An Autonomy Software Testbed Simulation for Ocean Worlds Missions

L. J. Edwards[1], U. Y. Wong[1], K. M. Dalal[2], C. S. Kulkarni[2], A. Rogg[2], A. Tardy[2], T. R. Stucky[3], O. M. Umurhan[3], D. Catanoso[4], and T. M. Welsh[5]

[1]NASA Ames Research Center, MS 269-3, Moffett Field, CA 94035; 1st author email: laurence.j.edwards@nasa.gov
[2]SGT, LLC. at NASA Ames Research Center, Moffett Field, CA 94035
[3]SETI Institute at NASA Ames Research Center, Moffett Field, CA 94035
[4]Universities Space Research Association at NASA Ames Research Center, Moffett Field, CA 94035
[5]Logyx LLC at NASA Ames Research Center, Moffett Field, CA 94035

ABSTRACT

One of NASA's goals is to identify and characterize potentially habitable environments in the Solar System and beyond. While the search for evidence of extant and extinct life, and conditions supporting life in the Solar System, has primarily focused on Mars, there is strong evidence that a number of outer solar system moons (e.g., Europa, Enceladus, and Titan) harbor subsurface oceans. These "ocean worlds" may be the best places to search for extant life beyond Earth. However, due to the extreme environmental conditions and communications constraints, current mission operations practice of pervasive ground control supervision will be unproductive, and substantially autonomous operations will be required. This paper provides an overview of an autonomy software testbed simulation facility, the Ocean Worlds Autonomy Testbed for Exploration Research and Simulation (OceanWATERS) that NASA Ames Research Center is developing to spur the development of such autonomous operation capabilities.

INTRODUCTION

In the search for habitable environments and evidence of extant and extinct life in our Solar System outside of Earth, moons and other icy bodies of the outer solar system have recently received increased attention, as there is strong evidence that a number of these icy bodies harbor subsurface oceans (e.g., Europa, Enceladus, and Titan).

Surface exploration missions to these "ocean worlds" will be amongst the most challenging to date. Ocean world environmental conditions and operational constraints are extreme. For example, one-way communication delays to Europa range between 35 and 52 minutes (cf., 3 and 22 minutes for Mars). Also, ocean world moons orbit gas giants and receive a heavy bombardment of radiation from planetary magnetospheres, which shortens electronics life spans. Such conditions motivate the use of significantly higher levels of autonomy than current practice in robotic planetary surface exploration.

The Ocean Worlds Autonomy Testbed for Exploration Research and Simulation (OceanWATERS) project led by NASA Ames Research Center (ARC) is developing an autonomy software testbed simulator that will spur the development and maturation of autonomy technologies enabling increased science operations productivity for ocean worlds surface missions (see Figure 1). In particular, autonomy that enables "fail-active" operations (continued operations without ground control intervention in the presence of sub-system failures) is of primary interest, and promises to substantially increase mission productivity.

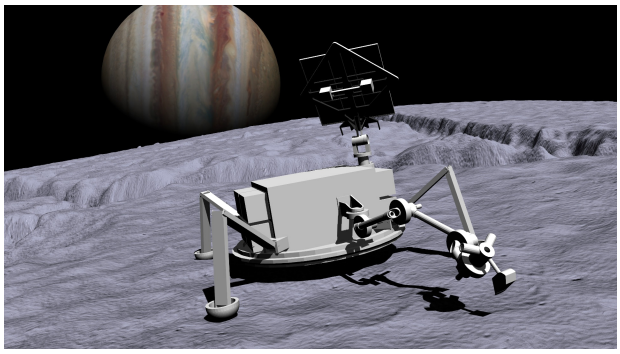

Figure 1. The OceanWATERS simulation environment with Europa Lander model.

NASA ARC has been involved in the research and development of visualization and simulation techniques for mission science operations and mission technology development since the early 1990s. Examples include the Mission Simulation Facility (MSF) [Flückiger, et al. 2002], the Mars Exploration Rover (MER) mission's "Viz" science operations planning software, [Edwards, et al. 2005], and a high fidelity "conops" (concept of operations) simulator developed for the proposed Resource Prospector (RP) mission, [Allan, et al. 2019]. From an implementation standpoint, OceanWATERS is most closely related to the RP conops simulator, but is closest in concept to the MSF. However, the MSF was designed to have general applicability in the field of robotics, whereas OceanWATERS is being developed with a specific focus on simulation of landers operating within an ocean worlds environment.

Complementary to the NASA ARC OceanWATERS effort, the NASA Jet Propulsion Laboratory (JPL) is developing a physical autonomy testbed. NASA intends to make these testbeds available to researchers from US academic institutions, companies, and/or other government agencies to perform development and testing of autonomy technologies.

APPROACH – AUTONOMY TESTBED SIMULATOR

The OceanWATERS testbed simulator provides environment (e.g., lighting and surface material properties) and lander (hardware and software) simulation capabilities against which autonomy software can be tested. To focus testbed development, the proposed Europa Lander mission was selected for initial lander and environmental modeling, but the testbed is designed to be readily adaptable to other planetary bodies and other lander configurations. The testbed is built exclusively from open source software, and in particular software that is well known to the robotics community. The software is in

active development and to this point a breadth first approach has been taken to rapidly develop system components and capabilities that cover the range of expected uses. While basic components have been integrated, full system integration is in progress, and further refinement of components and prioritization will be guided by feedback from early adopters and end users. In the following we describe the overall architecture, current component implementations, and capabilities of the testbed.

Architecture and Simulation Infrastructure

OceanWATERS is designed to be a modular, extensible, multi-process system. This facilitates the integration of component software written in different programming languages and software produced by disparate development efforts and vendors. This approach also provides flexibility to more readily adapt to unforeseen use cases that may come to light as the software is made available to end users. A high-level data flow diagram is shown in Figure 2, each component in the diagram denoting a separate process.

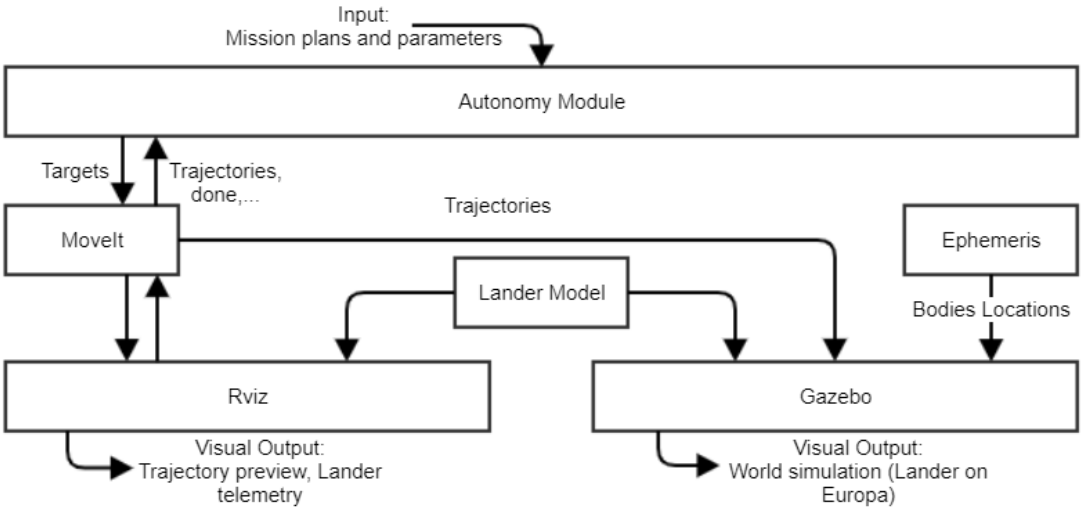

Figure 2. OceanWATERS architecture diagram

The Robotic Operating System (ROS) [ROS 2019] and the Gazebo simulation environment [Gazebo 2019], were chosen as the foundational software infrastructure for OceanWATERS simulation development. Both are widely used in the robotics community for research and development. ROS is essentially middleware for inter-process communication (IPC), but the open source effort that develops ROS provides a number of components from various sources that have been adapted to utilize ROS IPC (including Gazebo). Gazebo provides a framework for physical and visual simulation of robotic systems. Gazebo integrates various physics engines, including the Open Dynamics Engine, [ODE 2019], which is currently used by OceanWATERS. For rendering, Gazebo integrates the high-level OGRE open source rendering engine, [OGRE 2019], which utilizes open standard OpenGL graphics [OpenGL 2019] for low level rendering.

For the purposes of illumination and shadow modeling, communication constraint modeling, and celestial sphere appearance modeling, the relative location of the Sun and planetary bodies of the Solar System must be known. To this end, we utilize "SPICE" ephemerides and software developed by NASA JPL's Navigation and Ancillary Information Facility (NAIF) [NAIF 2019]. For a given observer location and time, SPICE software determines the position, velocity, and orientation of a given target body relative to the observer.

An autonomy module is being developed for OceanWATERS to exercise the testbed, and as an example implementation. For this purpose, the open-source PLEXIL plan execution system [PLEXIL 2019] (discussed later) was selected and incorporated as a ROS component.

Visual Appearance

Visual simulation is required to deliver simulated camera images to computer vision algorithms and as a diagnostic aid for human operators. We evaluated three main options for rendering images: rasterization, ray tracing, and hybrid solutions. Each of these has a different set of trade-offs regarding quality, performance, and ease-of-implementation. Rasterization is the real-time graphics method implemented in OpenGL and 3D graphics hardware, and was ultimately chosen for a variety of reasons: image quality was acceptable, it was the most performant, and it was by far the easiest to implement.

To deliver final images with as much realism as possible, we attempt to accurately simulate lighting, terrain reflectance, and digital camera characteristics. Custom OpenGL Shading Language (GLSL) [Khronos 2019] shaders are used to compute lighting and material properties.

Illumination. We simulate direct light from the sun, direct light from lander-mounted lights, and indirect light from other sources. As we are currently only concerned with the visual spectrum, light intensities in the following are defined in units of lux. Sunlight intensity I is calculated using the formula $I = E_{sc} \cdot (AU/D)^2$ where E_{sc} is the solar illuminance constant (128 kilolux), AU is one astronomical unit (1.496×10^{11} m), and D is the distance from the sun as provided by our ephemeris model. We currently only simulate mission sites on Europa and so have not implemented an atmospheric attenuation term.

Lander lights are implemented as spotlights, point sources that project a texture in a specified direction with a specified cone angle. The texture can be used to simulate the nonuniform lensing commonly observed in headlights and flashlights. We have not yet chosen a real-world light from which we can derive a final texture or light intensity, and the user may want to do this in order to customize the simulation. Lander lights are currently set to produce 1000 lux at a distance of 1 meter.

Other light sources that contribute to our images are starlight and reflected light. Reflected light includes light that has bounced off terrain or planets in the sky. For example, Jupiter covers about 12 degrees of visual angle as seen from Europa and

Saturn covers about 27 degrees of visual angle as seen from Enceladus, making them significant sources of reflected light at some mission sites. Reflected light is rendered into an irradiance environment map as a pre-render pass. Irradiance environment mapping provides a fast approximation of the overall intensity of indirect light. It precomputes light reflected from a perfect Lambertian surface and stores it in a cube map. For any point on a surface, an approximation of light contributed by many light sources and diffused at that point can be looked up using the point's surface normal as an index into the cube map. Starlight provides a very small contribution (2.2×10^{-4} lux). However, it is rendered into the irradiance environment maps, as it might provide detectable illumination for some sensors in the absence of other light sources.

Shadows. The sun is currently the only light source that casts shadows in the simulation. As the brightest source of direct light, it is the best first choice for shadows. We use Gazebo and Ogre3D's built-in shadow mapping solution with some customization to increase resolution and improve smoothing of shadow edges. Lander-mounted lights would be a good second choice for casting shadows, but software constraints in Ogre3D currently prevent this.

Eclipses occur every day on Europa and frequently on some other ocean moons. Simulating an eclipse on a moon essentially requires us to cast a shadow from the parent planet onto the moon's surface. Using our ephemeris model, we compute the fraction of the sun visible at a mission site and scale the sunlight intensity accordingly. This darkens the entire scene evenly but does not simulate the subtle gradation of a planet's penumbra. This is a reasonable approximation because these particular penumbras would be hundreds of kilometers across.

Reflectance. The formulation of a detailed reflectance model is in its initial stages, and we use a simple placeholder for now: surface reflectance for Europa is modeled with a surface albedo of 0.64 [Hamilton 2009], and variation is implemented with a normal map which is not physically based.

Surface Geometry

Synthetically generated terrain can model surface features beyond the resolution of currently available orbital data for testing a range of landing site conditions. However, as there has not yet been direct observation of features at the lander scale (< 1m), there is wide uncertainty as to the nature and distributions of features relevant to our simulator. The icy surface might take on the nearly flat, pebble laden texture of a desert landscape or the relatively smooth, high albedo characteristics of the surface of a large glacier, like in Antarctica. Owing to the atmosphere-free conditions, theoretical arguments suggest that the surface may be covered by spike-like ice structures called penitentes that can range from 0.1-10 meters scales shaped by differential insolation and sublimation processes [Hand 2019]. Given the recent observation of surface salts, the icy flats of Europa might express vast fields of salt efflorescence like the gnarled structures of Devil's golf course found on Death Valley's floor (see Figure 3).

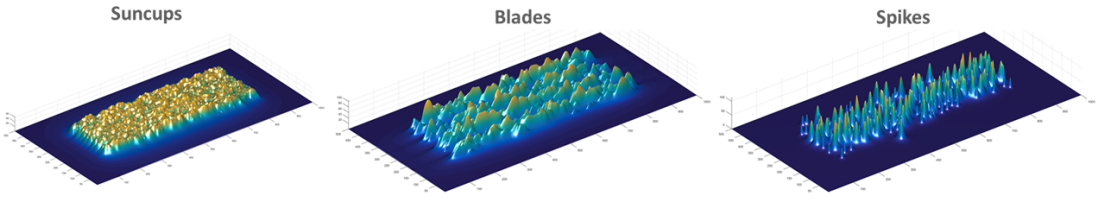

Figure 3. Icy penitente features can manifest in a myriad number of visual appearance classes. These can be simulated by using unique parameter combinations in a single numerical generator function.

We take a two-pronged approach to modeling icy terrain by utilizing a combination of fractal noise perturbation for small details (< 10 cm) and randomized procedural generation of monolithic features using analytical models (> 10 cm). This builds on software elements for Lunar terrain modeling developed by our team [Allan, et al. 2019], but with additional work to make the terrain better resemble ocean world environments.

The noise approach is appropriate for high-resolution features representing the environment within the near-field of the lander and necessary for physical interaction, while procedural feature generation is best suited for far-field terrain that provides aesthetic realism. Our approach is grounded by methodical analysis of low-resolution orbital information and terrestrial analog examples.

Generation of Far-field Features. Far-Field features primarily provide visual fidelity and improved immersion for our simulation. Terrain at distances of 10s to 100s of meters from the landing site will drive the appearance of the horizon line, determination of terrain occlusions, and interaction with planetary illumination (e.g. shadows). We will utilize gross-scale (10-50 m scale) Europa DTMs to represent the general rolling character of the terrain [Schenk 2008]. The DTMs will also influence secondary illumination phenomena (e.g., multiple reflections), though these are not currently implemented. We have modeled geometric features including hills, valleys, craters, blocks, penitentes and crevasses. These features are simulated on terrain with a nominal area of 200 m x 200 m and has changes in elevation as high as 30 m. The procedural generator operates on a 2.5D voxel grid that serves as the base plane, a digital terrain model (DTM), where the spatial resolution is 10 cm per voxel.

The process to generate a terrain is sequential and involves one step per feature type (see Figure 4).

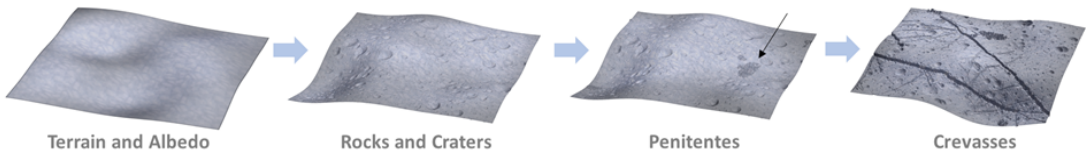

Figure 4. The procedural terrain generation process adds elements sequentially.

The procedural generator first creates an underlying map of hills and valleys by growing elevation seeds on a flat plane with morphological operations and adding noise. Alternatively, there is the possibility of loading a priori orbital maps at this stage. Craters and rocks are then added using separate uniform background distributions

which are of exponential size-frequency. Craters of a certain size will spawn ejecta fields which increase the probability of encountering rocks in specific areas. Penitente features are generated as anisotropic noise on a heightfield with 10 parameters controlling size, isotropy, density, variability, among other qualities. Lastly, fracture networks (e.g. crevasses) are generated using a tree model. There is also a phenomenological model that generates a range of plausible visual results, and is not based on real mechanical models of ice. Trunks, the largest fractures/crevasses, are randomly generated on the map by controlling dominant direction, thickness, depth, length and other variables. Along the trunk's length, smaller branches are allowed to spawn based on a branch factor and spatial probability. Example synthetically generated terrain models are shown in Figure 5.

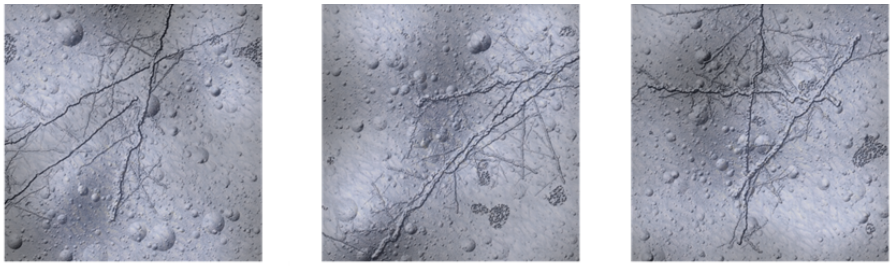

Figure 5. Three examples of randomly generated terrain using the same appearance parameters. All features are generated as offsets added to a heightfield. A wavelet method was devised to quickly blend features with an existing DTM, removing artifacts while preserving the fidelity.

Near-field Features. There is a particular lack of prior research on the disposition of terrain at the 1-10mm scale that is suitable for sampling simulation. In order to provide realistic references for building the near-field geometry, we took advantage of NASA field opportunities to Death Valley, California and the Atacama Desert, Chile in order to observe and record analog sites. Scenes of interest were preselected from satellite views based on previously known and partially surveyed regions of salt efflorescence, [Umurhan 2019]. At each site up to 8 locations were selected with the aim of capturing a maximal range of textural and color characteristics. A high-resolution laser scanner (with built-in color camera) was used to digitize 5m x 5m patches of terrain at 2.5mm resolution. Figure 6 illustrates examples of patches collected that span a range of salt and rock feature distributions. A total of 11 scenes were collected from which we plan to release a curated dataset of DTMs to the research community.

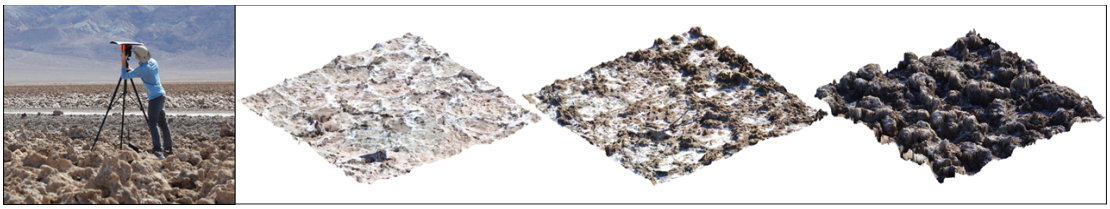

Figure 6. We used laser scanning technology to digitize high-resolution 3D models of analog terrain in Death Valley and the Atacama desert.

Utilizing the reference DTMs, we have been developing a near-field terrain generator based on the principles of fractal expansion. Perlin noise is generated using parameters visually tuned to match the salt, regolith and rock appearance. This process is

recursively applied to up-sample the terrain to higher and higher resolutions. A scripting framework automates this process as a series of passes similar to a graphical shader.

Camera Simulation

Images are initially rendered into an intermediate floating-point texture. We use a GLSL shader to post-process this texture and give it the characteristics of an image captured by a digital camera (see Figure 7).

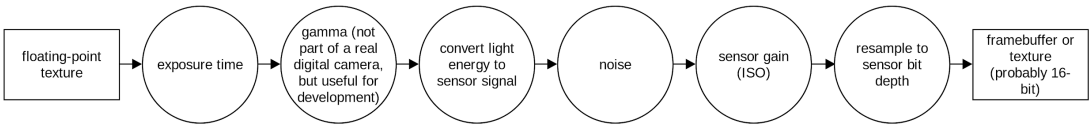

Figure 7. Our digital camera simulation pipeline.

The goals of the camera simulation are a) to allow a properly exposed image to result from realistic exposure time and sensor gain settings, b) to add a realistic distribution of noise to the image, and c) to quantize the image in such a way as to simulate the bit-depth of a real camera. The first can be achieved by inputting an image texture with realistic values in units of lux and by choosing a reasonable multiplier for the "convert light energy to sensor signal" step. Achieving the other goals is simpler because they can be simulated by distinct steps in the pipeline with no coupling to other steps or the input image.

After the pipeline is configured, a human operator or autonomy algorithm can control exposure time and sensor gain. Final images are written directly to the user's display or to a 16-bit texture, though the content might be quantized to a lower bit-depth to simulate a specific camera. (OpenGL does not provide any options for render targets between 8 and 16 bits.)

Lander Kinematics and Dynamics Modeling

A 6 degrees of freedom (DoFs) robotic arm is added to the simulator, with a Phoenix inspired trenching end-effector. The underlying path-planning tool used in the simulator is MoveIt, a ROS package. When planning an arm trajectory, MoveIt considers a simplified collision model to avoid interference with the lander body parts, as well as the masses and inertias of the rigid links. When executing a trajectory, Gazebo simulates both the kinematics and dynamics of the arm.

In order to emulate planetary operations, Rviz (the ROS data visualizer) is used to display the planned trajectory for user validation. At the same time, it displays the arm telemetry coming back from the lander (simulated in Gazebo or the physical hardware, should it be available).

Lander/Surface Interaction Modeling

Estimation of force response on the lander body, robotic arm, and arm end-effector due to collisions with the terrain allows testing of the force feedback control loops, collision fault contingencies, and sample collection strategies. Considering that analytical models are unable to account for general scoop geometries, terrain surface geometry, excavator fill, and heterogeneity of the terrain, and that experimental models present limited flexibility in the represented scenario, a computational Discrete Element Method (DEM)-based solution has been chosen to simulate the terrain/lander interaction. Providing force feedback from the terrain through the summation of all contact forces between each particle and the end effector, DEM is accurate and generalizable to all terrain interaction scenarios; however, is computationally intensive and cannot typically run in real-time. To provide high-fidelity terrain interaction force feedback for real-time simulations in the OceanWATERS testbed, DEM was used to record results of different terrain interaction scenarios for reference by simulator in the form of a force and torque lookup table.

Discrete Element Method Simulations. DEM is used to simulate different variations on the terrain/scoop interaction. Given the uncertainties in Europa surface material, the simulations were performed for three material types: snow simulant, ice fragments simulant, and dry sand simulant. The resultant force and torque that acted on the scoop's center of mass during the simulations were logged into a lookup table that is parameterized by scoop depth and material type. This lookup table is accessible by the OceanWATERS testbed to provide real-time approximation of terrain-scoop interactions.

Shear modulus and particle size optimization was performed to speed up the simulations. Complex shape particles are represented with clumps of spheres of different sizes, approximating sharp edges with concentrations of small-size spheres. However, the presence of smaller spheres greatly increases computation time. For this reason, simple particle configurations have been chosen and they are represented in Figure 8a, [Catanoso, et al. 2020]. The current OceanWATERS testbed version implements a single sample collection technique which consists of excavating the granular terrain through five consecutive passes. For each pass, the scoop executes the same trajectory consisting of a first circular motion for approaching the terrain, then a linear translation parallel to the terrain, and finally a second circular motion to leave the terrain. Figure 8b shows the three phases of the scoop trajectory during the single pass. The scoop configuration currently used in OceanWATERS, visible in Figure 8b, is a simplified version of the Phoenix Mars Lander scoop [Arvidson, et al. 2009].

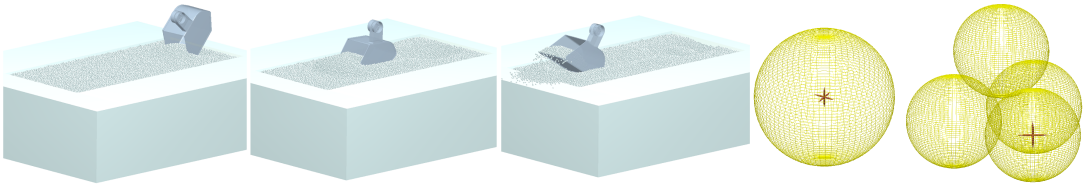

Figure 8. a) Circular-linear-circular sample collection strategy. b) Particle configurations used in DEM simulation: single sphere for snow and sand, four spheres approximating a tetrahedron for ice fragments. In all cases, the sphere diameter is 3.5mm.

A comprehensive analysis of the DEM simulation results and a study focused on reducing computation time are presented in [Catanoso, et al. 2020].

Power Modeling

The Europa lander is powered by a set of battery packs to perform all its operations. In the absence of fatal system faults, end of mission occurs upon complete discharge of the battery pack. Under these circumstances it is crucial to understand how batteries behave under different loading conditions and to capture that knowledge in the form of models that can be used by monitoring, diagnosis, and prognosis algorithms. For the simulation, an electrochemistry-based model of lithium-ion batteries is implemented that captures the significant electrochemical processes, is computationally efficient, and is of suitable accuracy for reliable end-of-discharge (EOD) prediction under different operating conditions of the lander. The algorithm output provides information which can be utilized by the autonomy module to command specific actions. These can be based on the time of operation, state of charge of the battery, temperature, etc.

Battery Modeling. In order to predict onboard EOD as defined by a voltage cutoff, the model computes voltage as a function of time given the current drawn from the battery. In this simulation a lumped-parameter ordinary differential equations form is being used, so it is efficient and usable for on-line prognostics. The model specifically is for a single Li-ion 18650 batteries with an average nominal voltage of 3.7V and nominal capacity of 2200mAh, however, the model is still general enough that with some modifications it may be applied to different battery chemistries. The battery pack is made up of several of these standard li-ion cells in series-parallel configuration to achieve required voltage and current ratings. Details of the model are discussed in [Daigle & Kulkarni 2013].

Autonomy Module

As an autonomy testbed it is not in the scope of the OceanWATERS project to develop a comprehensive autonomy solution for ocean world surface missions. In consequence, the system will include only basic autonomy capabilities that exercise the testbed and provide baseline functionality. The present model is that all required autonomous behavior is specified in pre-written *plans* to be stored and executed onboard the lander. Extensions of this model could include a facility to modify or replace existing plans as the mission progresses, with plan updates coming from Earth or generated by automated planners onboard and/or on Earth. Autonomy requirements are presently

being derived from a Europa reference mission description developed at JPL and thus far are met solely with static onboard plans. The plans are written in *PLEXIL*.

Developed by NASA, and available as open source software, PLEXIL (Plan Execution Interchange Language) is a language for representing plans for automation. It is a synchronous language with a strong formal semantics, [Dowek, et al. 2007]. A PLEXIL plan is a hierarchy of nodes whose execution is governed entirely by conditions and events. The PLEXIL executive executes the lander plans and is incorporated in OceanWATERS by embedding it in a ROS node. Commands and *lookups* (PLEXIL's means of reading the environment) are mapped to the corresponding ROS messages, services, and actions via an adapter that shares a common interface to both the simulation and physical testbeds (see Figure 9).

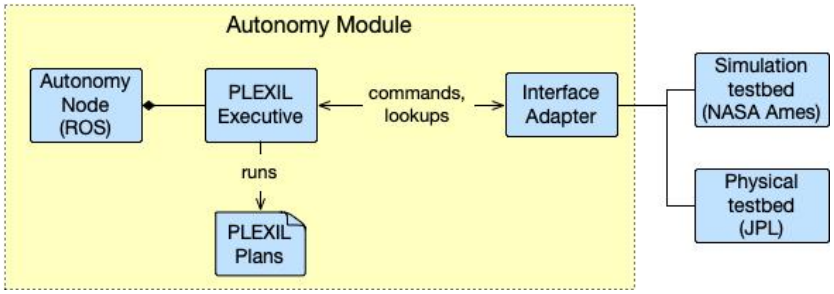

Figure 9: OceanWATERS autonomy module

At present only Sol 0 (day one) of the Europa reference mission is being modeled. The root PLEXIL node of this mission segment decomposes the day into its various tasks, which consist of both sequential and concurrent activities. The leaf nodes of the plan are typically invocations of lander operations such as arm movement.

CONCLUSION

The OceanWATERS project is developing a modular testbed simulator for the development of autonomy software relevant to ocean worlds surface exploration missions. This effort is initially focused on lander missions, and is using the proposed Europa Lander mission to drive implementation of capabilities. The testbed currently provides surface terra-mechanics modeling, as well as illumination and surface reflectance modeling, with simulated lander image sensor data and articulation telemetry. Although there is relatively little known about the surface conditions on ocean worlds, for the initial Europa simulation, every attempt has been made to incorporate the latest data and hypotheses regarding Europa surface conditions.

The testbed implements a multi-process architecture utilizing ROS as middleware and Gazebo to generate environment "ground truth". A reference autonomy module has been implemented to exercise the system and provide an example of interfacing to the simulator via the API.

The intent is to make the autonomy testbed simulator available for use by academia, government, and industry.

Future Work. In order to test the ability of autonomy software to flexibly re-plan operations in the face of faults or unexpected environmental conditions, a facility for injecting faults into the simulation will be developed. In addition, facilities for interrogating the system state will be further developed to enhance understanding and diagnosis of autonomy software behavior.

Europa contains ice-mixed compounds of either salt or sulfuric acid [Trumbo 2019], and there are observations of other trace chemicals, e.g., H_2O_2 [Trumbo 2019]. We plan to visually simulate a variety of these materials, and have the ability to vary the bidirectional reflectance distribution function (BRDF) depending on the material. We also plan to maintain the correct average albedo for each world we simulate.

On the arm-surface interaction side, the currently implemented terrain-scoop interaction lookup table, only handles the case of surfaces consisting of unconsolidated granular material. To additionally handle scenarios with crusty snow or solid polycrystalline ice, pre-processing of the surface with cutting or drilling end effectors will be simulated, and force/torque up tables generated. We are also working on the integration of an open-source DEM software solution into the OceanWATERS testbed. This will allow users to customize the simulation for novel types of terrain, end-effector geometry, and sample collection strategy.

REFERENCES

Allan, M., et al., (2019). "Planetary Rover Simulation for Lunar Exploration Missions". IEEE Aerospace Conference.

Arvidson, R. E., Bonitz, R. G., Robinson, M. L., Carsten, J. L., Volpe, R. A., Trebi-Ollennu, A., ... & Shaw, A. S. (2009). Results from the Mars Phoenix lander robotic arm experiment. *Journal of Geophysical Research: Planets*, *114*(E1).

Catanoso, D., Stucky, T., Case, J., and Rogg, A. (2020). "Analysis of Sample Acquisition Dynamics Using Discrete Element Method". IEEE Aerospace Conference, Yellowstone Conference Center, Big Sky, Montana, USA, March 7-14. (in review)

Dowek G., Muñoz C., and Pasareanu C. (2007). "A Formal Analysis Framework for PLEXIL", 3rd Workshop on Planning and Plan Execution for Real-World Systems.

M. Daigle and C. Kulkarni (2013). "Electrochemistry based Battery Modeling for Prognostics", Annual Conference of the Prognostics and Health Management Society (PHM 2013), October 2013, New Orleans, LA.

Edwards, L., Bowman, J., Kunz, C., Lees, D., Sims, M., (2005). "Photo-realistic Terrain Modeling and Visualization for Mars Exploration Rover Science Operations," Proceedings of IEEE SMC 2005, Hawaii, USA October.

L. Flückiger and C. Neukom (2002). “A new simulation framework for autonomy in robotic missions”. Proceedings of IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland (pp 3030 -- 3035).

Gazebo (2019). Gazebo opensource software website, http://gazebosim.org.

Hamilton, C. J. (2009). Europa Statistics, http://solarviews.com/eng/europa.htm.

K. Hand, A. Murray, J. Garvin, W. Brinckerhoff, B. Christner, K. Edgett, B. Ehlmann, C. German, A. Hayes, T. Hoehler et al. (2017). “Europa lander study 2016 report: Europa lander mission,” NASA Jet Propulsion Lab., La Canada Flintridge, CA, USA, Tech. Rep. JPL D-97667, 2017.

Khronos (2019). Khronos Group consortium website, https://www.khronos.org/opengl/wiki/Core_Language_(GLSL).

NAIF (2019). The Navigation and Ancillary Information Facility website, https://naif.jpl.nasa.gov/naif.

ODE (2019). Open Dynamics Engine open source software website, https://www.ode.org.

OpenGL (2019). OpenGL website, https://www.opengl.org.

PLEXIL (2019). Plan Execution Interchange Language (PLEXIL) open source software website, http://plexil.sourceforge.net.

ROS (2019). Robotic Operating System (ROS) open source software website, https://www.ros.org.

Schenk, P. (2008). “Cartographic and Topographic Mapping of the Icy Satellites of the Outer Solar System”. Proceedings ISPRS XXXVII, Commission IV, WG IV/7, 2008.

Trumbo, S.K., Brown, M.E., and Hand, K.P. (2019). “Sodium chloride on the surface of Europa”. Science, Volume 5, Issue 6, id. aaw7123 (2019).

Umurhan, O.M., Allan, M.B., Edwards, L.J., Tardy, A. Welsh, T.M., and Wong, U. (2019). “High resolution digital elevation models of The Devil's Golf Course: a possible terrestrial analog of Europa's surface.” AGU Fall Meeting 2019, P53C-3468.