An KML Interface for Dynamics Simulation of Robotic Planetary Exploration

Thomas M. Howard, Jonathan Cameron, Steven Myint, Hari Nayar, and Abhi Jain
Mobility and Robotics Systems (347)
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109
{Thomas.M.Howard/Jonathan.Cameron/Steven.Myint/Hari.Nayar/Abhi.Jain}@jpl.nasa.gov

Abstract—Efficient exploration of the Moon, Mars, and other celestial bodies will require careful coordination between assets in human-robot teams. Such complexity requires mission planning tools to accurately simulate resource consumption, communications, and maneuverability/traversability for multiple vehicles throughout proposed expeditions. This paper presents a Keyhole Markup Language (KML) interface for simulation of complex robotic vehicle missions with the Lunar Surface Operations Simulator (LSOS), a dynamics-based rover simulator based on the Rover Analysis, Modeling, and Simulation (ROAMS) software. This technique outlines simple Schemas that describe complex events in a KML file and replaces manual simulation script construction. Several experiments and simulation studies that utilized this method are presented.

TABLE OF CONTENTS

1. INTRODUCTION

Multi-robot mission simulation is a difficult task that requires two core technologies: high-fidelity simulation and mission scripting tools. The fundamental capacity to model interactions between mobile robot systems and the environment is necessary to accurately predict traversability and power consumption. Mission scripting tools are necessary to control the behavior of autonomous agents. As mission complexity increases, scripting these event-driven simulations becomes more difficult. Writing custom scripts for individual simulations is time-consuming and prone to errors. A common high-level language for describing sequences of common mission events with an easy-to-use interface is a valuable tool for simulating complex missions.

The Lunar Surface Operations Simulator (LSOS) (Figure 1) has been developed for simulation of rovers, habitats, communication, and power assets in Lunar and Lunar-analog environments [1]. The tool has been used to perform power analyses, path traversability, task durations, lighting analyses, and line-of-sight communication between vehicles, habitats, and the Earth.

Figure 1: A view of a simulated rover traversing lunar-like terrain in the Lunar Surface Operations Simulator.

This paper presents the development and application of a KML interface for LSOS. The general approach is shown in Figure 2.

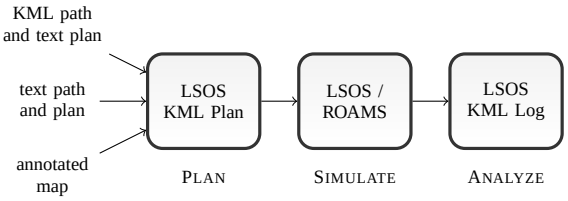


Figure 2: The plan, simulate, analyze cycle using LSOS for dynamics simulation and KML for mission definition and analysis.

First, a general description of the task is provided through a combination of KML, text, map, or spreadsheet and is converted into a single LSOS KML plan. LSOS then loads the LSOS KML plan to initialize and perform a dynamics simulation of the mission. The log files produced by LSOS are then converted back into a KML format that can be overlaid

[1] 978-1-4244-3888-4/10/$25.00 ©2011 IEEE.
[2] IEEEAC Paper #1100, Version 1, Updated 10/12/2010.

on top of the original LSOS KML plan to study quantities of interest and deviations from predicted behavior.

This paper describes development of the LSOS KML Schema specification, discusses applied LSOS technologies, and presents three simulation studies that apply these tools to evaluate mobile robot performance in example missions.

2. Related Work

Associated work comes from two areas in robotics: markup and execution languages and dynamics simulation. Markup and execution languages generally provide a high-level interface for describing complex behaviors. KML [2] is a XML-based language designed to associate information with two and three-dimensional coordinate systems (latitude, longitude, altitude) using Placemarks. Schema types and ExtendedData, SchemaData and SimpleData elements can be defined to encode specific information with a geometry describing geographic positions, paths, or regions. Another example of an XML-based language is the Plan Execution Interchange Language (PLEXIL), which can be used to efficiently describe hierarchical command sequences [3]. The Task Description Language (TDL) similarly can convert task-level controls into C++ code for robot operations [4].

Dynamics simulation of planetary mobile robots is an entirely separate area of research that attempts to approximate real-world dynamics in a computationally efficient manner. [5] describes the ORSIS tool that includes a 3-D multibody vehicle model, wheel-soil interaction and terrain models. The RCAST project [6] describes a mobile robot simulation development effort for the ExoMars rover with reference [7] describes an experimental setup for the validation and tuning of rover simulation software. Reference [8] describes the use of analytical modeling of wheel-terrain interaction modeling together with an experimental test bed for traction studies. The simulations developed in this paper make use of the ROAMS mobile robot simulator described in references [9], [10].

Our approach utilizes KML Schema types and SchemaData elements to describe parameterized environments, assets, and events in LSOS. Parameters are defined using KML SimpleData elements to customize the behavior of a particular Placemark to the mission specification. The KML files with LSOS-specific Schemas are read and interpreted by LSOS to initialize a finite state machine that governs robot behavior throughout the simulated mission.

3. Technical Approach

Three core technologies enable the LSOS KML interface for complex robotic vehicle simulation. First, Schemas are defined to describe important aspects of mission environments, assets, and events. Placemarks are used to define geometry and are augmented with SchemaData to describe specific scenario information (planet, date, time), robots (All-Terrain Hex-Limbed Extra Terrestrial Explorer (ATHLETE) [11], Space Exploration Vehicle (SEV) [12]), or events (extravehicular activity, traverse, pause, resource consumption, recharge, etc.) LSOS parses the unordered KML Placemarks and extracts SchemaData and geometry information to identify, sort, and order events through an event label, a robot identifier, and a sequence number. LSOS automatically synthesizes a multi-tiered finite state machine to control robot behaviors throughout the simulated mission. The logged data is then converted back into KML format to provide an overlay with the original LSOS KML plan for analysis. The framework is designed to be extensible so that additional environments, assets, and events can be easily added to the existing set as new mission requirements are defined.

LSOS KML Schema

Utilizing existing KML extensions enables the use of existing tools for viewing and editing KML files. The KML markup language is organized by defining Placemarks that contain geometry and extended data types. Within the extended data types, Schemas can be defined that contain information particular to that Placemark. For LSOS scenarios, we have defined three different varieties of SchemaData types: MissionInfo, Rovers, and Events.

MissionInfo— In order to initialize the simulation environment, a simulator must know when and what the environment is going to be. A *MissionInfo* Placemark describes the planet, location, and starting time required to initialize the dynamic simulation. The geometry of the MissionInfo Placemark is not used in the simulation construction.

Rovers—*Rovers* are the simulation objects whose behavior is controlled by the LSOS KML events. Each *Rover* has a unique RoverId parameter that is used by the paths and events to construct the finite state machines, a RoverType parameter to determine which type of model to use, and a RoverConfig parameter to initialize the rover configuration. The geometry of the *Rover* Placemark is a Point (latitude/longitude/altitude triple) that defines the initial position of the rover at the start of the simulation.

Events—*Events* are Placemarks that describe some rover behavior. Examples of events include performing extracurricular vehicle activities (EVA), waiting, deploying communication assets, sample collecting, or drilling. *Event* Placemark geometry is typically a line or point used to describe a route or location of an event. Table 1 describes several implemented LSOS KML events that were used in later experiments. All of these events are parameterized using SimpleData fields to adapt general behaviors for particular scenarios. Repeatedly applying parameterized event logic ensures that the plan applies identical logic to execute similar behaviors.

LSOS Simulation

Once the mission has been fully described, the LSOS KML plan is simulated using the Lunar Surface Operations Simula-

Event	Description
Route	Drive the rover along the path sequence of waypoints
PathPlanRoute	Plan a route between each waypoint and drive the rover along the planned path
Wait	Stop at a particular point until a particular time is reached
Pause	Stop at a particular point until a particular chage in time has elapsed
Recharge	Change the energy level and/or energy rate
EVA	A sequence of wait events that represent the exit, activity, and entry of EVA activities
DeployCA	Deploy a rover-based comm asset
StowCA	Stow a rover-based comm asset
DeploySP	Deploy a rover-based solar panel
StowSP	Stow a rover-based solar panel

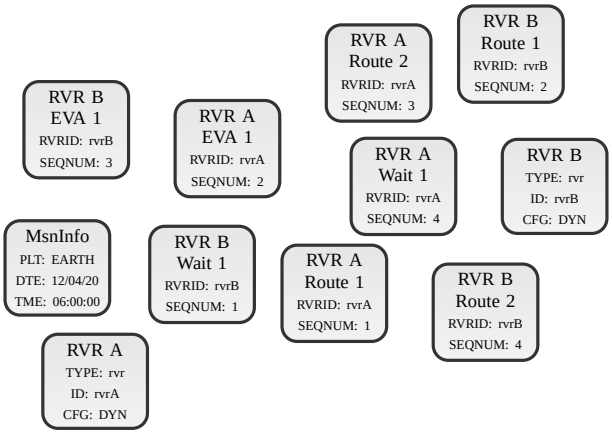
Table 1: A list of several implemented LSOS KML Events with short descriptions of their behaviors

tor (LSOS), an extension of the ROAMS simulation environment. This section describes component technologies used in simulation of the LSOS KML plan.

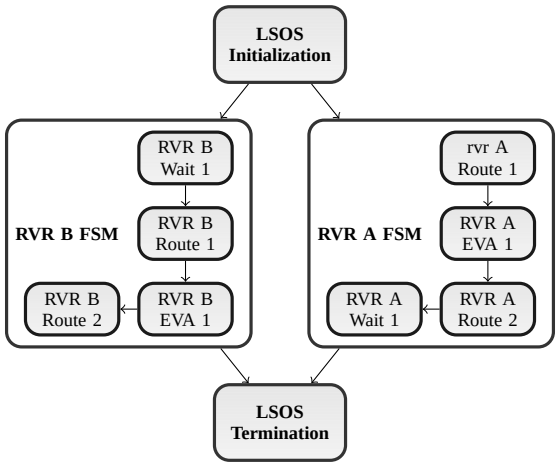
Finite State Machine Synthesis—LSOS converts KML Placemarks to construct multi-level finite state machines to govern the behavior of all simulated agents in the simulation. Figure 3 illustrates the process for constructing the state machines and states necessary to simulate the robotic planetary exploration mission. In this example, a mission is described for two rovers and eight events.

LSOS extracts information necessary to initialize the simulation environment from the MissionInfo Placemark (planet, date, time, etc.) and determines that two independent finite state machines are needed to control the two simulated rovers. Each event is sorted by a rover identifier to match the event to a particular asset. The rover identifier is used to sort all events with particular rovers. Once each event group is formed, the events are ordered based on a sequence number embedded in all event SchemaData types. The resulting finite state machine includes a LSOS initialization state to construct and set the simulation environment, two sub-finite state machines that control each robot through their mission of planetary exploration, and a LSOS termination state that reports recorded data and shuts down the simulation when all rover-level finite state machines have completed.

Dynamics Simulation—The planetary rover vehicle modeled in ROAMS [9], [10], [13] was originally developed for the rocker/bogie class of six-wheeled rovers used for planetary surface exploration. ROAMS has been subsequently extended to support a much broader class of suspension types including legged platforms, skid-steered vehicles, and wishbone suspension vehicles such as the SEV. The simulator can support several variations on the basic design in terms of the location of the differential, the number of steerable wheels



(a) Unordered KML Placemarks with LSOS KML SchemaData



(b) Finite state machine generated by interpreting LSOS KML SchemaData in unordered KML Placemarks

Figure 3: An illustration of finite state machine synthesis from LSOS KML Placemarks. The unordered list of KML Placemarks with LSOS KML SchemaData in (a) is grouped and sequenced by the rover identifier and the sequence number to produce the finite state machine shown in (b).

and the various mechanical dimensions of the rover. While the underlying DARTS multi-body engine supports very general multi-body topologies, ROAMS specializes this using parameterized templates. The templates provide entries for the basic kinematics and inertia properties of the rover, its arm (if any), inertial sensors, number of steerable wheels etc. The templates simplify the definition and addition of new rover vehicle models. For existing suspensions, the templates are filled in with numeric values specific to the rover.

ROAMS makes some assumptions to reduce the number of unknowns. First, ROAMS assumes single point contact for each wheel. This assumption eliminates half of the unknown variables since torque cannot be exerted at a point contact. We are still left with three unknown forces at each wheel contact point. To solve for these forces, ROAMS uses two separate and independent compliance systems at each wheel contact to compute the unknown forces. One compliance system is

used to compute the normal force and the second system is a two-degree of freedom system used to compute the tangent force. The spring and damping coefficients for these compliance systems are parameters of the compliant contact model. The compliance model [13] uses Terzaghi's terra-mechanics equations to compute the maximum available traction force at each contact and to estimate the wheel sinkage into the terrain.

Path Planning—Blind drives to KML coordinates may prove difficult (if not impossible) in complex terrains. Path planners may be necessary to avoid small hazards not visible in KML editing programs. The *PathPlanRoute* LSOS KML Event takes the sequence of coordinates and applies the path planning technique described in [1] to generate a more detailed route considering the high-fidelity terrain model used in LSOS. Route path planning is performed during finite state machine construction and currently does not react to changes in the environment.

Power Modeling— Power modeling in LSOS is primarily driven by the energy consumption of rovers and energy production of solar panels. The LSOS simulator implements dynamic models of power generation and consumption that run concurrently during simulations. At frequent intervals, power generation is determined by measuring the exposure of the solar panels on the respective vehicle to the sun. Total power consumed is the sum of the energy consumed to operate on-board systems and the driving load due to the motion of the vehicle. Driving load is the energy needed to overcome rolling resistance of the wheels in contact with the ground and the energy needed to climb slopes [1]. A surplus energy in the energy balance equation charges the on-board battery and a net loss drains the battery. The vehicle rolling resistance parameters were determined by calibrating the simulated vehicle model with data obtained from field trials of the SEV and ATHLETE vehicles. Other approximate parameters used in the power model, including battery capacity, solar panel size, efficiency, vehicle mass and on-board systems power load, were provided by mission planners at the NASA Langley Research Center.

Energy production is calculated based on the amount of sunlight that hits the solar panels. The solar panel power level computation takes into account the panel orientation with respect to the sun and also any objects that may occlude sunlight. The computation is done using the three-dimensional visualization system, allowing it to be accelerated by the GPU. When the power level of a panel is queried, the visualization system colors that panel with an emissive green color. The solar panel is then rendered from the point of view of the sun using orthographic projection. The rendered image is then examined, counting the number of solar panel pixels that are visible (Figure 4). Factors that could decrease the count include vehicles, other solar panels, or the terrain. The solar incidence ratio is determined by dividing the number of visible pixels by the maximum number of solar panel pixels from the viewpoint. The power generation level is computed by multiplying the solar incidence ratio by the optimal power level for the solar panel model.

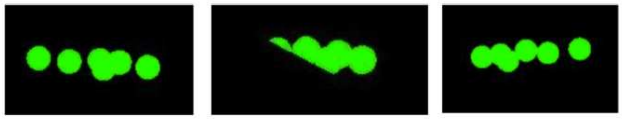


Figure 4: An example of rendering solar panels for use in computing their power level. Green pixels represent the visible portions of the solar panels.

LSOS Analysis

The final step in the presented procedure is the data analysis. A subset of the simulation data is logged in LSOS at a fixed interval throughout the simulation. This logged data is then post-processed into KML format to provide an easily decipherable view of the simulation results. All elements of the simulation data are contained in uniformly distributed Placemarks that can be queried using a KML viewing program. An example of this technique for simulation data analysis is shown later in Figure 8.

4. EXPERIMENTS

The LSOS KML interface has been applied for simulation of several candidate multi-day, multi-platform Human-robot missions in Lunar and Lunar-analog environments. In each experiment, LSOS-specific SchemaData was fused with Placemark geometry to describe particular behaviors as outlined by mission specifications. LSOS simulated and recorded resource consumption, traverse distance and duration, energy and power consumption, velocity, attitude, elevation, proximity, and line-of-sight between other robotic vehicles. Line-of-sight was computed by rendering a view of the one platform from the other and searching the scene for representative ornamental geometry. The 3D visualization information was serialized to file for post-simulation traverse analysis and to improve simulation efficiency. The logged data was also post-processed into KML format to provide an overlay for detailed mission analysis.

Example Lunar Traverse Simulation

To demonstrate LSOS KML mission simulation, an example traverse composed of previously described events was planned to explore regions of the south pole of the Moon. A synthetic lunar terrain that applied a terrain enhancement technique to a filtered version of the LOLA terrain [1] was used for the experiment to approximate a complex lunar surface. An example mission plan (Figure 5) was composed by defining a multi-coordinate route and adding several solar panel recharge events along the traverse. The Placemark on the left labeled "SEV 1" marks the starting location of the robot for the simulation, the lines define the high-level path

that the rover will traverse, and the letter-labeled balloons mark other non-route LSOS KML events. The SchemaData for each Placemark, describing LSOS events, can be queried by selecting a particular Placemark.

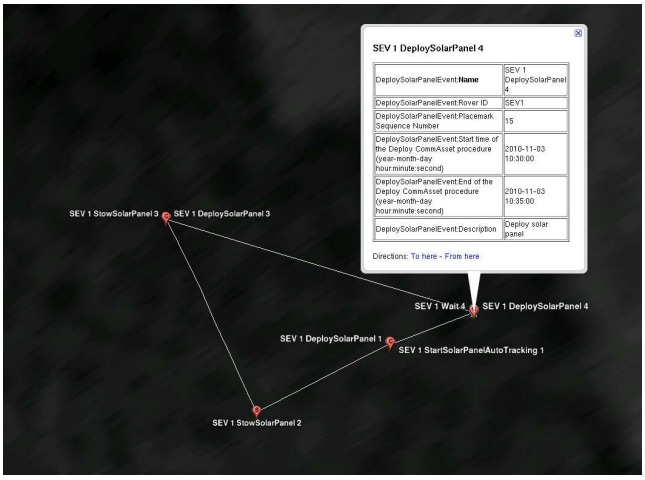


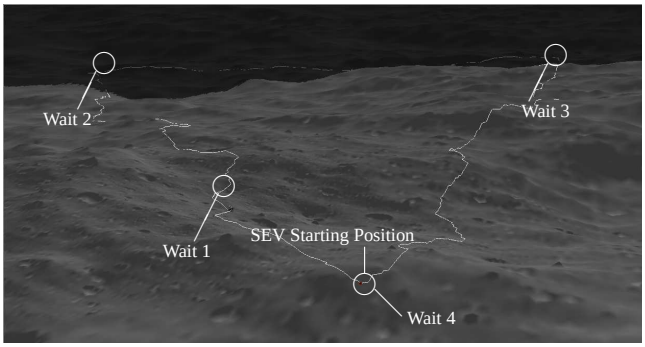
Figure 5: A view of the LSOS KML file used to control a dynamics simulation of an example lunar traverse.

The example LSOS KML plan was simulated with LSOS to determine the energy production/consumption, distance traveled, and time required to perform each traverse throughout the example lunar traverse. The SEV state was recorded at 0.5 Hz for the entire duration of the five-hour example mission. Figure 6 shows an overview of the simulated mission and the rover at two solar panel recharge events.

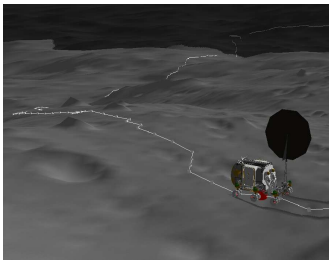
This experiment was designed to exercise two particular behaviors. First, this scenario was used to test the effectiveness of the solar panel power analysis by purposely planning recharge events in the sun and shade during the mission. If the solar panel power analysis is performed correctly, no power should be generated during the shaded recharge events. Second, the path plan route event must plan a path through the environment to dodge the many rover-sized craters that infiltrate the landscape.

Several plots showing data from the example lunar traverse simulation are shown in Figure 7. Figure 7a is a record of the position of the SEV during the simulated lunar traverse. The jagged nature of the path shows that the path planner effectively planned a route through the environment to minimize exposure to rough terrain and high slopes. Figures 7b, 7c, and 7d show the elevation change, energy change, and the energy rate as a function of time. A one-dimensional graph of planned events are shown in Figures 7b, 7c, and 7d to assist in interpreting the simulation data.

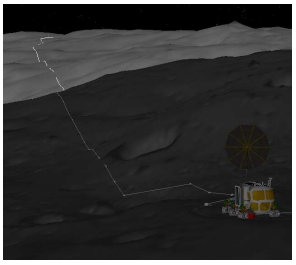
Notice in Figure 6a that the first and fourth solar panel recharge event is in the sun while the second and third are in the shadow cast by the rim of a nearby crater. This is accurately reflected in the recorded energy rate in Figure 7d where the energy rate is positive during the first and fourth solar panel recharge event, meaning that the solar panel is illuminated and generating power (Figure 6b). Similarly, the energy rate during the second and third solar panel deployment events are all remain at the static power draw as expected because no power is generated when the solar panels are in the shadow of a nearby crater (Figure 6c). This type of information can provide mission planners valuable data to estimate traverse durations and required energy resources to execute a lunar traverse.



(a) An overview of the resulting LSOS KML mission plan applied to the simulation terrain. The shadow covering the right portion of the traverse is cast by a nearby crater.



(b) A view of the SEV at the second solar panel recharge during event "Wait 2".



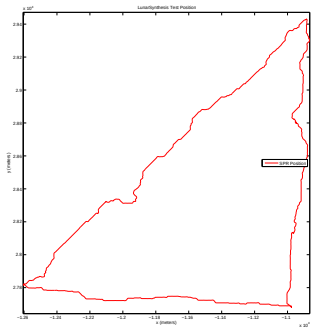
(c) A view of the SEV at the third solar panel recharge during event "Wait 3".

Figure 6: A view of the Space Exploration Vehicle (SEV) traversing a planned path during the Lunar traverse experiment.

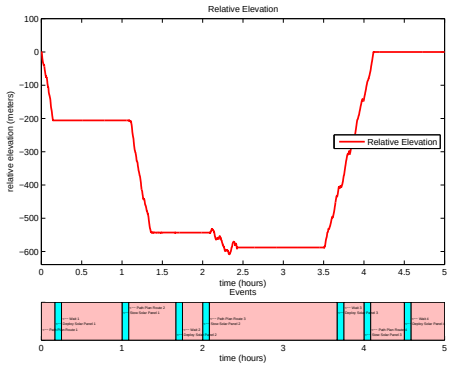
Figure 8 shows a KML file that LSOS generates from the logged data overlaid on top of the original LSOS KML plan. As previously mentioned, the path plan route events plan a motion that deviates from the originally intended path, but ROAMS also simulates terrain effects (e.g. wheel slip) and the response of the path following algorithm. The ability to overlay high-fidelity simulation logged data on top of the original plan is also a useful tool to approximate how closely a candidate mission may execute.

DRATS Traverse Simulation

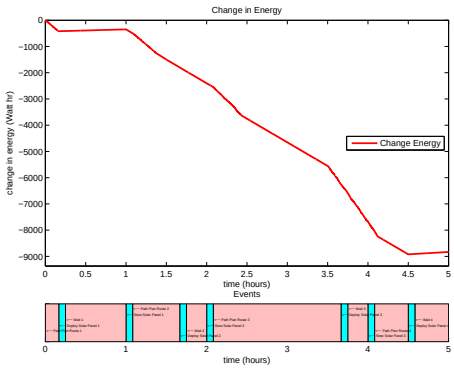
The two-week Desert Research and Technology Studies (DRATS) traverse LSOS experiment sought to quantify distance traveled, energy dissipation, and line-of-sight commu-



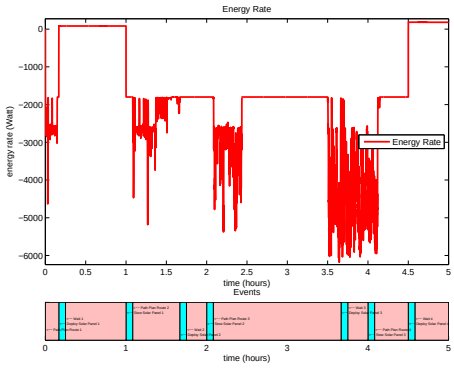
(a) Position plot



(b) Relative elevation versus time plot with event record



(c) Energy versus time plot with event record



(d) Energy Rate versus time plot with event record

Figure 7: Several data products produced by LSOS using the LSOS KML interface during the example Lunar traverse simulation.

Figure 8: A view of the KML file generated from the logged simulation data overlaid on top of the LSOS KML plan.

nication for two Space Exploration Vehicle (SEV) rovers during candidate science sorties near Black Point, Arizona. The DRATS traverse was simulated in LSOS beforehand to provide estimates of traversability, power consumption, feasibility, and distance between rovers. The complexity of the two-week mission drove the development of the LSOS KML interface to provide a compact scenario scripting language that could interface with LSOS. Figure 9 shows the LSOS KML plan and analysis overlay for one day of the two-week experiment.

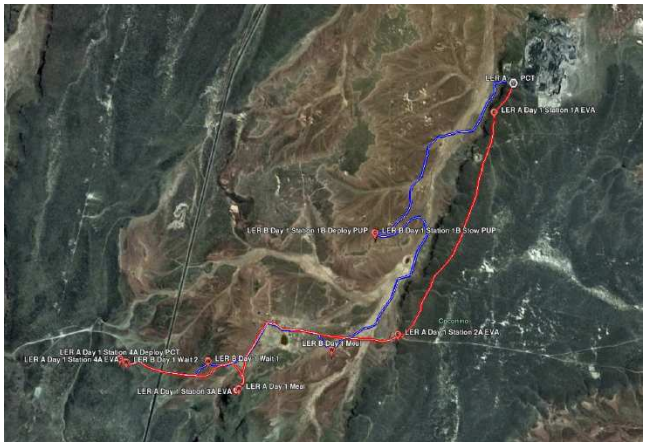


Figure 9: A view of a candidate mission plan and analysis overlay for one day of the DRATS experiment. The red line indicates the simulated path of the SEV "A" rover and the blue line shows the simulated path of SEV "B" rover.

For this particular mission, both rovers start from the same region and explore above and below a ridge, stopping to perform extravehicular activities (EVA) along the way. These experiments demonstrate that the LSOS KML interface has the capability to perform multi-vehicle dynamics simulation.

Figure 10 shows the initial step from the LSOS simulation for the two SEV rovers.

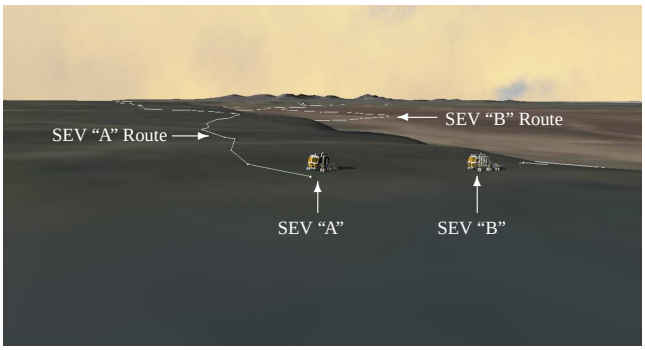


Figure 10: A view of the SEV rover simulation from the mission plan depicted in Figure 9. In this scenario, each SEV must travel several kilometers stopping at several points along the way to deploy and stow communication assets, perform extravehicular activities, and pause for other mission-related tasks.

Lunar Traverse Simulation

A four-week Lunar traverse LSOS experiment similarly sought to estimate distance traveled, energy dissipation, and elevation change for two SEVs and one All-Terrain Hex-Limbed Extra Terrestrial Explorer (ATHLETE) robot during exploration near the South Pole of the Moon. Figure 11 shows the LSOS KML plan and analysis overlay for one day of the four-week experiment.

Figure 11: A view of the mission plan and analysis overlay for two simulated SEV rovers and one ATHLETE robot during one day of the Lunar traverse experiment. The red line indicates the simulated path of the SEV "A" rover, the blue line shows the simulated path of SEV "B" rover, and the green line shows the simulated path of the ATHLETE rover.

The Lunar traverse required new events beyond what was implemented for the DRATS traverse simulation, specifically events for deploying, stowing, and steering rover-based solar panels. These events were implemented and added to the existing library of LSOS KML events for future mission simulations. The elevation map used for the Lunar traverse simulations was based on data taken by the Lunar Orbiter Laser Altimeter (LOLA) on board the Lunar Reconnaissance Orbiter (LRO) spacecraft. Figure 12 shows the initial step from the LSOS simulation for the ATHLETE robot.

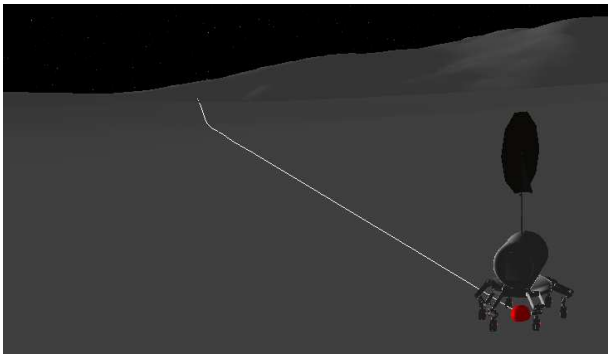


Figure 12: A view of the simulation of the ATHLETE traverse from the mission plan depicted in Figure 11 during a solar panel recharge event.

5. Conclusions

These experiments and analyses demonstrated that KML, when augmented with effective Schema types and tools for automatic finite state machine synthesis, is an effective tool for initializing simulation of robotic planetary exploration missions. Defining each scenario with LSOS KML increased the speed for defining and performing large analyses including the two-week DRATS traverse and a four-week Lunar traverse. Future work in this area involves expanding the library of rovers, events, and environments for future simulations and adapting the automatic finite state machine synthesis tools to other markup languages for non-terrestrial simulations.

Acknowledgments

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under contract to the National Air and Space Administration. We thank our sponsor, Doug Craig from NASA Headquarters, and our collaborators at the United States Geological Survey (USGS), NASA Glen Research Center, NASA Langley Research Center, NASA Johnson Space Center, NASA Ames Research Center, NASA Marshall Space Flight Center and Jet Propulsion Laboratory for support in the development of LSOS.

References

[1] H. Nayar, B. Balaram, J. Cameron, M. DiCicco, T. Howard, A. Jain, Y. Kuwata, C. Lim, R. Mukherjee, S. Myint, A. Palkovic, M. Pomerantz, and S. Wall, "Surface Operations Analyses for Lunar Missions," in *AIAA Space 2010 Conference and Exibition*, August 2010.

[2] "OGC KML 2.2 Standard." [Online]. Available: http://www.opengeospatial.org/standards/kml/

[3] V. Verma, T. Estlin, A. Jonnson, C. Pasareanu, R. Simmons, and K. Tso, "Plan Execution Interchange Language (PLEXIL) for Executable Plans and Command Sequences," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, 2005.

[4] R. Simmons and D. Apfelbaum, "A Task Description Language for Robot Control," in *Proceedings of the Conference on Intelligent Robotics and Systems*, October 1998.

[5] C. Harnisch and B. Lach, "Off Road Vehicles in a Dynamics Three-Dimensional Realtime Simulation," in *Proceedings of the 14th International Conference of the International Society of Terrain-Vehicle Systems*, October 2002.

[6] R. Bauer, W. Leung, and T. Bafoot, "Development of a Dynamics Simulation Tool for the ExoMars Rover," in *Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[7] R. Krenn, M. Apfelbeck, A. Gibbesch, B. Schafer, A. Chilian, and M. Griner, "MBS Simulations and Performance Testing of Planetary Rover Locomotion," in *International Conference on Robotics and Automation 2010 Planetary Rover Workshop*, May 2010.

[8] K. Yoshida, K. Nagatani, and J. Yusa, "Traction Performance of Wheel and Track for Soft-Soil Traversal," in *International Conference on Robotics and Automation 2010 Planetary Rover Workshop*, May 2010.

[9] T. Huntsberger, A. Jain, J. Cameron, G. Woodward, D. Myers, and G. Sohl, "Characterization of the ROAMS Simulation Environment for Testing Rover Mobility on Sloped Terrain," in *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, January 2008.

[10] A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, and R. Steele, "ROAMS: Planetary Surface Rover Simulation Environment," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, May 2003.

[11] B. Wilcox, "ATHLETE: Lunar Cargo Handling for International Lunar Exploration," in *Proceedings of the AIAA Space 2010 Conference*, no. AIAA-2010-8803, September 2010.

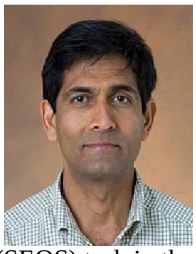
[12] R. Ambrose, "Human-Robotics Interactions Field Test Experiences from a collaborative ARC, JPL and JSC Team," in *Proceedings of the AIAA NASA 3rd Space Exploration Conference*, February 2008.

[13] G. Sohl and A. Jain, "Wheel-Terrain Contact Modeling in the ROAMS Planetary Rover Simulation," in *Proceedings of the 5th International Conference on Multibody Systems, Nonlinear Dynamics and Control*, September 2005.

Thomas M. Howard received his Ph.D. in Robotics from Carnegie Mellon University's Robotics Institute in 2009 and B.S. degrees in Mechanical Engineering and Electrical and Computer Engineering from the University of Rochester in 2004. He is a Research Technologist at the Jet Propulsion Laboratory in the Robotic Software Systems Group. His research interests include model-predictive motion planning, navigation, and control of mobile robots and underconstrained manipulators.

Jonathan Cameron Jonathan Cameron received his Ph.D. in Mechanical Engineering from Georgia Institute of Technology in 1993. He is currently employed at Jet Propulsion Laboratory (JPL) where he does modeling and simulation of robotic vehicles for space exploration. His research interests include robotics, dynamics, kinematics, and software development.

Steven Myint Steven Myint is a Member of the Technical Staff in the Robotic Software Systems group. Steven has previously worked on safeguarded teleoperation of the PackBot robot. He currently works mainly on visualization for various projects in the Darts Simulation Lab.



Hari Nayar Hari Nayar is Supervisor of the Robotics Modelling, Simulation and Visualization Group in the Mobility and Robotics Systems Section at JPL. He has worked on a variety of robotics research projects at JPL. He manages and develops algorithms and software for the Surface Exploration Operations Simulator (SEOS) task in the Darts Simulation Lab.

Abhi Jain Abhi Jain is a Senior Research Scientist in the Mobility and Robotics Systems Section at JPL. His research interests include high-performance simulations, computational multibody dynamics, underactuated systems and nonlinear control.