Planetary Rover Simulation for Lunar Exploration Missions

Mark Allan
KBRwyle
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-2
mark.b.allan@nasa.gov

Uland Wong
KBRwyle
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
uland.wong@nasa.gov

P. Michael Furlong
KBRwyle
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
padraig.m.furlong@nasa.gov

Arno Rogg
KBRwyle
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
arno.rogg@nasa.gov

Scott McMichael
KBRwyle
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
scott.t.mcmichael@nasa.gov

Terry Welsh
Logyx LLC
425 N. Whisman Road #400
Mountain View, CA 94043
terence.m.welsh@nasa.gov

Ian Chen
Open Robotics
170 S. Whisman Road
Mountain View, CA 94041
ichen@openrobotics.org

Steven Peters
Open Robotics
170 S. Whisman Road
Mountain View, CA 94041
scpeters@openrobotics.org

Brian Gerkey
Open Robotics
170 S. Whisman Road
Mountain View, CA 94041
gerkey@openrobotics.org

Morgan Quigley
Open Robotics
170 S. Whisman Road
Mountain View, CA 94041
morgan@openrobotics.org

Mark Shirley
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
mark.h.shirley@nasa.gov

Matthew Deans
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-1
matthew.c.deans@nasa.gov

Howard Cannon
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-1
howard.n.cannon@nasa.gov

Terry Fong
NASA Ames Research Center
Moffett Field, CA 94035
MS 269-3
terry.fong@nasa.gov

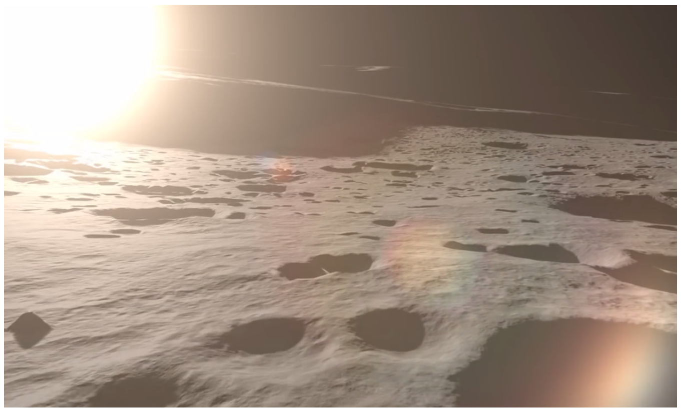
Abstract—**When planning planetary rover missions it is useful to develop intuition and skills driving in, quite literally, alien environments before incurring the cost of reaching said locales. Simulators make it possible to operate in environments that have the physical characteristics of target locations without the expense and overhead of extensive physical tests. To that end, NASA Ames and Open Robotics collaborated on a Lunar rover driving simulator based on the open source Gazebo simulation platform and leveraging ROS (Robotic Operating System) components. The simulator was integrated with research and mission software for rover driving, system monitoring, and science instrument simulation to constitute an end-to-end Lunar mission simulation capability.**

Although we expect our simulator to be applicable to arbitrary Lunar regions, we designed to a reference mission of prospecting in polar regions. The harsh lighting and low illumination angles at the Lunar poles combine with the unique reflectance properties of Lunar regolith to present a challenging visual environment for both human and computer perception. Our simulator placed an emphasis on high fidelity visual simulation in order to produce synthetic imagery suitable for evaluating human rover drivers with navigation tasks, as well as providing test data for computer vision software development.

In this paper, we describe the software used to construct the simulated Lunar environment and the components of the driving simulation. Our synthetic terrain generation software artificially increases the resolution of Lunar digital elevation maps by fractal synthesis and inserts craters and rocks based on Lunar size-frequency distribution models. We describe the necessary enhancements to import large scale, high resolution terrains into Gazebo, as well as our approach to modeling the visual environment of the Lunar surface. An overview of the mission software system is provided, along with how ROS was used to emulate flight software components that had not been developed yet.

Finally, we discuss the effect of using the high-fidelity synthetic Lunar images for visual odometry. We also characterize the wheel slip model, and find some inconsistencies in the produced wheel slip behavior.

TABLE OF CONTENTS


(a) A Lunar scene produced by the RP Simulator.

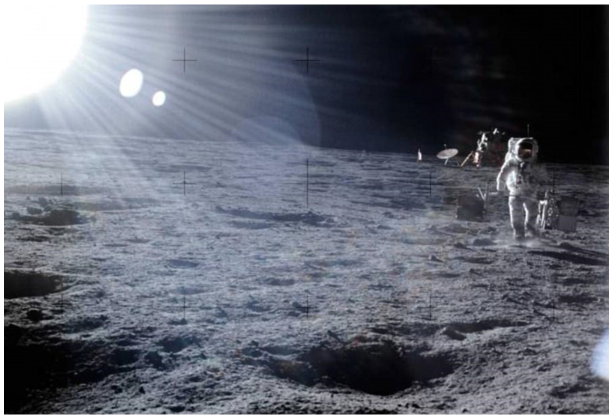

(b) A Lunar scene from the Apollo 12 mission.

Figure 1: One of the goals of the simulator was to create a Lunar environment that not only simulated the interaction of the vehicle with the Lunar terrain, but was also visibly similar to the Lunar terrain.

1. INTRODUCTION

Over the last twenty years, new observations of the Moon by orbital and surface missions have provided evidence of a Lunar water system that is dramatically more complex and rich than previously believed [1]. The potential to exploit water resources for *in situ* utilization motivates low-cost robotic prospecting missions to the surface of the Moon.

However, the human perceptual system is unaccustomed to operating in the Lunar environment. The Apollo astronauts noted difficulty in judging distances, estimating sizes, and tracking landmarks [2]. Further compounding this is the danger that can be posed from the operating interfaces used, which can influence the types of errors which are committed during operations [3], [4].

It is desirable to avoid mission-terminating errors in even the lowest-cost missions, but that requires training in representative environments. Replicating the Lunar environment at scale for practice operations is not feasible, but a viable alternative is to construct a simulation environment which presents representative visible characteristics, as well as approximations of the expected rover-terrain interactions.

Additionally, early in mission development there are many unknowns. As the various mission elements work to address their specific requirements within budget and technology constraints, the trade space is large and a design decision in one element may have ripple effects that impact overall system performance. One way to reduce uncertainty about design alternatives and to answer questions about system performance, particularly from an operational perspective, is through simulation. For example, it may be obvious that a skid steer mobility system would be simple, cheap, and robust from a mechanical perspective but that dead reckoning from an explicit steer platform would give superior localization results. But how would the driving behavior of these two platforms compare with respect to solar charging rates over a traverse?

Our goal was to create a simulation environment in which we could rapidly create approximate prototypes of complete Lunar rover systems to estimate characteristics and performance. Due to the broad scope of the simulation and the desire to have a functional, end-to-end Lunar driving simulation as quickly as possible, a simulation framework that offered a rich set of off-the-shelf capability was necessary.

We built our simulation environment using the open-source Gazebo platform and tools from the ROS ecosystem (Section 3). ROS is used in conjunction with in-house software to simulate both the on- and off-board rover mission software (Section 7). Considerable effort was devoted to producing realistic visual renderings of Lunar environments, with limited processing burden on the simulation (Section 5). The structure of the rover prototypes was modeled using ROS tools, and we developed models in the Gazebo simulation environment for approximating wheel slip in loose soil(Section 6).

Although applicable to a broad range of scenarios, our Lunar rover simulation was built initially to exercise and evaluate our reference mission's approach to Lunar surface operations. The key aspects of that approach were (a) teleoperation of the rover by a pair of drivers on Earth and (b) continuous, real-time interpretation of instrument data by a small science team that is tightly integrated within mission operations followed by limited adjustment of the mission plan. This paper focuses on the development and capabilities of the simulator itself and how it enabled evaluation of the driving task, but we touch on the science data interpretation task (Section 8).

2. BACKGROUND

Reference Mission

The Resource Prospector (RP) is a Lunar rover mission concept developed by NASA [5], [6]. RP is intended to characterize the nature and distribution of subsurface volatiles in permanently shadowed regions (PSR) located in one of the Moon's polar regions. RP is also intended to demonstrate the feasibility of *in situ* resource utilization by extracting and processing volatiles (e.g., hydrogen) that could prove valuable for future exploration missions. RP directly addresses several of NASA's "Strategic Knowledge Gaps" for Lunar exploration as well as the Global Exploration Roadmap's strategic goal of using local resources for human spaceflight.

Hydrogen and other volatiles have the potential to be an enabling resource for human missions to the Moon and beyond. The distribution of Lunar subsurface volatiles drives the RP mission requirement for mobility. In particular, the spatial distribution of cold-trapped volatiles is hypothesized to be governed by impact cratering with the top 0.5m being patchy at scales of 100m. The mixing time scale increases with depth (less frequent larger impacts). Consequently, increased mobility (greater range) reduces the depth requirement for sampling (i.e., drilling or excavation depth).

For the RP mission, a solar-powered, Lunar rover equipped with a drill and a science payload would be used to: (1) locate surface and near-subsurface volatiles; (2) excavate and analyze samples of the volatile-bearing regolith; and (3) demonstrate the form, extractability and usefulness of the materials. The payload contains two prospecting instruments, the Neutron Spectrometer System (NSS) and the Near Infra-Red Volatile Spectrometer System (NIRVSS), which can sense hydrogen at concentrations as low as 0.5 WT% to a depth of 1 m and can capture signatures of bound H_2O/OH. The payload also contains a processing system that can heat regolith samples to 450 C and analyze evolved gases using a gas chromatograph / mass spectrometer.

RP would operate on the Moon for 7 to 10 days during a single Lunar day at a location where both continuous solar illumination and direct-to-earth (DTE) communication conditions are present [7]. The rover is designed to be capable of operating in shadowed regions for up to 6 hours before needing to return to sunlight for recharge. Because the rover is not designed to survive the cold of the Lunar night, the mission focus is to traverse sufficient distance to determine the lateral and vertical distribution of volatiles in a target region before the Lunar day ends.

The short duration of the RP mission, combined with operational constraints and limited knowledge about the surface environment (e.g., high-tempo, interactive rover commanding and real-time science operations) create unique challenges for mission operations [8]. These challenges have the potential to contribute to significant levels of operator workload for the team responsible for maneuvering the rover on the Lunar surface. These challenges may also impact the ease of acquiring and maintaining situation awareness, the ability to identify and handle contingencies, and ground control performance (rover driving, system monitoring, etc.)

Reference Mission Operations

Lunar surface operations provide for qualitatively different mission operation scenarios than what is possible in the Martian surface. An important aspect is the broad swings in temperature on the Moon. From Lunar day to Lunar night there can be swings in temperature of over three hundred degrees. Building a robot to survive Lunar night comes with additional costs and reliance on heat sources, like the radioisotope heating used in the Lunokhod missions [9]. This can increase mission complexity substantially.

If the rover is not designed to survive Lunar night then the mission either has the time pressure to complete within a Lunar day, or the operational pressure to chase sunlight to peaks of eternal light, as proposed in [10]. The Resource Prospector mission planned to operate within the constraints of the Lunar solar day, creating a need for high-tempo mission operations.

The Moon is not without its operational advantages. First, the relative closeness of the Moon means that the time-of-flight communications delay is only 2.5 seconds, and not on the order of minutes for a Mars mission. Second, through the use of the Deep Space Network, it is possible to be in constant communication with a Lunar surface vehicle, barring shadows from terrain features. Constant availability of a rover supports the high-tempo mission cadence missions like Resource Prospector.

Taking advantage of the high communications availability, we designed our autonomy system to be distributed between the rover on the surface of the Moon and the remote operations team on Earth. The architecture would place lower-level components of an autonomous system – e.g., waypoint following, relative pose estimation, vehicle safeguarding – on-board the robot. Higher-level algorithms, like terrain analysis and absolute localization through terrain matching, are operated on the ground. The division of software modules is given in Fig. 16.

3. SIMULATOR APPROACH

After examining the options available in the industry, we chose to base our work on the robot simulator Gazebo [11][2]. We based this decision on three key characteristics of Gazebo: maturity, breadth, and openness. With work beginning in 2002, Gazebo is among the oldest robot simulators that is still actively developed and widely used today. In that time, Gazebo has been used to simulate a wide variety of robotic systems in myriad types of indoor and outdoor environments [12], with notable examples including the DARPA Robotics Challenge [13] and the NASA Space Robotics Challenge [14]. Finally, Gazebo is an open source platform released under the permissive Apache 2.0 License, which means that we can freely use it for our work and modify it to fit our specific needs. For similar reasons, we chose the widely used and also open source ROS (Robot Operating System) platform[3] as the basis for our efforts in prototyping the vehicle control software [15].

Driving planetary rovers on the Lunar poles is not a common use case, however. Consequently, significant modifications, extensions, and new development was needed in order to implement our target scenarios. First, high resolution terrain representative of the potential landing sites for the reference mission was required. The simulation environment must be capable of supporting large, high resolution terrain in order to enable traverses on the order of hundreds of meters. The synthetic camera views rendered by the simulation must have sufficient fidelity to test human perception and situational awareness, as well as be suitable as test data for computer vision researchers. Although high fidelity mechanism modeling was not required for our purposes, we would need flexible mechanism configuration to explore the impact of design elements such as steering limits and camera placement. Finally, the simulation would need to integrate with mission operations software, or analogs of such software, in order to conduct end-to-end driving simulations.

4. SYNTHETIC TERRAIN GENERATION

To produce a viable Lunar driving experience the simulated world must be representative of the target physical environment. Specifically, the morphology of the terrain should accurately represent terrain features that would be considered either positive or negative obstacles for the rover. Although Digital Elevation Models (DEMs) are available for much of

[2]http://gazebosim.org
[3]http://www.ros.org

the Lunar surface, the resolution of these models are not sufficient to represent rover-scale hazards. To make matters worse, the best resolution Lunar DEMs are generated from stereo orbital imagery, yet the reference mission was targeting areas with permanently shadowed regions. Existing terrain models of the Lunar surface are on the order of meters, yet a resolution on the order of centimeters is required to adequately represent rover-scale hazards. We addressed this disparity by developing tools to synthesize high resolution terrain suitable for rover driving.

On the Moon, the principal obstacles of interest to rovers are rocks and craters. Getting the ”density” right was one of the primary objectives of this work as the distributions of obstacles have significant impact on mission traverse paths and schedule. While mean values are published in literature [16] for Highlands/Mare/Polar regions, these size-frequency distributions vary widely according to locality. Parameters such as proximity to large or fresh craters, in particular, are significant modulating factors for rocks. We attempted to model these variations by procedural placement of craters and ejecta fields and simulating the processes of Lunar terrain formation with the most current orbital information.

In our process, crater distributions are first sampled to produce an estimate of size-frequency per unit area. For a scene of specific dimension, we can predict how many craters of a particular diameter are present. These distributions take an inverse-exponential form introduced in [16]. We have updated parameters for Polar regions based on manual identification in LRO images and curve fitting. Variation from uncertainty is allowed by estimating sigma from fitting residuals. In the case where the scene is based on a prior DEM, large craters are identified by hand and this is factored into the sampled distribution so that double counting does not occur. Next, an age is assigned pseudo-randomly to each crater with the probability of a crater being old decreasing with size such that small craters are uniformly distributed [17]. The sampled distribution is ordered sequentially in time and assigned a flag based on whether it is large enough to generate ejecta. Working forward in time, ejecta generating craters are ”placed” by assigning a uniform random (x,y) location while their shape is stenciled onto a spatial probability map used to generate ejecta rocks. Shape consists of an interior, rim, and ejecta blanket which are all tunable physical parameters. Small craters do not generate ejecta and are thus only assigned spatial coordinates without contributing to the ejecta map.

Unlike craters, it is difficult and time consuming to identify existing rocks from orbital imagery for each terrain. Moreover, small rocks are under image resolution limits and must be extrapolated from models. Thus, all rocks are generated by randomly sampling from distributions of size-frequency similar to craters. There are two populations associated with rocks, one for ejecta thrown during crater formation, and a second uniform background distribution for terrain between large craters. For the Polar regions, the background distribution is several orders of magnitude sparser than ejecta. During the process of sampling rocks and assigning spatial coordinates, the ejecta probability map is consulted in order to modulate local density. At each time step, old rocks have a probability of being covered by regolith, and a new crater forming will cover old rocks under its extent. When the number of generated rocks and craters equals the expected number plus some variation, the sizes, ages, and coordinates of the features are passed into the fractal terrain generator.

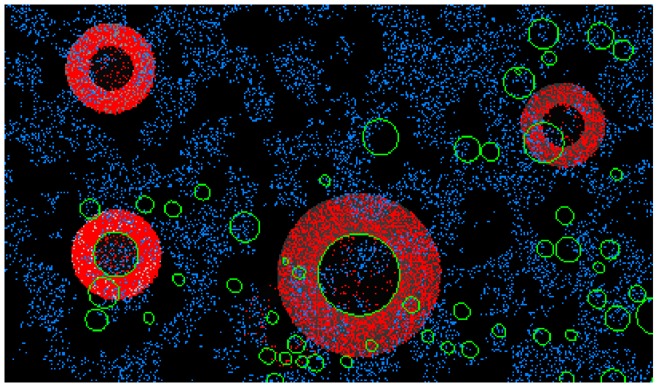


Figure 2: Graphic showing manually placed craters (green), background rock distribution (blue), and ejecta rock distribution (red).

Our fratal terrain generation process synthetically increases resolution of DEMs and inserts craters utilizing techniques established by [18]. With this approach, the process starts with a low resolution DEM of the area of interest. A fractal expansion of the region is performed by iteratively applying the diamond-and-square algorithm to reach the desired resolution. Craters are placed by fitting a plane to a patch of terrain, inserting the crater bowl profile into the plane, and blending the ejecta blanket into the surrounding terrain. The crater profile is defined by four connected polynomials for which a ”freshness” parameter controls crater degredation.

The diamond-and-square method of fractal terrain generation is straightforward to implement and a reasonable choice for adding higher frequency details. However, the method is known to create interpolation artifacts when constrained [19], particularly on low frequency features like aged crater rims. We employed two approaches to mitigate these artifacts. First, we replace linear interpolation with Catmull-Rom splines in the diamond-and-square algorithm which markedly reduces the severity of the artifacts but does not eliminate them. Second, we use an approach similar to that used in [20] whereby the terrain is split into high and low frequency components. The high frequency component is scaled up with the diamond-and-square algorithm whereas the low frequency component is scaled up with bilinear interpolation. Craters are inserted into the low frequency component before adding the high and low frequency components back together to reconstitute the DEM. (Fig. 3)

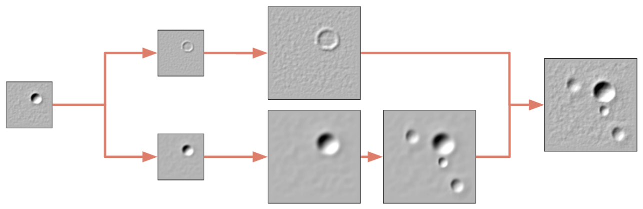


Figure 3: Terrain is split into high and low frequency components during upscaling process to avoid interpolation artifacts

Rocks are inserted into the DEM as a final step. For each rock to be inserted, a model is randomly selected from a library of high resolution rock hightfields. The model is scaled to the desired size, then overlaid onto the terrain.

In addition to the enhanced GeoTIFF DEM, the synthetic terrain generation process produces other assets that will be used for appearance modeling in Gazebo. A rock mask maps which cells in the DEM correspond to regolith and which cells correspond to rocks. Next, an albedo map is generated by inserting crater splash patterns using crater location, size, and freshness parameters. The excavated regolith from a fresh crater impact is lighter in shade due to lack of exposure to solar radiation, and over time the ejecta blanket weathers and blends into the surrounding terrain. An exponential function applied to the freshness parameter determines ejecta ray lightness. Finally, we generate a normal map which captures the surface normal at each DEM posting.

5. Lunar Visual Environment

Our simulator placed an emphasis on high fidelity visual simulation because visual cues are fundamental to driving and localization tasks. Human operators must be able to perceive hazards and maintain adequate situational awareness in order to safely drive a rover, and the development of robust computer vision software requires images representative of those that would be received during the mission.

The visual environment at the Lunar poles is unlike anything found on Earth. The sun is very low on the horizon (5 to 10 degrees elevation in our scenarios) which causes extremely long shadows in one direction and blinding sunlight in the opposite direction. There is no atmospheric scattering which, on Earth, can provide important depth queues [21] and leads to Lunar shadows appearing sharper and darker. Additionally, the Lunar regolith that covers the surface of the Moon has unique reflective properties.

Lunar regolith is formed through a microbombardment processes which pulvarizes rock into soil [22]. Because the Lunar soil is not produced by the erosion processes we are accustomed to on Earth - the particulate matter is sharp and has reflectance properties unlike Earth soils, which can pose a challenge to both human and automated perception systems. Notably, there is a significant opposition effect [23] in which the surface brightens and loses contrast when the view angle and illumination angle approach coincidence.

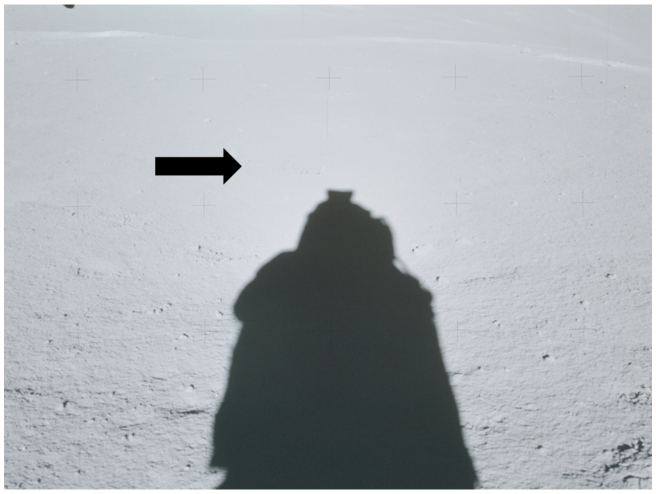


Figure 4: The opposition effect is illustrated in this surface image of the moon from Apollo 16. Inability to perceive terrain detail occurs when viewing away from the solar direction.

By accurately modeling (within the technical limitations of real time computer graphics and our budget) the appearance of the Lunar surface, we desired to build a grounded intuition about what it would be like to drive a rover at the Lunar poles. To do this, we enhanced Gazebo to be capable of rendering high resolution, large scale terrains. We developed a surface shader that models the reflective properties of Lunar regolith, and we draw the rover wheel tracks into the terrain so drivers can see where they have been. We improved Gazebo's real-time shadow rendering for our use case, and developed tools to pre-render large scale static shadows with accurate penumbras. We used ephemeris data to accurately place the Sun and Earth in the sky for proposed mission landing dates. Expected scene irradiance was modeled in order to approximate global illumination effects, and we extended Gazebo's camera modeling capabilities by adding dynamic exposure and high bit depth rendering.

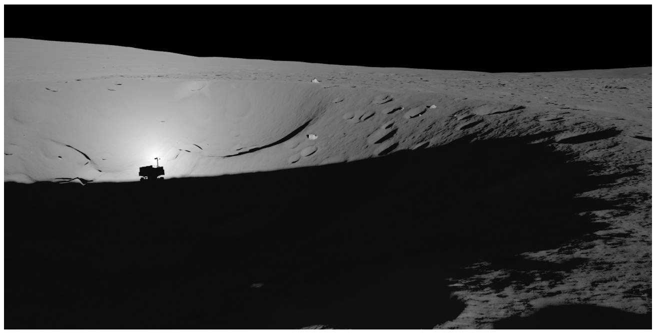


Figure 5: Opposition effect in simulation.

Gazebo Terrain Improvements

As high resolution synthetic terrains were generated and used in simulation, it became evident that the existing terrain visualization support in Gazebo had performance limitations. At the highest resolution of 8K in our experiments, the DEM can take up to several minutes to load in Gazebo, and once loaded, a significant drop in framerate was observed. To address these performance issues, we added two features to Gazebo: 1) Level-Of-Detail (LOD) support and 2) caching of terrain LOD data to disk. Gazebo uses the Ogre3D rendering engine, which conveniently has support for both of these features.

LOD—The most noticeable drop in runtime rendering performance occurs when the entire terrain is within the camera view, in which case, no parts of the terrain are frustum-culled. While the camera sensors onboard the rover do not necessary see the entire terrain, the degraded performance was affecting normal usage of the simulator such as basic user interaction with the simulation environment. Integrating LOD support helped to reduce the triangle count and hence improves the frame rate in this scenario (see figure 6). The quad-tree based LOD implementation in Ogre3D lets users configure LOD transitions to some degree by specifying the maximum screen pixel error allowed when rendering. In general, a node higher up the tree, i.e. lower detail, can only be rendered if its screen space pixel error, computed based on the height characteristics of the node and the current camera distance, does not exceed the specified maximum.

Terrain data caching—It was observed that a large portion of the Gazebo startup time was attributed to the loading and parsing of DEM data into Ogre3D internal terrain tile data format. Adding the ability to cache these data means that the overhead would only need to be incurred once. All subsequent Gazebo sessions using the same DEM file and textures will fetch and load the generated Ogre3D terrain cache from disk, bypassing the DEM parsing and terrain tile generation process. In our tests, the Gazebo startup time reduced by a factor of 10 (from 5 minutes to 30 seconds) when loading a simulation environment constructed from an 8K resolution DEM.

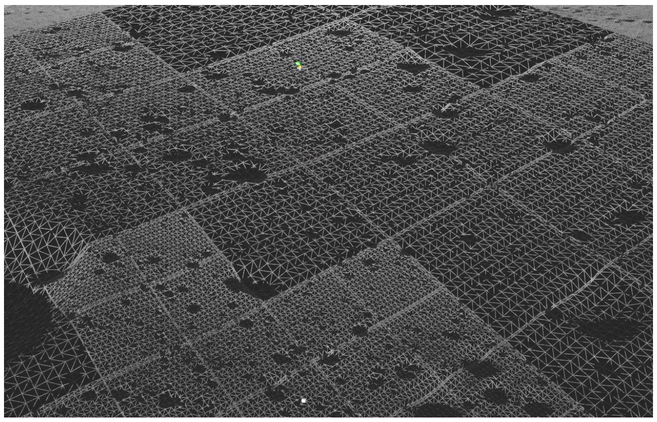


Figure 6: Wireframe visualization of terrain with LOD enabled

Enabling LOD helped resolved framerate issue but in some cases, it introduced noticeable popping effects during LOD transitions. This issue was mitigated by pre-subdividing the DEM into smaller terrain chunks. In doing so, we also had to make sure that the UV coordinates of each terrain chunk are transformed so that the terrain texture maps can be sampled correctly during shading.

Background Terrain Meshes—Having terrain that extends to the horizon is critical because human drivers often use distant terrain features to get their bearings and the RP navigation team was investigating automated methods of localizing using the horizon line. Even with the LOD enhancements to Gazebo, the render performance and memory load of the high resolution terrain made it impractical to extend to the horizon.

Consequently, we split the terrain into two regions, a high resolution ”drivable region” and lower resolution ”backround tile” meshes. The drivable region uses a high resolution DEM, typically in the range of 3-5cm per posting, and employs a detailed shader. The background tiles begin at the boundary of the drivable region and fall off at progressively lower resolution toward the horizon. Because of the distance between the eyepoint and background tiles, there is no distinguishable transition despite the significantly lower resolution and simplified shader (Fig. 7).

GLSL Shader

A shader is a programmable replacement for parts of the real-time rendering pipeline on a graphics processing unit (GPU). Using the OpenGL Shading Language (GLSL), we have built a shader that simulates lighting, shadows, material properties for regolith and rock, and camera exposure. This effort ultimately produces a final color value for every pixel in our camera images. Much of the shader combines standard computer graphics techniques or slight variations on them. However, our material simulation of regolith is not commonly found in shaders.

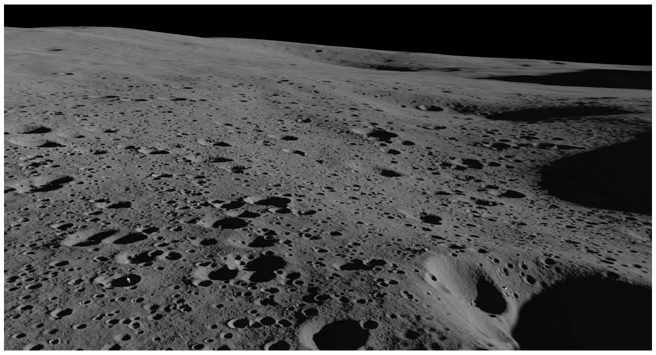


Figure 7: Drivable region in foreground blends seamlessly with lower resolution background tiles.

Regolith is the layer of powdery dust which covers most of the moon except for the rare vertical rocky surface. As such, correctly simulating the appearance of regolith via the bidirectional reflectance distribution function (BRDF) is a major factor in achieving visual fidelity. The reflectance of Lunar regolith and regolith-like materials have been comprehensively studied and the Hapke functions remain the most accurate physically-based models [24], [25]. However, the Hapke BRDF has several drawbacks when used for rover-scale rendering including mathematical complexity, non-invertibility, and edge artifacts [26]. We implement an approximation to the full BRDF suggested by Hapke, which combines coherent-backscatter and shadow hiding into a single term [25]. Furthermore, our Hapke approximation shader is combined with a Lambertian BRDF with a nadir-dependent angle to reduce rendering artifacts. This is a similar strategy used by [27] in developing the Lunar-Lambertian model derived from Lommel-Seeliger reflectance. Our Hapke parameters are estimated from prior literature for Highlands regolith, which is a stand-in for Polar regions of the moon. Our model achieves the desired loss of contrast and opposition surge while behaving properly under direct illumination.

Wheel Tracks

Any rover will leave wheel tracks in the powdery Lunar regolith. We wanted to include these tracks in our visual simulation for realism and as an orientation aid for operators. It would be possible to reshape the terrain geometry to represent wheel tracks, but this method is difficult to implement and would provide more fidelity than required. An alternative is to use a technique called bump mapping to give the illusion of wheel tracks on flat geometry.

Our shader employs bump mapping to give the regolith its fine detail and we expand on this to add wheel tracks. We apply a 1-channel texture to the drivable area of the terrain. It starts out with every texel set to the maximum value, 255. Wherever a rover wheel travels we draw a groove in the texture, setting texels to 0 or some medium value. Wherever this texture is less than 255, the shader mixes a surface normal into the bump mapping. To approximate a surface normal it compares neighboring texels in the wheel tracks texture to find the slope in the x- and y-directions. Assuming a z-value of 1.0, the three values are normalized to produce a final surface normal.

To draw the wheel tracks in the texture we use a Gazebo plugin. We use Gazebo’s knowledge of the rover model and the terrain layout to compute what part of the texture each wheel position corresponds with. Everywhere each wheel travels, we draw an anti-aliased dot in the texture of a size in texels that appears to match the wheel size when the texture is projected on the terrain. These dots overlap to give the appearance of a continuous groove made by each wheel.

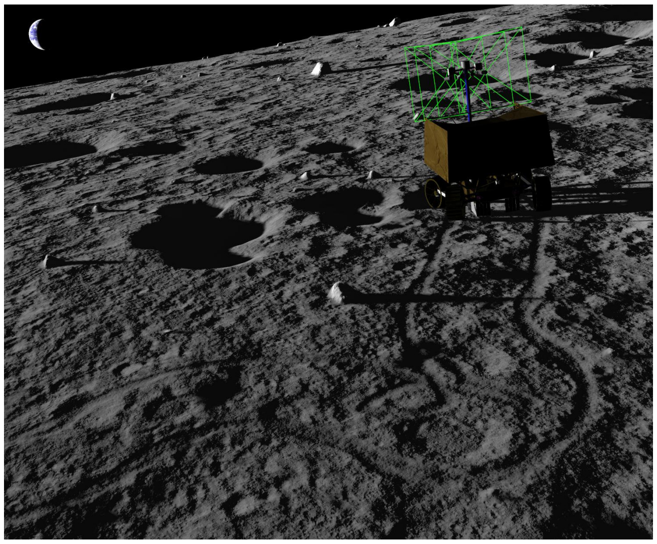


Figure 8: Rover wheel tracks

Shadows

Shadow rendering provides countless challenges and has been under continuous development since the advent of computer graphics. Like most aspects of real-time computer graphics, a good solution involves finding a set of approximations that work well for your particular problem. Our problem was to render shadows that would appear as realistic as possible from the perspective of the rover cameras. Additionally, we needed a shadow solution that would support low illumination angles and long cast shadows as the reference mission scenarios have the sun at just a few degrees above the horizon.

For our purposes, we were only concerned with shadows cast by the sun. On a rover, any lights would necessarily be mounted close to the rover's cameras, so the shadows they cast would be mostly hidden from view.

The dominant real-time shadow technique and the one we used is called shadow mapping. It involves rendering a texture map that contains the depth of objects in a scene from the viewpoint of a light source and then comparing those depths with object positions when rendering the scene from the viewpoint of the virtual camera. These comparisons reveal which surfaces are illuminated by the light and which are in shadow. There are other shadow techniques, but shadow mapping provides a popular combination of speed and flexibility.

One common challenge related to shadow mapping is providing an appropriate amount of shadow map resolution in every part of a scene. Ideally, shadow detail decreases with distance from the user's point of view such that there is a unique shadow map texel for every pixel in the final image. In a small scene such as an indoor room, a uniform distribution of shadow resolution can work well. A uniform distribution of shadow resolution is impractical from the standpoint of memory consumption when rendering a landscape out to the horizon, however.

It is possible to improve shadow quality if you can determine in advance which shadows in the scene are static and which are dynamic. If the light source moves then all the shadows it casts are dynamic. The orientation of the sun relative to the moon changes so slowly that we can assume it is effectively static in our simulation. The only moving object is the rover itself. Therefore, we can pre-compute shadows cast by the terrain (often referred to as baked shadows), and we only need to compute real-time shadows for the rover itself.

Baked Shadows— Because the sun moves slowly (∼0.5 deg/hour), we elected to pre-render the shadows generated by the terrain shadowing itself, capturing the result in a lightmap that was used by the shader to render static shadows with very little runtime overhead. This approach produces high-quality shadows over the whole terrain, with smoothly varying penumbras cast by distant features and tiny shadows cast by one pixel bumps.

To generate the lightmap, a 2-dimensional form of ray-tracing called ray-casting was used because the terrain was modeled as a height field with no overhangs. The lightmap was calculated at the same 4 cm resolution as the terrain, and the sun was modeled as an extended source with rays cast from 280 points within the sun's disk.

While the sun moves slowly, it's not fixed. The sun moves its apparent width in one hour, so any terrain pixel can potentially go from fully lit to fully unlit or vice versa within that hour. This only happens at the edges of shadows, however, and is rarely important to rover operations because the rover's path is planned to avoid these areas. Our approach to this slow movement is to generate one lightmap for every hour and to linearly interpolate linearly between adjacent lightmaps in the shader, however, this enhancement has not been implemented yet.

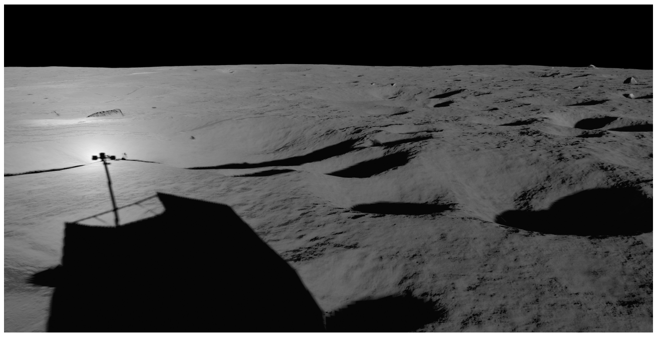


Figure 9: Terrain shadows are pre-rendered while vehicle shadows are rendered in real time.

Gazebo Real-Time Shadow Improvements— Gazebo originally used the Ogre3D default shadow mapping algorithm which is a hybrid of Parallel Split Shadow Maps (PSSM)[28] and Light-Space Perspective Shadow Maps (LiSPSM)[29]. PSSM is a form of cascaded shadow maps in which multiple depth maps are rendered in order to reduce shadow aliasing artifacts. LiSPSM is a perspective reparameterization technique that warps the shadow map in order to improve the distribution of shadow texels in the rendered scene.

While the default shadows in Gazebo worked well for typical scenes in which the sun is overhead, it produced severe artifacts for our use cases. We identified the cause of these artifacts to be LiSPSM and a hard coded axis assumption. The LiSPSM implementation in Ogre3D has a singularity when the view vector and light vector are close to coincident which causes a severe drop in shadow quality. In our scenes, real-time shadow mapping is only used for the rover and when the rover cameras are observing the rover shadow, the light vector and view vector are necessarily close to coincident. Additionally, Ogre3D assumes all scenes are oriented with the Y-axis pointing up while Gazebo adopts a Z-up convention. This decreased the usable region of each shadow map and often caused shadow edges to be more jagged than necessary and ”swim” when the viewpoint changed.

We corrected both problems by adding code to Gazebo that bypasses these troublesome parts of Ogre3D. We implemented a standard PSSM algorithm that provides a good distribution of shadow resolution and corrected all hard coded Y-up assumptions. To further refine the PSSM implementation, we added hooks to the Gazebo API to allow aspects of the shadows to be tuned through Gazebo plugins.

These modifications mitigated the most severe artifacts, but there were still some features on the rover, such as the tall, thin mast, that were not casting shadows consistently. The shader that Gazebo was using to apply shadows was smoothing shadow edges using Percentage Closest Filtering (PCF) with a 3x3 grid of sample positions. This worked well for smoothing jagged shadow edges but, due to the filter's uniformity, sometimes small features would fall through the cracks and not affect the final shadow intensity. We replaced PCF with something known as Hardware PCF, a common graphics driver hack that accomplishes similar smoothing with little performance impact. We then applied a 9-sample Poisson filter which further smoothed the shadows and does a much better job of preserving small features than the uniform filter.

Ephemeris

The position of the Sun and Earth relative to the moon is computed using the SPICE C toolkit provided by the Navigation and Ancillary Information Facility (NAIF) [4]. Using the timestamp maintained by the simulator, this toolkit generates the transformations between the Moon, the Sun, and the Earth at the current simulated time. We are not concerned about the locations of stars because they are not drawn in the simulator. By chaining these transformations with the rover's current position on the surface of the moon we are able to compute the positions of the Sun and the Earth relative to our local coordinate system and position them accurately in the simulated environment.

Rover Lights

Gazebo was extended to allow lights to be attached to articulated rover components specified in the Unified Robot Desription Format (URDF), which is the file format commonly used to describe ROS robot models. Ogre3D material files were used to pass light parameters from Gazebo to our GLSL shader. Our terrain shader simulates up to four spotlights having brightness, cone angle, and distance falloff that can be specified as part of the Gazebo light description. We also added a texture projection feature for simulating the detailed pattern projected by a real-world light source. This infrastructure gives us the ability to simulate a variety of lighting hardware and placement before settling on a final solution.

[4]https://naif.jpl.nasa.gov/naif/toolkit.html

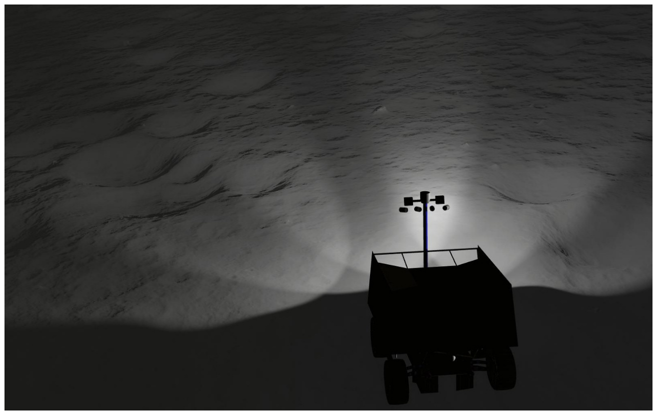


Figure 10: Rover driving in shadowed crater

Photometry

The reference mission required training in use of photographic systems coupled with active illumination for driving. Developments were made to model complex illumination effects from natural sources, rover-based lighting, and sensor optics in order to produce an accurate camera image. To accomplish this in a real-time manner, we used physically-based raytracing to produce exemplar scenes which were then approximated with shader tricks.

The dynamic range of scenes on the Moon drives the performance of stereo safeguarding, human situational awareness, and the design of rover lighting. To accurately simulate dynamic range, global illumination must be modeled. If direct sunlight were the only source of illumination, shadowed areas would be pitch black. However, we know this is not the case from images of the Lunar surface, despite the high dynamic range. Light reaches shadowed areas from multiple natural sources including diffuse interreflection of the Lunar surface, Earthshine, and starlight. Using ray tracing, we simulated the contribution of indirect scattering in ideal surface craters given particular phase angle combinations. Our findings indicate that an average of 3% of the direct illumination reflects into shadows in polar scenes. This effect was approximated in rendering by illuminating the scene with a low-intensity virtual source opposing the sun in azimuth. Analytically we also showed that light from the star field is insignificant and that Earthshine, while potentially a fraction of indirect scattering, is complexly dependent on the phase of Earth. Thus, neither these other sources were implemented.

Camera Exposure and Bit Depth

In order to take advantage of the dynamic range of our simulated scene, we extended Gazebo to emulate camera exposure and enabled high bit depth rendered images. Our modifications to Gazebo permit plugin authors to inspect all shaders in the scene, search for shader variable names, and set uniform variable values. We emulate exposure with a linear multiplier on the output color component of fragment shaders in the scene. The camera exposure Gazebo plugin has a ROS interface to accept camera exposure commands and modifies a commonly named exposure variable in all shaders in the scene.

A standard bit depth of 8 is adequate for humans viewing images on a nonspecialized monitor, but computer vision algorithms can make use of higher depths. The reference mission cameras under consideration have a bit depth of 12,

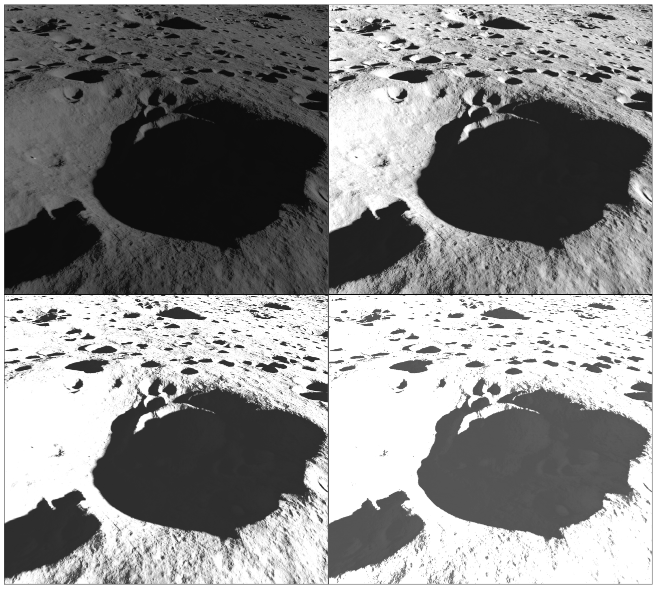


Figure 11: Four camera exposure levels showing dynamic range and approximation of global illumination.

so we need to render images of the same depth to accurately simulate our intended image pipeline. We extended Gazebo to capture 16-bit integer images. Internally, Gazebo already supports floating point textures as they are used to store data generated by depth and other image-based range sensors. So the task of simulating 16-bit grayscale and RGB camera sensors was relatively straightforward, essentially by rendering to textures in the these formats. Direct 12-bit image capture is not an option in OpenGL, but a 16-bit image can be downsampled to 12-bit as a post-render step.

6. Vehicle Simulation

The first prototype of the Resource Prospector robot (Fig. 12a) is an explicit steered vehicle, which is capable of being skid-steered. During the early stages of the project we were exploring which driving style (skid-steered vs explicit steering) would result in more effective operations.

Explicit steering permits the robot to drive in any direction and maintain the orientation of the solar panels to maximize the insolation of the solar panels. Further, the ability of the vehicle to "walk" out sand traps improves the robustness of the vehicle to environmental hazards. The skid-steered version of RP would present a simpler control problem, and simpler mechanical construction, reducing cost and complexity of the vehicle.

Simulating Vehicle Mobility

We developed the physical simulation of the robot in Gazebo in order to test skid-steered and explicit steered driving. This was to aid the development of the mission's concept of operations.

It had yet to be determined what steering mechanism should be used for the vehicle. The RP team wanted to evaluate skid-steered and explicit steered vehicles. To do this we simulated two different versions of the robot. The first version of the robot was simulated using the Husky Gazebo model [30], the second version was simulated with a model of NASA Ames' KRex2.

The Husky robot is only capable of skid-steering, which does not reflect the full capabilities of the RP vehicle. However, since the RP vehicle is capable of skid steering, the Husky provided a platform to quickly mount the sensors needed for conducting missions operations tests and provide a reasonable approximation of vehicle motion.

To model explicit-steered driving, we used a modified version of the NASA Ames Intelligent Robotics Group's KRex2 robot. The KRex2 robot is, like RP, an explicit-steer vehicle. We modified the KRex2 model to reflect the offset steering which is present on the RP rover, but not on the actual KRex2 model.

However, where RP can independently lower and raise its wheels, KRex2 cannot, and relies on a differencing bar to maintain contact with the terrain, as shown in Fig. 13. We chose to use a differencing system rather than to fully model the RP suspension, because it was a simpler to model, and did not involve modelling the control system of the RP rover in order to achieve full terrain contact with the wheels.

Using KRex2 as a proxy for the RP vehicle enabled the use of the explicit steering capacity for missions operations testing. Using the URDF language to describe the robot, it was easy to modify the sensors that were added to the earlier Husky model to the modified KRex2 model.

To both of these vehicles we added a stereo camera pair that was mounted on a gimbal at the top of the mast, seen in Fig. 12a, as well as the solar-panel superstructure. We also added a simulated IMU and tracked wheel odometry for use in pose estimation.

It was important to be able to place the robot superstructure on the vehicle, which was made easy with the URDF language. By placing the envelope of the robot structure in the scene, we could test how much the field of view of the operators would be occluded.

Wheel Slip

Localization algorithms that employ wheel odometry use measurements of wheel rotation to estimate robot motion. These algorithms work best when the wheel/terrain relative shear velocity (also known as wheel slip) is small. In most cases, wheel slip can occur when a sufficient torque is applied to a wheel, though it can be complex to model the behavior [31]. To reduce the computational burden, the Open Dynamics Engine (OpenDE) physics solver is used with the Gazebo simulator. This solver treats the 3D terrain as a rigid surface and discretizes contact as a wrench acting at contact points with a specified normal direction. At each contact point, a Coulomb friction model limits the maximum tangential force to the product of the normal force and a surface-dependent friction coefficient. To further reduce the computation burden, the ”friction cone” is approximated as a ”friction pyramid” [32]. The axes of the friction pyramid are aligned with the longitudinal and lateral axes of each wheel so that anisotropic parameters can be specified for the wheel axes.

The OpenDE friction model has parameters for the friction coefficient and slip compliance and are illustrated in figure 14. The ratio of tangential friction force to normal force is limited by the wheel/soil friction coefficient, and its behavior for small amounts of slip is governed by the slip compliance.

(a) The Resource Prospector (RP) robot is capable of independently steering each of its wheels. Additionally, each wheel can be raised and lowered independently, permitting the robot to "walk" out of sand traps.

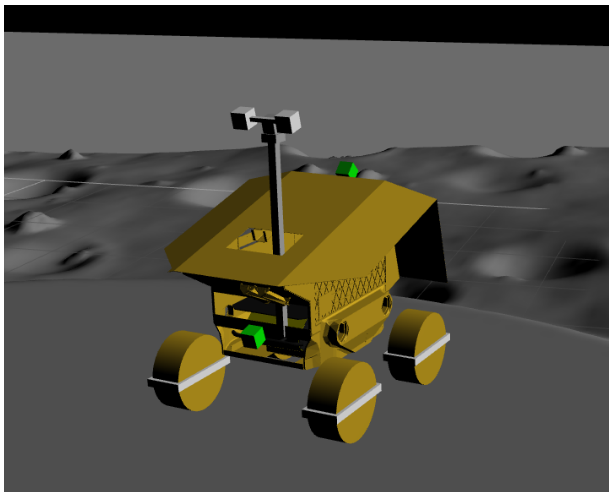

(b) We used the robot modelling framework of Gazebo to generate simulated versions of the RP rover. First, we used a skid steer model of the rover, based on the Husky robot model. Second, we based the steering mechanism on a model of the KRex2 robot.

Figure 12: The RP prototype (Fig. 12a) was constructed at Johnson Space Center, the rover was simulated (Fig. 12b) using Gazebo and ROS

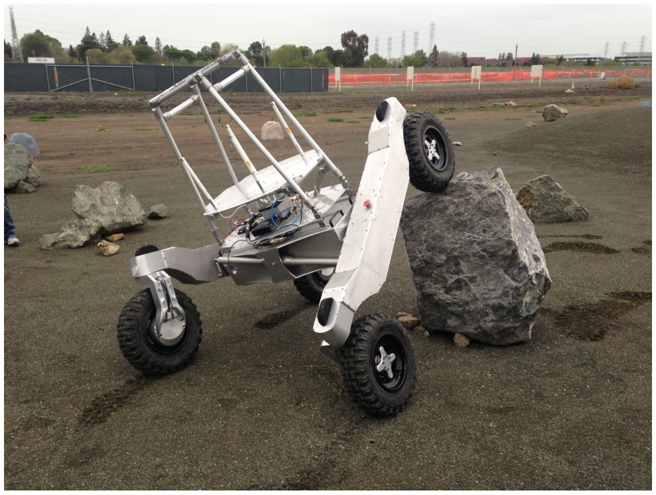


Figure 13: KRex2 exercising the differencing bar to maintain terrain contact for all wheels.

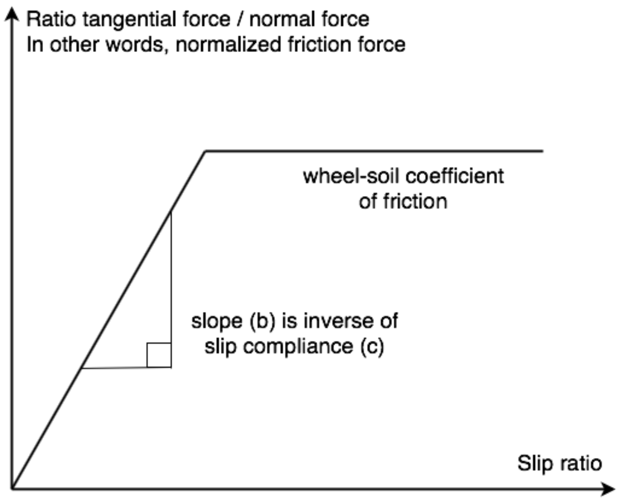


Figure 14: Wheel/soil interaction model used in the simulator

The slip compliance has units of the inverse of a damping coefficient, such that the value of 0 leads to infinite damping and no slip, while positive values allow for some slip. The slip compliance corresponds to the inverse slope of the curve near the origin.

In the real world, an input drive command will result in a non-zero slip of the rover wheel on loose terrain which will generate drift in localization algorithms. In order for this drift to be plausible in simulation, a wheel slip plug-in was developed and implemented for Gazebo that adjusts the slip compliance parameter based on a test results from two tests: RP single wheel drawbar pull test and RP Lunar mass equivalent test unit, MGRU (see figure 15) in the SLOPElab facility at NASA Glenn Research Center (GRC) [33]. The simulant for these test was GRC-1 [34], which has similar mechanical characteristics as the Lunar regolith. The slip compliance tuning parameter enables simulation of loose material generating slip greater than zero on shallow angle slopes.

Although not perfect at simulating the wheel/soil interaction, this plug-in enables the simulated rover to have high level slip behavior relatively close to what would be expected on Lunar terrain. In order to increase the fidelity and give more realism for wheel sinkage and rover embedding, this simple model would need to be replaced by a discrete element method (DEM) model which is very computationally expensive [35]. Additionally, existing DEM solutions lack validations for lunar terrain.

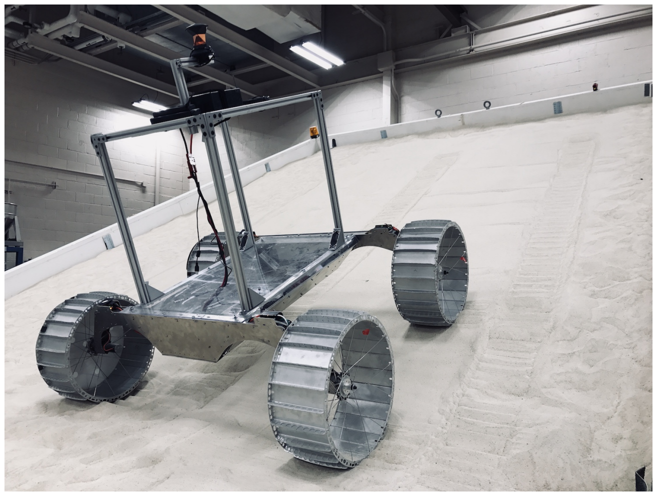


Figure 15: A picture of the RP Lunar mass equivalent rover at the SLOPElab in NASA GRC.

7. Flight Software Prototype

A major aspect of the Resource Prospector project was the use of distributed autonomy. The decision to distribute the software was possible because of the high communications availability between the rover and the Earth.

The autonomy software for the vehicle was divided into on-board (Flight) software, which was responsible for safeguarding the vehicle and executing simple waypoint following, and the off-board (Ground) software, which acted as assistive tools for the remote human operators, illustrated in Fig. 16.

In order to test operations we used a combination of ROS software components and NASA-developed software to build an end-to-end mission simulator. As illustrated in Fig. 16, we split the software into on-board (Flight) and off-board (Ground) software components.

To validate the distributed autonomy concept we needed to model the anticipated communications delays and interruptions in the Earth to Moon communications link. We built a communications bridge between the operations center and the simulated Lunar rover.

Autonomy algorithms (mapping, planning) carry a substantial burden for validation and verification. Operating the higher-level algorithms on the ground as assistive tools, instead of on-board the rover in a decision-making capacity, mitigates the validation and verification burden, enabling faster and more cost-effective deployment and development.

Flight/Ground Software split

As discussed in Section 2 we divided the rover software into low-level control and safeguarding components, which were deployed on-board the rover ("flight software"), and higher-level autonomy or assistive tools which operated on Earth ("ground software"). The division of those software components are given in Fig. 16.

The role of the rover was played by the modified simulated RP rover, as described in Section 6. The flight software consisted of the components that were designated the minimum viable components to complete the mission. The flight software components include:

Rover Kinematics The software that translates vehicle body motion commands into motor commands. We modelled the Rover Kinematics module with the move_base[5] module with additional explicit steering control.
Mobility The mobility module is responsible for closed-loop control on waypoint navigation. This functionality was also supplied by the move_base module.
Localization The pose estimation solution which integrated wheel odometry, a star tracker, and the IMU. We modelled the Localization module with ROS' robot_localization[6] package.
Camera/Gimbal Pointing The RP navigation cameras and the high-gain antenna are mounted on gimbals on the mast. This module is responsible for closed-loop control of those gimbals. In our simulation we have primarily concerned ourselves with camera pointing and not the effects of antenna pointing, although planned improvements to communications system simulation fidelity would take antenna pointing into account.
Wheel, IMU, Gimbal, and Camera I/O These modules responsible for low-level hardware interfaces. This functionality was supplied by the ros_control[7] module.
Rover Mode Manager This software component is responsible for preventing the robot from executing illegal or unsafe actions, and ensuring that all on-board modules are in the appropriate operating mode. While this module did exist on the prototype RP rover, we did not implement it for the simulated rover operations, and as such we did not seek a ROS equivalent.
Virtual Bumper The virtual bumper is designed to keep the rover safe from navigation obstacles, as per [36]. This module was not implemented in the simulated rover operations. The prototype virtual bumper implemented for the RP project is described in [37], with a simple safeguarding scheme described in [38].

The ground software components represented higher-level software, which integrated data to improve the situational awareness of the rover drivers. The particular modules included in the ground software are:

Advanced Localization The advanced localization module integrated the on-board pose estimates with terrain relative navigation, mono- and stereo-scopic visual odometry. We again used the robot_localization module to integrate the different sources of motion estimation. We did not implement the terrain relative navigation in this simulated version, however, the extensible nature of the robot_localization package would make this addition straightforward.
Stereo Reconstruction Produces a metric reconstruction of the scene from the rover's stereo cameras. We used OpenCV[8] to perform Stereo Reconstruction.
Mapping Uses the estimated pose and reconstructed scenes to build DEMs of the Lunar surface.
Hazard Detection Provides two forms of hazard detection. The module processes *a prior* elevation maps to identify geometric hazards to the rover, as well as point clouds from the stereo cameras to identify potential vehicle hazards that may not appear in the terrain geometry. A NASA developed terrain analysis module produced a cost map which was injected into the ROS costmap_2d[9] module to provide this functionality.
Planning The planning module takes goals issued by the

[5]http://wiki.ros.org/move_base
[6]http://wiki.ros.org/robot_localization
[7]http://wiki.ros.org/ros_control
[8]http://wiki.ros.org/vision_opencv
[9]http://wiki.ros.org/costmap_2d

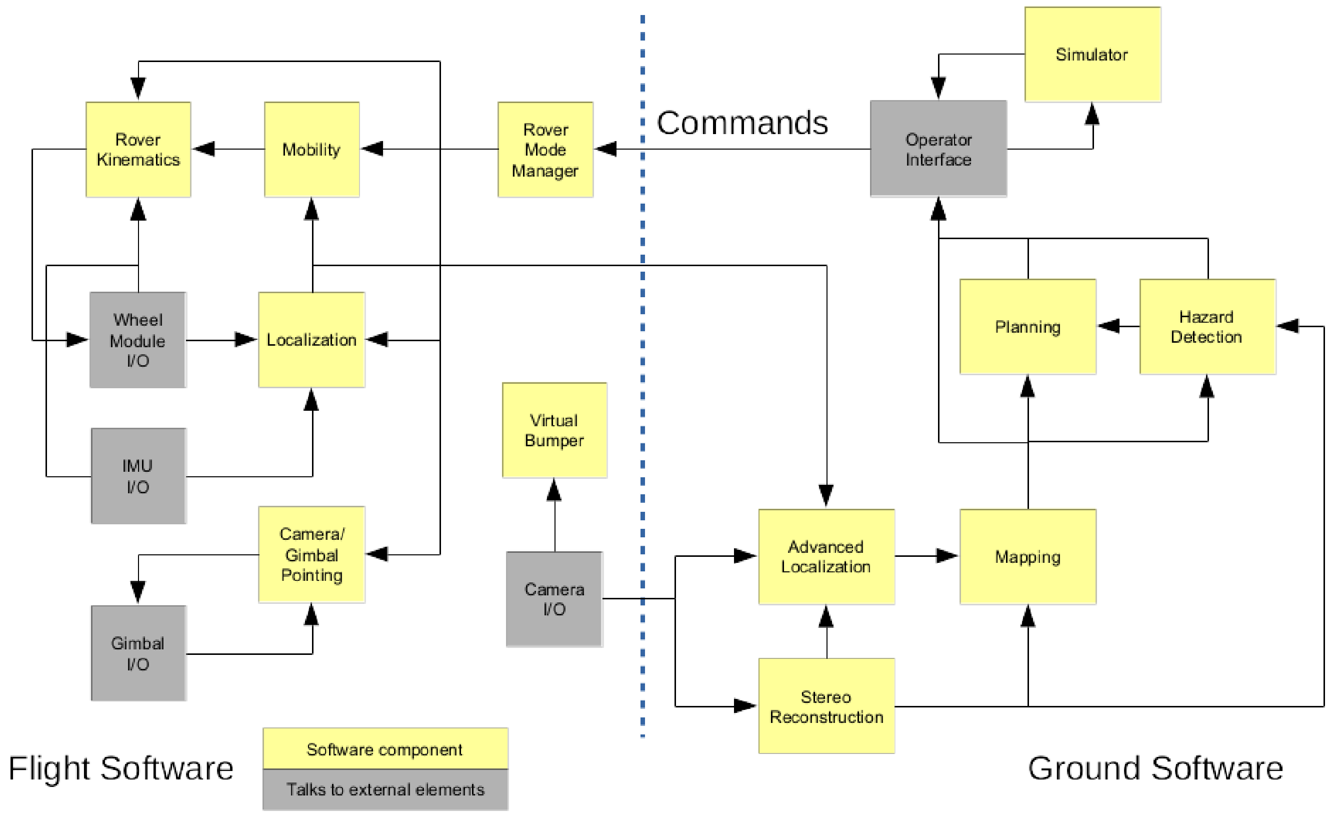


Figure 16: On the left hand side we include the software modules that would operate on-board the robot. These include low-level components, including waypoint following and simple pose estimation fusing wheel odometry and the vehicle's IMU. On the right hand side of the diagram are the more computationally intensive algorithms, which are proposed to run on the Earth. Exploiting greater computing resources on the ground lets the robot access algorithms which would not be feasible on on-board radiation-hardened computers.

drivers and determines a sequence of waypoints which safely navigate the cost map produced by the Hazard Detection model. We used the move_base module to plan trajectories between waypoints.

Simulator The simulator acted as an assistive tool for the operators, to ensure that the commands being sent to the rover were the commands that were intended. We used the Gazebo simulator with the modifications described in this paper.

Operator Interface The operations interface was provided through two modules. Direct and immediate control of the rover was accomplished through NASA Ames Intelligent Robotics Group's VERVE software. Overall missions operations, and plan-level contextual information was provided through the WARP system. Both of these modules are described in greater detail below, and communicated to the ROS system using a software bridge developed for mission simulation experiments.

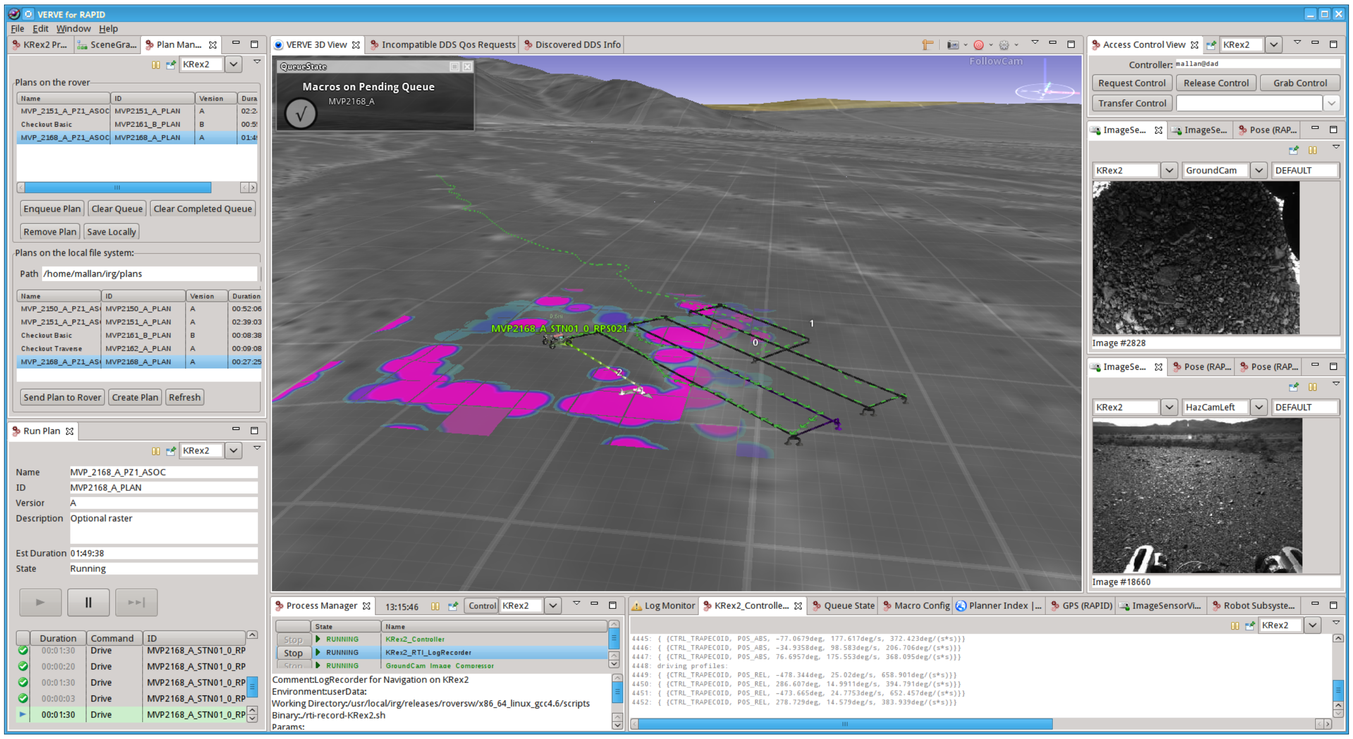
8. Operations Software

To enable teams of flight controllers to participate in driving simulations and to improve simulation fidelity, we interfaced the rover model with the mission control software intended for RP. This consisted of the robot driving software, VERVE, the science instrument monitoring displays, LabVIEW, and an integrated mission control software framework (WARP) that provides a wide variety of functions. To make these components speak to each other we developed a bridge that translated messages between the different communications protocols used by the operations software and the simulated rover. The bridge primarily converted between ROS and the RAPID communications protocol[39] used by some NASA-developed software.

In addition to protocol translation, the bridge models time delay and jitter to serve as a space comms emulator. Each data topic can be assigned a fixed time delay and/or assigned to a logical data channel with set bandwidth. The logical data channels use a token bucket to restrict data flow and data payload size is either taken directly from the ROS message or from a configuration file for cases where ROS payload size is significantly different from its flight counterpart. The bridge reproduces the expected time-of-flight, bandwidth, and ground processing delays under nominal conditions so that simulated operations have an appropriate cadence. Future work on the bridge will introduce off-nominal conditions such as comm dropouts and variable bandwidth.

VERVE— The rover driving software is based on the NASA Visual Environment for Robotic Virtual Exploration (VERVE) [40]. VERVE provides an interactive 3D user interface for remote monitoring and commanding of robotic systems (Fig. 17) and has been used for several Lunar analog field tests [41] [42] as well as commanding robotic assets from the International Space Station [43].

LabVIEW and WARP—LabVIEW[44] is commercial software and was used during the development and testing of



Controller: mallan@dad

Plans on the rover

Name	ID	Version	Dura
MVP_2151_A_PZ1_ASOC	MVP2151_A_PLAN	A	02:2
Checkout Basic	MVP2161_B_PLAN	B	00:5!
MVP_2168_A_PZ1_ASOC	MVP2168_A_PLAN	A	01:4

Plans on the local file system:

Path /home/mallan/irg/plans

Name	ID	Version	Duration
MVP_2150_A_PZ1_AS	MVP2150_A_PLAN	A	00:52:06
MVP_2151_A_PZ1_AS	MVP2151_A_PLAN	A	02:39:03
Checkout Basic	MVP2161_B_PLAN	B	00:08:38
Checkout Traverse	MVP2162_A_PLAN	A	00:09:08
MVP_2168_A_PZ1_AS	MVP2168_A_PLAN	A	00:27:25

Name	MVP_2168_A_PZ1_ASOC
ID	MVP2168_A_PLAN
Versior	A
Description	Optional raster
Est Duration	01:49:38
State	Running

	Duration	Command	ID
✓	00:01:30	Drive	MVP2168_A_STN01_0_RP
✓	00:00:20	Drive	MVP2168_A_STN01_0_RP
✓	00:01:30	Drive	MVP2168_A_STN01_0_RP
✓	00:00:03	Drive	MVP2168_A_STN01_0_RP
▶	00:01:30	Drive	MVP2168_A_STN01_0_RP

13:15:46

	State	Name
Stop	RUNNING	KRex2_Controller
Stop	RUNNING	KRex2_RTI_LogRecorder
Stop	RUNNING	GroundCam Image Compressor

```
Comment:LogRecorder for Navigation on KRex2
Environment:userData:
Working Directory:/usr/local/irg/releases/roversw/x86_64_linux_gcc4.6/scripts
Binary:./rti-record-KRex2.sh
Params:
```

```
4445: { {CTRL_TRAPECOID, POS_ABS, -77.0679deg, 177.617deg/s, 372.423deg/(s*s)}}
4446: { {CTRL_TRAPECOID, POS_ABS, -34.9358deg, 98.583deg/s, 206.706deg/(s*s)}}
4447: { {CTRL_TRAPECOID, POS_ABS, 76.6957deg, 175.553deg/s, 368.095deg/(s*s)}}
4448: driving profiles:
4449: { {CTRL_TRAPECOID, POS_REL, -478.344deg, 25.02deg/s, 658.901deg/(s*s)}}
4450: { {CTRL_TRAPECOID, POS_REL, 286.607deg, 14.9911deg/s, 394.791deg/(s*s)}}
4451: { {CTRL_TRAPECOID, POS_REL, -473.665deg, 24.7753deg/s, 652.457deg/(s*s)}}
4452: { {CTRL_TRAPECOID, POS_REL, 278.729deg, 14.579deg/s, 383.939deg/(s*s)}}
```

Figure 17: VERVE was used as the robot control interface for the Mojave Volatiles Prospector (MVP) project. MVP tested high-tempo prospecting operations which would be applicable to missions like Resource Prospector.

RP's instruments. The simulation provides inputs that flows through the real data processing pipeline and into the LabVIEW displays (Fig. 18).

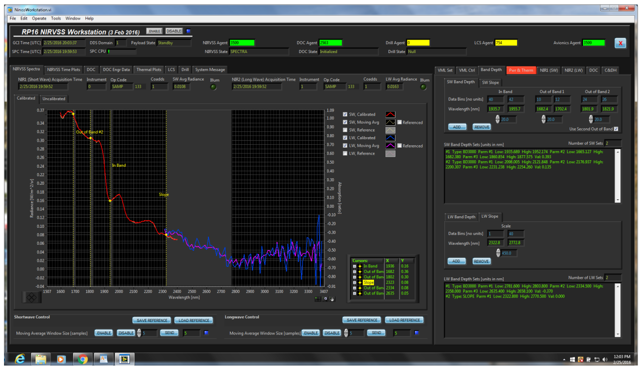


Figure 18: Instrument LabVIEW displays

WARP [45] provides a wide-variety of displays needed by flight control teams including real-time and historical telemetry display, caution and warning and event display, access to telemetry dictionaries, shift logs, timelines and activity plans, support for multiple missions, and base maps and map layers including the rover's path and heat map displays of instrument data (Fig. 19). WARP is built on OpenMCT [46], a web-based framework developed by NASAs Advanced Multi-Mission Operations System (AMMOS), Ames Research Center (ARC) and the Jet Propulsion Laboratory (JPL) and is used by several JPL missions including the next Mars rover (M2020).

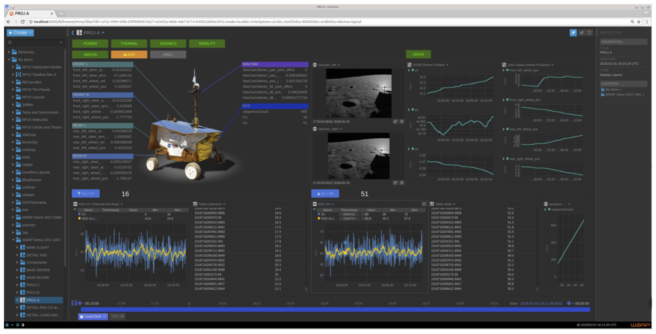
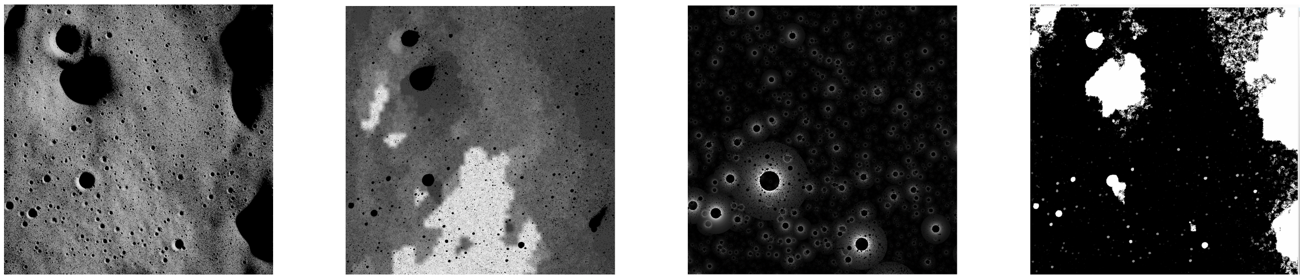


Figure 19: Web-based mission control displays (WARP/OpenMCT). Figure shows one among many display pages available.

Science Simulation and Software

The goal of Resource Prospector was to take measurements to estimate the water content at and under the Lunar surface over a 2500 square meter area and to repeat this data collection in several areas that differ by temperature profile. This is the scale over which water would be mined to support a potential, future Lunar base. The real-time monitoring of instrument data had two purposes. The first was to ensure the quality of the data collection and to adjust or correct instrument settings quickly to that end. The second was to make tactical decisions about where data collection time was best spent, because the total mission duration available to a solar powered rover at polar sites does not allow a completely methodical data collection pattern.

World Model—The science simulation consists of a world model and several instrument models. The world model



(a) Surface Temperature (b) Depth of dry layer (c) Subsurface Ice Concentration (d) Surface Frost

Figure 20: Data layers describing what instruments could see as the rover drives

describes the 'truth' that RP was intended to estimate in the form of several interrelated physical quantities that are measured by RP's instruments. Two of the instruments, NSS and NIRVSS, were introduced above. NSS measures hydrogen concentration in the regolith immediately in front of the rover, and NIRVSS measures surface frost, minerology, and the concentration of ices in regolith brought up by the drill. NIRVSS also contains a Longwave Calibration Sensor (LCS) that measures surface temperature.

The world model provides data for these instruments that varies as the rover moves in ways that must be consistent with physics and with theories about how ices have been emplanted or removed. It did this by modeling the distribution of several physical quantities, some as 2D functions (variation over the surface) and others as 3D functions (variation within the top meter of regolith).

The quantities modeled are: (a) the surface temperature, (b) the area concentration of water ice as a surface frost, (c) the area concentration of OH adhered to the surface, (d) the thickness of the upper layer of regolith that must be dry based on its known thermal history, (e) the water ice concentration in the layer below that as a weight percentage, (f) the volumetric concentration of hydrogen in hydrogen-bearing minerals, and (g) the surface distribution of several minerals. The most important of these were the water ice surface and volumetric concentrations. The others were included to present a consistent picture and to provide realistic sources of noise and other variations that could make interpreting the data more difficult.

The 2D and 3D functions implemented as a set of 30720 x 30720 arrays generated before simulation and representing a 4cm grid covering a 1.5 km2 area that would be driven over during the simulations. This approach was used because the distributions of the quantities (except the minerals) were generated by modeling the physical processes that could have created them over time. Mineral concentrations were modeled as Perlin noise.

A sequence of arrays was generated, with each array calculated using only the arrays before it. The DEM was first, followed by the lightmap and then the surface temperature. Regolith is an excellent insulator, so the surface temperature can be approximated as a function of sun angle and shadows (ignoring Earthshine and heat rising from the interior). Since the DEM was synthetic, the locations and sizes of the craters was known. OH concentration is modeled a function of surface temperature. The thickness of the dry layer is a much more complex function of the time history of the surface temperature, and Earthshine, heat rising from the interior and the exchange of infrared energy between nearby parts of the surface (crater interiors) very much do need to be modeled; this array came from published work[47]. The volumetric distribution of water ice was inspired by a model of impact gardening published in [48] and is a function of the distance to the rim of nearby craters above a certain size. Surface frost only occurs in permanent shadow and its concentration is a function of the contiguous area of each frost patch. As mentioned above, mineral concentrations were modeled as Perlin noise and could have been calculated at simulation time, but they were converted into arrays to keep the implementation more uniform. Fig. 20 shows several of these arrays.

Instrument Models—During simulation, the rover's ground truth pose was used as a lookup into these arrays and the results were broadcast at several Hz to the instrument models. The instrument models added noise and converted these quantities into the data format generated by the sensors within the instruments. This allowed most of the real data processing pipeline for each instrument to be used.

Science Data Displays—Instrument data was presented to the scientists using the displays that RP intended to use in flight. This involved a combination of LabVIEW displays, which were primarily used by the scientists responsible for the health of the instrument and the quality of the data collection, and a web-based set of displays [46] that were primarily used by scientists responsible for data interpretation and tactical decision-making. (Both kinds of displays were used by both groups.) The web-based displays presented the most important information available in the LabVIEW displays and also presented instrument data as heat maps on top of 2D site maps so scientists could see how the data was distributed spatially. This followed the approach of xGDS [40] and xGDS components would have been used during flight. Fig. 18 and Fig. 19 show examples of these displays.

9. Experiments

Stereo Visual Odometry

One of the goals of the visual simulation was to provide realistic data for algorithmic development. One such algorithm is the stereo visual odometry we implemented for Resource Prospector's ground software to aid in navigation. Our approach generates 3D point clouds from successive stereo camera images, aligns them using the Point Cloud Library's (PCL)[10] implementation of the Iterative Closest Point algorithm, and uses the transforms between point clouds to track

[10]http://pointclouds.org/

changes in the rover's movements. This module interacts with other software and sensors to estimate the rover's absolute pose.

To refine the mission's concept of operations, we wanted to know how much overlap is required between camera images for stereo visual odometry to be effective. The easiest way to lose overlap between successive images is by yawing the camera or rover, so that is the first test we ran. We learned that with camera rotations of 50-degrees or less, stereo visual odometry yields more accurate results than the fused wheel and star tracker odometry we had been using (shown in Fig. 21).

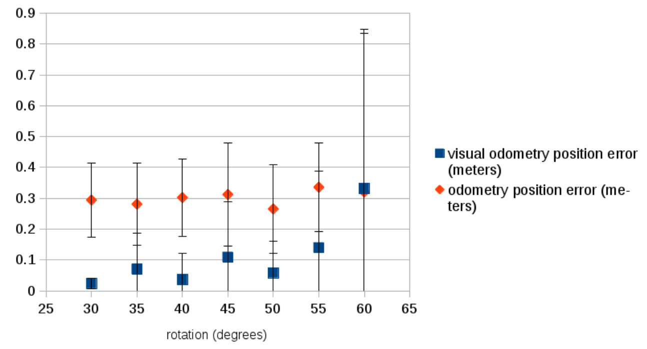


Figure 21: Position error vs. camera rotation angle for stereo visual odometry and wheel-and-star-tracker odometry

There are occasionally incorrect point cloud alignments, informing the need to improve performance or to estimate alignment quality for rejection. It is important to note the results for stereo visual odometry are preliminary; however, our simulator has provided the only opportunity to develop and test such modern robotic approaches with any confidence - Apollo surface images being non-digital and terrestrial analogs being unrealistic. There are further tests to run and improvements to be made to both our visual simulation and visual odometry. For example, after a specific camera is chosen for the mission we will be able to refine our camera simulation. This will affect point cloud generation and require us to run these tests again.

We also learned that using rover lights in shadowed areas hurts our visual odometry results. As expected the quality of our stereo point clouds is greatly reduced when the light sources are near the cameras because the Hapke reflectance model simulates strong, contrast-reducing backscatter. We have not attempted to quantify results in this area as it is still considered ripe for refinement.

Wheel Slip

In order to characterize the `wheel_slip_plugin`, we set up a test harness to gather data on wheel slip behavior over multiple runs. The purpose of this effort was twofold; to determine whether the overall slip behavior was in line with expectations for wheels in contact with unconsolidated soil, and to determine what parameter values should be used for the plugin to most closely match results from the physical testbed.

We created a Gazebo world with ramps from 0 to 30 degrees

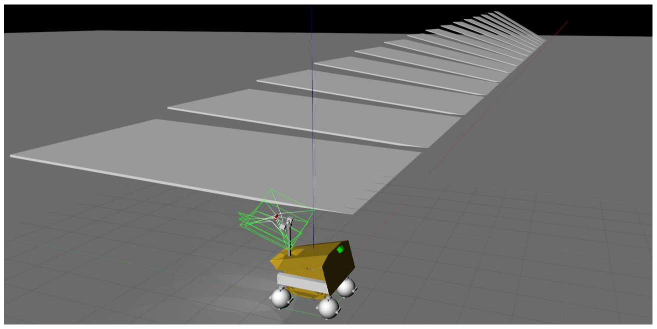


Figure 22: Gazebo simulation for the slip vs slope test with the RP rover.

at 2 degree increments (see figure 22). A `rostest`[11] script would spawn the rover at the base of each ramp and command it to drive 10 meters forward. Wheel slip was measured by comparing dead reckoning distance driven to ground truth and test runs covered sets of parameter values for longitudinal compliance, physics engine solver iterations, and drive speed.

Overall, the `wheel_slip_plugin` demonstrated the expected behavior and compared favorably with physical testbed results as can be seen in figure 23, which compares wheel slip measured from MGRU with the `wheel_slip_plugin`.

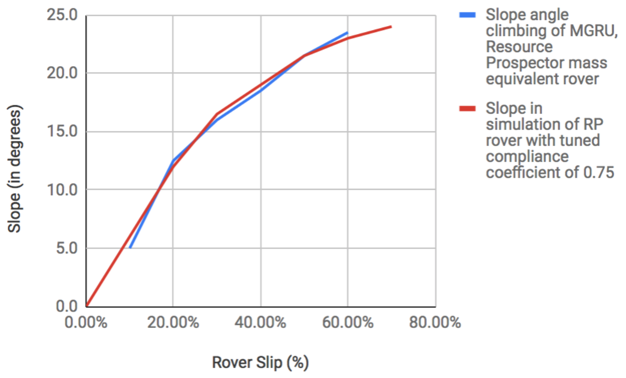


Figure 23: Comparison of the RP Lunar mass equivalent rover slip on GRC-1 simulant and RP rover slip in the Gazebo simulator.

However, our test results show that at certain angles and compliance values, there is a wide variation in measured wheel slip (see figure 24). Those disparities are particularly large on a flat terrain. The cause of this variation is unknown and is currently under investigation.

10. CONCLUSION

This project represents a fusion of COTS, open-source software, with domain specific knowledge that resulted in the production of a high-fidelity Lunar rover driving simulation. It presents an approach to rapidly developing visual and physical simulations that enable the holistic evaluation of a mission, from concept of operations to algorithm and mechanism design.

[11] http://wiki.ros.org/rostest

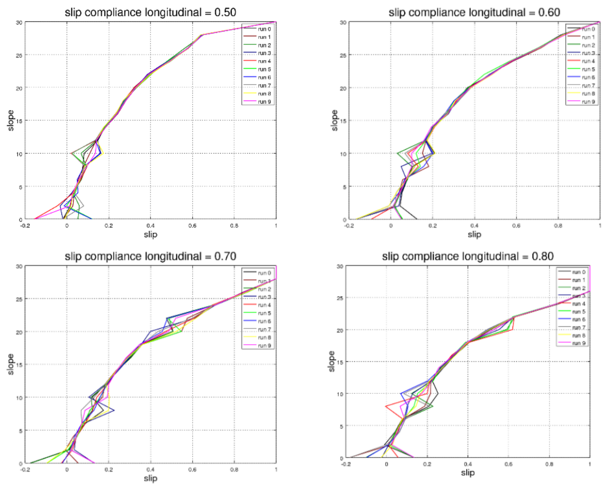


Figure 24: Graph of the slip of the RP rover in Gazebo showing the disparities in the amount of slip.

The simulator permitted us to develop and test three major areas for the mission operations. First, we were able to simulate the physical environment the robot would be operating in. This includes medium fidelity wheel/terrain interactions and high fidelity visual simulation of the moon. Second, due to the increased physical verisimilitude, we were able to test rover navigation and perception algorithms on the simulation data with increased confidence in the accuracy of the results. Third, by virtue of being a high-fidelity simulation of rover mechanism and software behaviour, we were able to close the loop for the communications and operations infrastructure for the mission. Then we were able to test different concepts of operations, and even rover configurations rapidly, and build confidence in the mission design.

11. Acknowledgements

The authors would like to thank the NASA Human Exploration and Operations Mission Directorate (HEOMD) and Advanced Exploration Systems (AES) division for the opportunity to advance the Resource Prospector mission concept. Many thanks are due to Tony Colaprete and Rick Elphic for sharing their knowledge of selenology and giving guidance on Lunar appearance modeling. Finally, thanks to all the mission element leads who displayed extraordinary leadership and to Mission Systems Engineering who tied it all together.

References

[1] A Colaprete, R Elphic, D Andrews, J Trimble, B Bluethmann, J Quinn, and G Chavers. Resource prospector: An update on the lunar volatiles prospecting and isru demonstration mission. 2017.

[2] David Vaniman, Robert Reedy, Grant Heiken, Gary Olhoeft, and Wendell Mendell. The lunar environment. *The lunar Sourcebook, CUP*, pages 27–60, 1991.

[3] Douglas E McGovern. Human interfaces in remote driving. Technical Report SAND88-0562, Sandia National Laboratories, 1988.

[4] Alonzo Kelly, Nicholas Chan, Herman Herman, Daniel Huber, Robert Meyers, Pete Rander, Randy Warner, Jason Ziglar, and Erin Capstick. Real-time photorealistic virtualized reality interface for remote mobile robot control. *The International Journal of Robotics Research*, 30(3):384–404, 2011.

[5] Daniel R Andrews. Resource prospector (rp)-early prototyping and development. In *AIAA SPACE 2015 Conference and Exposition*, page 4460, 2015.

[6] A Colaprete, RC Elphic, D Andrews, G Sanders, A McGovern, R Vaughan, J Heldmann, and J Trimble. Resource prospector: Mission goals, relevance and site selection. 2015.

[7] Jay Trimble and Carvalho Robert. Lunar prospecting: searching for volatiles at the south pole. In *14th International Conference on Space Operations*, page 2482, 2016.

[8] Becky L Hooey, Jason JNT Toy, Robert E Carvalho, Terrence Fong, and Brian F Gore. Modeling operator workload for the resource prospector lunar rover mission. In *International Conference on Applied Human Factors and Ergonomics*, pages 183–194. Springer, 2017.

[9] Stephan Ulamec, Jens Biele, and Ed Trollope. How to survive a lunar night. *Planetary and space science*, 58(14-15):1985–1995, 2010.

[10] Nathan Otten, David Wettergreen, and William Whittaker. Strategic autonomy for reducing risk of sun-synchronous lunar polar exploration. In *Field and Service Robotics*, pages 465–479. Springer, 2018.

[11] Nathan P Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, volume 4, pages 2149–2154, 2004.

[12] Steffi Paepcke and Louise Poubel. Whats new in gazebo? upgrading your simulation user experience! In *ROSCon*, 2016.

[13] Carlos E Agüero, Nate Koenig, Ian Chen, Hugo Boyer, Steven Peters, John Hsu, Brian Gerkey, Steffi Paepcke, Jose L Rivero, Justin Manzo, et al. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *IEEE Transactions on Automation Science and Engineering*, 12(2):494–506, 2015.

[14] Kimberly A Hambuchen, Monsi C Roman, Amy Sivak, Angela Herblet, Nathan Koenig, Daniel Newmyer, and Robert Ambrose. Nasa's space robotics challenge: Advancing robotics for future exploration missions. In *AIAA SPACE and Astronautics Forum and Exposition*, page 5120, 2017.

[15] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*. Kobe, Japan, 2009.

[16] Surveyor Investigator Teams. Surveyor project part 2 - science results final report. In *Technical Report JPL-TR-32-1265-PT-2*. Jet Propulsion Lab, 1986.

[17] Caleb I. Fassett and Bradley J. Thomson. Crater degradation on the lunar maria: Topographic diffusion and the rate of erosion on the moon. *J. Geophysical Research Planets*, 119:2255–2271, 2014.

[18] SM Parkes, Iain Martin, Martin Dunstan, and D Matthews. Planet surface simulation with pangu. In *Space OPS 2004 Conference*, page 389, 2004.

[19] Gavin SP Miller. The definition and rendering of

terrain maps. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 39–48. ACM, 1986.

[20] Uday J Shankar, Wen-Jong Shyong, Thomas B Criss, and Dewey Adams. Lunar terrain surface modeling for the alhat program. In *Aerospace Conference, 2008 IEEE*, pages 1–10. IEEE, 2008.

[21] Glenn A Fry, CS Bridgman, and VJ Ellerbrock. the effects of atmospheric scattering on binocular depth-perception. *Optometry and Vision Science*, 26(1):9–15, 1949.

[22] Grant Heiken, David Vaniman, and Bevan French. *Lunar Sourcebook - A user's guide to the moon*. Cambridge University Press, 1991.

[23] Bruce Hapke. Bidirectional reflectance spectroscopy: 4. the extinction coefficient and the opposition effect. *Icarus*, 67(2):264–280, 1986.

[24] Michael Shepard and Paul Helfenstein. A test of the hapke photometric model. In *Journal of Geophysical Research*, volume 112, 2007.

[25] Bruce Hapke, Brett Denevi, Hiroyuki Sato, Sarah Braden, and Mark Robinson. The wavelength dependence of the lunar phase curve as seen by the lunar reconnaissance orbiter wide-angle camera. *Journal of Geophysical Research: Planets*, 117(E12), 2012.

[26] Uland Wong. Lumenhancement: Exploiting appearance for planetary modeling. In *PhD Thesis*. Carnegie Mellon University, 2012.

[27] A. S. McEwen. A precise lunar photometric function. *Lunar and Planetary Science*, 27:841–842, 1996.

[28] Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings VRCIA 2006 ACM International Conference on Virtual Reality Continuum and its Applications*, pages 311–318, 2006.

[29] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light space perspective shadow maps. *Rendering Techniques*, 2004:15th, 2004.

[30] Ryan Gariepy, Prasenjit Mukherjee, Paul Bovbel, and Devon Ash. husky_gazebo. `http://wiki.ros.org/husky_gazebo`. Published: 2015-02-21 Accessed: 2018-09-21.

[31] K. Yoshida and H. Hamano. Motion dynamics of a rover with slip-based traction model. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 3155–3160 vol.3, May 2002.

[32] John M. Hsu and Steven C. Peters. Extending open dynamics engine for the darpa virtual robotics challenge. In Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, pages 37–48, Cham, 2014. Springer International Publishing.

[33] NASA Content Administrator. husky_gazebo. `https://www.nasa.gov/centers/glenn/events/tour_erb_slope.html`. Published: 2018-01-04 Accessed: 2018-10-03.

[34] Chunmei He. *Geotechnical characterization of lunar regolith simulants*. PhD thesis, Case Western Reserve University, 2010.

[35] Rahul Soni, B.K. Mishra, and R Venugopal. A review on discrete element method (dem): Steps, methodologies, development and applications in simulation of granular flow. 05 2012.

[36] Larry Matthies, Tucker Balch, and Brian Wilcox. Fast optical hazard detection for planetary rovers using multiple spot laser triangulation. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 859–866. IEEE, 1997.

[37] Ara Nefian, Uland Y Wong, Michael Dille, Xavier Bouyssounouse, Laurence Edwards, Vinh To, Matthew Deans, and Terry Fong. Structured light-based hazard detection for planetary surface navigation. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2665–2671. IEEE, 2017.

[38] P Michael Furlong, Michael Dille, Uland Wong, and Ara Nefian. Safeguarding a lunar rover with wald's sequential probability ratio test. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5411–5418. IEEE, 2016.

[39] R Jay Torres, Mark Allan, Robert Hirsh, and Michael N Wallick. Rapid: Collaboration results from three nasa centers in commanding/monitoring lunar assets. In *Aerospace conference, 2009 IEEE*, pages 1–11. IEEE, 2009.

[40] Susan Y Lee, David Lees, Tamar Cohen, Mark Allan, Matthew Deans, Theodore Morse, Eric Park, and Trey Smith. Reusable science tools for analog exploration missions: xgds web tools, verve, and gigapan voyage. *Acta Astronautica*, 90(2):268–288, 2013.

[41] Matthew Deans, Terrence Fong, Pascal Lee, Kip Hodges, Mark Helper, Rob Landis, Steve Riley, Maria Bualat, Estrellina Pacis, and Linda Kobayashi. Robotic scouting for human exploration. In *AIAA Space 2009 Conference & Exposition*, page 6781, 2009.

[42] Terrence Fong, Maria Bualat, Matthew Deans, Bryon Adams, Mark Allan, Martha Altobelli, Xavier Bouyssounouse, Tamar Cohen, Lorenzo Fluckiger, Joshua Garber, et al. Robotic follow-up for human exploration. In *AIAA SPACE 2010 Conference & Exposition*, page 8605, 2010.

[43] Maria Bualat, Terrence Fong, Mark Allan, Xavier Bouyssounouse, Tamar Cohen, Lorenzo Fluckiger, Ravi Gogna, Linda Kobayashi, Grace Lee, Susan Lee, et al. Surface telerobotics: development and testing of a crew controlled planetary rover system. In *AIAA Space 2013 Conference and Exposition*, page 5475, 2013.

[44] What is labview?, 2018.

[45] Jay Trimble and George Rinker. Open source next generation visualization software for interplanetary missions. In *14th International Conference on Space Operations*, page 2348, 2016.

[46] J. et al. Trimble. Openmct: Open source mission control technologies, 2018.

[47] Matt Siegler, David Paige, Jean-Pierre Williams, and Bruce Bills. Evolution of lunar polar ice stability. *Icarus*, 255:7887, Oct 2015.

[48] Dana M. Hurley, David J. Lawrence, D. Benjamin J. Bussey, Richard R. Vondrak, Richard C. Elphic, and G. Randall Gladstone. Two-dimensional distribution of volatiles in the lunar regolith from space weathering simulations. *Geophysical Research Letters*, 39(9), May 2012.

Biography

Mark Allan *is a senior research engineer at KBRwyle in the Intelligent Robotics Group at NASA Ames Research Center. He received his B.S. in Biology in 1994 from University of California Santa Barbara and his M.S. in Information Systems in 2008 from Santa Clara Univerity. His interests include teleoperation interfaces, visual simulation, and robotics middleware.*

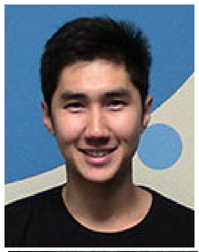
Uland Wong *is a senior computer scientist in the Intelligent Robotics Group at NASA Ames Research Center through KBRwyle. His research interest lies at the intersection of computational imaging and mobile robotics. Uland received his PhD in Robotics from Carnegie Mellon University in 2012.*

P. Michael Furlong *received his B.E. degree in Computer Engineering in 2005 from Memorial University of Newfoundland, and a Ph.D. in Robotics from Carnegie Mellon University in 2018. He is currently a senior researcher for KBR-Wyle at NASA Ames' Intelligent Robotics Group. His interests include science autonomy and theoretical neuroscience.*

Arno Rogg *received his B.S. in Microengineering in 2014 and his M.S. in Microengineering in 2016 from the Swiss Institute of Technology in Lausanne. He is currently a robotics engineer in the Intelligent Robotics Group at NASA Ames Research Center. His interests include smart mechanism for space robotic application, rover systems engineering and integration, rover mobility and terramechanics.*

Scott McMichael *received his B.S. and M.S. degrees in Computer Engineering from Case Western Reserve University in 2006 and 2008. He is currently a research engineer at the NASA Ames Research Center where he maintains the Ames Stereo Pipeline software project.*

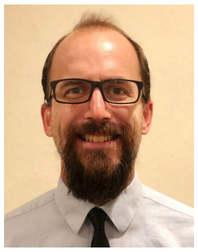
Terry Welsh *received his B.S. in Physics in 1996 and his M.S. in Computer Engineering in 1999 from Iowa State University. He has worked in a variety of industries and is currently a Senior Software Engineer in the Intelligent Robotics Group at NASA Ames Research Center. His interests include real-time computer graphics and visual simulation.*



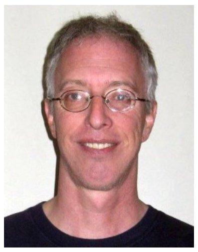
Ian Chen *received his B.E. and Ph.D. in the Electrical and Computer Engineering Department of University of Auckland in 2007 and 2011 respectively. He is currently a research engineer at Open Robotics. His key research interests are in robot simulation and visualization.*

Steven Peters *received his Ph.D at the Massachusetts Institute of Technology in 2012. He currently works at Open Robotics and contributes to the Gazebo simulator with a focus on physics simulation. His interests include high-fidelity simulation of robot and vehicle dynamics as well as implementation of real-time control algorithms on passenger vehicles and robotic manipulators.*

Brian Gerkey *received his Ph.D. in Computer Science from the University of Southern California (USC) in 2003, his M.S. in Computer Science from USC in 2000, and his B.S.E. in Computer Engineering, with a secondary major in Mathematics and a minor in Robotics and Automation, from Tulane University in 1998. He is cofounder and CEO of Open Robotics.*

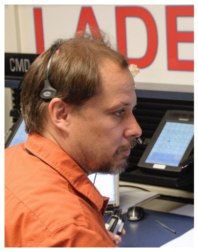


Morgan Quigley *received a Ph.D. in Computer Science from Stanford University in 2012. He is currently the Chief Architect at Open Robotics. His research interests include collaborative open source software, scalable distributed embedded systems, and hardware/software co-design.*

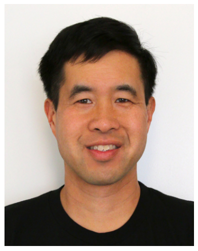


Mark Shirley *received B.S. and M.S. degrees in Electrical Engineering and Computer Science in 1983 and a Ph.D. in Computer Science with a concentration in Artificial Intelligence from the Massachusetts Institute of Technology in 1989. He is currently a member of the research staff at NASA's Ames Research Center. His interests include diagnostic algorithms, satellite ground systems and distance learning. He previously worked at Xerox PARC and GenRad Inc.*

Matthew Deans *received his PhD in Robotics from Carnegie Mellon University in 2005. His research interests include robotic mapping, localization, navigation, machine vision in unstructured environments, remote operation of rovers, and field robotics especially in terrestrial planetary analogs. On Resource Prospector he served as the lead for Rover Navigation.*



Howard Cannon *is a Computer Engineer at NASA Ames Research Center. He received his B.S. degree in Mechanical Engineering from Bradley University in 1988, and a M.S. degree in Robotics from Carnegie Mellon University in 1998. On Resource Prospector, Cannon served as the lead for the rover software, and as the project's software systems engineer.*



Terry Fong *received his B.S. and M.S. degrees in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 1988 and 1990. He received his Ph.D. in Robotics from Carnegie Mellon University in 2001. He is currently the Chief Roboticist at the NASA Ames Research Center. His interests include space robotics, human-robot interaction, and planetary exploration.*