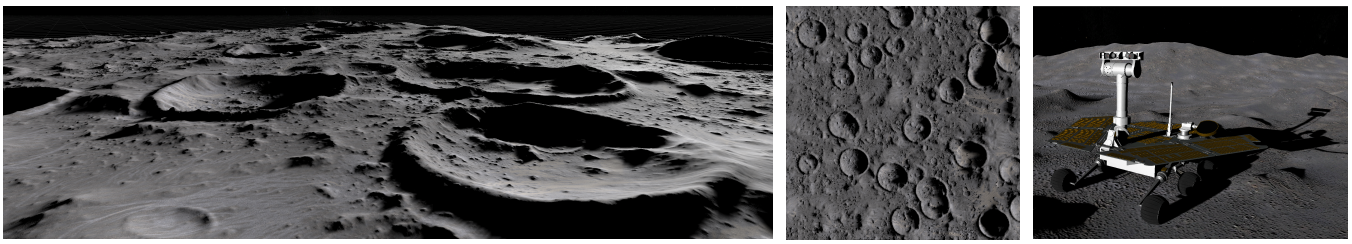
Comparison of SLAM algorithms on Unity-ROS(Robot Operating System) Simulator for Space Applications(URSSA)

Midhun S. Menon[1] and Daniel Koris[2] and Daniel Szafir[2,3] and Jack Burns[1]



(A) (B) (C)

Fig. 1: We present a test-bed setup for evaluating vision-based navigation for lunar operations. (A) Virtual lunar topography within the simulation testbed with synthetic terrain and accurate photometric responses modeled after the surface of the moon. (B) Top-down view of the entire 1 km $\times$ 1 km lunar terrain. (C) The MER 1 rover dynamic model used in our case study simulation.

***Abstract*— Robots are critical enablers for space exploration as they help offset safety risks associated with a human astronaut, aid in precursor missions prior to manned missions, provide critical on-mission and post-hoc support. However, designing robotic systems to navigate, map and localize itself in hostile and unknown extraterrestrial worlds remains open challenge. The prime reason for this being the testing the semi-autonomous agents for robust performance in such environments. In this paper, we address this issue by using Unity-ROS(Robot Operating System) Simulator for Space Applications(URSSA) for compare performance of three popular Visual Intertial Odometry (VIO) Simultaneous Localization And Mapping (SLAM) algorithms on lunar surface. The test architecture, agent modelling, rationale for test cases and generation of ground truth data is discussed. Paper concludes with results from simulations comparing the algorithms and discusses on failure reasons and potential directions for improving algorithms for better applicability such environments. We believe such a comparative study can give vital pointers for future research directions and help accelerate development of specific algorithms for SLAM in extra terrestrial environments.**

I. INTRODUCTION

The last decade has seen major space faring nations chart out focussed plans for deep space exploration starting with the moon. The Artemis I[1], Chang'e-4 mission[2] and Chandrayaan-2[3] are some of the ongoing and completed precursor missions to put human explorers on lunar surface. These missions have robotic systems onboard to do remote investigation,survey and data collection of the lunar surface. The follow-up missions to all of these missions have a increasing component of robotic systems of which ground based mobility systems are central. Wheeled rovers are the most commonly used platform for planetary surface mobility. To date, they have been used for surface exploration, mapping and scientific investigation. However, future missions call for advanced mission operations including periodic monitoring of geo-spatially distributed scientific assets, remote assembly and In-Situ Resource Utilization(ISRU)[4] tasks. The first of these call for navigation, mapping and localization capabilities over large ranges and the ISRU tasks require advanced manipulation and dexterity capabilities. More importantly, these algorithms need to be tested under a rich and varying environment to make sure they perform reliably. In this paper, the focus is on the testing of algorithms for navigation, mapping and localization. Currently, space robotics community relies on mission analogs for evaluation and testing of such on-board algorithms. These are operations carried out on Earth in a mock-up of the true mission environment. However, mission analogs are incapable of simulating certain critical, unique and finer aspects of the extraplanetary environment (e.g., gravity, photometric anomalies etc.) which impacts testing aefficacy. As an example, visual artefacts present in target environment but not simulated in a mission analog may artificially inflate expectations and over-estimate algorithm performance during testing. On similar lines, locating a matching topography for a mission analog on earth can also prove increasingly challenging as the geological processes governing surface formation and evolution will differ entirely in most cases.

In this research, we wish to address the problem of designing a virtual analogs for extra-terrestrial environments within a multiphysics framework for advancing research in algorithms that govern low-level autonomy and support interactive trade-offs between various levels of supervisory control. Furthermore, such virtual environments may enable future explorations into the design of new interfaces that support ground control and/or astronaut operation of surface robots from orbital stations to significantly improve critical space exploration missions.

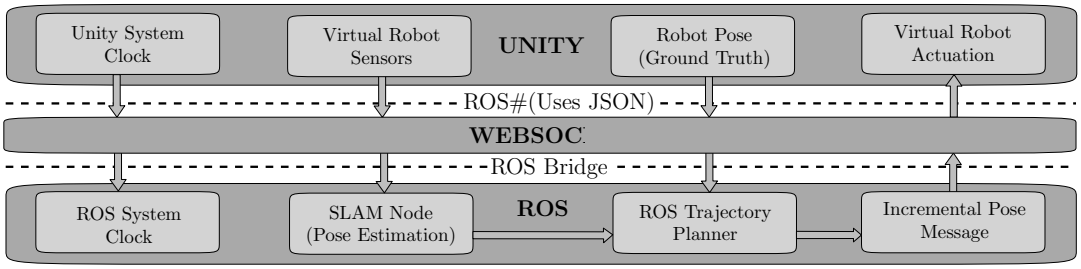


Fig. 2: URSSA system diagram detailing our modular architecture for bidirectional Unity–ROS communication.

II. Current Work

Virtual terrain and environments have been in use in the modeling-simulation and computer graphics community for many years. One of the earliest applications of virtual terrain was to model terrain wheel interactions([5], [6], [7], [8]). In [9], virtual environments are used for training purposes in space tasks and mission operations. However, both these applications were limited in scope of modeling only the topography of the terrain. Some of the first applications of accurate photometric models were in better scientific visualization of celestial bodies([10], [11], [12]) and for investigating perceptual errors made by human beings when infering scales and distances on extra-terrestrial environments[13]. More recently, a commercially available software platform called Planet and Asteroid Natural Scene Generation Utility(PANGU) uses accurate photometric and optical modeling to create high fidelity simulation of terrains and Entry-Descent-Landing(EDL) simulations([14], [15]). However, it is a highly specific software meant only for visualization and the best achievable frame rates are less than 10Hz.

To the best of our knowledge, the only other framework for simulating the rover-environment interaction is developed inhouse by NASA[16] for moon. However, it has been designed for high-fidelity simulation of lunar environment vision-based navigation algorithms and are limited in scope, scalability and their ability to provide perceptually real-time simulation. In addition, other frameworks are mostly based on classic simulation platform called Gazebo[17], which is not always as efficient, realistic, or scalable as modern engines such as Unity ([18], [19]).

This is the reason for simulators becoming increasingly based on frameworks like ROS[20] and Unity([21], [22]). Hence, in this paper, in-house developed simulation framework Unity-ROS Simulator for Space Applications(URSSA)[23]**(DS:How to cite the RSS paper?)** is used(Fig. 1), that leverages the Unity game engine as a virtual environment simulator to take advantage of its advanced rendering pipeline, support for reliable physics engines and scalable memory management capabilities. ROS is used as the framework to model the autonomous/semi-autonomous robotic systems interacting with the virtual world in Unity. We demonstrate the comparative testing of three popular Visual and/or Intertial Odometry (VIO)[24] Simultaneous Localization And Mapping (SLAM)[25] algorithms for surface exploration of moon. Only visual and inertial sensing modalities have been considered here as these are the ubiquitous sensors reliably flight tested and proven with minimal mass penalties on multiple missions to date by space agencies across the world. However, it may be noted that URSSA has no such restrictions.

III. Simulator model

The simulator architecture is shown in Fig. 2 and consists of three main components. Unity is used for simulating the environment, topography, photometry, rover dynamics and terrain interaction. The second main component is ROS and it is used to simulate the agent behaviour, which essentially takes in the cognitive information of the environment through the simulated sensors, and generates appropriate inferences(SLAM) and actions on how to interact with the environment(path planning). These two independent components are coupled through a communication framework called ROS#([22]), which also takes care of time synchronization and guarantees across the components. The details of the modelling are in [23]. The following sections give a brief overview each of these components.

A. Environment model(Unity)

The virtual environment is modeled as multiple interacting sub-systems inside Unity environment.

1) Topographic Model: The lunar topography is simulated using high-resolution synthetic terrain modeled on actual lunar surface data from Lunar Reconnaissance Orbiter(LRO)-Narrow Angle Camera(NAC) Digital Terrain Models(DTM) which are available to a resolution of 0.5m(at best). Sub-millimeter super-resolved terrain is generated from these DTMs using observed lunar surface roughness parameters([26], [27]), crater distributions([28], [29]) and boulder distributions [30] from the Apollo missions to generate physically realistic terrain by fractal expansion algorithms. Rocks of envelope diameters in range of 0.5m–2.0m have also been procedurally generated and uniformly distributed on the terrain as these are not captured by LRO-NAC DTM in its resolution limits(Refer Fig. 1(A,B)).

2) Photometric Model: Custom shaders are developed based on Hapke photometric model[31] to simulate the photometric response of moon called Opposition Surge(OS)[32](Fig. 3(A)), a specific behaviour exhibited by moon at low phase angles which induces visual artefacts like glare, lens flares and sensor saturation in optical imaging systems. Such artefacts can drastically affect mapping algorithms and perception-aided planners [33].

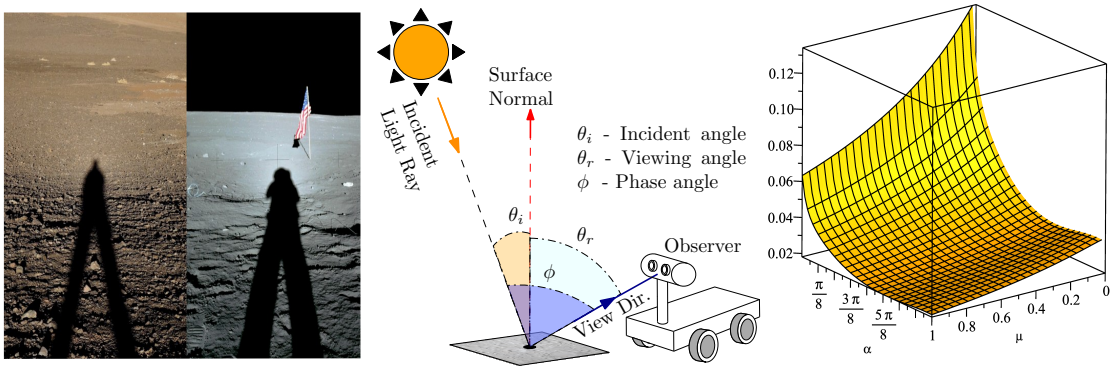


Fig. 3: (A)Comparig similar images taken on earth(left, Atacama desert and moon(right, Apollo 12), Opposition Surge is visible in brightening aroung head shadow Image credit:Molaro/NASA (B)Variables in a BRDF (C)Hapke BRDF plot

The Hapke Bidirectional Reflectance Distribution Function(BRDF)(Fig. 3(C)) is implemented in the Unity Physically Based Rendering(PBR) pipeline[34] and the parameters for the same are derived from lunar parameter maps[35]. The variables in the BRDF are the incident angle θ_i, the reflected angle θ_r and the phase angle ϕ(Fig. 3(B)). The plot of the Hapke BRDF is shown in Fig. 3(C), which shows the OS when the phase angle approaches zero.

3) Rover Mechanical Model: The rover is modeled as a multi-body system based off the design of Mars Exploration Rover(MER). The rover geometry has been modeled inside Unity(Fig. 1(C)) along with a stereo camera pair at the mast head and Inertial Measurement Unit(IMU) placed on the body. Every attempt is made to match the model rover design and parameters to the MER[36], [37]. However, the rocker-bogie suspension stiffness/damping parameters and the wheel masses have been tuned to minimize disturbances due to terrain irregularities on the rover body. Thus the suspension in-effect acts as a mechanical low-pass filter to terrain disturbances and also helps guarantee minium three point contact at all times. The six independently powered wheels are distributed as three pairs - front, middle and back. Of these, the front and back pairs act as steering wheels. Moreover, rover left and right rockers counter rotate actively to self balance the rover when moving over obstacles, as implemented in MER.

4) Rover Sensor Models: An exploratory rover operating in an unknown/partially known enviroment will likely have multiple sensors that collect data across various modalities [38], [39], ranging from hyper-spectral optical imagers (visible, infrared, ultraviolet, radio-spectrum etc.), haptic sensors, 3D ranging sensors (structured light sensors, stereo imagers, LIDARs etc.), odometry sensors, inertial sensors, etc. While the framework theoretically enables simulating any of these sensor modalities, two types of sensors are used in this paper : visible spectrum cameras (monocular and binocular) and Inertial Measurement Unit (IMU), the reasons being that these are the most reliable and least space/mass penalizing sensors on extra-terrestrial mobility platforms.

The IMU is modeled with based on sensor models suggested by [40] and [41] and elucidated in [42]. In short, the ground truth pose data of the frame is corrupted with noise from double integration of bias, scale factor uncertainties, and Additive White Gaussian Noise (AWGN) on angular rates (gyros) and accelerations (accelerometers).

The imaging sensor is modeled as a pinhole camera with radial distortion. The camera parameters are kept at the same values as the MER mast camera stereo pair(refer table in p4,[37]).

5) Wheel-Terrain Interaction Model: The wheels on the rover have been modeled as ellipsoids so as to simplify collider computations. Wheeled vehicles on lunar regolith experiences skid, slip and sinkage[43]. There are various traction models in literature to predict the contact forces at the wheel soil interface based on continuum methods([44], [45], [46]), Discrete Element Models([47], [48], [49], [50], [48]) and empirical models([51], [52]). However, as a first level simplification, in this paper, slip alone is considered with Coulomb dry friction model, wherein traction force is proportional to the normal reaction at the wheel-soil interface. Subsequent studies will incorporate some of the more advanced traction models mentioned above.

B. Agent Model(ROS)

The virtual robotic agents which are the processing and decision making portion of the robot is implemented as nodes in ROS. The SLAM algorithm runs as a ROS node, which estimates the robot's pose and trajectory. A ROSbridge node[53] acts as the modem between ROS and Unity. ROSbridge is used to stream sensor data from Unity to ROS nodes and send drive commands ROS to Unity.

C. Communication Framework(ROS#)

The ROS# framework is used to enable communication between Unity and ROS. The framework enables Unity to publish and subscribe to topics on a ROS nodes by encoding/decoding information using JSON format and communicating through websockets.

1) Time Synchronization: Right out-of-the-box, the system does not have time synchronization as the time-scales vary between ROS and Unity, due to which time flows at

different rates. As the publisher-subscriber model is asynchronous, time needs to be explicitly synchronized as all estimates are time stamped. Our architecture leverages a feature of ROS to coordinate timing across its distributed architecture as a properly designed ROS node may use all internal ROS timing mechanisms rather than machine-level timing mechanisms. These ROS timing mechanisms can be driven by an external synchronizing entity. In our case, since Unity is providing the simulation environment and is seen as the potential timing bottle-neck in the system, we publish a topic from Unity to drive the ROS internal timings. Since all data is time-stamped on both ends, and Unity is the singular clock generating timestamps, our framework is robust in terms of distributed temporal synchronization; in other words, if there is a delay on the Unity side, that delay will be propagated in the ROS system (although we note in our testing to-date, the system runs in perceptually real time).

2) Real Time compliance: If the simulator is to plugin to an external agent with independent clock, the combined system will loose time-synchronization. A typical scenario is the case where the simulator is interfaced with a independent SLAM algorithm running its own clock internally, in which case estimation results will be wrongly time stamped causing accumulated error over time. In order to avoid this, the WebRTC architecture[54] is implemented for the Unity websocket communication. The WebRTC is a Real-Time Compliant websocket communication architecture with hard-realtime guarantees, thereby helping to mitigate the timing issue and making the simulator robust.

3) Fixed Update: The *FixedUpdate* method is used for the physics calculations in Unity by which fidelity of the physics calculations are guaranteed at a fixed rate independent of the rendering frame rate. This helps make sure that physics computations are not skipped when the rendering frame rate gets throttled, making the simulation more robust to numerical errors and instabilities from large integration steps.

IV. Test System

The objective of this test system is to evaluate and compare performance of various SLAM algorithms on extra-terrestrial planetary bodies. Data is recorded over preset trajectories on the terrain using rosbags and fed as input to the SLAM algorithms, who then estimate the pose based on these inputs. Due to the probabilistic nature of many SLAM algorithms, statistical measures over multiple iterations of the same trajectory are generated and reported for performance evaluation. The main variables in this model are (a) Sun-Camera geometry (b) Shader Hapke coefficients (c) Path type. The test system has a total of 54 test cases as detailed below.

A. Sun-Camera Geometry

To study performance under varying extremes of sun-camera geometry, two positions of sun's elevation-10°(dawn/dusk) and 90°(noon) are chosen. And at each of these elevations, three azimuths(with respect to rover camera) are tested-0°(anti), 90°(cross) and 180°(pro).

B. Shader Hapke coefficients

To evaluate algorithm performance under varying shader backscatter conditions, three test cases are proposed - adversarial, nominal and benign. This will help evaluate performance and sensitivity to shader parameters.

C. Path type

For comparing performance under different types of trajectory shapes, the algorithms have to be run with three types of trajectories - straight, circular and rectangular trajectories.

V. Simulation

For this case study, we leveraged the modular architecture of URSSA to run a simulation using desktop PCs to distribute the compute load of virtual environment rendering and running SLAM. One machine (Intel 3.6 GHz processor, 32 GB RAM, NVIDIA GeForce RTX 2080 8 GB graphics card) ran the Unity environment simulation using Microsoft Windows 10 while the other (Intel 4 GHz processor, 8 GB RAM, NVIDIA GeForce GTX 1070 8 GB graphics card) performed the back-end SLAM estimation and robotic control using ROS Kinetic on Ubuntu 16.04. In the virtual lunar environment, the only source of illumination was sunlight, which was modeled as a directional white source (R:255,G:255,B:255). The virtual $1km \times 1km$ lunar terrain was modeled with a resolution of 0.5mm to ensure it has enough fidelity even when viewed from a height of 0.8m–1.0m (i.e., the typical height of a mast camera for a micro-rover).

VI. Results and Discussion

To be Done

VII. Conclusion

To be Done.

References

[1] M. J. Sundahl and C. D. Johnson, "Setting the stage: Introduction to nasas artemis program and the basics of space law," 2020.

[2] Y. Jia, Y. Zou, J. Ping, C. Xue, J. Yan, and Y. Ning, "The scientific objectives and payloads of change- 4 mission," *Planetary and Space Science*, vol. 162, pp. 207–215, 2018.

[3] V. Sundararajan, "Overview and technical architecture of india's chandrayaan-2 mission to the moon," in *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 2178.

[4] R. D. Green and J. E. Kleinhenz, "In-situ resource utilization (isru) living off the land on the moon and mars," 2019.

[5] Y. Yang, J. Bao, and Y. Jin, "Realistic virtual lunar surface simulation method," *Journal of System Simulation*, vol. 19, no. 11, pp. 2515–2518, 2007.

[6] Y.-c. Yang, J.-s. Bao, Y. Jin, and Y.-I. Cheng, "A virtual simulation environment for lunar rover: framework and key technologies," *International Journal of Advanced Robotic Systems*, vol. 5, no. 2, p. 16, 2008.

[7] L. Ding, H. Gao, Z. Deng, P. Song, and R. Liu, "Design of comprehensive high-fidelity/high-speed virtual simulation system for lunar rover," in *2008 IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, 2008, pp. 1118–1123.

[8] H. Gao, Z. Deng, L. Ding, and M. Wang, "Virtual simulation system with path-following control for lunar rovers moving on rough terrain," *Chinese Journal of Mechanical Engineering*, vol. 25, no. 1, pp. 38–46, 2012.

[9] E. Andreev, M. Nikolova, V. Radeva, and G. Bochev, "Creating moon port and spaceship simulations in a virtual environment," in *AIP Conference Proceedings*, vol. 2048, no. 1. AIP Publishing LLC, 2018, p. 020028.

[10] P. Van der Torren and P. Vreeburg, "Investigation and application of graphics shader technology in space simulation," 2011.

[11] E. Wright, "Preparing for a lunar impact," *ACM SIGGRAPH Computer Graphics*, vol. 45, no. 1, pp. 1–9, 2011.

[12] H. W. Jensen, F. Durand, J. Dorsey, M. M. Stark, P. Shirley, and S. Premože, "A physically-based night sky model," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 399–408.

[13] C. T. Oravetz, L. R. Young, and A. M. Liu, "Slope, distance, and height estimation of lunar and lunar-like terrain in a virtual reality environment," *Gravitational and Space Research*, vol. 22, no. 2, 2011.

[14] M. Dunstan, I. Martin, S. Parkes, and M. S. Gestido, "Pangu v4: A software tool for testing vision-based guidance and navigation systems for in-orbit, entry descent and landing and surface mobility operations," in *Data Systems in Aerospace Conference, DASIA 2018*, 2018.

[15] I. Martin, M. Dunstan, and M. S. Gestido, "Planetary surface image generation for testing future space missions with pangu," in *2nd RPI Space Imaging Workshop*, 2019.

[16] 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.

[17] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[18] U. G. Engine, "Unity game engine-official site," *Online][Cited: October 9, 2008.] http://unity3d. com*, pp. 1534–4320, 2008.

[19] A. Konrad, "Simulation of mobile robots with unity and ros: A case-study and a comparison with gazebo," 2019.

[20] 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.

[21] E. Babaians, M. Tamiz, Y. Sarti, A. Mogoei, and E. Mehrabi, "Ros2unity3d; high-performance plugin to interface ros with unity3d engine," in *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium*. IEEE, 2018, pp. 59–64.

[22] M. Bischoff, "ROS#," June 2019. [Online]. Available: https://github.com/siemens/ros-sharp/releases/tag/v1.5

[23] "Urssa: A unity-ros simulator for space applications," 2020.

[24] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery, "Augmenting inertial navigation with image-based motion estimation," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02Ch37292)*, vol. 4. IEEE, 2002, pp. 4326–4333.

[25] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles," in *Robotics Research*. Springer, 1996, pp. 613–625.

[26] M. S. C. (US), *Analysis of Apollo 10: Photography and Visual Observations*. Scientific and Technical Information Office, National Aeronautics and Space , 1971, vol. 232.

[27] L. C. Rowan, J. F. McCauley, and E. A. Holm, "Lunar terrain mapping and relative-roughness analysis," 1971.

[28] G. Neukum, B. König, and J. Arkani-Hamed, "A study of lunar impact crater size-distributions," *The moon*, vol. 12, no. 2, pp. 201–229, 1975.

[29] G. H. Heiken, D. T. Vaniman, and B. M. French, "Lunar sourcebook-a user's guide to the moon," *Research supported by NASA,. Cambridge, England, Cambridge University Press, 1991, 753 p. No individual items are abstracted in this volume.*, 1991.

[30] R. Watkins, K. Mistick, B. Jolliff, and S. Lawrence, "Boulder distributions around young lunar impact craters: Case study of south ray crater," in *Lunar and Planetary Science Conference*, vol. 49, 2018, p. 1146.

[31] B. Hapke, *Theory of reflectance and emittance spectroscopy*. Cambridge university press, 2012.

[32] T. Gehrels, T. Coffeen, and D. Owings, "Wavelength dependance of polarization. iii. the lunar surface." *The Astronomical Journal*, vol. 69, 1964.

[33] K. Otsu, A.-A. Agha-Mohammadi, and M. Paton, "Where to look? predictive perception with applications to planetary exploration," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 635–642, 2017.

[34] A. Pranckevičius and R. Dude, "Physically based shading in unity," in *Game Developer's Conference*, 2014.

[35] H. Sato, M. Robinson, B. Hapke, B. Denevi, and A. Boyd, "Resolved hapke parameter maps of the moon," *Journal of Geophysical Research: Planets*, vol. 119, no. 8, pp. 1775–1805, 2014.

[36] J. A. Crisp, M. Adler, J. R. Matijevic, S. W. Squyres, R. E. Arvidson, and D. M. Kass, "Mars exploration rover mission," *Journal of Geophysical Research: Planets*, vol. 108, no. E12, 2003.

[37] J. Maki, J. Bell, K. E. Herkenhoff, S. Squyres, A. Kiely, M. Klimesh, M. Schwochert, T. Litwin, R. Willson, A. Johnson, *et al.*, "Mars exploration rover engineering cameras," *Journal of Geophysical Research: Planets*, vol. 108, no. E12, 2003.

[38] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," *Journal of robotic systems*, vol. 14, no. 4, pp. 231–249, 1997.

[39] S. Ruocco, *Robot sensors and transducers*. Springer Science & Business Media, 2013.

[40] P. G. Savage, "Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms," *Journal of guidance, control, and dynamics*, vol. 21, no. 1, pp. 19–28, 1998.

[41] ——, "Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 208–221, 1998.

[42] D. Titterton, J. L. Weston, and J. Weston, *Strapdown inertial navigation technology*. IET, 2004, vol. 17.

[43] L. Ding, H. Gao, Z. Deng, K. Yoshida, and K. Nagatani, "Slip ratio for lugged wheel of planetary rover in deformable soil: definition and estimation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3343–3348.

[44] M. G. Bekker, "Mechanics of locomotion and lunar surface vehicle concepts," *Sae Transactions*, pp. 549–569, 1964.

[45] J.-Y. Wong, "Behaviour of soil beneath rigid wheels," *Journal of Agricultural Engineering Research*, vol. 12, no. 4, pp. 257–269, 1967.

[46] L. Ding, Z. Deng, H. Gao, J. Tao, K. D. Iagnemma, and G. Liu, "Interaction mechanics model for rigid driving wheels of planetary rovers moving on sandy terrain with consideration of multiple physical effects," *Journal of Field Robotics*, vol. 32, no. 6, pp. 827–859, 2015.

[47] K. Yoshida, T. Watanabe, N. Mizuno, and G. Ishigami, "Terramechanics-based analysis and traction control of a lunar/planetary rover," in *Field and Service Robotics*. Springer, 2003, pp. 225–234.

[48] H.-J. Jeong, J. H. Lim, and J.-W. Kim, "Development of a coarse lunar soil model using discrete element method," *Journal of the Korean Society for Aeronautical & Space Sciences*, vol. 47, no. 1, pp. 26–34, 2019.

[49] D. Rodríguez-Martínez, M. Van Winnendael, and K. Yoshida, "High-speed mobility on planetary surfaces: A technical review," *Journal of Field Robotics*, vol. 36, no. 8, pp. 1436–1455, 2019.

[50] M. Jiang, Y. Dai, L. Cui, and B. Xi, "Experimental and dem analyses on wheel-soil interaction," *Journal of Terramechanics*, vol. 76, pp. 15–28, 2018.

[51] H. Huang, S. Xu, Z. Meng, J. Li, and J. Zhang, "The sinkage characteristics and prediction of a planetary rover based on a similarity model experiment," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 10, pp. 3762–3774, 2019.

[52] J. Y. Wong, "Predicting the performances of rigid rover wheels on extraterrestrial surfaces based on test results obtained on earth," *Journal of Terramechanics*, vol. 49, no. 1, pp. 49–61, 2012.

[53] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Robotics Research*. Springer, 2017, pp. 493–504.

[54] S. Loreto and S. P. Romano, *Real-time communication with WebRTC: peer-to-peer in the browser.* " O'Reilly Media, Inc.", 2014.