OceanWATERS Lander Robotic Arm Operation

Damiana Catanoso
USRA
NASA Ames Research Center
Moffett Field, CA 94035
damiana.catanoso@nasa.gov

Anjan Chakrabarty
KBR, Inc
NASA Ames Research Center
Moffett Field, CA 94045
anjan.chakrabarty@nasa.gov

Jason Fugate
KBR, Inc
NASA Ames Research Center
Moffett Field, CA 94035
jason.fugate@nasa.gov

Ussama Naal
KBR, Inc
NASA Ames Research Center
Moffett Field, CA 94045
ussama.naal@nasa.gov

Terence M. Welsh
Logyx LLC
NASA Ames Research Center
Moffett Field, CA 94045
terence.m.welsh@nasa.gov

Laurence J. Edwards
NASA Ames Research Center
Moffett Field, CA 94045
laurence.j.edwards@nasa.gov

Abstract*—Ocean Worlds Autonomy Testbed for Exploration Research and Simulation (OceanWATERS) is an open-source simulator for developing onboard autonomy software for robotic exploration of ocean worlds, such as Europa, Enceladus, and Titan, built on the Robot Operating System (ROS) and Gazebo simulation environment. Inevitable ground communication delays increase demand for a high degree of autonomy during excavation, collection and transfer of samples to scientific instruments for in-situ analysis. This paper offers a detailed discussion of the robotic arm design and operation for such autonomous surface exploration, taking as reference the Europa Lander mission. The lander arm, which is designed primarily to acquire icy surface and subsurface samples within the arm's workspace, is a 6-degree-of-freedom manipulator with two end effectors: a sample excavation tool and a trenching end-effector. The robotic arm's modes and operations can be summarized as follows: stowed arm, intended as the lander arm default configuration characterized by zero-power consumption; un-stowed arm, target arm configuration after its first deployment; selection and deployment of the end-effector to use next; guarded move, to detect ground level at the desired trenching location; drill ice using the grinder; dig trench at a particular location using the scoop; deliver sample to the sample transfer dock; discard redundant samples. The motion planning tool used for the lander arm is MoveIt, a ROS package. MoveIt uses sampling-based planning and collision checking libraries to determine safe paths. The Rapidly Exploring Random Trees* (RRT*) has been chosen as default planning algorithm as it provides optimal plans with an exponential speed and is guaranteed to find a solution, if feasible solutions exist. Furthermore, this work quantifies and discusses the energy requirements for excavating and collecting samples. In OceanWATERS, force feedback from the terrain, which influences the arm dynamics, is modelled using a discrete element method (DEM) simulation. The DEM and Gazebo software run in parallel and communicate through a co-simulation plugin. This paper presents an analysis and comparison of three DEM open source software (YADE, ESyS-Particle, Project Chrono) for implementation in OceanWATERS and motivates the choice of YADE as most suitable candidate.

TABLE OF CONTENTS

U.S. Government work not protected by U.S. copyright

1. INTRODUCTION

OceanWATERS - Ocean Worlds Autonomy Testbed for Exploration, Research and Simulation - is a simulation testbed aimed to spur the development of autonomy software for future robotic missions to planetary bodies in the solar system characterized by an icy surface. The software has been developed at NASA Ames Research Center (ARC), released as open source and available on GitHub [1]. Missions to "ocean worlds", such as Europa, Enceladus and Titan, will require a high level of autonomy in performing missions tasks because of their distance from Earth and hostile environmental conditions. Given that one-way communication delays between our planet and Europa range between 35 and 52 minutes, ground controlled operations would require a long mission lifetime. On the other hand, constant exposure of electronic components to the high radiation levels characterizing Jupiter's atmosphere sensibly reduces available mission lifespan. Thus, robotic operations to icy worlds require a higher degree of autonomy, as well as suitable testbeds for development and testing, in comparison, for instance, to missions to Mars or other neighboring planets. OceanWATERS inherits NASA ARC's significant experience in the field of simulation techniques and visualization for interplanetary missions technology development and operations. Notable projects developed at ARC include, but are not limited to, the Mission Simulation Facility [2], the "Viz" science operations planning software [3], developed for the Mars Exploration Rover mission [4], and a high fidelity "conops" (concept of operations) simulator developed [5] for the proposed Resource Prospector (RP) mission [6].

OceanWATERS offers a virtual platform where software robustness, as well as faults handling capabilities, can be developed and tested. The OceanWATERS simulator is built on the Robot Operating System (ROS) [7] and the Gazebo simulation environment. The simulation takes as a reference the Europa Lander mission [8], including a model of the lander to execute operations dictated by the autonomy module. The current OceanWATERS' Gazebo world includes a digital elevation model of the surface of Europa, built starting from point cloud measurements acquired during field trips to the Atacama desert in Chile and the Death Valley desert in the United States [9]. These locations have been chosen because they accurately represent the salt efflorescence

and penitentes structures observed on Europa. Accurate source of illumination is provided by sunlight and Juipiter's albedo, whose relative location is provided by the SPICE ephemerides module developed by NASA JPL's Navigation and Ancillary Information Facility (NAIF) [10]. Terrain-lander interaction during sample collection is accurately simulated using discrete element method (DEM) [11]. Current implementation of the interaction includes a ROS plugin that reads a lookup table containing force feedback from the terrain to the scoop whenever contact is detected. In [11], Catanoso et. al. presented an analysis of the terrain-lander interaction using discrete element method. The analysis includes DEM parameter tuning to achieve a compromise between accuracy and computation time, simulation data validation using a physical testbed, and a thorough description of the sample collection strategy adopted in OceanWATERS. The team is now working to include a co-simulation plugin to run the ROS-Gazebo and DEM simulations in real time. Figure 1 shows a snapshot of the OceanWATERS simulation. Figure 2 shows the Europa Lander simplified model, which consists of the main body, an antenna, landing stabilizers, anchored to the ground, and the robotic arm. The lander arm is designed primarily to acquire icy surface and subsurface samples within the arm's workspace. The 6-degrees-of-freedom robotic arm concept from the Europa Lander Report [8] has been modified and included in OceanWATERS. The shoulder location on the lander body has been changed and one more link added to increase the number of DOF to 6. The arm has multiple terrain excavation tools as end effectors. Currently, two available end effectors have been incorporated in the simulation: the sample excavation tool and the trenching tool. The sample excavation tool is modeled as a conventional grinder, which is capable of excavating icy materials at depths greater than 10 cm. The grinder included in OceanWATERS does not coincide with the actual ice-breaking tool that will be used in the Europa Lander mission. It serves as a placeholder for any tool that is able to break ice into fragments and create a particle bed that will be collected by the scoop. The trenching tool, modeled as the Phoenix Mars Lander [12] scoop, is responsible for aggregating samples excavated from the surface at the target depth, collecting samples and transferring them to the sample transfer dock. The arm can be programmed to move in any particular orientation and direction to excavate and acquire samples. The main arm operations, designed for the initial software release, are: stowed arm, intended as the lander arm default configuration characterized by minimal power consumption; un-stowed arm, the target arm configuration after its first deployment; guarded move, to detect ground level at the desired trenching location, moving along a given direction; grind, to drill ice using the grinder; dig circular, to dig a trench at a particular location using the scoop, following a circular trajectory; dig linear, to dig trench at a particular location using the scoop, following a linear trajectory; deliver sample to the sample transfer dock; discard sample. Each arm motion has to be first planned and then executed. The motion planning tool used in OceanWATERS is MoveIt [13], a ROS based tool. MoveIt uses sampling-based planning and collision checking libraries to determine safe paths. Among MoveIt's multitude of planning algorithms, the Rapidly Exploring Random Trees* (RRT*) has been chosen as default planning algorithm as it provides optimal plans with an exponential speed. Furthermore, RRT* is a complete planner, i.e. it is guaranteed that it finds a solution, if feasible solutions exist [14]. In OceanWATERS, the chosen optimization objective is the mechanical work, which directly affects the power consumption.

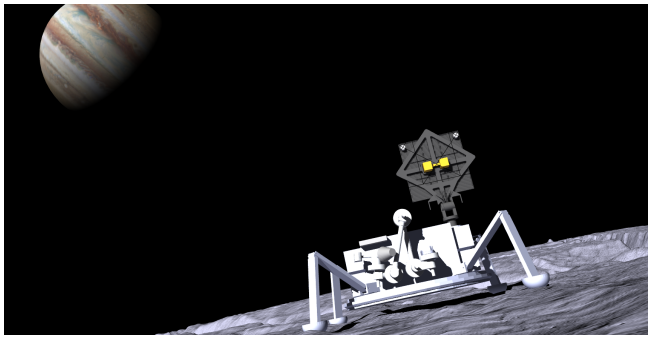

Figure 1. OceanWATERS simulation in the Gazebo virtual environment. The lander is located on the Europa terrain with Jupiter in the background.

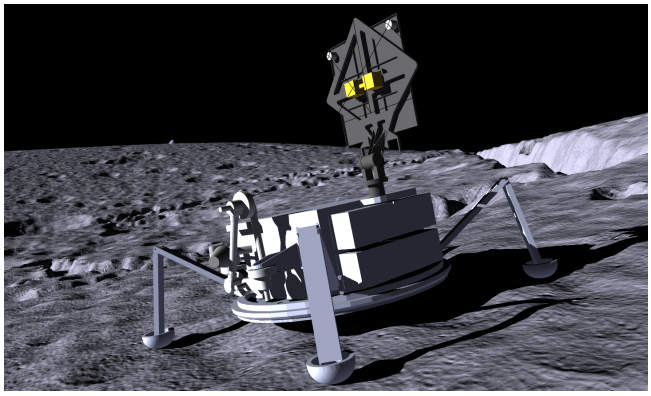

Figure 2. Europa Lander model in OceanWATERS with stowed arm.

In this paper, the work relative to the OceanWATERS' robotic arm is presented. The paper includes accurate description of: physical and mechanical properties of the system and the arm (Section 2), planning of the different modes of operation using ROS services and MoveIt (Section 3), energy requirements calculation for excavating, collecting and delivering samples (Section 4), a comparison between multiple open source DEM software candidates for co-simulation implementation in OceanWATERS (Section 5), conclusions and future work (Section 6).

2. System Description

As shown in Figure 2, OceanWATERS' Europa Lander model consists of a simplified body, four landing stabilizers (legs), one two-degree-of-freedom antenna, a stereo camera, and one 6-degree-of-freedom robotic arm supporting two end effectors. A proportional-derivative-integral (PID) position controller commands the motion of the arm's and antenna's revolute joints. The PID controller parameters have been tuned with the aid of ROS inbuilt plugins, accessible through the rqt graphical user interface. For each of the joints, the tuning method consists of publishing a known trajectory using the "Message Publisher" plugin, observe the extent to which the joint, under the PID control action, follows the given trajectory. This can be done using the rqt plotting tools. To achieve the desired system behavior, the PID constants can be dynamically changed using the "Dynamic Reconfigure" plugin, until the output trajectory overlaps the desired one. Figure 3 shows the rqt window configuration in the specific case of tuning the shoulder yaw joint. Here, the Message

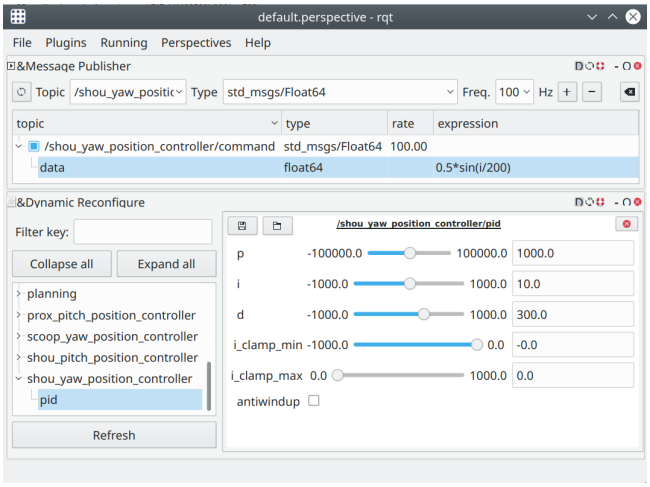

Figure 3. Rqt interface for tuning the shoulder yaw joint PID controller parameters. The interface includes the Message Publisher and Dynamic Reconfigure plugins.

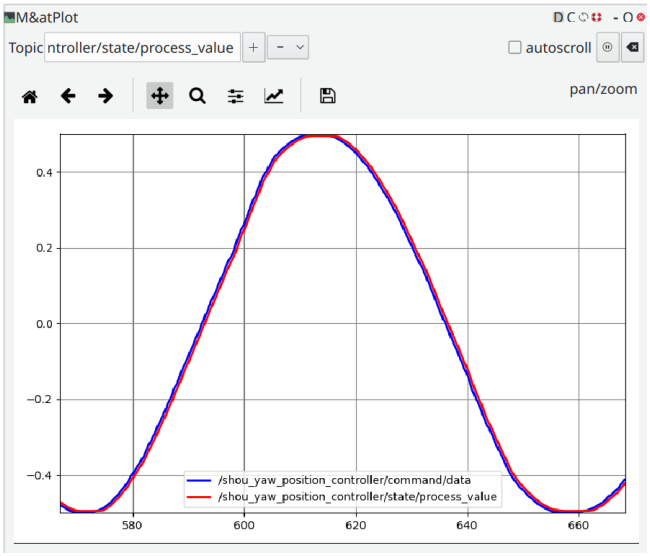

Figure 4. Shoulder yaw joint angle plots when commanding a sinusoidal trajectory: in blue the desired value and in red the actual system response.

Publisher is publishing a sinusoidal joint trajectory with a publishing frequency of 100 Hz. The current PID parameters for the shoulder yaw joint appear in the Dynamic Reconfigure section of the window. Real time plots of the joint state are visualized in Figure 4: the blue line represents the value that the joint angle, in radians, should have when the sinusoidal trajectory is published, while the red line represents the actual joint state. The following subsections provide more details on the mechanical properties and configuration of the antenna and arm subsystems.

Antenna

The lander is designed to communicate with Earth through a relay orbiting the ocean world (Europa, Enceladus or Titan). The lander is equipped with a high gain antenna (HGA) for communication. Since the relay will be orbiting the planetary body, the HGA is designed to provide full sky coverage. The HGA is mounted on a pan/tilt head. It is possible to point the antenna at a particular direction in the sky using the pan and tilt position controller.

Robotic Arm

The robotic arm includes a series of five links and two end effectors. The arm is attached to the lander body through the shoulder link. The shoulder is followed, in order, by the proximal, distal, wrist and hand links. Figure 5 shows the links in detail together with the sample transfer dock, where the sample is delivered. Figure 6 shows the frames associated to each link, where the Red-Green-Blue axes colors represent respectively the X-Y-Z axes. Moreover, in Figure 6, the so called "base_link" appears at the center of the lander body. The base_link is the main frame of reference for all the coordinates used in the motion planning and execution algorithms, and has the X axis parallel to the sides of the lander body, pointing towards the arm, the Z axis parallel to the gravity vector, pointing upwards, and the Y axis resulting from the cross product between Z and X, according to the right hand rule. Revolute joints allow relative rotational motion of one link with respect to another. Table 1 provides each arm's joint name and the two links it connects.

3. Modes Of Operation

The OceanWATERS team developed a digging procedure for the autonomous mission on Europa. After landing and deploying on Europa's surface, the lander will screen the surrounding terrain with the stereo camera, and identify/select the most scientifically significant terrain location within the arm workspace. After deploying the arm, the lander will learn the relative terrain altitude with respect to the lander at the selected location by performing a guarded move towards the terrain. Terrain layers adjacent to the surface are less significant under a scientific point of view. This is because Europa flies through Jupiter's magnetosphere, which is characterized by high radiation levels and this strong radiation might have destroyed potential biological signs on the surface. For this reason, the lander will use the grinder to create a 10 cm deep trench of loose ice particles, collect the contents of the resulting trench and discard it, dumping it in a remote part of the workspace. A second grinding action will be required to fragment the ice at the bottom of the empty trench. Once more loose material is available, the lander will use the scoop to collect the sample and deliver it to the sample transfer box. Upon completion of the sample delivery, the arm will return to the initial stowed configuration. OceanWATERS implements the arm motion planning and execution through ROS services. While the motion planning for each of the operation modes is implemented through a dedicated service, executing any planned trajectory is carried out by a single service. To date, the available motion planning ROS services are listed and described in Table 2. Services concerned with grinding and digging perform the respective end effector selection before planning for the motion. The motion planning problem is set by writing Python scripts containing a series of target states in Cartesian or joint space that yield the successful execution of the given motion. MoveIt generates a series of consecutive plans, including a set of joint positions that are later interpolated to provide the desired trajectory that the arm has to follow. The interpolation is conducted by MoveIt's "fake controller". Three fake controllers are available: last point, via points and interpolate. OceanWATERS adopts the interpolate fake controller to guarantee smooth motion between the target states, with an interpolating frequency of 2.5 Hz. While the fake controller interpolates the plan points, a real-time visualization of the desired trajectory is

Table 1. Robotic arm joints and connected links.

Name	First link	Second link
Shoulder yaw	Lander body	Shoulder
Shoulder pitch	Shoulder	Proximal
Proximal pitch	Proximal	Distal
Distal pitch	Distal	Wrist
Hand yaw	Wrist	Hand
Scoop yaw	Hand	Scoop
Grinder yaw	Hand	Grinder

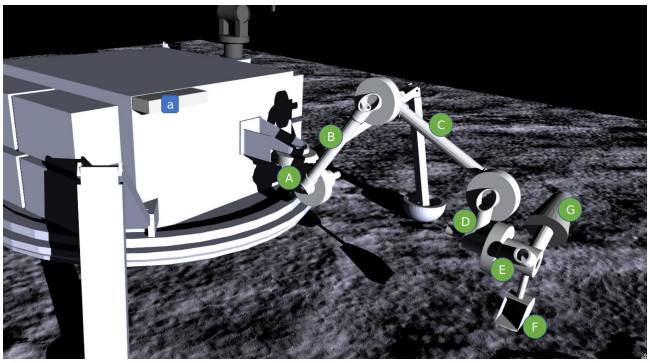

Figure 5. Links: Shoulder (A), Proximal (B), Distal (C), Wrist (D), Hand (E), Scoop (F), Grinder (G), Sample transfer dock (a).

visualized in rviz, a 3D interactive visualization tool available in ROS. Figure 7 shows the lander in the rviz window: the solid arm represents the current arm configuration, while the "ghost" arm moves according to the calculated plans. The final product of the planning phase is a joint space trajectory, which is written and stored in a .csv file. The publish_trajectory service is responsible of reading and publishing each row of the .csv file, with a given rate, on an apposite ROS topic. Publish_trajectory is called whenever a planned arm trajectory needs to be executed. The arm controllers in Gazebo read the target states in real time and power the arm. As a result, the lander will move the arm following the planned trajectory in Gazebo. Figure 8 shows the lander performing a scooping motion in Gazebo. The planning services seen in Table 2, are sufficient to perform all the tasks required by a 21-sol autonomous missions. Each of these services, together with the service's own message field, is introduced individually in the rest of this section. ROS services operate by means of a customized ROS message, whose fields can be specified accordingly to the needs. The message can carry information on the target location and direction of the commanded movement. Message field values are provided upon service call by the user, if the service is called individually, or by autonomy, if the service is part of a autonomous plan. A list of the message fields and types belonging to each planning service is available in Table 3.

Un-stow

The unstow service is meant to be called at the beginning and at the end of a sample collection session. It brings the arm from the stowed configuration to a "safe" arm configuration before executing any other motions. It is also the desired configuration that the arm should have before calling the stow service, i. e. before folding the arm back to the

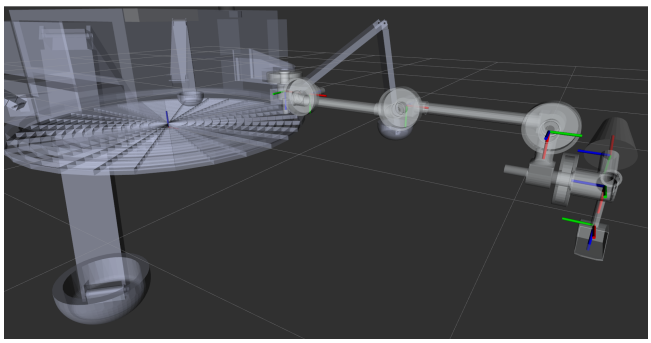

Figure 6. Rviz view of Base_link, located at the center of the lander body, and the robotic arm link frames.

Table 2. Arm motion planning ROS services.

Name	Planned movement
Unstow	Unstow arm
Guarded_move	Perform a guarded move
Grind	Grind solid ice to create a trench
Dig_circular	Collect sample with circular motion
Dig_linear	Collect sample with linear motion
Deliver_sample	Sample delivery and discard
Stow	Stow arm

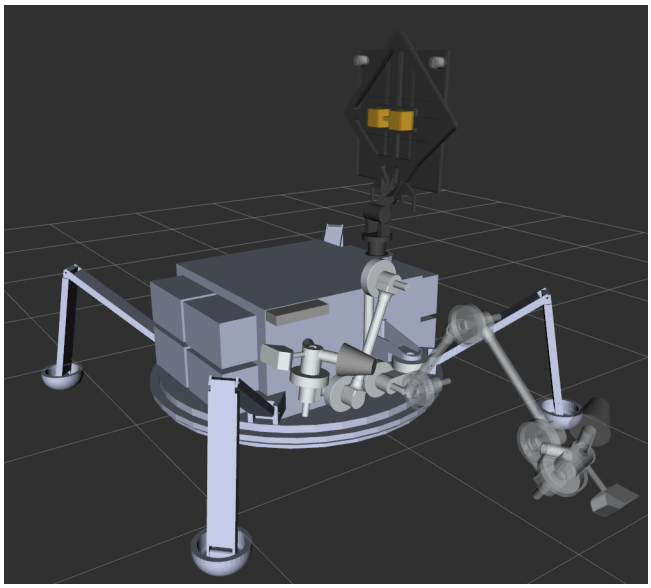

Figure 7. Rviz visualization of the lander arm: the solid arm shows the current configuration, the transparent (ghost) arm moves according to the planned trajectory.

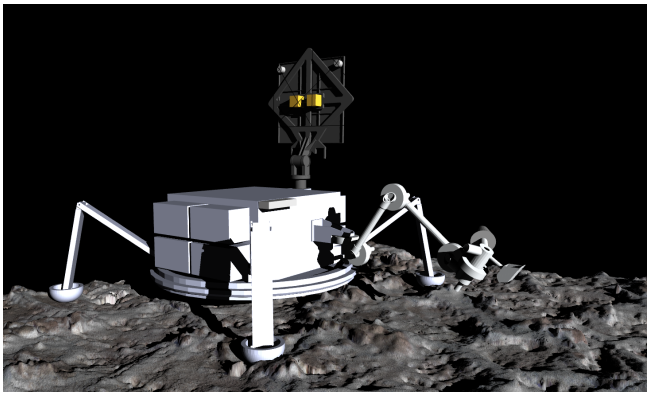

Figure 8. Arm configuration while executing the planned sample collection motion in Gazebo.

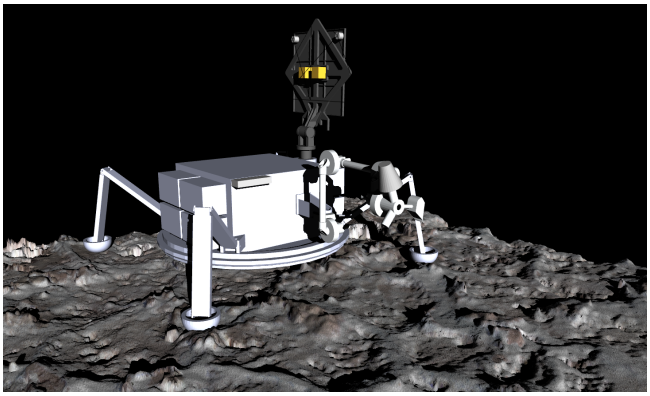

Figure 9. Lander arm in the un-stowed configuration.

stowed position. This service is introduced as a matter of precaution, to avoid the demand for reconfigurations that are too complicated and could cause collisions. Figure 9 shows the "safe" arm configuration, that the arm will have after calling unstow service. Even though it is meant to be used at the beginning and end of the sample collection, the service can be called at any moment during the mission and will have the same result of taking the arm to the configuration shown in Figure 9. Like stow, the target arm configuration is not a user-input parameter and does not change. So, also in this case, the unstow message field is only del_prev_traj as shown in Table 3.

Guarded move

The objective of the guarded move command is to detect the location of the ground or any kind of obstruction with respect to the lander at a predetermined location within the workspace. After selecting the scoop as end effector to perform the maneuver, the scoop is brought to a predetermined location in space, which is the maneuver starting point. Once reached the starting point, the scoop starts moving slowly, approaching the terrain along a given direction until it finds the ground or an obstacle. After the ground is found, a ROS message containing the X-Y-Z coordinates of the detected obstacle/ground is published on a dedicated topic. The service's message fields are reported in Table 3, and consist of the X-Y-Z coordinates of the starting point of the maneuver, the X-Y-Z coordinates of the unit vector along which the scoop approaches the obstacle, and the desired distance to cover along the given direction. The Z component of the scoop's velocity vector during the movement is tracked and smoothed

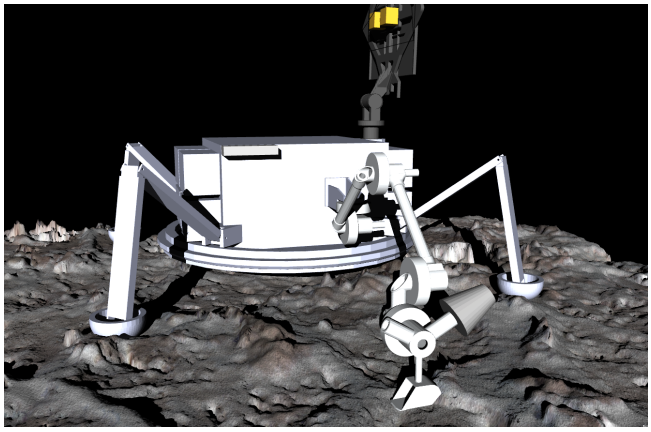

Figure 10. Lander arm at the end of guarded move execution.

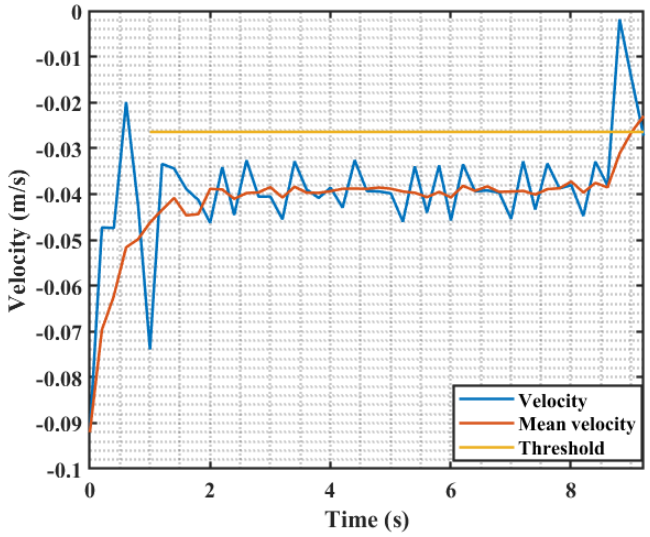

Figure 11. Ground detection process. Ground is detected when the mean velocity (orange line) intersects the threshold (yellow line).

with a moving average over a window of five samples. The smoothing is needed to subdue the fluctuation in velocity induced by the PID controller. A threshold is calculated as the sum of the mean and standard deviation of the first five velocity samples. A contact with the ground is posted when the effective velocity exceeds the threshold, i. e. when it exceeds the mean of the first five samples by a margin of one standard deviation. When this happens, the arm movement is suspended and the location of the end effector (scoop) is reported with respect to the base link using ROS TF2 package [15]. Figure 11 shows how the ground is detected when the mean velocity (orange line) intersects the threshold (yellow line), which is calculated starting from the first five samples.

Grind

During mission lifetime, the lander is likely to encounter hard ice surface. In this case, it will have to grind the ice and collect the resulting particles. The grind service plans the arm motion to create a trench of a defined length and depth, starting from a defined location, following a specified direction. Figure 12 shows a snapshot of the arm finishing grinding a trench parallel to the arm. ROS message fields

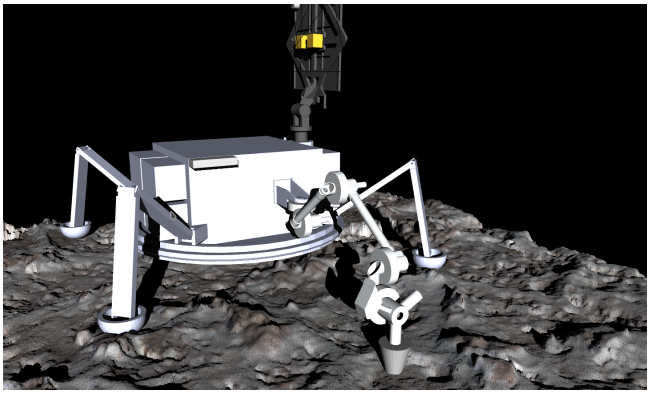

Figure 12. Lander arm while grinding terrain to create a trench.

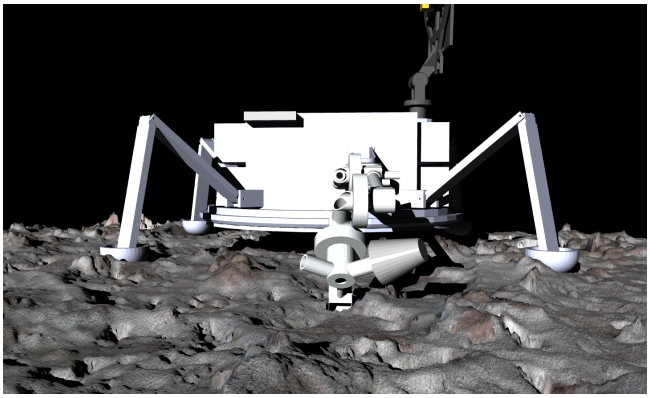

Figure 13. Sample collection through circular scoop motion.

in this case are several and listed in Table 3. The trench starting point is specified by the x and y coordinates. The boolean variable parallel gives the direction, which can be parallel to the arm, if parallel=True, and perpendicular to the arm, if parallel=False. The grinding motion starts at the starting point and, if the desired direction is parallel, it moves outwards. If the direction is perpendicular, it moves to the lander's right. Since the lander works in the base_link coordinate frame, z=0 does not correspond to the terrain height. Thus, the terrain height has to be provided to the service as the terrain z coordinate in base_link. The lander learns the terrain height at a specific location by means of another service described later in this section: guarded_move. When use_defaults is set to True, the service ignores all the values assigned to the other parameters and assigns a set of hard coded default values.

Dig circular

After grinding the terrain, the lander collects a sample using the scoop end effector and decides whether the sample has to be discarded or not. The lander collects the sample following two possible strategies: a circular scoop trajectory or a rasping-like motion. The dig_circular service implements the first trajectory. Here, the scoop reaches the x and y coordinates of the starting point and rotates around the hand joint, if the desired direction is perpendicular to the arm, or the wrist joint if parallel=True. Table 3 shows dig_circular's message fields, which have the same meaning of the grind service. Figure 13 shows a snapshot of the circular digging executed in Gazebo.

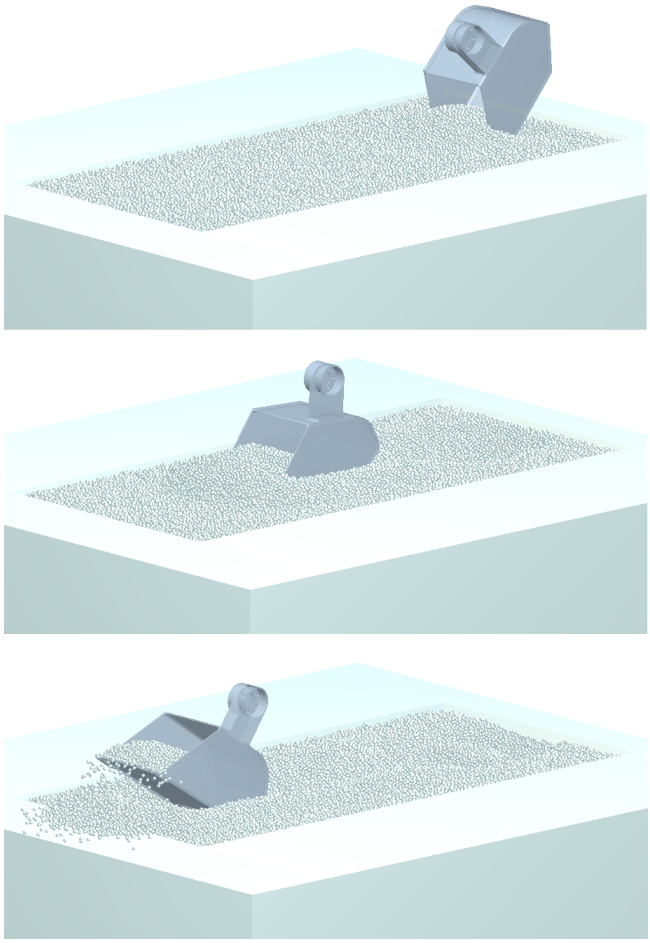

Figure 14. Capture of the linear sample collection strategy [14]. Top image shows the scoop entering the particle bed. Middle image shows the scoop traversing parallel to the surface. Bottom image shows the scoop exiting the granular material.

Dig linear

Dig_linear is the second digging service that implements a linear scoop trajectory that wants to resemble a rasping motion. In [11], Catanoso et. al. accurately describe the scoop motion, which is summarized in Figure 14. Table 3 shows the service message parameters, which have same meaning described earlier. The length parameter refers only to the length of the linear path in the terrain. There is no parallel variable here, since only the radial direction has been implemented until now. Figure 15 shows the scoop while it executes the linear part of the trajectory.

Deliver sample

Once the scoop fills up with material, the sample is delivered or discarded. Discarding a sample means collecting it and dumping it in a remote part of the workspace, chosen because not scientifically interesting. Coordinates of the dumping point are provided to the service through the parameters x, y and z in Table 3. Figure 16 shows such scenario. When the sample is considered meaningful, under a scientific point of view, the sample is delivered to the sample transfer dock, as shown in Figure 17. In this case, the user has to call the service using use_defaults=True.

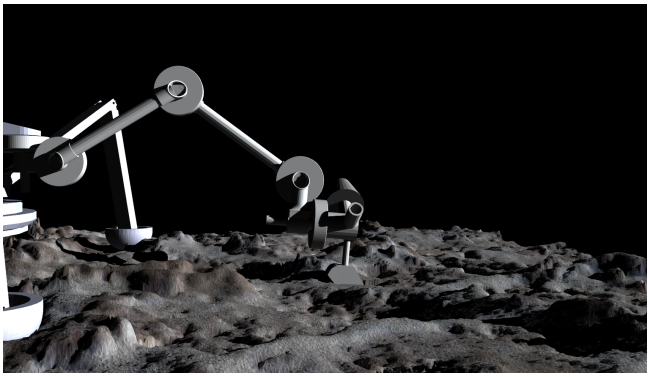

Figure 15. **Capture of the lander arm while collecting a sample executing a linear trajectory.**

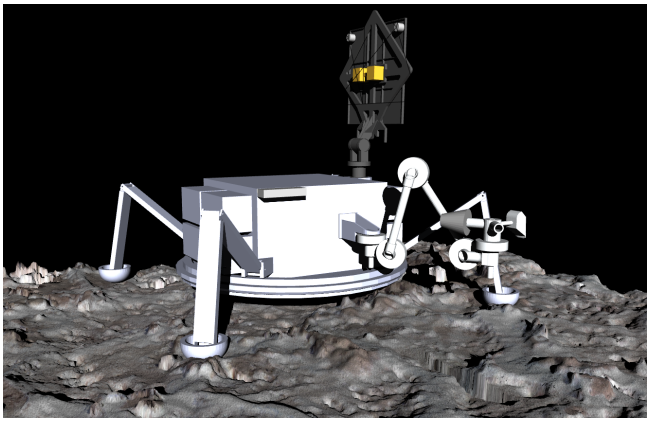

Figure 16. **Sample discard to remote workspace location.**

Stow

The stow service plans for taking the arm to a folded configuration starting from any other arm configuration. The stowed arm position has already been seen in Figures 1 and. The stowed configuration represents a minimum energy consumption configuration, so in non-operative phases of the mission it is desirable to keep the arm folded. Since the service's target arm configuration does not change, the stow service does not require any extra information from the user. The boolean variable delete_prev_traj is common to all the services. When set to True, it deletes all the .csv trajectory files present in the folder.

4. POWER CONSUMPTION

Energy consumption is one of the critical factors for mission design. Energy consumption will decide on-board energy storage necessities and will determine the total life for the mission. Hence it is important to accurately model the power requirements and energy consumption for each of the pre-determined modes of operation of the lander arm. It is assumed that the torque and angular velocity of each joint remains constant during the small time interval(k) of receiving messages. The instantaneous power consumed, at the k-th instant, by the i-th joint can be calculated as

$$P(k)_i = \tau_i(k)\dot{\theta}_i(k), \tag{1}$$

where τ_i is the torque generated by the i-th joint motor and $\dot{\theta}_i$ is the i-th joint's angular velocity. The total instantaneous

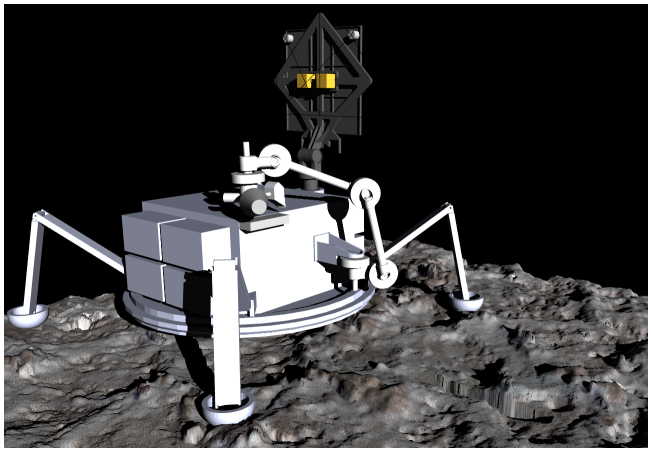

Figure 17. **Sample delivery to sample transfer box.**

power for all the joints can be calculated as

$$P(k) = \sum_{i=1}^{n} P(k)_i + P_0 \tag{2}$$

where n is the total number of joints and P_0 is the auxiliary power required (i.e. internal losses and power required to run ancillary equipment). The total energy consumption can be calculated as

$$E = \int_{t_0}^{t_f} P(t)dt, \tag{3}$$

which, when discretized in time, becomes

$$E = \sum_{t_0}^{t_f} P(k)dt_k. \tag{4}$$

Here dt_k is the length of the k-th time interval. Energy consumption and power requirements will be used for determining mission longevity and peak power requirements from the battery. Figure 18 shows the instantaneous mechanical power drawn for the guarded move operation. As it can be seen from the figure, the lander power increases dramatically while interacting with the ground. One of the tasks of the autonomy module is to tackle such high power draining maneuvers.

5. ARM-TERRAIN INTERACTION

An integral portion of the OceanWATERS testbed is the ability to model interactions between an end effector and the terrain. An accurate force feedback from the terrain on the scoop is required by fault-detection and autonomous decision-making algorithms to identify when the requested torque on the robotic arm's joints exceeds the maximum available torque. Knowledge of the terrain force feedback helps significantly in evaluating the structural properties of the arm's links and in properly selecting actuators for the joints [11]. An integrated discrete element method (DEM) modeling approach is an effective way to develop an understanding of these interactions, since DEM evaluates elastic forces, cohesion, and friction at the particle level while allowing for customized environmental properties and particle shapes. There are a variety of accessible, open-source platforms that embrace a DEM approach for simulating interactions between solid bodies and granular material; however,

Table 3. Message fields and types for each available planning service.

Service name	Message fields	
	BOOL	FLOAT32
Unstow	delete_prev_traj	
Guarded_move	delete_prev_traj, use_defaults	x, y, z, direction_x, direction_y, direction_z, search_distance
Grind	delete_prev_traj, use_defaults, parallel	x, y, depth, length, ground_position
Dig_circular	delete_prev_traj, use_defaults, parallel	x, y, depth, ground_position
Dig_linear	delete_prev_traj, use_defaults	x, y, depth, length, ground_position
Deliver_sample	delete_prev_traj, use_defaults	x, y, z
Stow	delete_prev_traj	

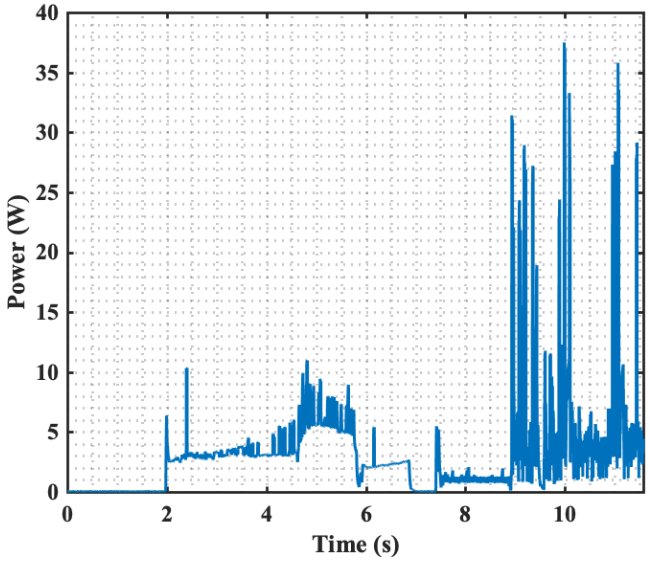

Figure 18. Power consumption while the arm performs a guarded move to find ground location.

due to the many unknowns of terrain characterization of ocean worlds, the ideal tool should be capable of various terrain specification features in addition to being easily integrable and accurate. Users may also have to interact with the software to some degree in order to adapt the model to a specific use case outside of what is specifically presented by OceanWATERS. Of the many viable options, YADE [16], ESyS-Particle [17], and Project Chrono [18] were considered for integration with OceanWATERS. A case study was performed in order to quantify and compare the performance and capabilities of each open source DEM platform. Weights were applied to each comparison category to compile a score between zero and one for each of the three options based on the requirements for users of the OceanWATERS testbed. In addition to identifying the standard capabilities of each software, numerical comparisons were made to both experimental testbed data and the commercial DEM solver, EDEM [19], for a representative simulation. A thorough description of the testbed is provided in [11]. The representative simulation consists of having the scoop digging in terrain, following the linear trajectory described in [11]. The chosen bulk material is sand, represented by spherical particles with a Ø 3.5 mm, and the depth reached by the scoop during the linear phase of the motion is 3 mm. Characteristics of the scoop motion are summarized in Table 4. Figure 19 shows plots of the Z component of the force acting on the scoop center of mass,

Table 4. Scoop displacements and velocities during each phase of the linear trajectory.

Event	Speed	Total Angle/Distance
First Rotation	30 deg/s	90 deg
Linear Translation	0.1 m/s	0.3 m
Second Rotation	30 deg/s	90 deg

Table 5. DEM simulation parameters for sand bulk material.

Parameter	Value
Poisson's Ratio	0.25
Shear Modulus (Pa)	1e+7
Rolling Friction	0.05
Static Friction	0.35
Coefficient of Restitution	0.88
Solid Density (g/cm^3)	1.6
Terrain Model	Elastic, non-cohesive

in the scoop link frame seen in 6, resulting from running the same reference simulation previously described in the open source software YADE, ESyS-particle, and Project Chrono, as well as in the EDEM commercial software. Sand was chosen as the bulk material due to its ease of accessibility for experimental implementation when compared to terrain more likely to be found on an ocean world such as snow or ice. The normalized root-mean-square deviation (NRMSD) was identified as the ideal parameter for comparing the force feedback data for each simulation as it allows for an overall comparison throughout the pass. Experimental test data are used as a baseline in order to validate the approximate magnitude and trend of the force feedback data to be expected when using DEM solvers; however, heavier weights were applied to the EDEM force comparisons due to the physical limitations of the experimental testbed itself. The resulting normalized error statistics and other performance characteristics that were considered in evaluating the open-source platforms can be seen in Table 6. Ultimately, YADE was identified as the ideal solver by a narrow margin and selected for implementation in OceanWATERS.

Table 6. Parameters, weights and total score for comparison, evaluation and selection of most suitable open source DEM software for integration in OceanWATERS.

Parameter name	EDEM	YADE	ESyS-Particle	Chrono	Weight
Scripting wrappers	1	1	1	0	0.65
C++ API	0	1	1	1	0
Limit in number of particles	0	0	0	0	0
Particle bonding	1	1	1	0	1
Polyhedral particle shape	0	1	0	1	0.5
Multi-sphere particle	1	1	1	0	0.8
Parallel computations	1	1	1	1	0
Super-computer suitable	0	0	1	1	0.4
Clear documentation	1	1	0.5	1	0.6
Active community	1	0.8	0.6	0.7	0.75
GPU capable	1	1	0	0.5	0.5
Fx normalized Score Compared to EDEM	1	0	1	0.56	0.9
Fy normalized Score Compared to EDEM	1	0.36	0	1	0.2
Fz normalized Score Compared to EDEM	1	0	0.68	1	0.7
Fx normalized Score Compared to Testbed	0.68	1	0	0.23	0.45
Fy normalized Score Compared to Testbed	1	0	0.25	0.92	0.05
Fz normalized Score Compared to Testbed	0	0.42	1	0.77	0.15
Total	0.84	0.68	0.67	052	N/A

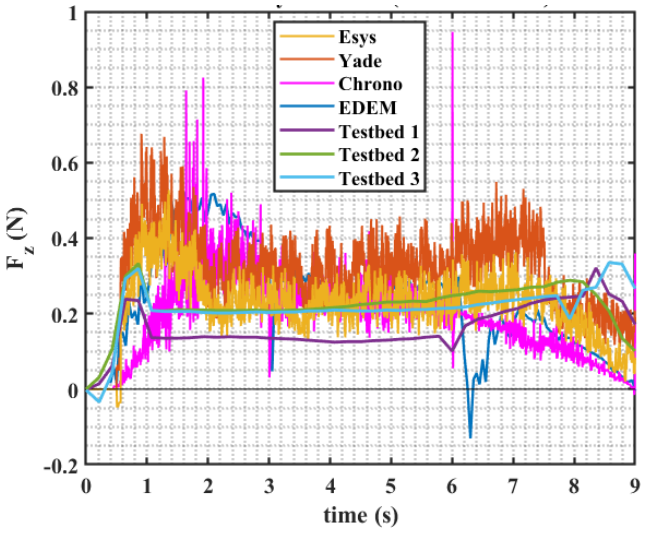

Figure 19. Comparison between Z force component in the scoop's Gazebo link frame resulting from running the same simulation scenario on EDEM, Yade, Project Chrono, ESyS-Particle, and three experiments on the physical testbed.

6. CONCLUSIONS

The Ocean Worlds Autonomy Testbed for Exploration, Research and Simulation (OceanWATERS), developed at NASA Ames Research Center, is a software simulation testbed that enables autonomous mission operations on the surface of ocean worlds. The simulation, based on the Robot Operating System (ROS) and the Gazebo virtual environment, selects as initial lander and environmental modeling the Europa Lander mission. The Europa Lander robotic arm has six degrees of freedom and is equipped with two end effectors for grinding hard ice and collecting loose particles, which are respectively: the sample excavation tool (scoop) and the trenching tool (grinder). The arm's links are: shoulder, proximal, distal, wrist, hand, scoop and grinder. The joints are controlled by a proportional-derivative-integral (PID) controller, whose parameters have been tuned using the rqt ROS interface. Arm motion for autonomous tasks execution consists of a planning phase and an execution phase, both accomplished by means of ROS services call. While each task planning requires a dedicated service, executing any of the planned trajectories is accomplished by the same trajectory publisher service. The arm operations included in the current versions of OceanWATERS are: stow arm, un-stow arm, perform a guarded move, circular digging, linear digging, grinding, deliver or discard sample. The planned trajectory is visualized on the rviz interface, executed by a "ghost" arm. Future releases of OceanWATERS will include a transition from the currently used Joint Position Controller to the Joint Trajectory Controller. While the former only controls the individual joints position values, the latter controls position and velocity for all the joints at once, guaranteeing a smooth motion and unifying the two-step planning and execution processes into one. The planning services currently include the parameter "delete_prev_traj", in the message definition, that, if set to True, all the trajectory files will be deleted. The team is planning to remove this parameter and add a safer functionality to clean up the trajectory folder. Definition of the terrain-lander interaction consists of estimating the forces and torques acting on the end effector while grinding and digging. In OceanWATERS, an accurate force feedback from the terrain is currently achieved running a discrete element method (DEM) simulation offline, filling a lookup table and include it Gazebo through a ROS plugin. With the intent of implementing a OceanWATERS-DEM co-simulation, investigation and selection of a DEM open source software has been carried out. The selected DEM open source software is Yade and it's going provide real time force feedback, running in parallel with OceanWATERS. Yade and Gazebo

will interact through a customized ROS plugin.

ACKNOWLEDGMENTS

The authors thank the entire OceanWATERS team for their effort in successfully delivering the first software release.

REFERENCES

[1] "Oceanwaters on github," last accessed 15 January 2021. [Online]. Available: https://github.com/nasa/ow_simulator/

[2] L. Fluckiger and C. Neukom, "A new simulation framework for autonomy in robotic missions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 3030–3035.

[3] L. Edwards, M. Sims, C. Kunz, D. Lees, and J. Bowman, "Photo-realistic terrain modeling and visualization for mars exploration rover science operations," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2. IEEE, 2005, pp. 1389–1395.

[4] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias *et al.*, "Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission," *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 8–12, 2004.

[5] M. Allan, U. Wong, P. M. Furlong, A. Rogg, S. McMichael, T. Welsh, I. Chen, S. Peters, B. Gerkey, M. Quigley *et al.*, "Planetary rover simulation for lunar exploration missions," in *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–19.

[6] D. R. Andrews, A. Colaprete, J. Quinn, D. Chavers, and M. Picard, "Introducing the resource prospector (rp) mission," in *AIAA SPACE 2014 Conference and Exposition*, 2014, p. 4378.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[8] K. Hand, A. Murray, J. Garvin, W. Brinckerhoff, B. Christner, K. Edgett, B. Ehlmann, C. German, A. Hayes, T. Hoehler *et al.*, "Europa lander study 2016 report: Europa lander mission," *NASA Jet Propuls. Lab., La Cañada Flintridge, CA, USA, Tech. Rep. JPL D-97667*, 2017.

[9] O. M. Umurhan, M. B. Allan, L. J. Edwards, A. Tardy, T. M. Welsh, and U. Wong, "High resolution digital elevation models of the devil's golf course: a possible terrestrial analog of europa's surface," *AGUFM*, vol. 2019, pp. P53C–3468, 2019.

[10] C. Acton, N. Bachman, B. Semenov, and E. Wright, "Spice tools supporting planetary remote sensing," 2016.

[11] D. Catanoso, T. Stucky, J. Case, and A. Rogg, "Analysis of sample acquisition dynamics using discrete element method," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–11.

[12] R. Arvidson, R. Bonitz, M. Robinson, J. Carsten, R. Volpe, A. Trebi-Ollennu, M. Mellon, P. Chu, K. Davis, J. Wilson *et al.*, "Results from the mars phoenix lander robotic arm experiment," *Journal of Geophysical Research: Planets*, vol. 114, no. E1, 2009.

[13] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[14] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[15] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, April 2013, pp. 1–6.

[16] J. Kozicki and F. V. Donze, "Yade-open dem: An open-source software using a discrete element method to simulate granular material," *Engineering Computations*, 2009.

[17] D. Weatherley, "Esys-particle v2. 0 user's guide," 2009.

[18] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *International Conference on High Performance Computing in Science and Engineering*. Springer, 2015, pp. 19–49.

[19] "Edem simulation," last accessed 16 October 2020. [Online]. Available: https://www.edemsimulation.com/

BIOGRAPHY

Damiana Catanoso *is a Software Engineer, and USRA employee, working in the Intelligent Systems Division at NASA Ames Research Center. She received her B.S. degree in Aerospace Engineering from La Sapienza University of Rome in 2015 and a double M.S. degree in Space Automation and Control from Wuerzburg University and Lulea University of Technology in 2019.*

Anjan Chakrabarty *received his B.S. from Jadavpur Universiry, India. He completed his MS(2010) and PhD(2014) from the Pennsylvania State University in Aerospace Engineering. Anjan Chakrabarty is currently a Research Engineer with the Advanced Control and Evolvable Systems (ACES) Group in the Intelligent Systems Division at NASA Ames Research Center (ARC). He is employed by KBR (formerly SGT Inc). Prior to joining KBR, Anjan was a NASA Postdoctoral Program (NPP) Fellow.*

Jason Fugate *received his B.S. (2015) and M.S. (2017) degrees in Aerospace Engineering from The Ohio State University. He earned a second M.S. degree in Aeronautical Mechanics & Energetics from ISAE-ENSMA (Poitiers, France) in 2017. Jason joined the workforce at NASA Ames in 2017 and has worked on a variety of aeronautics and aerospace robotics projects as a modeling and simulation specialist since.*

Ussama Naal *received his B.S. degree in Informatics Engineering in 2007 from University of Aleppo and a M.S. in Electrical and Computer Engineering from the University of Oklahoma in 2011. His expertise lies within areas of image processing, data visualisation, computer graphics and simulation. Ussama is now part of Intelligent Robotics Group at Ames Research Center focusing on robotics and simulation.*

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

Laurence J. Edwards *is a research scientist at the NASA Ames Research Center leading surface reconstruction in the Autonomous Systems and Robotics Area. His current research focuses on automated techniques for Digital Terrain Model generation from orbital and surface imagery. Dr. Edwards received his Ph.D. and M.S. in Mechanical Engineering from Stanford University, and a B.S.E. in Aeronautical and Mechanical Sciences from Princeton University.*