Open-Source Simulation for Planetary Surface Robotics

Edward Tunstel, Russell Turner[‡], Christopher Olson
Space Department / [‡]Milton Eisenhower Research Center
Johns Hopkins University Applied Physics Laboratory
Laurel, MD 20723
Edward.Tunstel@jhuapl.edu

Abstract – *This paper highlights options for open-source software tools germane to 3-D physics-based simulation for planetary surface robotics studies. Several commercial and open-source tools of comparable capability are briefly described. An open-source tool used as a simulation component of a planetary robotics software development environment is discussed. The efficacy of the tool as a cost-effective basis for development and evaluation of concepts and algorithms is proven through an example application. It was applied to modeling, simulation, and motion control of a rover model performing a notional task in a virtual lunar surface environment. Basic features of the tool are exercised as well as interfaces for adding custom control software and sensor models.*

Keywords: Planetary surface robotics, physics-based simulation, open-source software.

1 Introduction

Technology development efforts to produce robotic products are best pursued with a combination of hardware prototypes and computer simulation tools. While hardware prototypes are indispensable and simulation tools are not, simulation capabilities significantly enhance productivity and success. Significant progress can be made with simulation tools of medium to high fidelity. Tools in mind are those that would facilitate modeling of robotic systems, their physical operating environments, and components thereof (sensors, actuators, science instruments, terrain, reduced gravity fields, man-made structures, other moving equipment and humans). This includes modeling that accounts for dynamics sufficient to allow study of robotic motion in a variety of environments and in a realistic way.

When required hardware or software infrastructure is unavailable, existing simulation environments are viable alternatives. A focused survey of existing tools that could be useful for planetary robotics development was performed to identify promising candidates. The focus was on commercial and open-source tools that specifically support planetary surface robotics or have capabilities that could be tailored for that purpose. *Open-source* refers to software that is provided with original source code that can be added to and/or improved by users according to some agreed upon programming standard. While not required to, developers typically share their improvements or additions to the software with a user community. Most tools surveyed require additional development of specialized features and elements before they could effectively support simulation capabilities needed for planetary robotics.

Modeling and simulation capabilities for developing and testing planetary robot design concepts and control software are integral to conceiving robotic solutions for planetary surface missions. Conventional robot computer simulation tools only satisfy a subset of requirements for planetary robotics simulators rendering them incomplete with respect to important domain-specific features or elements. Alternative sources for mature capabilities of planetary robotics modeling and simulation are rare, expensive, or not readily accessible (e.g., proprietary). Rare sources that are most germane and economically accessible can be leveraged as a baseline upon which to tailor a custom solution. This work identified an existing real-time, 3-D physics-based tool onto which specialized models and features were built to yield the foundation of a tailored simulation environment for planetary robotics development. In the remaining sections, this paper highlights the utility of robot simulation, several existing simulator options, and the application of an open-source software tool for planetary robotics simulation-based development.

2 Simulation for Planetary Robotics

Beyond solid modeling and design of robotic systems, simulation plays an important computer-aided engineering role in planetary robotics. It is beneficial in several ways as a means to: explore conceptual designs and feasibility; develop technology in lieu of available hardware; predict behavior in planetary environments that are expensive to physically emulate; support command sequence testing, validation, and anomaly investigation for robotic systems in operation; and support design or functional refinements during the life cycle of robotic systems. Furthermore, simulators support development of autonomous control algorithms and software as well as architectures for mission concepts of operation. Planetary robotics simulation environments allow researchers to begin developing capabilities and concepts for future missions before it becomes feasible or affordable to experiment on robotic hardware. Simulation investments yield returns over time in cost-savings for development and equipment, capability enhancement as well as sponsored research and missions.

3 Simulation Options

Simulation for robotic systems is a very active area of late as research groups at universities and national laboratories, as well as industry, have become very interested in pursuing robotics. Most groups tend to start from the ground up and opt to establish a footing in the area using simulation tools before investing in hardware procurement or fabrication (or for lack of resources and expertise needed to launch their research with hardware). This holds true across domains of application whether for Earth or space robotics. The popularity of robotics and the demand for guidance on how to get started in the field has led to a number of surveys covering available commercial and open-source options for simulation environments.

A recent survey of 14 widely available simulation tools for unmanned systems concludes that developing a simulator from the ground up is no longer necessary given the availability of open-source and commercial simulators as well as 3-D video game engines (for physical simulation and visualization) [1]. Another recent survey reported on 9 open-source mobile robot development environments that are freely available, i.e., unencumbered by licensing costs and available for free download from the Internet [2]. Some are mainly robot software/architecture development environments while others are purely simulation-based (to be interfaced with control software developed by other means). The environments were evaluated and compared using criteria corresponding to typical stages of application development, including features and usability. Regarding features, the highest-ranking environments were those that provided high-fidelity 3-D simulators that could be used to model robotic mechanisms. Environments receiving highest usability rank included easily accessible simulators for architecture implementation and debugging [2].

While the evidence is clear that simulators are a staple of serious robotics development, most available simulators only support a subset of features needed for planetary robotics. Where basic elements do exist to build required features, substantial effort is often required. For example, representing effects of wheel-soil interactions in simulation enables prediction of rover locomotion performance. Most available 3-D simulators for robotics do not provide such representations and are less relevant for terrains of interest outside of planetary robotics (e.g., indoor floors, paved roads, and otherwise flat/benign surfaces).

For planetary robotics, simulation models that capture such important effects would need to be available and configurable to represent interactions between wheels/ legs/tools and different terrain/soil/regolith types. Models can be adopted from examples in the literature or available commercially and tailored as needed to account for planetary surface conditions of interest. Alternatively, special models could be derived experimentally using specific prototype wheels/legs and soil simulants for incorporation into the simulator. Once representative hardware is available, it is also important to validate and refine simulator models via comparison and tuning with respect to measured system behavior [3, 4]. This example is representative of the situation as it relates to other important features mentioned above that are not supported "out of the box" by most simulators. Similarly, the ability to represent or import models of robot onboard subsystems (e.g., power, thermal, etc) and science instruments [5] is also important for planetary robotics simulations. Several commercial and open-source simulators that are suitable for planetary robotics purposes are briefly discussed below.

3.1 Commercial simulators

Vega Prime is a toolkit for 3-D visual simulation applications and has been commonly used for military and construction training simulators [6]. Its price can be on order of several tens of thousands of dollars including most useful functional modules, associated licenses, and annual support. *Vortex Simulation Toolkit* is a dynamics engine for building physics-based interactive simulations that can cost several thousands of dollars [7]. A combination of these tools is a good candidate for robotics simulation particularly if working with scenarios on a theater scale. Aside from the overall expense, the closed nature of the software development libraries forces more dependence than desired on the vendors for special feature additions.

Adams, by MSC Software, is a popular family of interactive motion simulation software modules for multi-body mechanical systems and is considered a standard in certain industry sectors. Users can start with the *Adams* basic package and choose from a wide range of specialized extension modules (e.g., controls, vibrations, car vehicle dynamics) [8]. The basic package costs tens of thousands of dollars within a range governed in part by license type. From a robotics perspective, *Adams* can be considered to be more of a stand-alone, preliminary mechanical design and analysis tool than an algorithm development and evaluation tool that can be easily interfaced with custom robot control software. However, similar tools sometimes permit interfaces to complementary tools like Matlab.

Microsoft *Robotics Studio* provides a service-based architecture framework for robotics development including a simulator with a high-fidelity physics engine [9]. It supports a short list of commercially available educational and hobby robot models limited to built-in sensor and actuator models. It is licensed for non-commercial use at no charge (and with limitations), while commercial licenses cost several hundred dollars. Considerable effort would be required to bridge the feature gap between *Robotics Studio* and a simulation environment well suited for planetary robotics. The *Webots*™ mobile robot simulator [10] offers similar features including a well-integrated simulation and software development environment for several thousands of dollars.

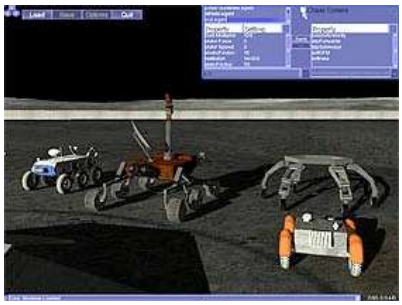
Many generic 3-D simulators are available but would require substantial investment of time and effort to make them useful for planetary robotics simulations. The commercially available options among them require major up front financial investments on top of the time and effort.

3.2 Open-source simulators

Player/Stage/Gazebo [11] is an open-source project that provides a 2-D (*Stage*) and 3-D (*Gazebo*) simulation environment for robot testing that communicates through an interface (*Player*) with virtual robots and sensors. *Gazebo* supports physics simulation and a low-fidelity visualization of the environment. This is one of the most commonly used open-source simulation environments by robotics researchers [2]; it requires a build-up of substantial add-ons to be useful for planetary robotics simulation.

Mission Simulation Toolkit (MST) is a software system for robotic autonomy research released under the NASA Open Source Agreement [12, 13]. It was developed by the NASA Ames Research Center and the Jet Propulsion Laboratory (JPL) to facilitate development of autonomous planetary robotic missions and it provides a simulation framework and tools. MST is a strong candidate offering useful autonomy software infrastructure elements and features germane to planetary robotics. Fig. 1 (left) shows an MST screenshot including a NASA K-9 rover model.

Mission Simulation Toolkit, NASA



Digital Spaces

Figure 1. Screenshots from simulation tools created using open-source tools funded by NASA.

Digital Spaces (DSS) [14] is an open-source simulation environment that can support mission engineering design and operations as well as development. It has been used in NASA centers for simulations and visualization ranging from International Space Station training to Mars exploration. DSS was developed as a NASA Small Business Innovation Research project and used to support the NASA Robotic Lunar Exploration Program (RLEP2) and Exploration Systems Mission Directorate exploration architecture studies [15]. For the RLEP2 studies, DSS enabled lunar surface modeling and mobility studies of four classes of vehicles (Fig. 1, right) including a rover concept derived from military Multifunction Utility Logistics Equipment vehicle designs, a NASA Mars Science Laboratory rover, a JPL ATHLETE limbed rover, and a joint NASA Marshall Space Flight Center–Johns Hopkins University Applied Physics Laboratory (JHU/APL) rover concept. As such, DSS is a strong candidate with many features and existing models germane to planetary surface robotics and has also been applied to in-space simulations.

3.3 Other simulators

In addition to open-source tools such as MST and DSS, sophisticated planetary surface robotics simulation tools exist in a number of research laboratories. An issue with many is that they are somehow proprietary or closed to customization by outside developer-users. One example that is representative of the state-of-the-art in this area is the Rover Analysis, Modeling, and Simulation (ROAMS) tool [4]. ROAMS is developed at NASA JPL and features rover, sensor, and terrain models as well as an interface to a JPL open-source robotics control software framework. ROAMS has not been readily accessible to researchers outside of its development team or who are not participants in active NASA projects.

4 Open-source Simulator Selection

Advances in electronics supporting 3-D computer graphics (e.g., powerful 3-D chip-sets) and associated cost-effectiveness have enabled their use as standard components in common desktop PCs and laptops. This has further enabled development of powerful real-time 3-D simulation tools that can run on inexpensive PCs. Such advancements in 3-D graphics hardware and software technology fueled by the computer and video gaming industry has enabled open-source software that is comparable to the best commercial simulation tools.

For certain research projects or programs, the approach of adopting as a baseline an open-source tool designed for planetary robotics work is more efficient and economical than purchasing commercial simulation software. Practical candidates would not require major up front investment, would offer substantial features and models for planetary robotics relative to generic 3-D simulators, and would be flexible and open enough to allow customization and expansion. A focused survey limited primarily to the tools mentioned above led us to conclude that existing open-source simulation tools are sufficiently well equipped to serve as a baseline for planetary robotics work. Our survey deliberately narrowed the space of options to those expected to require least time and effort to develop desired features and models to enrich our simulations.

DSS was evaluated with emphasis on practical attributes such as built-in features, computing resource requirements (qualified by a preference for the tools to run on modest desktop and laptop PCs), documentation and user support. DSS is available via download from the Internet and can be installed and run on modern desktop PCs. Its Application Programming Interface (API) enables

users to develop custom software-based logic to control motion of 3-D robots and objects in a virtual environment. Popular high-level programming languages such as C++ or Python can be used for this purpose. DSS currently only supports Microsoft Windows operating systems, however. Since DSS tutorials and a set of software demonstration projects available to users made exclusive use of Python, custom software to control simulated robots was developed in Python as well for this evaluation.

We set out to develop custom features and models germane to our near-term planetary robotics interests that would augment DSS to yield an evolving 3-D simulation environment for research at JHU/APL. Sufficient understanding of the tool was gained to permit custom functionality to be added for an example planetary surface robotics application described below.

5 Example Application

DSS version 0.12.4 was used on desktop computers (DELL Optiplex 745) and a high-end gaming laptop (Dell XPS M1730). Fig. 2 illustrates the resulting architecture consisting of DSS at its core surrounded by third-party tools and add-ons representing the simulation-based planetary surface robotics development environment. DSS can make use of synthetic terrain meshes as well as meshes created using terrain data derived from planetary missions (and available from sources such as the NASA Planetary Data System or the USGS Planetary GIS Web Server). Blender, an open-source 3-D solid modeling tool, was adopted for creating original content for use in virtual scenes and for importing, exporting, and converting various graphics file formats for use in the environment.

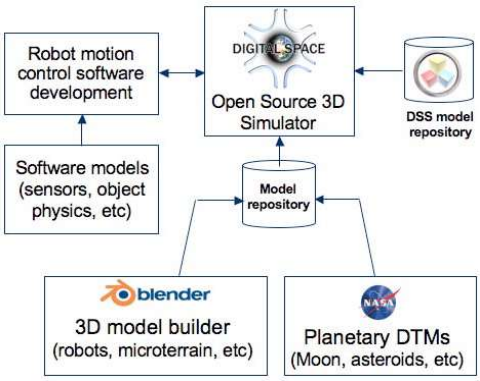


Figure 2. Digital Spaces-based simulation environment.

5.1 Simulation models and ingredients

To validate the utility of DSS for flexible development of robotics software and algorithm evaluation a use-case was formulated for implementation that called for basic ingredients representative of surface robot applications. The use-case focused on real-time simulation of a robotic vehicle performing a navigation and manipulation task that included the following main ingredients: a virtual planetary surface, a 3-D rover model with robotic arm, control software to guide rover motion, and a sensor model for detecting mobility obstacles.

Rover and lunar surface models from the DSS demonstration repository were explored for use our application. Experimentation with several wheeled rover models converged to one for which four wheels could be both driven and steered independently and for which five robotic arm joints could be independently moved. As pre-configured, the rover model could be maneuvered interactively using keyboard entries or a PC joystick to control wheel and robotic arm motions. Toward the objective of robot software development as a feature of the desired simulation environment, the simulation was reconfigured so that mobility could be controlled via original software. This is enabled by the DSS API and its means for interfacing control software to the movable joints of robot models that obey configurable robot kinematics and rigid body dynamics associated with physics of interaction with the environment (e.g., gravity, wheel-terrain friction, and collisions). Robotic arm motion remained under control of the keyboard/joystick, emulating remote control by a human through teleoperation. A teleoperator could move robotic arm joints and a gripper individually or several could be moved simultaneously to perform telemanipulation for sample retrieval.

An approach to emulating non-contact range sensing using built-in ray casting functions of DSS was established. A range-finding sensor model was implemented for attachment to the rover model. It is comprised of multiple rays arranged radially in a plane around the rover body to emulate a range sensor with discrete 360º coverage. Each ray is configured to have a finite length representing the reliable range of detection for the sensor (the collection of rays). This is functionally representative of a collection of infrared range sensors or discrete laser rangefinders, which are among the simplest navigation sensors typically employed on mobile robots. Whenever any ray intersects an obstacle in the simulation scene, the distance along the ray to the obstacle becomes available to the rover motion control software indicating a detected obstacle.

Basic motion control logic was encoded for the simulated rover to determine the appropriate driving and/or steering action to take (or not) to avoid collision with detected obstacles while resulting in the least deviation from paths directed toward a designated navigation goal in the scene. Software enabling straight driving to a goal, or surface coordinate, combined with range sensor-based obstacle avoidance enabled surface navigation in the simulated lunar scene [16]. The sensor model thus served as a body-mounted navigation sensor supporting traverses. Additional software was written to acquire the range measurements from this sensor model while the rover was in motion performing point-to-point navigation in relatively open terrain among detectable mobility obstacles.

The focus of this work was on the application of DSS to implement realistic simulations and enabling interfacing of original control software to simulated robots. As such, less emphasis was placed on optimizing the performance of the range sensor model and motion control algorithm; these could be further improved as necessary.

5.2 Use-case scenario

A notional lunar surface scene including two rovers and a lander was configured as the focal environment for the scenario. It was comprised of a set of terrain meshes and 3-D scene objects available within DSS, including rover and lander models. All objects in the simulation scene were set to be under influence of lunar gravitational acceleration. An illustrative lunar sample retrieval scenario was adopted for execution within this scene (Fig. 3).

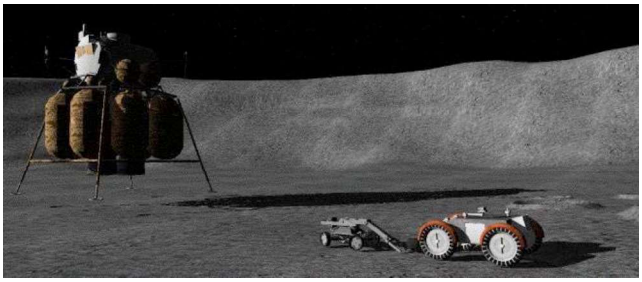


Figure 3. Lunar surface scene showing (left to right) ascent vehicle, sample retrieval rover, and rover with rock sample.

The scenario is described as follows: A robotic precursor mission is presupposed to have resulted in a successful rover sample acquisition and caching operation. The rover possessing a lunar rock sample remains on the lunar surface in a nonfunctional state since the end of its mission. A follow-on mission successfully delivers a sample retrieval rover and ascent, or Earth-return, vehicle to the location of the nonfunctional rover that holds the sample cache. The simulation focuses on a simplified task of retrieving the sample and delivering it to the ascent vehicle. A similar scenario was executed in past NASA/JPL field tests with Mars sample retrieval as a focus [17]. This particular scenario exercises the navigation and telemanipulation functions of the sample retrieval rover while demonstrating the capability of the environment to simulate the associated physics behavior of the main 3-D object models involved. Fig. 4 shows screenshots of different phases of execution in the scenario.

This use-case served to demonstrate integrated use a variety of DSS features for a representative application. While it represents only a fraction of what could be done with the simulation environment and underlying open-source simulation tools, it successfully illustrates the following: operation of representative object and physical environment models; sensor model and control software add-ons; sensor-based control of a virtual robot; and execution of a traverse and telemanipulation task.

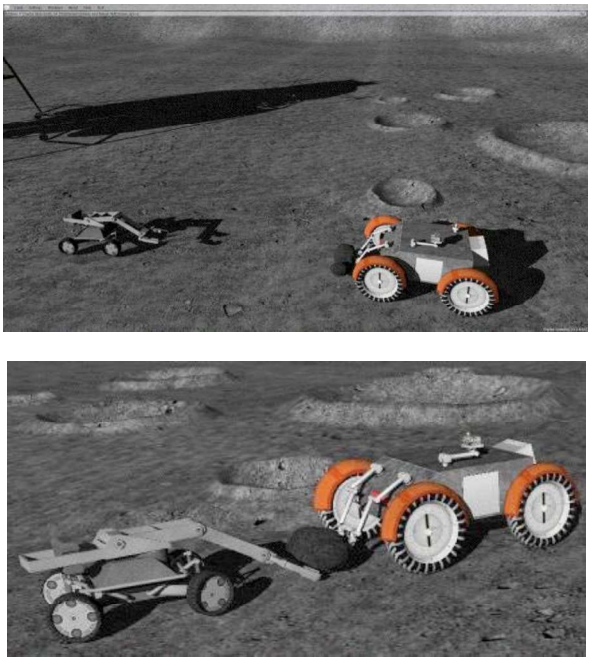


Figure 4. Lunar sample retrieval scenario showing the retrieval rover approaching the nonfunctional rover (top) and start of simplified teleoperated sample retrieval using the robotic arm.

5.3 Future work

By design, the simulation environment remains a work in progress to be enhanced over time via use on future projects that demand new or refined features and capabilities. With an eye toward relevance to JHU/APL planetary robotics work with physical rover prototypes future work will model actual JHU/APL robotic vehicles, subsystems, and science instruments. This will enable the simulation model to be validated to some degree against the physical prototypes and improved as a result.

With the DSS tool it is also possible to develop software to enable visually realistic simulation of the manner in which loose terrain (soil/regolith) interacts with robot wheels/legs and tools. This will be pursued in future work. Emulating such terrain interactions would require real-time modification of terrain mesh files such that they locally deform according to the physics of interaction accounting for user-configurable parameters governing attributes such as friction, restitution, angle of repose, etc. This would enable simulation of soil/terrain deformation in reaction to robotic activities such as surface sampling and excavation. The feasibility of implementing this capability using DSS has been proven in related work [18]. Associated models can be based on empirical studies, or existing models could be applied that are documented in the terramechanics and planetary surface robotics research literature [3, 19-21].

6 Conclusions

Robotic hardware is essential for planetary robotics research and technology development. It is also expensive to acquire, build, and operate. In lieu of available hardware, and for lesser up front investment, physics-based simulators permit research progress including algorithm and concept evaluation. Open-source simulation tools, in particular, offer sufficient features and capabilities on par with expensive commercial software along with open intellectual property from a community of developers.

A select few open-source simulation tools exist that are sufficiently well equipped to serve as a baseline for planetary robotics work. They deliver mature and solid foundations upon which to build domain-specific models and leverage other relevant models from other tools. This work has leveraged one of these candidates, funded by NASA to produce planetary surface system simulations, to seed a development environment for surface robotics. The utility and efficacy of Digital Spaces and its underlying tools, including the popular Object-Oriented Graphics Rendering Engine [22] and Open Dynamics Engine [23], was validated by application to typical planetary robotics simulation tasks.

Acknowledgment

This work was supported by the JHU/APL Civilian Space Business Area. We thank B. Damer and P. Newman of DigitalSpace Corporation for DSS application support.

References

[1] J. Craighead, R. Murphy, J. Burke and B. Goldiez, "A survey of commercial & open source unmanned vehicle simulators," Proc. IEEE Intl. Conf. on Robotics and Automation, Rome, Italy, 2007, pp. 852-857.

[2] J. Kramer and M. Scheutz, "Robotic development environments for autonomous mobile robots: A survey," *Autonomous Robots*, Vol. 22, No. 2, pp.101-132, 2007.

[3] R. Bauer, W. Leung and T. Barfoot, "Experimental and simulation results of wheel-soil interaction for planetary rovers," Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, Edmonton, Alberta, Canada, 2005.

[4] T. Huntsberger, A. Jain, J. Cameron, G. Woodward, D. Myers, G. Sohl, "Characterization of the ROAMS simulation environment for testing rover mobility on sloped terrain," Proc. 9th Intl. Symp. on AI, Robotics and Automation in Space, Los Angeles, CA, 2008.

[5] P. Poulakis, L. Joudrier, S. Wailliez, K. Kapellos, "3DROV: A planetary rover system design, simulation and verification tool," Proc. 9th Intl. Symp. on AI, Robotics and Automation in Space, Los Angeles, CA, 2008.

[6] Presagis, "Vega Prime," http://www.presagis.com/products/visualization/details/vegaprime/.

[7] CMLabs Software, "Vortex Simulation Toolkit," http://www.cm-labs.com/ software/vortex/.

[8] MSC Software, "Adams – Overview," http://www.mscsoftware.com /products/adams.cfm.

[9] Microsoft, "Microsoft Robotics," http://msdn.microsoft.com/en-us/robotics/.

[10] N. Uebelhart, S. Michaud and O. Michel, "Modelling and animation of virtual planetary rovers in simulated world for evaluation of locomotion structures," Proc. 1st International Conference on Dextrous Autonomous Robots and Humanoids, Yverdon-les-Bains, Switzerland, 2005.

[11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, An open-source multi-robot simulator," Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, Sendai, Japan, 2004, pp. 2149-2154.

[12] G. Pisanich, L. Plice, C. Neukom, L. Flückiger, and M. Wagner, "Mission Simulation Facility: Simulation support for autonomy development," Proc. 42nd AIAA Aerospace Sciences Conference, Reno, NV, 2004.

[13] NASA, "Mission Simulation Toolkit," http://opensource.arc.nasa.gov/project/mission-simulation-toolkit/.

[14] DigitalSpace Corporation, "Virtual moonyard lunar hazard obstacle course," http://www.digitalspace.com/projects/dss/moonyard/index.html.

[15] B. Blair, B. Damer, D. Rasmussen and P. Newman, "Design simulation in support of exploration and ISRU for NASA's lunar exploration program and the mining industry," Proc. Space Resources Roundtable VIII, Colorado School of Mines, Golden, CO, 2006.

[16] E. Tunstel, H. Danny, T. Lippincott, and M. Jamshidi, "Fuzzy behavior-based navigation for planetary micro-rovers," Proc. NASA University Research Centers Technical Conf., Albuquerque, NM, 1997, pp. 729-734.

[17] P.S. Schenker, T.L. Huntsberger, P. Pirjanian, E.T. Baumgartner, E. Tunstel, "Planetary rover developments supporting Mars exploration, sample return and future human-robotic colonization," *Autonomous Robots*, Vol. 14, 2003, pp. 103-126.

[18] N.A. El Samid, T. Jekanthan, J. Richard, D. Boucher and G.M.T. D'Eleuterio, "Infrastructure robotics: A technology enabler for lunar in-situ resource utilization, habitat construction and maintenance," Proc. 59th Intl. Astronautical Congress, Glasgow, Scotland, 2008.

[19] V.I. Balovnev, *New Methods for Calculating Resistance to Cutting of Soil*, Published for U.S. Dept. of Agriculture and National Science Foundation, Washington, D.C., Amerind Pub. Co., New Delhi, 1983.

[20] G. Andrade, F. Ben Amar, P. Bidaud and R. Chatila, ”Modeling Robot-Soil Interaction for Planetary Rover Motion Control,” Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, Victoria, B.C., Canada, 1998.

[21] R. Gaskell, L. Ekroot Husman, J.B. Collier and R.L. Chen, “Synthetic environments for simulated missions,” *IEEE Aerospace & Electronic Systems Magazine*, July 2007, pp. 14-20.

[22] G. Junker, *Pro OGRE 3D programming*, Apress, New York, NY, 2006.

[23] R. Smith, “Open Dynamics Engine,” http://www.ode.org/, 2000-2008.