Autonomous Ocean World Exploration: Advancement of a Software Testbed

K. Michael Dalal[1], Chetan S. Kulkarni[2], Thomas R. Stucky[3]
KBR Inc. at NASA Ames Research Center, Moffett Field, CA, 94035, USA

Terence M. Welsh[4]
Logyx LLC at NASA Ames Research Center, Moffett Field, CA, 94035, USA

The search for life signs and potentially habitable bodies in our solar system and beyond is one of NASA's top priorities. The prime destinations for such exploration are bodies containing liquid water, such as Jupiter's moon Europa and Saturn's moon Enceladus. Initial missions to these "ocean worlds" will be robotic, and because of long communication lags and blackouts, harsh environments, and limited battery life, a high degree of onboard autonomy will be required. To this end NASA has developed the *Ocean Worlds Autonomy Testbed for Exploration, Research, and Simulation* (OceanWATERS), a software testbed for surface mission autonomy. Under grants from NASA's *Autonomous Robotics Research for Ocean Worlds* (ARROW) and *Concepts for Ocean worlds Life Detection Technology* (COLDTech) programs, collectively 6 research teams at universities and companies across the United States have developed autonomy solutions using OceanWATERS, tackling a variety of surface mission challenges and employing state-of-the-art AI-based solutions. This paper provides a self-contained high-level description of OceanWATERS while detailing enhancements made since its 2020 release. It provides a high-level summary of the projects undertaken by participants in the ARROW and COLDTech programs and how these efforts have helped shape OceanWATERS. Finally, potential future work and directions for the testbed are listed.

I. Introduction

The search for life (extinct or extant) and potentially habitable bodies in our solar system and beyond is one of the 12 priority science questions outlined in the National Academies' 2022 decadal survey [13]. Extraterrestrial destinations containing liquid water present an opportunity to search for life as we know it, and in recent years an increasing number of such locations have been discovered within our solar system. Several Jovian moons—Europa, Ganymede, and Callisto [20]—and the Saturnian moons Enceladus [17] and Titan [19] are known or suspected to harbor massive subsurface oceans. Of these "ocean worlds", Europa is the focus of a planned NASA orbiter mission, Europa Clipper [10], as well as an early lander mission concept, the Europa Lander [7][8].

Whereas most robotic missions to the Moon and Mars (e.g. orbiters, rovers, landers) have had ground controllers on Earth tightly involved in mission operations, missions to more distant worlds will require a high degree of onboard autonomy due to long communication lags and blackouts, harsh environments (radiation, cold), and limited battery and hardware life. The past decade has seen great advances in both AI technologies and computing scalability and performance that offer promising solutions for spacecraft autonomy and motivate the software system and research programs described in this paper.

---

[1] Computer Scientist, NASA Ames Planning and Scheduling Group; Project Lead, OceanWATERS.
[2] Scientist, NASA Ames Diagnostics and Prognostics Group, AIAA Associate Fellow.
[3] Software Engineer, NASA Ames Intelligent Robotics Group.
[4] Senior Software Engineer, NASA Ames Intelligent Robotics Group.

II. Overview of OceanWATERS

Ocean Worlds Autonomy Testbed for Exploration, Research, and Simulation (OceanWATERS) [5] is a software package intended as a virtual environment for testing lander autonomy solutions. It was in active development at the NASA Ames Research Center from 2018-2024 and has been available as open source since 2020 (https://github.com/nasa/ow_simulator). OceanWATERS provides a physical and visual simulation of a prototypical lander on the icy surface of Europa (Figure 1). It is built on the Robot Operating System (ROS) and Gazebo simulator, which are long-standing open-source robotics standards, and runs on commercial-grade Linux workstations. Primarily aimed as a testbed for high-level autonomy (e.g. applications of artificial intelligence), OceanWATERS does include a relatively low-level autonomy component: a facility for executing plans (autonomy specifications) written in the *PLan EXecution Interchange Language* (PLEXIL) [24], an open-source software package also developed at NASA Ames. The intention was that external autonomy modules could generate PLEXIL plans and interact with the PLEXIL executive, or bypass PLEXIL altogether and interface directly with the ROS layer of the simulator.

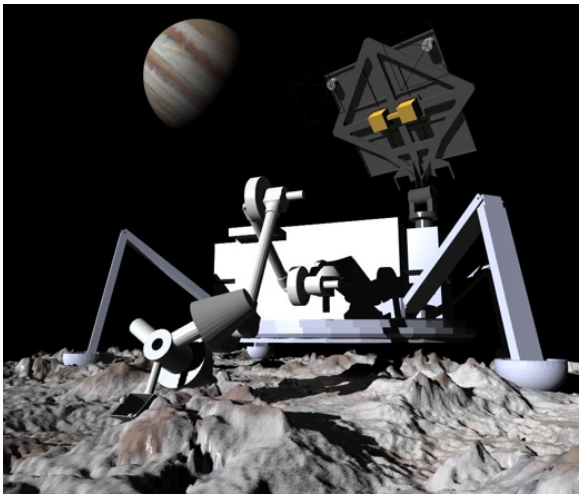


Figure 1. Lander on Europa as simulated in OceanWATERS. The scoop is collecting a sample from a trench. Jupiter is seen in the background.

The OceanWATERS lander was modeled after requirements and specifications made in JPL's Europa Lander Study of 2016 [8]. The lander's controllable parts include stereo cameras and spotlights mounted on an antenna mast that pans and tilts. A 6-degree-of-freedom (DoF) robotic arm with a force-torque sensor has two interchangeable end effectors: a grinder designed for digging trenches and a scoop for collecting ground material. A sample receptacle mounted to the lander can be filled with regolith and emptied on demand. The lander is powered by a simulated non-rechargeable battery pack.

The simulated environment includes a section of terrain upon which the lander sits. Several terrain models with a variety of surface and material properties can be selected, including one that is a digital replication of a portion of the Atacama Desert in Chile, a potential analog of Europa's surface [25]. Simulation of Europa's celestial sphere, which includes the position and movement of Jupiter and the Sun, is provided by JPL's SPICE ephemeris toolkit [1]. Direct and indirect lighting from the sun with associated surface illumination, reflectance, and shadows are also simulated.

Operations supported by OceanWATERS include panoramic and directed imaging of the environment and lander workspace, both Cartesian and joint-level arm commanding, grinding of the terrain surface (e.g. digging a trench), and scooping of ground material (Fig. 1) which can be discarded or collected as science samples; science operations themselves are not simulated except for their estimated power draw. These operations are realized as ROS actions and are complimented by a wide selection of telemetry as ROS messages continually published by each lander subsystem.

A diagram of the OceanWATERS architecture is given in Figure 2. A more detailed description of the elements in this section can be found in [5].

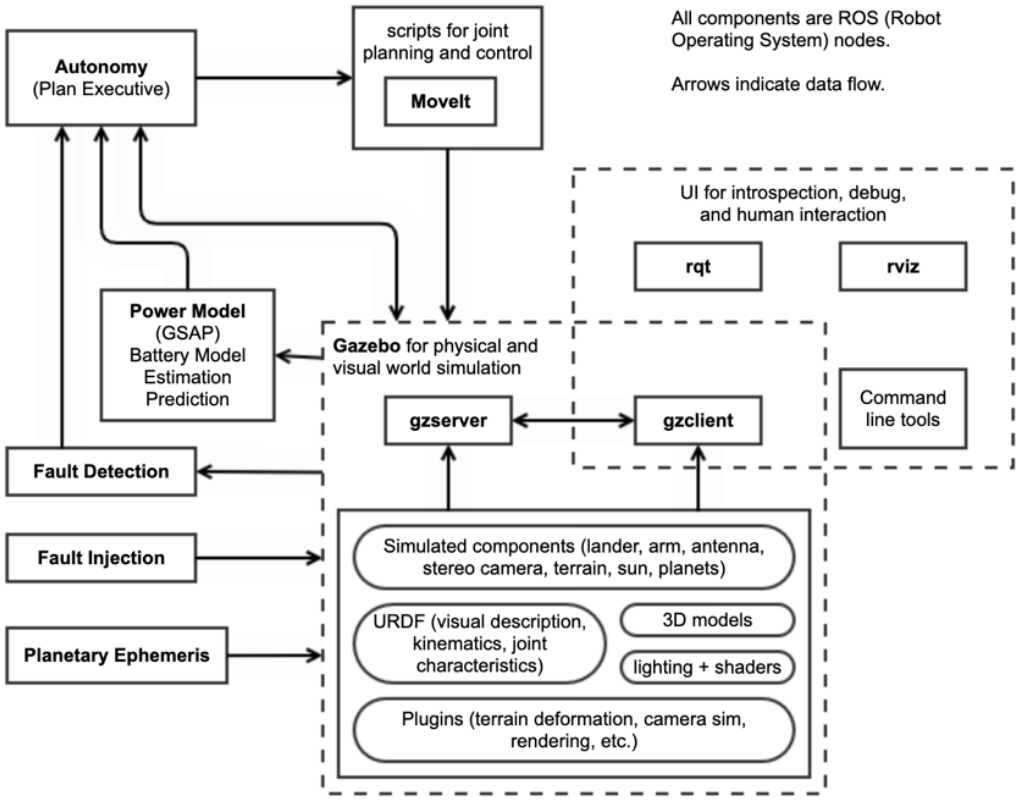


Figure 2. OceanWATERS architecture illustrating interaction of ROS and Gazebo components.

III. Advancement of OceanWATERS

OceanWATERS has had significant enhancements since its open-source release in 2020, described in [5]. Some of these were driven or shaped by feedback from the ARROW and COLDTech research teams and requirements of their projects, while others were generally useful new features or improvements of existing ones. The following subsections highlight these enhancements.

A. Unification with OWLAT

While all 6 of the ARROW/COLDTech research teams used OceanWATERS as a testbed for their technologies, half of them also scoped their work for demonstration on *Ocean Worlds Lander Autonomy Testbed* (OWLAT) [14], a complimentary physical testbed developed concurrently at NASA's Jet Propulsion Laboratory (JPL). OWLAT features a Barrett WAM robotic arm mounted on a Stewart platform and includes a pan-tilt unit that mounts an Intel RealSense camera (Figure 3). It operates on a physical "sandbox" that can be filled with a variety of planetary surface simulants. At the start of the ARROW program in 2020, the simulated lander in each testbed was disjoint in supported operations and did not share a common software interface. The testbed teams collaborated in the development of a unified interface to lander commands and telemetry that would allow autonomy solutions for one testbed to be adaptable with minimal effort for the other. One challenge here was the fundamental differences between the testbeds. For example, the OceanWATERS lander's 6-DoF virtual arm has two automatically interchanging end effectors while OWLAT's 7-DoF physical arm has 4 different end effectors that must be changed manually. The unification effort spanned many months and resulted in the testbeds having a large overlap in lander functionality and a highly unified ROS interface for commands and telemetry. Enhancements to OceanWATERS from this effort were numerous and are described throughout the remaining subsections.

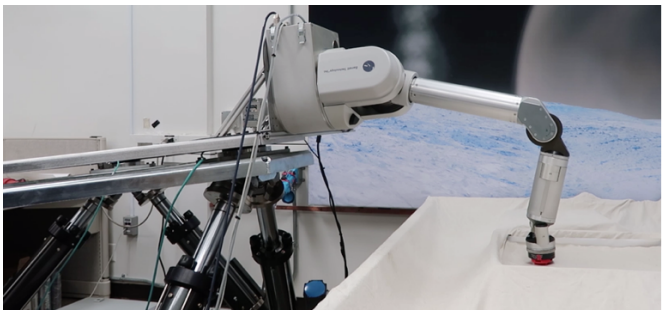


Figure 3. The OWLAT testbed at JPL.

B. Power System Modeling

The OceanWATERS lander is powered by a simulated lithium-ion battery, which began as a single 18650 cell model and was eventually upgraded to the envisioned 24-cell pack model using the 6S4P configuration. Due to the harsh environment of ocean worlds and other constraints of such a mission, the battery pack is non-rechargeable. Thus, battery life is a critical constraint for onboard autonomy since battery depletion effectively ends the mission. Monitoring the battery pack's state-of-charge (SOC) and temperature, estimating future states based on present and projected power load, and predicting the eventual end-of-discharge (EOD)—more usefully represented as remaining useful life (RUL)—are essential onboard capabilities. Such capabilities are provided by our prognostics framework.

The *Generic Software Architecture for Prognostics* (GSAP) [21] is an open-source prognostics engine developed at NASA Ames that was integrated into OceanWATERS and models the lander's battery pack. GSAP takes as input the battery's present power draw, temperature, and SOC estimate, and produces an RUL estimate that can be queried by autonomy. Details of the function and architecture of prognostics and GSAP are given in the Appendix. In practice, up to 4 instances of GSAP can be run with OceanWATERS, each performing prognostics for a subset of the battery pack for a more accurate overall prediction. Due to GSAP's significant CPU load, the number of instances can be specified by the user.

Battery prognostics would not be accurate without accounting for the lander's *entire* power draw. A power consumption model was added to all simulated subsystems, e.g. the mechanical effort of joint movement is translated into electrical power draw. Basic support for modeling the power consumption of the compute board, communications, heating, and other subsystems not otherwise simulated in OceanWATERS was provided in the form of user-settable parameters for baseline power draw.

C. Fault Modeling

Detection, isolation, and mitigation of faults is a critical capability in autonomous space systems and was a key focus in most of the ARROW/COLDTech research projects. A fault modeling framework in OceanWATERS was developed and matured throughout the research period. Two kinds of faults were addressed: *execution errors* and *goal errors*. Execution errors are hardware failures, such as a frozen or locked arm joint or a disconnected battery cell. Goal errors, always associated with a lander command, occur when there is a significant difference between a state expected after the command, and the actual state, e.g. the arm or antenna not ending at the specified position after being commanded to move. Execution errors will often cause goal errors. However, goal errors can occur in the absence of hardware problems, e.g. arm movement getting blocked by an unexpected obstacle.

OceanWATERS models a variety of execution and goal errors in all its lander's subsystems: arm, antenna, camera, and power system. Execution errors can occur in two ways. First, they can be *injected* by the user or an external module. Fault injection is a means of "faking" a fault and has the effect of modifying relevant telemetry (e.g. the battery's temperature) by substituting a stream of error-indicating values. Second, execution errors can occur "naturally" during lander operation, e.g. by overstressing a component. In contrast, goal errors cannot be injected; they only occur when a command's objective is not accomplished. The predefined execution errors are automatically detected by a module (ROS node) that continuously monitors telemetries and sets *fault flags* in a published ROS message when anomalies are detected. Goal errors are detected in the implementation of commands, e.g. the Pan command checks whether the antenna reached the commanded angle. Goal errors are also published via fault flags.

Fault detection in OceanWATERS is primitive, representing what might easily be implemented at a low level in a real lander system. Atomic faults are detected by simple tests on individual telemetry streams. Sophisticated fault detection and isolation (when data suggests multiple possible problem sources) was out of scope for the testbed, but

exactly the kind of problem intended for its users. While an autonomy module attached to the OceanWATERS simulation (including the provided PLEXIL executive) can see and react to the simple faults detected by the simulator, more advanced algorithms would be needed in autonomy to detect a variety of subtle, intermittent, or transient faults and to identify root cause conditions outside the scope of the simulator's ontology. For example, a locked arm joint might be caused by ice buildup as demonstrated in the CARL project [4].

In general, a fault renders a component inoperable in its current and/or future operation until the component is repaired or the condition causing the fault is removed. In OceanWATERS, injected faults (execution errors) are removable (clearable) only by the user. They cannot be cleared by autonomy, just as a broken physical system in general cannot be "fixed" by a command from autonomy; a repair procedure, or even human intervention, may be needed. In contrast, goal errors can *only* be cleared by autonomy, and OceanWATERS provides a command (ROS action) for this purpose.

D. Lander Commands and Telemetry

The OceanWATERS simulated lander had numerous improvements since its initial release, including an expanded dictionary of commands, new and improved mission-like telemetry, and improved arm trajectory planning. Its command implementation was refactored into an extensible Python package.

The largest improvement to OceanWATERS, which resulted from the unification effort with OWLAT, was the addition of many arm commands. There are new commands for directing the arm to a specified pose, including a "guarded" version that stops arm movement as soon as a force or torque threshold is breached. Joints can now be moved individually or collectively with commands that also have a guarded version. Touching or "poking" to find the ground position can now be performed from any angle. A command to stop the arm in motion was added to support interruption and is automatically called when an arm fault occurs.[2]

The stereo camera interface has been improved with the inclusion of a command to capture images and another to set camera exposure; previously these were possible only through manual ROS message publishing and parameter settings, respectively. The lights on the lander's antenna mast can now be individually adjusted in brightness or turned off via a command. Another command can be given a point in space and the lander's antenna mast will pan and tilt to face it. The lander sample collection dock can be emptied to support multiple sample collections in one simulation session. Finally, to support more sophisticated fault handling that requires goal errors to be cleared before issuing the next command, the FaultClear command has been added.

The effort to unify the testbeds also improved the OceanWATERS telemetry dictionary. Use of enumerations and bitflags for the several different fault telemetry ROS messages were standardized, and constants for interpreting bitflag or enumeration values are now provided in the message definitions themselves rather than left solely to documentation. Dedicated ROS topics for arm motor telemetry were made available for arm joint positions, velocities, accelerations, and efforts. Additionally, the arm's scoop end effector's position in 3D space is published at a regular rate on its own dedicated topic.

A new and extensible ROS action server framework was factored out of disparate Python scripts that originally implemented the lander commands. The new framework presented an opportunity to rework the planning of lander arm command trajectories for a higher degree of accuracy and control. The previous arm command trajectories relied upon ad-hoc commanding of individual joint angles which resulted in slightly different trajectories depending on where in the lander's workspace the arm was directed. For the same reason, commanded dig depths were typically not adhered to and circular motion of the scoop for an arbitrary radius was impossible. These shortcomings were addressed by building a workhorse Python class that handled necessary geometry calculations and calls to the MoveIt planner to string together any series of end-effector translation (linear or circular) or joint translations into a single trajectory object that could be passed onto MoveIt for arm trajectory execution. These improvements were then used to rework every arm command using Cartesian coordinates and translations where necessary to ensure consistent trajectories were generated regardless of workspace location.

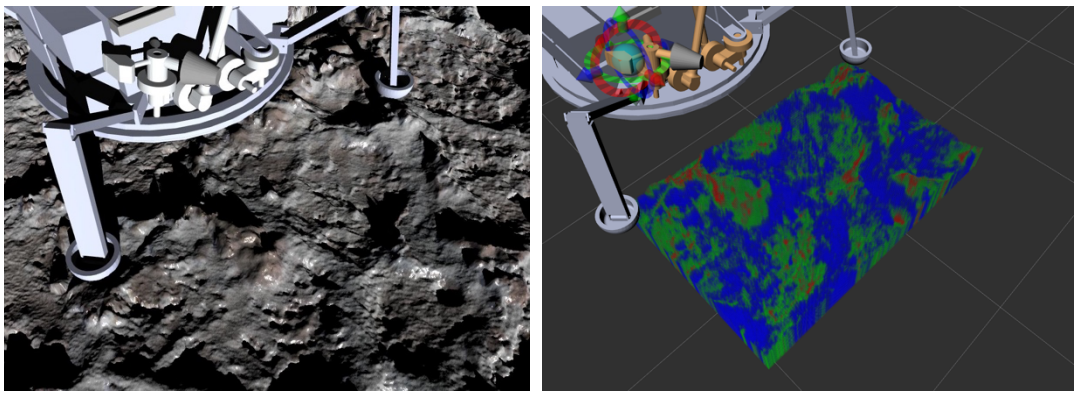
An unforeseen benefit of the command and trajectory refactoring was a Python package that facilitates the creation of new lander commands, even with arm movements, so users may customize the lander command dictionary as needed for their studies.

---

[2] This behavior did not suit the REASIMO project [23], which did its own low-level arm control and wanted its controllers to respond to faulty telemetry directly. An option to allow the arm to continue movement during arm faults was added to OceanWATERS.

E. Terrain Interaction and Sampling

The 2020 release of OceanWATERS could simulate the digging of trenches with a combination of the grinder end-effector to process terrain into tailings and the scoop end-effector to collect those tailings. This was done by tracking the trajectory of both end-effectors and reducing the terrain's height wherever they intersect with it. The grinder end-effector primarily modifies unseen collision geometry whereas the scoop end-effector only modifies the visualized terrain. As a result, the scoop end-effector is incapable of passing through terrain that has not yet been processed by the grinder end-effector.



(a) **(b)**

Figure 4. (a) The workspace on the front side of the lander in the atacama_y1a world. (b) An Rviz visualization of the material grid generated from terrain surface colors. The different visualization colors red, green, and blue are chosen to represent the different materials present.

The terrain interaction model has been expanded to include dig force, collectable regolith particles, and a multi-material terrain model that correlates material concentrations to terrain surface colors. The result of these improvements is the capability within OceanWATERS to simulate an end-to-end sample collection sequence consisting of sample site identification based on surface color, trench preparation, sample collection, and sample delivery. The remainder of this section describes the enabling software.

The multi-material terrain model is implemented as a grid of *voxels*, which are the 3D analogue to pixels. The user defines a domain in the Gazebo world that intersects the terrain where the lander is expected to dig; this domain is known as the workspace. The workspace is subdivided by voxels of constant side length. Each voxel within the workspace is populated with a blend of materials from a list than can be defined by the user. Figure 4 shows an example voxel grid generated for a particular workspace. At present, material properties only vary in terms of their density and color. The density is reflected in the mass of the regolith particles that result from digging, and the color acts as a reference point in RGB color space relative to the surface colors of the terrain. The concentration of each material in a voxel is relative to the RGB color space distance between the material's color and the surface color directly above it. This means the voxel grid will contain a material distribution that correlates with the color of the terrain, which enables the testing of autonomous sample site selection that will be reflected in the material concentrations of what is sampled.

Each simulation frame in which the scoop or grinder end-effectors modify the terrain results in two images published on a ROS topic: one is the differential image and the other is the result image. Figure 5 shows an example of these images during a dig. Within only the region modifications occurred, the differential image encodes changes in heights to the terrain for that frame and the result image encodes final heights after the modification; therefore, both images added together give the shape of the terrain in that region prior to the modification. These two images are used to calculate the volume displaced by that modification frame. By integrating over the material concentrations of each voxel within that volume, an overall concentration for that modification frame can be computed. The concentrations and volumes are published onto another ROS topic and received by the regolith system that handles spawning and tracking of regolith particles. This allows for the tagging of each regolith particle spawned with the material concentration of where it was sampled.

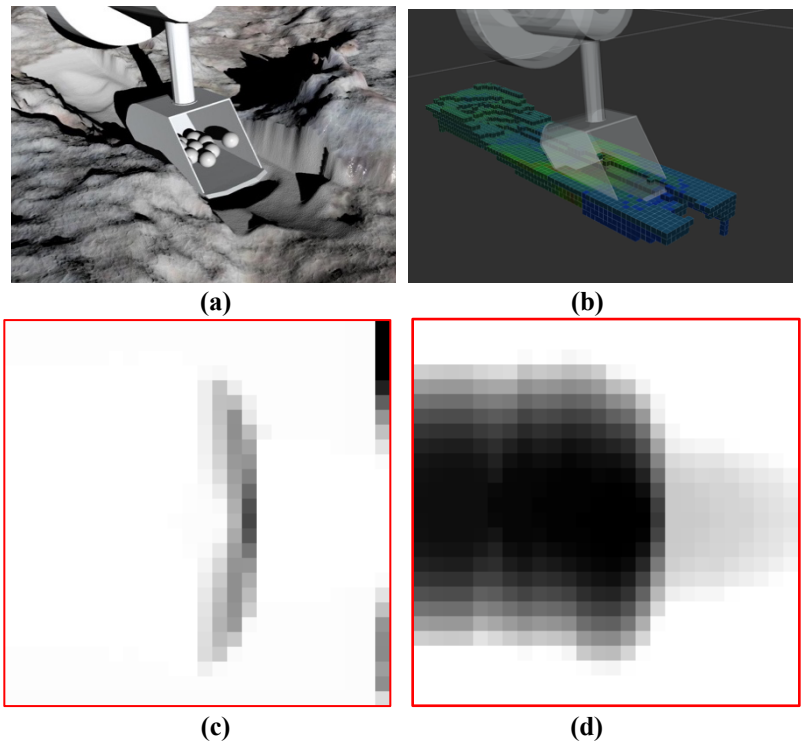


Figure 5. (a) The scoop end-effector excavating tailings. (b) An Rviz visualization of the voxels within the material grid the scoop is collecting from. The colors displayed are based on the concentration of materials within the voxels being scooped. (c) and (d) are the differential and result images, respectively, generated by the dig action seen in (a) and (b). The right-hand side of (c) and (d) is the front facing direction of the scoop.

To calculate the forces experienced by the scoop end-effector during a dig, the Balovnev bucket force equations [2] are used which provide an analytic estimate for the forces on the scoop both in the direction of digging and in the direction orthogonal to the terrain surface. This same dig force equation was used by *Gallo et al* to estimate dig forces on their prototype lunar vehicle as it plowed through simulant material [6]. The same differential images as described in the previous paragraph are fed into the dig force estimator to compute the depth of dig for each modification frame. The maximum of each image is fed into a moving maximum filter to the smooth results, and the resulting value is fed into the bucket force equation to estimate the dig force and inject it into Gazebo's dynamics. This allows for the power consumed by a dig operation to be proportional to the depth being dug.

F. Visual Simulation

Visual simulation is required as feedback to users and to deliver simulated lander camera images to other components, such as autonomy. At the beginning of the project, images were rendered using Ogre3D, the default renderer in Gazebo, with custom shaders programmed in OpenGL Shading Language (GLSL). Later work added many small refinements and some significant additions to the shader code and Gazebo plugins that support it.

The original Lambertian shading model was replaced by Oren-Nayar [16]. Oren-Nayar adds a variable backscatter component, which can more believably simulate materials found in nature, such as rock, ice, and salt.

Achieving realistic shadows with real-time graphics is difficult and the details are highly dependent on the scene being rendered. The initial implementation of sun shadows had many visual artifacts, such as shadows disconnected from their casters and light bleeding into shadowed regions. A high level of refinement was achieved by switching from Gazebo's default back-face shadow maps to front-face shadow maps and adding custom shadow map blurring and biasing to the shaders.

The digital camera simulation is achieved by post-processing images to add camera effects, such as exposure time, sensor response, noise, and bit-depth. A significant addition has been lens flare, which is packaged with Gazebo as a plugin that uses Ogre3D image compositing to layer new imagery on top of existing images. However, adding it to

the simulation proved difficult. Having variable exposure and bit-depth requires rendering the original image to a floating-point texture, which was not originally supported by Gazebo. The team submitted modifications to Gazebo that allowed the lens flare plugin to function within a floating-point image pipeline.

During digging operations terrain can be deformed causing the heightmap used by Gazebo and Ogre3D to update. The normal map that describes the terrain's *surface normals* also gets updated; without this feature, surfaces facing away from the sun could appear to be directly lit. An example of a trenched region of the terrain can be seen in Figure 6. After trenching, modified terrain is rendered with a new material, giving surfaces processed by the grinder a rough, snowy appearance.

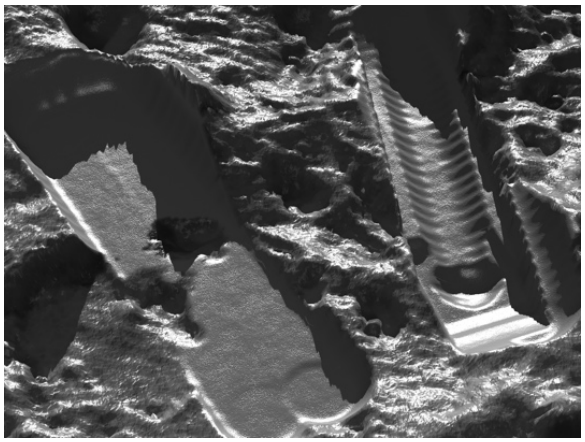


Figure 6. Two examples of modified regions. The region on the left was processed by the grinder end effector only. The region on the right was processed by the grinder and the tailings were removed by the scoop.

G. Autonomy Package

The OceanWATERS plan execution module, based on PLEXIL [24] and embedded in a ROS node, has been enhanced in numerous ways over the past several years. A significant architectural change was made to support plans that execute only on OceanWATERS, only on OWLAT, or on both testbeds. This division was necessary due to the testbeds having an autonomy interface that. while highly unified. is not identical. Dozens of new exemplary autonomy plans exercise every command in each testbed and implement a variety of test sequences and sample mission segments. Two abstractions on faults were added to the model described in Section C. First, plans can query for specific, atomic faults as defined by the enumeration of execution and goal errors; they can also query for "general" faults: the presence of any specific fault in a subsystem. Second, a user-specifiable model of fault dependencies allows the presence of any specific fault to render given plans or subsystems inoperable; for example, one can specify that a locked pan joint prevents the execution of any plan that requires panning the antenna. There are sets of plans that illustrate basic fault handling patterns as well as a rudimentary health monitor for the lander. One novel enhancement for OceanWATERS was made to the PLEXIL system itself: the ability to create persistent "checkpoints" of a plan's execution and enable the plan to resume where it left off in the event of an executive crash, e.g. a reboot of the lander's computer. A graphical interface for selecting from available plans and sequencing and monitoring their execution is shown in Figure 7 and greatly eased usability of PLEXIL. Finally, an ancillary project at NASA Ames addressing formal verification and validation (V&V) of PLEXIL used some OceanWATERS plans in testing a model checker for PLEXIL [18], and this effort resulted in insights for improving PLEXIL as well as the OceanWATERS plans.

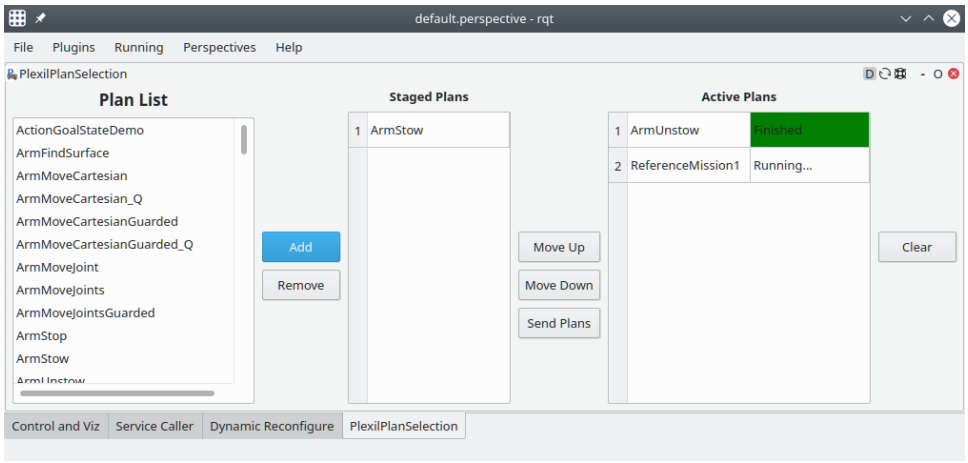


Figure 7. Interface for PLEXIL plan selection and execution.

IV. Research Programs

NASA's initial development of OceanWATERS and OWLAT was the first step in a plan for realizing candidate onboard autonomy solutions for ocean world landers. In 2020 NASA solicited applications for its *Autonomous Robotics Research for Ocean Worlds* (ARROW) program, and in 2021 the similar *Concepts for Ocean worlds Life Detection Technology* (COLDTech) program [12]. Collectively six research teams, based in universities and companies across the United States, were awarded grants to develop and demonstrate autonomy solutions on OceanWATERS and OWLAT. These 1–2-year projects are now complete, and a wide variety of autonomy challenges in ocean world surface missions have been addressed. This section provides a high-level summary of these efforts. The first two paragraphs describe ARROW projects, and the final 4 are COLDTech projects, each listed arbitrarily.

The *Resource Adaptive Software Purpose-Built for Extraordinary Robotic Research Yields – Science Instruments* (RASPBERRY SI) project led by a team at University of South Carolina explored techniques in machine learning, probabilistic planning, and causal analysis. They developed software algorithms and tools for root cause identification of faults, causal debugging, causal optimization, and causal-induced verification. They applied these technologies on in-house robots as well as the ocean world testbeds [9], and a portion of their OceanWATERS-based solution has been open-sourced at https://github.com/nasa-raspberry-si/autonomy.

Honeybee Robotics' *Stochastic PLEXIL* (SPLEXIL) project extended PLEXIL with stochastic decision-making by enabling PLEXIL plans to choose from a variety of policies (sub-plans) ranked by reinforcement learning in which the policies are tested and evaluated by an internal plan simulator employing partially ordered Markov Decision Processes (POMDP). A small portion of this work, a ROS action-based library for PLEXIL, was open-sourced at https://github.com/honeybee-robotics/plexil_actionlib.

A team based at Lockheed Martin developed and applied a technology called *Causal and Reinforcement Learning* (CARL) [4]. CARL is used to create an adaptive and resilient autonomous agent that learns the dynamics of an environment and causal connection between variables to generate behaviors and diagnose and circumvent faults, both expected and unexpected. They integrated a model of JPL's mission ready *Cold Operable Lunar Deployable Arm* (COLDArm) [15] into OceanWATERS and demonstrated a "fail-active" scenario that diagnosed ice buildup on the arm's end effector and resolved this fault by activating the heater.

The aim of *Robust Exploration with Autonomous Science On-board, Ranked Evaluation of Contingent Opportunities for Uninterrupted Remote Science Exploration* (REASON-RECOURSE) [11], a project based at the University of Colorado, was maximizing science return from the lander in the face of limited communications bandwidth and battery life. An onboard component (REASON) utilizes automated planning driven by temporal logic specifications to optimally schedule lander activities and prioritize downlink data. A ground component (RECOURSE) is a graphical tool that displays science data, and provides interfaces for managing uplinks and downlinks, and creating activity schedules for the lander.

The *aDaptive, ResIlient Learning-enabLed oceAn World AutonomY* (DRILLAWAY) [22] project based at the University of Illinois focused on sample site selection in unknown environments. Posing the problem as one of optimal policy selection in a Markov decision process, they produced results on both an in-house testbed and OceanWATERS that outperformed baseline approaches.

Robust, Explainable Autonomy for Scientific Icy Moon Operations (REASIMO) [23], a project led by the California Institute of Technology (Caltech), created an autonomy architecture aimed at increasing the science yield and robustness of ocean world landers by integrating existing tools from various sources as well as developing new ones. The architecture, connected via an established spacecraft framework, includes tools for planning, executing, and verifying lander actions; for state estimation and fault detection; and for machine learning based recalibration.

V. Conclusion

We have presented an overview of the OceanWATERS lander autonomy testbed and described its key enhancements over the past several years during which teams funded by NASA research grants developed and demonstrated autonomy solutions using this testbed as well as the complimentary OWLAT physical testbed at JPL. The interface unification effort between OceanWATERS and OWLAT and how it improved both testbeds has been highlighted, and the efforts of the diverse ARROW and COLDTech research teams have been summarized.

Although the OceanWATERS project and its 6 described research efforts have officially finished, there is ripe opportunity for continued use and improvement of OceanWATERS in the open-source community and in future NASA programs. Landed missions on ocean worlds and other distant bodies are an eventuality and will require both early simulation and advanced onboard autonomy. OceanWATERS is a unique testbed for the development of enabling technologies.

In most of the described research efforts, machine learning in some form is a primary technology, and indeed it dominates the world of AI today. Since machine learning thrives on training data, providing ways to vary the ocean world environment was a key driver in recent improvements to OceanWATERS. Much more improvement could be made in this area. For example, dynamic terrain generation would be very useful and was only explored and prototyped on a small scale.

Although the main objective of sophisticated onboard autonomy in distant space missions is to reduce the need for communication with Earth, *some* Earth communication will or could be needed and would be vital to the mission's success. Frameworks for modeling lander/Earth communications with associated time delay and mechanisms for time compression were designed and even prototyped for OceanWATERS in recent years, but there were insufficient resources to produce a usable implementation. This would be a worthy effort to continue.

Sophisticated autonomy can be encoded in plans that are pre-loaded on spacecraft. With their event and condition driven basis and reflective capabilities, PLEXIL plans can even implement logic that would ordinarily require replanning. Although OceanWATERS includes many exemplary autonomy plans, many more could be written, and existing ones improved, e.g., plans that model multiple, prioritized sample selection.

At the inception of the OceanWATERS project in 2018, Europa was considered the first choice in ocean worlds to further explore. The 2022 decadal survey [13] prioritizes Enceladus. OceanWATERS could be extended to model Enceladus and any planetary body in general. There are also many environment elements not modeled in OceanWATERS, such as surface temperature, radiation, tremors, and dynamic features such as plumes and soil shifting. Additional world and environment feature models would make for a richer autonomy testbed.

Finally, the version of ROS (Noetic) used by OceanWATERS will lose support from its official maintainers in 2025. Though a significant programming task, porting OceanWATERS to the newer ROS 2 would greatly extend its life.

Appendix

This section gives further details about prognostics and the application of GSAP in OceanWATERS; see Section B for background. 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. As mentioned earlier the battery pack is made up of 24 standard lithium-ion cells in a series-parallel configuration to achieve required voltage and current ratings for lander components. Details of the SOC and EOD estimation algorithms are discussed in [3].

In a model-based prognostics architecture, there are two sequential problems, (i) the estimation problem, which requires determining a joint state-parameter estimate $p(x(k); \theta(k)|y(k_0{:}k))$ based on the history of observations up to time k, $y(k_0{:}k)$,and (ii) the prediction problem, which determines at k_P, using $p(x(k); \theta(k)|y(k_0{:}k))$, a probability distribution $p(k_E(k_P)|y(k_0{:}k_P))$. The distribution for k_E can be trivially computed from $p(k_E(k_P)|y(k_0{:}k_P))$ by subtracting k_P.

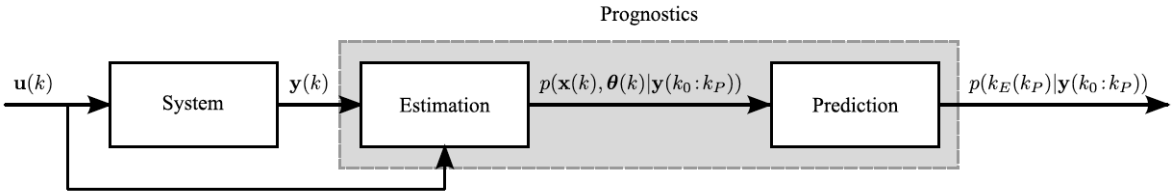


Figure 8. Schematic of prognostics architecture.

The prognostics architecture schematic in Figure 8 shows that at discrete time k, the system is provided with inputs u(k) and provides measured outputs y(k). The estimation module uses this information, along with the system model, to compute an estimate p(x(k); θ(k)|y(k_0:k)). The prediction module uses the joint state-parameter distribution and the system model, along with hypothesized future inputs, to compute the probability distribution p(k_E(k_P)|y(k_0:k_P)) at given prediction times k_P. Figure 9 abstracts GSAP's function in OceanWATERS, with "Decision Makers" denoting onboard autonomy.

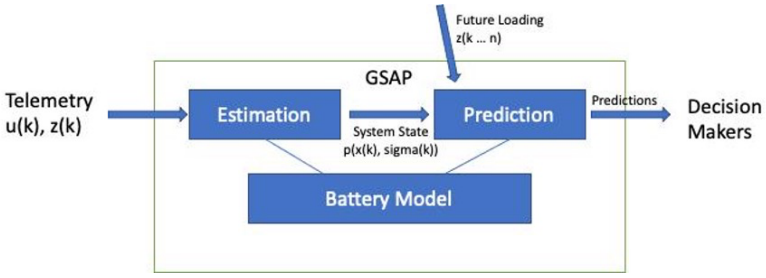


Figure 9. Schematic of the application of GSAP in OceanWATERS [21].

Acknowledgments

The authors express their gratitude to NASA's Science Mission Directorate for funding the ocean world testbeds and their associated ARROW and COLDTech projects. We appreciate the contributions from past team members who helped develop the OceanWATERS features highlighted in this paper: Milan Addepalli, Liam Carr, Damiana Catanoso, Anjan Chakrabarty, Jason Fugate, Anthony Koutroulis, Albert Kutsyy, Lanssie Ma, Ussama Naal, Keegan Nave, Molly O'Connor, Emily Schiede, Pranay Thangeda, Devin Wright, Yinan Xu, and, and Honghao Zhen. Finally, we would like to thank members of the OWLAT team at JPL (Hari Nayar, Ashish Goel, and Erica Tevere) and the ARROW and COLDTech teams for an insightful and fruitful collaboration.

References

[1] Acton, C. H., "NASA's SPICE System Models the Solar System," *International Astronomical Union Colloquium,* Vol. 165, pp. 257-262, Cambridge University Press, January 2009.

[2] Balovnev, V. I., *New Methods for Calculating Resistance to Cutting Soil*, Amerind Publishing (Translation), P. Data, translator and Rosvuzizdat, New Delhi, Available from National Technical Information Service, Springfield, VA 22161, 1983 and 1963.

[3] Daigle, M., and Kulkarni, C.S., "Electrochemistry-based battery modeling for prognostics," Annual Conference of the PHM Society. Vol. 5. No. 1. 2013.

[4] Debrunner, C., Dixon, E., Hockman, B., Fahandezhsaadi, S., Kolosa, D., Lee, T. E., ... and Steinbis, J., "Fail-Active Autonomy in Unknown-Unknown Situations for Deep Space Missions," *2024 IEEE Aerospace Conference,* pp. 1-10, IEEE, March 2024.

[5] Edwards, L. J., Wong, U. Y., Dalal, K. M., Kulkarni, C. S., Rogg, A., Tardy, A., Stucky, T. R., Umurhan, O. M., and Welsh, T. M., "An Autonomy Software Testbed Simulation for Ocean Worlds Missions," *Earth and Space 2021,* pp. 369-380, 2021.

[6] Gallo, Christopher A., Allen Wilkinson, R., Mueller, Robert P., Schuler, Jason M., Nick, and Andrew J., "Comparison of ISRU Excavation System Model Blade Force Methodology and Experimental Results," *2009 AIAA Aerospace Science Meeting,* AIAA, January 2009.

[7] Hand, K. P., Phillips, C. B., Murray, A., Garvin, J. B., Maize, E. H., Gibbs, R. G., ... and Maxwell, K. A., "Science Goals and Mission Architecture of the Europa Lander Mission Concept," *The Planetary Science Journal*, Vol. 3, No. 1, Jan. 2022.

[8] Hand, K.P., Murray, A.E., Garvin, J.B., Brinckerhoff, W.B., Christner, B.C., Edgett, K.S., Ehlmann, B.L., German, C.R., Hayes, A.G., Hoehler, T.M., Horst, S.M., Lunine, J.I., Nealson, K.H., Paranicas, C., Schmidt, B.E., Smith, D.E., Rhoden, A.R., Russell, M.J., Templeton, A.S., Willis, P.A., Yingst, R.A., Phillips, C.B., Cable, M.L., Craft, K.L., Hofmann, A.E.,

Nordheim, T.A., Pappalardo, R.P., and the Project Engineering Team, "Report of the Europa Lander Science Definition Team," Posted February, 2017.

[9] Hossen, M.A., Kharade, S., Schmerl, B., Cámara, J., O'Kane, J.M., Czaplinski, E.C., Dzurilla, K.A., Garlan, D., Jamshidi, P., "CaRE: Finding Root Causes of Configuration Issues in Highly-Configurable Robots," arXiv:2301.07690*,* 2023.

[10] Jackson, M., Bayer, T., Sheldon, C., Buffington, B., Harris, I., Jones-Wilson, L., ... and Wang, X., "Europa Clipper Mission: Onward to Implementation!" *2020 IEEE Aerospace Conference,* Big Sky, MT, USA, pp. 1-20, March 2020.

[11] McMahon, J., Ahmed, N., Lahijanian, M., Amorese, P., Deka, T., Muvvala, K., ... and Wakayama, S., "REASON-RECOURSE Software for Science Operations of Autonomous Robotic Landers," *2023 IEEE Aerospace Conference,* pp. 1-11, IEEE, March 2023.

[12] Mercer, C. R., Stephan, R. A., Voytek, M. A., Niebur, C., Paganini, L., Schulte, M. D., Nguyen, Q., Dalal, K. M., and Nayar, H. D., "NASA Science Technology Development Programs for Ocean Worlds Exploration," *Earth and Space 2022,* pp. 308-320, 2022.

[13] National Academies of Sciences, Engineering, and Medicine, "Origins, Worlds, and Life: A Decadal Strategy for Planetary Science and Astrobiology 2023-2032," *The National Academies Press,* 2023.

[14] Nayar, H., Goel, A., Boettcher, A., Hans, M., Sawoniewicz, J., Gaut, A., ... and Thomsen, M., "Development of a Lander Autonomy Testbed for Ocean Worlds Missions," *Earth and Space 2021,* pp. 531-540, 2021.

[15] Newill-Smith, D., Shatts, J., Dillon, R. P., Karras, J., Brinkman, A., Backus, S., ... and Fleischner, R., "Cold Operable Lunar Deployable Arm (COLDArm) System Development and Test, "2023 *IEEE Aerospace Conference*, pp. 1-19, March 2023.

[16] Oren, Michael., & Nayar, S. K. "Generalization of Lambert's Reflectance Model," *1994 Conference on Computer Graphics and Interactive Technique (SIGGRAPH),* pp. 239-280, July 1994.

[17] Patthoff, D. A., and Kattenhorn, S. A., "A Fracture History on Enceladus Provides Evidence for a Global Ocean," *Geophysical Research Letters*, Vol. 38, No. 18, 2011.

[18] Rocha, C., Cadavid, H., Munoz, C., & Siminiceanu, R., "A Formal Interactive Verification Environment for the Plan Execution Interchange Language," *Integrated Formal Methods: 9th International Conference,* IFM 2012, Springer Berlin Heidelberg, proceedings 9, pp. 343-357, Pisa, Italy, June 18-21, 2012.

[19] Sotin, C., Kalousová, K., and Tobie, G., "Titan's Interior Structure and Dynamics after the Cassini-Huygens Mission," *Annual Review of Earth and Planetary Sciences*, Vol. 49, No. 1, pp. 579-607, 2021.

[20] Spohn, T., and Schubert, G., "Oceans in the Icy Galilean Satellites of Jupiter?" Vol. 161, No. 2, pp. 456-467, 2003.

[21] Teubert, C., Daigle, M. J., Sankararaman, S., Goebel, K., and Watkins, J., "A Generic Software Architecture for Prognostics (GSAP)," *International Journal of Prognostics and Health Management,* Vol. 8, No. 2, 25 Sept. 2017.

[22] Thangeda, P., and Ornik, M., "Adaptive Sampling Site Selection for Robotic Exploration in Unknown Environments," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 4120-4125, IEEE, October 2022.

[23] Touma, T., Daş, E., Burdick, J. W., Clark, E. B., Mackey, R., Feather, M. S., ... and Prather, M., "Towards Robust, Resilient Ocean World Science Sampling Systems," *2023 IEEE Aerospace Conference,* pp. 1-9, IEEE, March 2023.

[24] Verma, V., Estlin, T., Jónsson, A., Pasareanu, C., Simmons, R., & Tso, K., "Plan Execution Interchange Language (PLEXIL) for executable plans and command sequences," *International Symposium on Artificial Intelligence,Robotics and Automation in Space (iSAIRAS)*. September 2005.

[25] Wong, U., Umurhan, O., Tardy, A., Welsh, T., Allan, M., and Edwards, L., "FROST Dataset: Features Relevant to Ocean Worlds Surface Terrain," NASA Ames Research Center, Dec. 2019.

GOVERNMENT RIGHTS NOTICE

This work was authored by employees of ***KBR Wyle Services, LLC*** *under Contract No.* ***80ARC020D0010*** *with the National Aeronautics and Space Administration. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for United States Government purposes. All other rights are reserved by the copyright owner.*