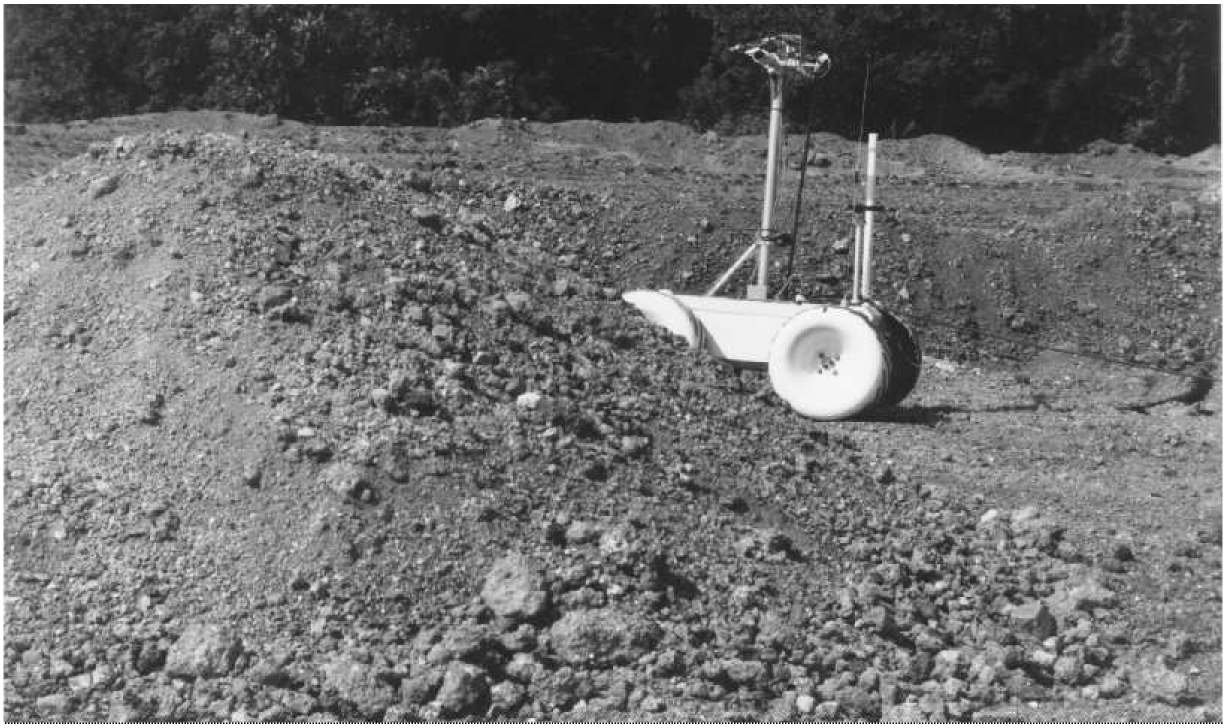


Figure 3. The Ratler at the Pittsburgh Slag Heap

proven to be very useful in understanding certain critical timing issues in the autonomous navigation system.

Concurrency is very important in achieving the levels of performance needed (1-2 Hz cycle time is desirable). In particular, perception and planning functions run simultaneously: while the obstacle avoidance planner is using one stereo elevation map to evaluate paths, the stereo system is processing another image. When stereo completes, it asynchronously sends the image to the planner. Likewise, the arbiter is getting asynchronous path evaluations from the planner and user interface, combining the most recent information to produce steering commands. Meanwhile, the controller module is receiving asynchronous commands and requests for pose information from different modules. While it is admittedly more difficult to develop and debug distributed, concurrent systems, they have great advantages in terms of real-time performance and modularity in design and implementation.

Experimental Results

We have done a number of tests with the Ratler and its navigation system, concentrating on pushing the autonomous aspects of the system. Most of the experiments were performed at a slag heap in Pittsburgh (Figure 3), on an undulating plateau featuring some sheer cliffs and sculpted features (mounds and ridges).

Early experiments tested the dead reckoning of the system and the ability to avoid discrete obstacles. After characterizing the response of the vehicle and improving the position estimation, the rover was able to navigate hundreds of meters with minimal human intervention. To date, our longest contiguous run has been 1,078 m, where 94% of the distance was traversed in autonomous mode and the rest in direct teleoperation mode. Average speed attained for that run was about 10 cm/s, although we have driven, with similar results, at over twice that speed. In all, over 3000 stereo pairs were processed and used by the planner for path evaluations.

In other experiments, we tested the ability of the arbiter to combine inputs from multiple sources: the obstacle avoidance planner and either the user or a path-tracking module. These experiences pointed out the need to improve the arbitration model to choose more intuitive steering commands (such as by biasing toward directions that have recently been chosen). This is particularly important when dealing with a human operator, who needs a fairly predictable model of the system behavior in order to feel comfortable and confident about interacting with it.

The experiments also pointed out the need to perform more sophisticated map merging, to reduce the effects of sensor noise, and demonstrated the possibility of reducing computation even further by even more judiciously requesting stereo data points needed by the obstacle avoidance planner.

Ongoing and Future Work

To achieve the ambitious goals of the mission scenario (1000 km traverse over two years), we need to harden and extend our techniques. Many changes to the current system merely involve incremental improvements: increasing the stereo field of view by using two pairs of cameras, making stereo more robust to noise, merging maps better to reduce sensor uncertainty, requesting stereo data points more judiciously, improving dead reckoning accuracy, and developing more natural arbitration schemes.

Other changes that our experience has shown to be necessary, or at least highly desirable, are of a more fundamental nature. One involves the addition of short-range proximity and tactile sensors to provide more reliable safeguarding. We are currently looking at laser-based proximity sensors that can provide coverage in front of the rover at ranges of one to two meters. Such sensors would be used to warn of impending dangers, especially cliffs and other drop-offs. Tactile sensors would detect collisions with the body segments, especially the underside of the Ratler (i.e., high centering).

We also plan to add more extensive internal monitoring of the robot's health and integrity. For example, we will monitor battery voltage and motor currents, and autonomously "safe" the vehicle if they exceed given thresholds. We will also make use of various sensor redundancies to detect sensor failure itself. For instance, we can use the wheel encoders to estimate the change in orientation of the robot, which provides confidence that the turn-rate sensor is performing adequately.

Our experimental work will take two directions: we will continue to demonstrate and quantify autonomous navigation capabilities using stereo, and we will also investigate more carefully issues of mixed-mode and safeguarded teleoperation. This includes quantifying the performance improvements gained by adding various technologies, such as safeguarding, high-level commands, etc. Our goal for 1995 is a 10 km safeguarded teleoperation and a 2 km autonomous traverse in rougher, more lunar-like terrain.

Conclusions

We have presented a system for autonomously and semi-autonomously driving the Ratler, a four-wheeled, articulated prototype lunar rover, in natural, outdoor terrain. The navigation system uses a combination of on-board and off-board computation to control the vehicle, process stereo images, plan to avoid obstacles, and integrate machine and human recommendations regarding the travel direction.

Our experiments have demonstrated basic competence in driving to avoid cliffs and mounds, but much more work needs to be done in order to produce a system that can behave reliably over many weeks and kilometers. In particular, we have targeted the areas of safeguarding and remote teleoperation as worthy of further investigation.

It is important to realize that safeguarding and autonomous navigation can have profound impact on the ease and reliability of remote driving of a lunar rover. On the other hand, such systems admittedly add complexity to the hardware and software requirements of a rover. We need to perform careful experiments to quantify the value added by these technologies, in order to demonstrate their effectiveness for near-term lunar missions.

References

[1] F. Cozman and E. Krotkov. Mobile Robot Localization using a Computer Vision Sextant. Submitted to IEEE Intl. Conf. Robotics and Automation, Nagoya, Japan, 1995.

[2] L. Katragadda, et al. Lunar Rover Initiative — Preliminary Configuration Document, Tech. Report CMU-RI-TR-94-09, Carnegie Mellon University, 1994.

[3] L. Katragadda, J. Murphy, and W. Whittaker. Rover Configuration for Entertainment-Based Lunar Excursion. In International Lunar Exploration Conference, San Diego, CA, November 1994.

[4] A. Kelly. A Partial Analysis of the High Speed Autonomous Navigation Problem. Tech Report CMU-RI-TR-94-16. Robotics Institute, Carnegie Mellon University, 1994.

[5] E. Krotkov, M. Hebert, M. Buffa, F. Cozman, and L. Robert. Stereo Driving and Position Estimation for Autonomous Planetary Rovers. In *Proc. IARP Workshop on Robotics In Space*, Montreal, Canada, July 1994.

[6] J. Purvis and P. Klarer. RATLER: Robotic All Terrain Lunar Exploration Rover. In *Proc. Sixth Annual Space Operations, Applications and Research Symposium*, Johnson Space Center, Houston TX, 1992.

[7] L. Robert, M. Buffa, and M. Hebert. Weakly-Calibrated Stereo Perception for Rover Navigation. In *Proc. Image Understanding Workshop*, 1994.

[8] R. Simmons. Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation*, **10:1**, Feb. 1994.