Study on Lunar Simulation Softwares

I. JPL Open Source Rover Project by NASA

The Open Source Rover project by NASA JPL primarily focuses on building a terrestrial rover, and its software is designed to control its movements and process sensor information. It is usually used on Earth rather than in a lunar or space environment. However, the principles and some aspects of the software could be adapted or extended to simulate or incorporate factors relevant to space exploration.

Figure 1: Example of a rover (Paddy-rover)

The current scope of the Open Source Rover project does not explicitly mention support for importing external robotic models in XML or 3D formats. However, if the software environment is expanded to include simulation tools like Gazebo, importing such models would be feasible. The provided software primarily controls physical rover hardware rather than simulating motion or interactions in a virtual environment. Python is the programming language that is usually used. ROS integration is possible by setting up ROS nodes on the Raspberry Pi and Arduino. Python also supports ROS integration.

II. NASA Lunar Terrain Visualization using Unreal Engine 5

The "NASA Lunar Terrain Visualization in Unreal Engine 5" presentation at Unreal Fest 2022 showcased how NASA uses Unreal Engine 5 (UE5) to create highly detailed and accurate visualizations of the lunar surface.

Figure 2: Apollo 17 image

Figure 3: Recreation using UE5

Accurate 3D models of the lunar surface are created using Digital Elevation Models (DEMs) and using detailed texturing techniques, the lunar regolith and various geological features are simulated. The software simulates the lighting and shadow conditions on the Moon, including the stark contrasts and deep shadows created by the low-angle sunlight, which is crucial for navigation and planning.

Three methods were used to generate lunar terrain:

Landscapes: Use GDAL to convert DEM file to a PNG and import directly into level using UE5 Landscape Mode.

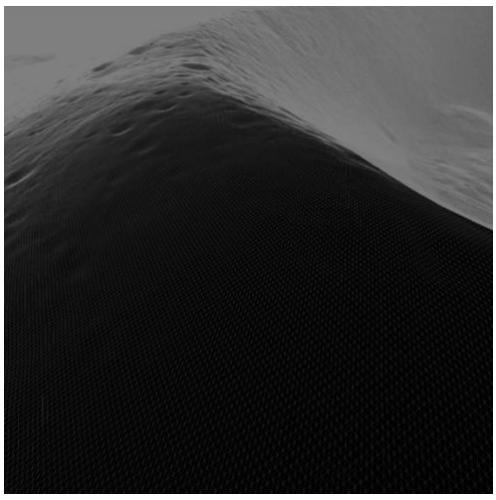

Figure 4: Landscape wireframe

It is easy to use and requires no additional plugins. However, it is difficult to add accurate lunar curvature and has performance limitations depending on memory, file size, etc.

Nanite meshes: Import DEMs directly one at a time with custom in-engine Heightmap Importer Tool. Tiles selected DEMs and builds static meshes from the data with Nanite enabled.

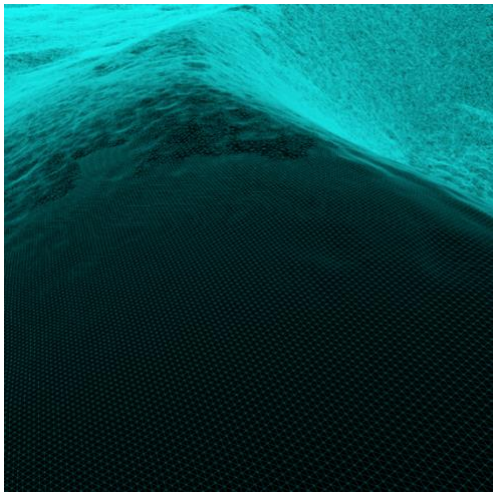

Figure 5: Nanite mesh wireframe

This method has good performance, can add lunar curvature and can be further optimized with virtual shadow maps. It is also easy to use. However, it takes time to generate and it is difficult to fix issues within the mesh. It also has more hardware requirements due to the virtual memory utilization.

Clip maps: During runtime DEMs are imported and continuously supply the clip map with data.

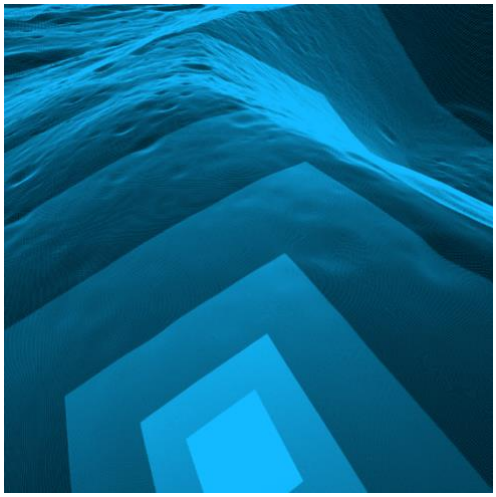

Figure 6: Clip map wireframe

This method also has good performance and can add lunar curvature. It is memory efficient and easy to modify. However, fast camera speeds can display artifacts as the terrain updates and shadows may not be fully formed.

UE5 supports importing various 3D model formats, like OBJ, which allows for the integration of robotic models and other assets. Moreover, UE5 includes a physics engine that allows for realistic modeling of motion and interactions with the environment. UE5 supports python, MATLAB, and ROS integration. The software can simulate various sensors used for robot navigation, including cameras, LIDAR, IMUs, and ultrasonic sensors.

III. OmniLRS

The OmniLRS is an open-source robotics simulator for lunar environment. It accurately models lunar terrains using data from real lunar and can generate terrain features like craters and rocks.

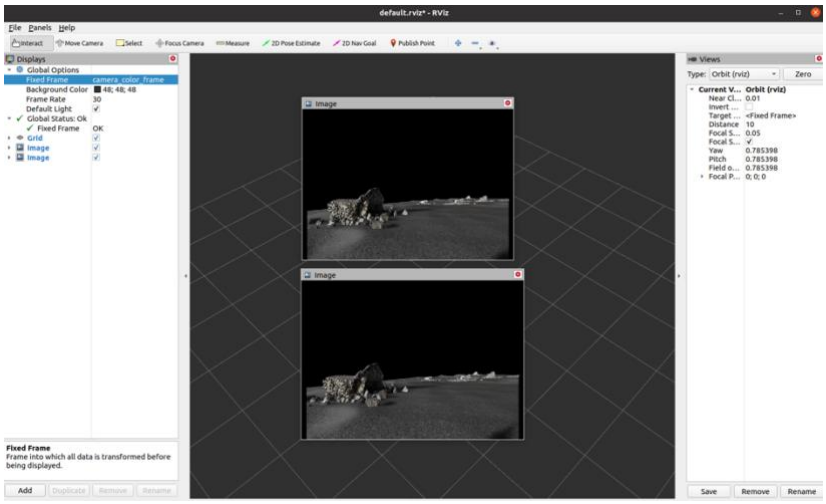

Figure 7: OmniLRS simulation

The simulation was created with the help of pre-made environments, like the LunaLab at the University of Luxembourg, which is a 6.5m x 10m lunar-analog facility. Images from the Lunar Reconnaissance Orbiter (LRO) were used to create 3km x 3km DEMs with a 1-meter resolution. The software supports modeling interactions with the environment like simulating the motion of rovers across the terrain and interactions with obstacles like rocks and craters. 100 half-crater profiles from lunar images were processed to create a library of crater shapes. Using these profiles, craters of various sizes and shapes can be generated and distributed using Poisson distribution. The terrain generated is fed into IsaacSim, a simulation application built on NVIDIA Omniverse. After generating the terrain, features like texture and rocks were added. The simulator offers 4 robot models equipped with onboard sensors ready for immediate use, including the Leo rover and EX1 rover. To accurately simulate onboard sensors like stereoscopic cameras, specific parameters and attachment positions are set. The simulator integrates environments and robots seamlessly with both ROS1 and ROS2. The robots utilize default sensors within Isaac, including RGBD cameras, 2D and 3D lidars, IMUs, Tfs (transformations), and joint states. Since the simulation uses IsaacSim, we can provide our own 3D models of robots.

References

https://github.com/nasa-jpl/open-source-rover
https://www.youtube.com/watch?v=VISq_V4W6LU&ab_channel=UnrealEngine
https://ntrs.nasa.gov/api/citations/20220013642/downloads/UnrealFest.pdf
https://arxiv.org/pdf/2309.08997v1
https://github.com/AntoineRichard/OmniLRS