

Article

LunarSim: Lunar Rover Simulator Focused on High Visual Fidelity and ROS 2 Integration for Advanced Computer Vision Algorithm Development

Dominik Pieczyński , Bartosz Ptak , Marek Kraft *  and Paweł Drapikowski 

Faculty of Control, Robotics and Electrical Engineering, Poznań University of Technology, 60-965 Poznań, Poland

* Correspondence: marek.kraft@put.poznan.pl

Abstract: Autonomous lunar exploration is a complex task that requires the development of sophisticated algorithms to control the movement of lunar rovers in a challenging environment, based on visual feedback. To train and evaluate these algorithms, it is crucial to have access to both a simulation framework and data that accurately represent the conditions on the lunar surface, with the main focus on providing the visual fidelity necessary for computer vision algorithm development. In this paper, we present a lunar-orientated robotic simulation environment, developed using the Unity game engine, built on top of robot operating system 2 (ROS 2), which enables researchers to generate quality synthetic vision data and test their algorithms for autonomous perception and navigation of lunar rovers in a controlled environment. To demonstrate the versatility of the simulator, we present several use cases in which it is deployed on various efficient hardware platforms, including FPGA and Edge AI devices, to evaluate the performance of different vision-based algorithms for lunar exploration. In general, the simulation environment provides a valuable tool for researchers developing lunar rover systems.

Keywords: space robotics; computer vision; deep learning; simulation



Citation: Pieczyński, D.; Ptak, B.; Kraft, M.; Drapikowski, P. LunarSim: Lunar Rover Simulator Focused on High Visual Fidelity and ROS 2 Integration for Advanced Computer Vision Algorithm Development. *Appl. Sci.* **2023**, *13*, 12401. <https://doi.org/10.3390/app132212401>

Academic Editor: Alessandro Di Nuovo

Received: 28 September 2023

Revised: 9 November 2023

Accepted: 14 November 2023

Published: 16 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Exploring the Moon and other planets has been a long-standing goal of space agencies and private companies around the world. The number of space launches has seen a steady increase in recent years. Lowering costs and the broader availability of key technologies have driven progress and made space exploration more accessible [1,2]. However, the growth in this sector may slow down because of issues related to the scalability of current technology. Although space-related innovations are certainly at the forefront of technical and scientific advancement, most missions still rely on human supervision and intervention. To overcome this issue, we can increase the degree of autonomy by implementing more advanced and independent systems. Autonomy is finding its way into satellites, which require relatively little operator attention. One can observe a trend of making the satellite hardware relatively more and more self-contained, with on-board processing and control capabilities [3,4]. The next logical step is applying autonomy to in-orbit operations and space robotics [5,6]. Complex in-orbit operations, or robot steering, require significant human involvement when carried out completely manually. With the increasing number of launches, one can expect the number of devices and vehicles performing such complex tasks to grow. Without some degree of autonomy, the sheer amount of equipment that needs to be supervised and tended to by qualified staff will impose serious limitations on the scalability of such deployments. Coupled with additional limitations for manual control, such as communication delays, this poses a problem that needs to be addressed—with new paradigms, methods, and algorithms facilitating automation and autonomy in space.

The number of space applications on the verge of disruption by increased autonomy is vast, including, among numerous others, robotic rovers. The number of deployments of such robots has increased significantly as of late [7–10]. To prepare for these missions, simulating the environments of these extraterrestrial bodies has become an essential part of the development of robotic technologies. Although many such solutions have been developed over the years, most of them are not up to the task of addressing the challenges of modern-day space autonomy. The recent wide adoption of deep learning [11,12] and the increased use of industry-standard robotic software [13–15] are slowly making their way into space-related applications. Moreover, space applications often use dedicated and heterogeneous computational resources, which creates the need for hardware-in-the-loop (HIL) simulation, allowing the testing of the complex interplay of hardware and software under possibly realistic conditions [16].

In this paper, we present LunarSim, a space simulator that is tightly integrated with ROS 2 [17], space-ROS [15], and micro-ROS [18], and is capable of performing simulations, using a wide range of hardware (programmable logic, embedded microcomputers, and microcontrollers). A sample view of its environment is demonstrated in Figure 1. The main motivation for the simulator’s development was the insufficient visual fidelity of the images generated by other simulators and/or their limited availability. The focus of the majority of simulators is on terramechanics, as they were mostly developed to analyse the problem of terrain traversal, with the quality of visualisations, the capabilities of HIL simulation, and software integration not being the focus of the developers. According to the scale proposed in [19], they could be classified as high fidelity in terms of terramechanics and physical and terrain simulation, but as low to medium categories of visual fidelity level. The presented simulator aims at a high fidelity level, in terms of visualisation, and a medium fidelity level of physical simulation. Phenomena such as slip, inertia, gravity, collisions, traces left by the moving robot, etc., are simulated, so that robot traversal is realistic enough to test mapping and navigation algorithms. We do not model soil compacting or settlement, as it is not crucial for vision-based situational awareness. In a similar way, terramechanics simulators do not provide configurable, photorealistic rendering, hardware-in-the-loop or standard software integration capabilities, as they are not relevant to their focus area.

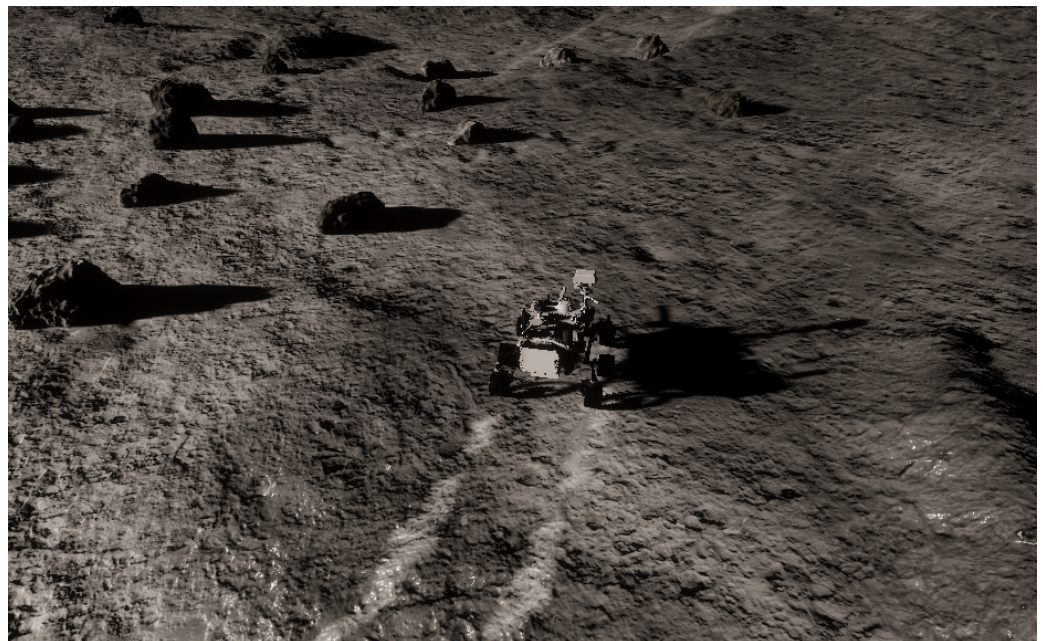


Figure 1. View from the LunarSim environment.

The simulator can generate any quantity of ground truth data for training and testing the onboard machine learning and deep learning image processing (segmentation, object

detection, depth maps) paired with realistic images. It can also support the development of vision-based navigation—visual odometry (VO) and simultaneous localisation and mapping (SLAM)—paving the way to increased AI-assisted rover autonomy. The simulator offers a range of desirable properties that are typically only partially present in other solutions.

The principal contributions of this paper are outlined as follows:

- Introduction of a simulator and toolkit achieving a high degree of rendered scene realism, which is demonstrated by improved performance of the segmentation neural network model, using generated synthetic data for transfer learning when compared to an alternative solution.
- Interface-level compatibility with ROS 2, an industry-standard robotics software platform enabling rapid development and testing of applications, which is shown by running and evaluating the visual odometry engine, using simulated data.
- Compatibility with a wide range of hardware and simulation scenarios, which is demonstrated using deep learning image analysis, sensor data processing, and navigation functions using real devices, ranging from embedded devices with integrated AI accelerators to complex system-on-chip devices, including custom, programmable hardware.

2. Related Work

The potential of simulation as a tool for aiding the development of space robots has long been recognised, with examples such as SpaceDyn [20], ROAMS [21], and 3DROV [22] paving the way for future, more advanced solutions. In their survey [23], Ding et al. gave an overview of simulators, almost a decade later. They noted the emergence of follow-up solutions, such as ROSTDyn [24] or the use of commercial tools for multibody simulation [25] integrated within more extensive toolsets [26]. A large part of the functionality of both of the aforementioned simulators is focused on accurately modelling the physical properties of the rover and the environment, such as rover–surface soil interactions, slip dynamics, thermal models, power system models, and characteristics of the rover’s frame, sensors, and actuators. The gathered information is very valuable: high-fidelity wheel–soil interaction and rover mobility simulations across various conditions have been crucial for developing successful motion control algorithms, facilitating extraterrestrial exploration in multiple missions.

With the first steps already taken, it was evident that future missions would require increased situational awareness and autonomy, including VO and SLAM. The feasibility of such an approach was proven in large-scale analogue missions [27]. Analogue missions are considered the most valuable source of information, with more recent examples being [28,29]. However, gathering real-world data is a time-consuming, laborious process, which is limited, in terms of the number of scenarios and control over physical world parameters. Aside from SLAM and VO, recent advances in machine learning for visual information processing have created the need for high-fidelity visual information generation in large volumes, preferably with automatically generated and associated ground truth. The ability to use widely adopted, standard software for control and broad hardware support is also welcome.

A game-engine-based simulator introduced by NASA [30] offers excellent visual fidelity in replicating lunar surface images, but focuses primarily on topography design tools, which lack machine learning support components. It is, however, compatible with an in-house tricklibrary for experiment simulation control and HIL capability. With the advent of Space ROS [15], compatibility with the ROS framework will be a significant advantage for robot rover simulators. The Gazebo simulator, typically paired with ROS [31], has been used successfully in various applications and provides seamless integration, but it was not designed with visual fidelity or machine learning support in mind. The research presented in [32] used Gazebo with custom extensions, to improve the image quality of the rendered scene. However, achieving an acceptable level of visual fidelity and performance required extensive modifications to the rendering engine, which were proprietary. The re-

cently proposed Martian terrain simulators [33,34] also used ROS and Gazebo, but were mostly directed at the realistic simulation of the rover and soil contacts and terra-dynamics. Modern simulation systems enable automated training data generation and integration with ROS: for example, Nvidia Isaac Sim [35], which enables accurate simulation with realistic visuals and ROS support. However, it lacks space environments (the existing environments are primarily focused on robotic manipulation and robot warehouses) and is a partially closed, company-developed solution that requires specific hardware and infrastructure. It is important to note that none of these simulators are specifically designed for the development of computer vision algorithms in the lunar environment.

Illustrative frames extracted from our simulator, juxtaposed with sample images generated by prior solutions, are shown in Figure 2.

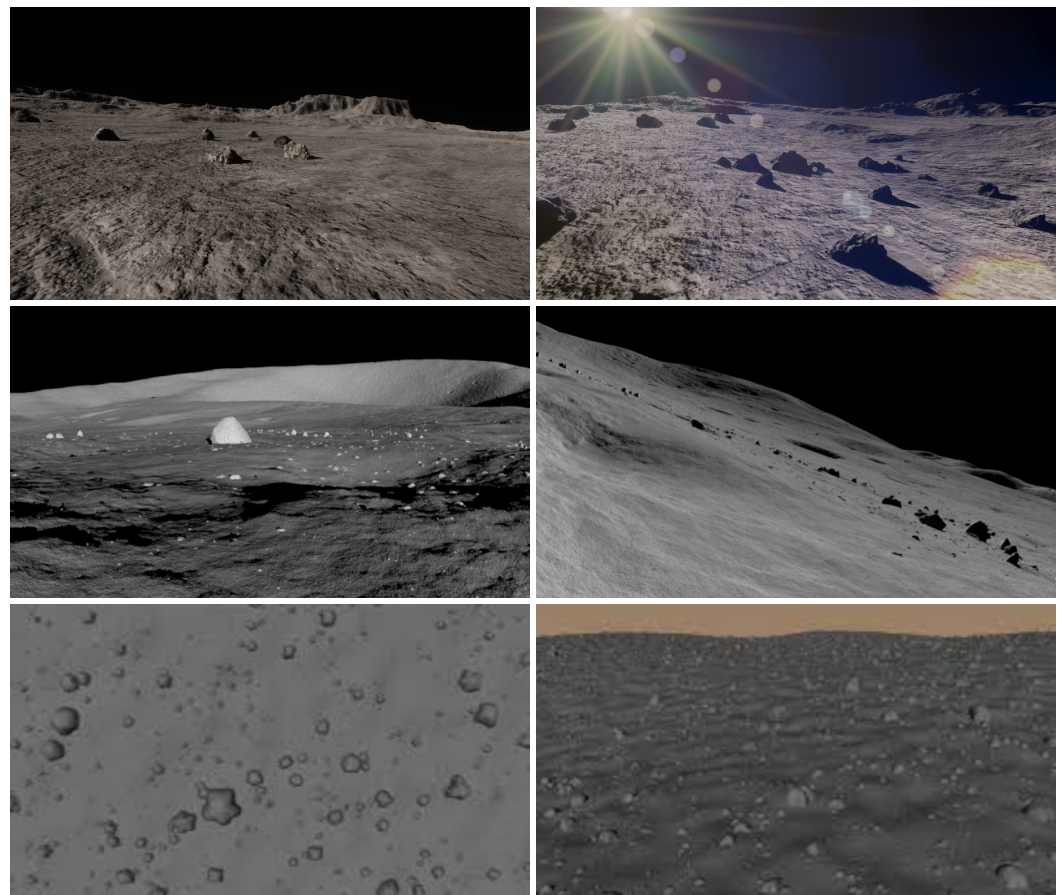


Figure 2. Example images from synthetic lunar and planetary environments. The proposed LunarSim (**top row**) generates more realistic images than those available in [36] (**middle row**) or [33] (**bottom row**) while also enabling ROS-compatible control and extensive hardware support, making it a viable solution for a broader range of applications than State-of-the-Art simulators.

3. Overview of the Solution

The presented simulator allows researchers to quickly bootstrap a lunar environment, to develop and test algorithms for autonomous rovers. This technology has the potential to significantly accelerate the development of lunar exploration systems by allowing researchers to generate synthetic data and efficiently test rover algorithms in a controlled environment. This is particularly relevant given the substantial costs and logistical complexities associated with real-world lunar missions or even analogue missions. Visual data also empower researchers to explore the potential impact of external factors on algorithmic performance. For instance, lighting conditions, visibility limitations, and variations in terrain textures can all influence the outcomes of the rover's perception and decision making algorithms. By manipulating these visual variables in the simulator, researchers can assess

the robustness of their algorithms and devise strategies to mitigate the adverse effects of real-world uncertainties. The block diagram of the complete system is shown in Figure 3.

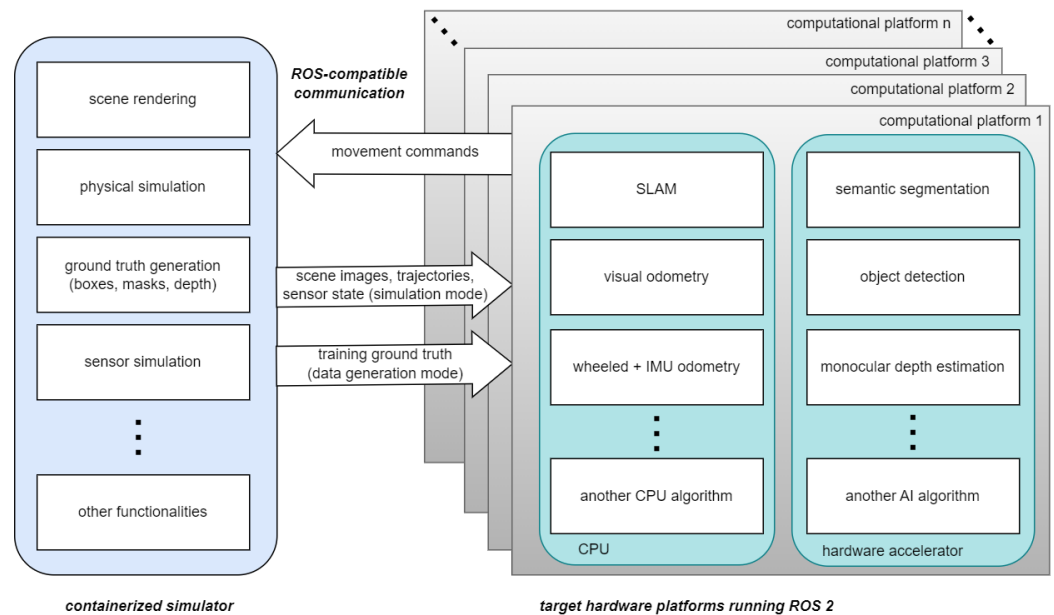


Figure 3. Block diagram of the implemented simulation solution.

To address concerns about the theoretical underpinning of the simulated lunar environment, our approach relies on the meticulous integration of empirical data and established lunar knowledge. The simulator's fidelity is achieved by incorporating satellite imagery and images from lunar missions. Moreover, to establish the simulator's theoretical validity, we have aligned its parameters with known lunar properties, such as gravitational conditions, illumination angles, and illumination temperature. The rover's physical attributes, including its size, weight, speed, and sensor capabilities, have been derived from real-world technologies and devices. These parameters were not only inspired by existing technology but also tuned to ensure that the use of the simulated rover would mirror sensors in the real world. This integration of established knowledge not only reinforces the simulator's authenticity but also provides a foundation for its representation of the lunar surface.

The Unity real-time development platform was selected as a simulation engine. The engine enables realistic scene rendering and accurate physical simulation out of the box. An artificial lunar landscape was built, using proprietary assets such as textures and models, as the game engine facilitates the use and import of models, objects, textures, and material data from various sources, both free and nonfree. The terrain contains some craters, hills, and flatlands, forming an area 950×950 m large. It is highly inspired by the Moon's surface but does not imitate any true part of the Moon. Similarly, rock models were created, textured, and manually placed in the terrain. A blending strategy was employed for the albedo (colour) maps and normal textures, making the integration of the lunar soil and other objects more natural. Additional post-processing effects, such as colour grading, ambient occlusion, contact shadows, and screen space reflections, were used, to further improve photorealism. It is worth noting that the scene's lighting can also be easily controlled, to simulate realistic effects like shadows, glare, lens flares, etc. The physical forces of the lunar environment are simulated and controlled by the NVIDIA PhysX engine [37]. The ability to freely model the lighting conditions is an important step in providing the necessary level of realism. Moreover, the ability to change the lighting-related parameters of the atmosphere and materials enables simulation in a range of realistic conditions, improving the robustness of the resulting solution. Dynamic lighting conditions and high dynamic range are significant problems in robot navigation [38]. Capturing environments under a range of lighting conditions is a crucial element of computer vision system evaluation in

many applications [39]. Introducing realistic textures is also an important step towards enabling robot situational awareness and navigation capabilities, using simulated input image data. Feature detection and tracking is a prerequisite and first step in almost all visual odometry and SLAM algorithms, and lack of texture might not only degrade the faithfulness of the simulation, it can also make testing the vision-based navigation algorithms impossible. Recent studies have highlighted the importance of textures in human perception and computer vision [40]. Moreover, the presence of sufficiently textured terrain has been highlighted as a necessary condition for Martian navigation [41].

In addition, numerous aspects of the simulation can be configured, using YAML configuration files. Specifically, various camera properties can be set to mimic real cameras, including focal length, sensor size, image resolution, noise, distortion, etc. The configuration also allows for switching simulation between the modes of generating data (Section 3.1) and ROS-connected (Section 3.2). The functionalities use built-in, easy-to-use, well-documented features of the utilised game engine, enabling easy, rapid changes and development. Having the configuration stored in files facilitates experimental result replication.

Finally, the simulator is containerised, using Docker software (ver. 24.0.5), and it is provided as a single image, to allow rapid use without the need to satisfy the dependencies, making it more accessible and portable for users. As shown in Figure 3, the containerised simulation is controlled by and capable of generating ROS-2-compatible signals, whose source or destination can be a variety of hardware platforms, which is demonstrated in detail by examples in Section 4. Containerisation also enables the lawful sharing of scenes and terrains containing not only free but also proprietary resources.

3.1. Dataset Generator Mode

The simulation can be configured to run in the data generation mode. A perception [42] library is used to capture the corresponding RGB and semantic segmentation frames with the information of the bounding boxes. Additionally, the simulation uses a simulated depth camera aligned with the left camera of the rover and saves depth information.

The semantic segmentation ground truth capture is configurable and can contain rocks, terrain, and rover parts masks. However, the bounding boxes are captured for rocks only. Both functions can be easily extended to process other objects of interest. All cameras simulate physical devices with configurable parameters. For the depth image, the clipping planes' settings are exposed, to enable the selection of depth range. This, however, can also be achieved when postprocessing the data. Another settable parameter is the registration frequency.

Moreover, the simulator's versatility is further enhanced by the incorporation of additional camera configurations, allowing it to generate essential visual ground-truth masks for intricate tasks, like bird's-eye-view (BEV) analysis. The capabilities of the simulator are showcased in Figure 4, in which a bird's eye view of the rover, with automatically generated semantic segmentation masks for the rocks and the rover, is presented. BEV algorithms for perception tasks are trending and drawing extensive attention both from industry and academia [43].

The data acquired using the simulator are synchronised and can later be used to train various machine learning algorithms. These include semantic scene segmentation, rock detection, monocular and stereo depth estimation, and optical flow or interest point location estimation. Additionally, we publish sample datasets, ready to use for the aforementioned tasks.

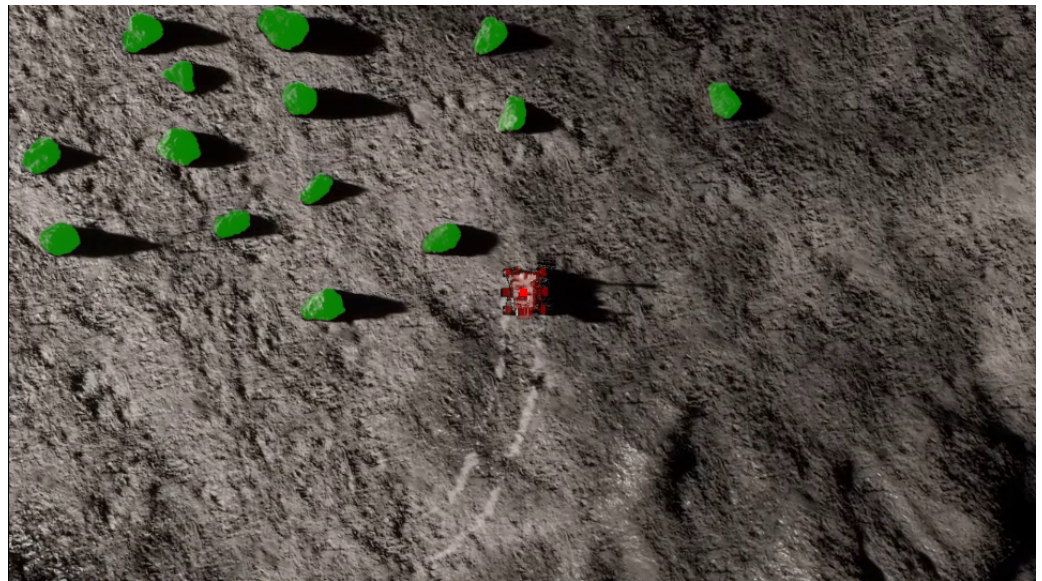


Figure 4. An example image with semantic labels for the bird's eye view (BEV) task.

3.2. Ros-2-Connected Mode

In the ROS-connected mode, the simulation does not save the cameras' data nor the ground-truth annotations to disk. Instead, ROS 2 for the Unity [44] library connects the application to the ROS system. It enables the simulator to receive and publish information simultaneously, emulating sensors and effectors from a real-world rover platform and accepting commands sent by ROS to control the rover.

ROS 2 provides a modular and distributed architecture that integrates different components and sensors, making it a flexible platform for building complex robotic systems. It offers a rich set of libraries, tools, and algorithms for various tasks, such as navigation, perception, manipulation, and planning, which can significantly reduce development time.

The simulation has been integrated with ROS 2 Humble, but there are currently no obstacles to upgrading it to newer versions in the future. This distribution was selected as the most recent version supporting modern Linux operating systems. It is also the basis of the recently proposed Space ROS system dedicated to space applications [15].

The rover accepts steering information as standard `geometry_msgs.msg.Twist` messages and, therefore, can be controlled using any capable ROS nodes. The publishing system sends the IMU (inertial measurement unit), stereo camera, depth sensor, and ground truth pose information with a configurable frequency, allowing the development and testing of perception and control algorithms. To make integration with other nodes straightforward, common ROS message types—such as `sensor_msgs.msg.Imu`, `sensor_msgs.msg.Image` or `sensor_msgs.msg.CompressedImage`, and `geometry_msgs.msg.PoseStamped`—are used. As mentioned before, the number, placement, and parameters of the sensors can be freely configured.

4. Real-World Use Cases with Sample Results

In the following section, we present examples of computer vision applications that can interact with the simulation. As shown in Figure 3, the simulator and ROS interface support a wide range of heterogeneous computational platforms. To demonstrate this capability, the applications were executed on efficient and low-power embedded hardware, facilitating the deployment of applications in resource-constrained environments, such as mobile and aerial robots. Please note that the presented results are not meant to set new standards in the presented use scenarios, but rather to comprehensively demonstrate the capabilities of the simulator on a broad range of target applications, using a broad range of hardware. The ability to integrate various commercial off-the-shelf (COTS) devices with the simulator aligns with the principles of the New Space paradigm [45]. The evolution of

this paradigm necessitates the creation of versatile tools capable of accommodating a wide range of hardware. The simulator can also easily be adapted to other devices, thanks to the use of industry-standard ROS interfaces.

We based this research on the artificial lunar landscape dataset (ALLD) [36]. It provides real lunar (ALLD (real)) and rendered (ALLD (synthetic)) images with ground truth segmentation masks. Similarly, we generated images from the simulator (our (synthetic)) and included a real subset of images collected at an analogue lunar research station (our (real))—the LunAres Research Station [46], which is a simulation facility for studying the challenges and requirements of space missions. The images were collected in various lighting conditions, using a camera mounted on the rover and directed 45 degrees down.

To test the simulator’s capability for rendering realistic images, a simple test was performed. An image descriptor, in the form of embeddings extracted by a convolutional neural network, was used to encode the images into vectors [47]. Generally, the closer the embedding vectors of two images lie in the embedding vector space, as measured by some metric, the more similar the images are. Using embeddings from single images would not be a good representation of a whole dataset, so after the embeddings were extracted, they were averaged, to represent all the images in a given dataset. The base MobileNetV2 model was used to generate the embedding vectors [48]. The model was pre-trained, using data from the ImageNet dataset [49]. The cosine distances of the averaged embeddings of the LunarSim and ALLD synthetic part to the average embedding of the ALLD real part were 0.347 and 0.360, respectively. This confirms that the LunarSim rendered images are a closer representation of the real images.

4.1. Object Detection—Embedded Device

Object detection is one of the fundamental tasks in the design of autonomous robots. It can provide information about “what objects are where”, providing a foundation for understanding the visual scene [50].

Although, given a well-performing scene segmentation algorithm, one can view object detection as redundant, it is often used alongside semantic segmentation, serving as a safety mechanism that provides results decoupled from the segmentation pipeline. Alternatively, it may be used to locate objects that are rare or small and, therefore, easily omitted by other approaches. Sometimes, object detection can also be used as a preprocessing step for semantic segmentation by first detecting the region of interest and then segmenting objects more accurately.

The presented simulator can be instructed to generate object detection annotations for any objects present in the scene, but it primarily generates them for rocks. This is mainly due to the lack of any significant objects of interest present in the lunar landscape. However, rock avoidance can be viewed as a crucial task for autonomous lunar rover operation or sample extraction. The size characteristics of the rocks are randomised, to generate diverse instances with varying sizes.

To demonstrate the accuracy and feasibility of the automatically generated object annotations, YOLOv7 [51] (tiny version) was trained, using simulation data. This specific algorithm was selected as it achieves State-of-the-Art results while maintaining an inference speed sufficient for on-board processing. Moreover, we quantised the model weights and benchmarked its accuracy and performance on the Edge AI device, Jetson Xavier NX, which is an embedded CPU with a GPU designed by Nvidia for edge computing devices. It is small, lightweight, energy-efficient, and equipped with software tools like TensorRT that make it easy to develop and deploy AI applications—in particular, based on convolutional neural networks (CNN) and computer vision (CV). Performed tests include floating-point, half-floating-point, and eight-bit integer quantisation done through minmax scaling. The most widely used metric, mean average precision (mAP), was applied. The results are shown in Table 1. Then, an ROS-based application was developed. It uses a colour image (/lunarsim/depth_camera/depth topic) to detect rocks and depth information (/lunarsim/left_camera/compressed topic), to estimate their real world size.

Table 1. Performance and speed comparison with different weights representation precision. YOLOv7 was used as an example object detection model for our synthetic data. Jetson NX was chosen as the reference platform.

Precision	mAP@0.5	Speed [img/s]
FP-32	0.424	40.27
FP-16	0.432	44.62
INT-8	0.411	48.96

4.2. Image Segmentation—FPGA Platform

Field programmable gate arrays (FPGAs) are a common computational platform choice for space robotics because they offer several key advantages over traditional microprocessors, including high performance and relatively low power consumption. They have already been used in space missions and, therefore, radiation-resistant variants are available [52]. The possibilities for using FPGAs in artificial intelligence have broadened with the availability of tools that allow for the quantisation and compilation of deep learning models.

“Identifying and avoiding terrain hazards is crucial for the safety of planetary rovers. [...] The rover must avoid driving over any potential hazards to its wheels, particularly angular embedded rocks”, as described in [53]. Therefore, the key task is terrain analysis, which can be defined as pixel-wise image segmentation. One of the obstacles that occur in the lunar environment is rocks. To demonstrate that our synthetic data are helpful and that the simulator provides meaningful, measurable aid in application development, we cross-validate datasets. We use synthetic data from LunarSum and ALLR to train the neural networks and real ALLD lunar data to validate them. For this purpose, we use the UNet [54] segmentation network, which was created with a resnet10t backend [55] and trained with recent improvements, such as the cosine learning rate schedule and data augmentation. The UNet architecture is a well-known neural network framework, designed to assign class membership to individual pixels within an image. Specifically, we opted for the UNet implementation, which adopts an encoder–decoder structure and leverages skip connections to facilitate learning of spatial correlations across corresponding encoding stages, as detailed in [56].

Using the synthetic data generated using the described simulator resulted in a better F1 score than in the case when the synthetic data was taken from the ALLD dataset. This indicates that the generated synthetic data more closely resembled real-life conditions. This was even more evident when both parts of the real life data set were used for testing, as the difference in the F1 score was even greater.

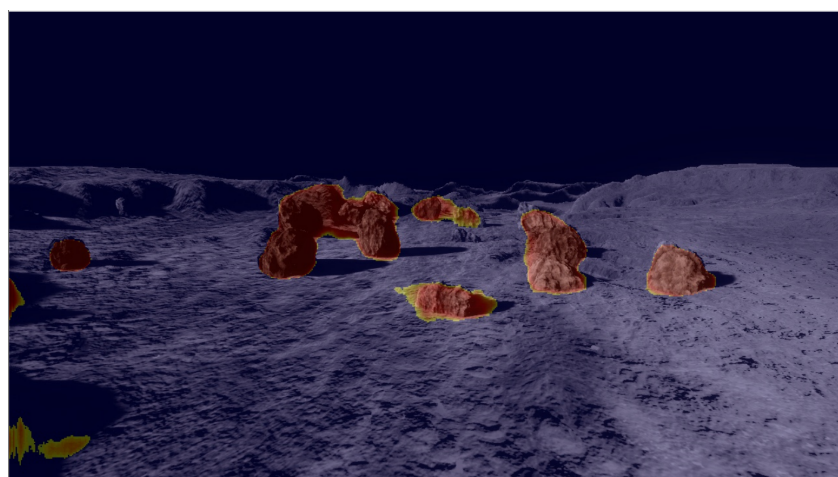
In the same way, as in [57], we used the Xilinx VCK190 developer kit to perform deep learning inference. We quantised and compiled the models trained using our synthetic data. After quantisation, the models reached the F1 score values of 0.459 and 0.541 for ALLD (real) and ALLD (real)+ our (real). Comparison between data set combinations (Table 2) shows improvement in performance on real images with our synthetic data being used for training. Example predictions of the network in real images are presented in Figure 6a (from the ALLD dataset) and Figure 6b (from the analogue lunar research station). Next, we built an ROS 2 node running on VCK190 that performed inference on the input images (/lunarsim/left_camera/compressed topic) and published a blended image with a heat map overlay for the presence of rocks (Figure 6c).



(a)



(b)



(c)

Figure 6. A result of rocks segmentation performed by VCK190 developer kit: (a) image from the ALLD (real image part) dataset; (b) image from the analogue lunar research station; (c) image from our synthetic dataset.

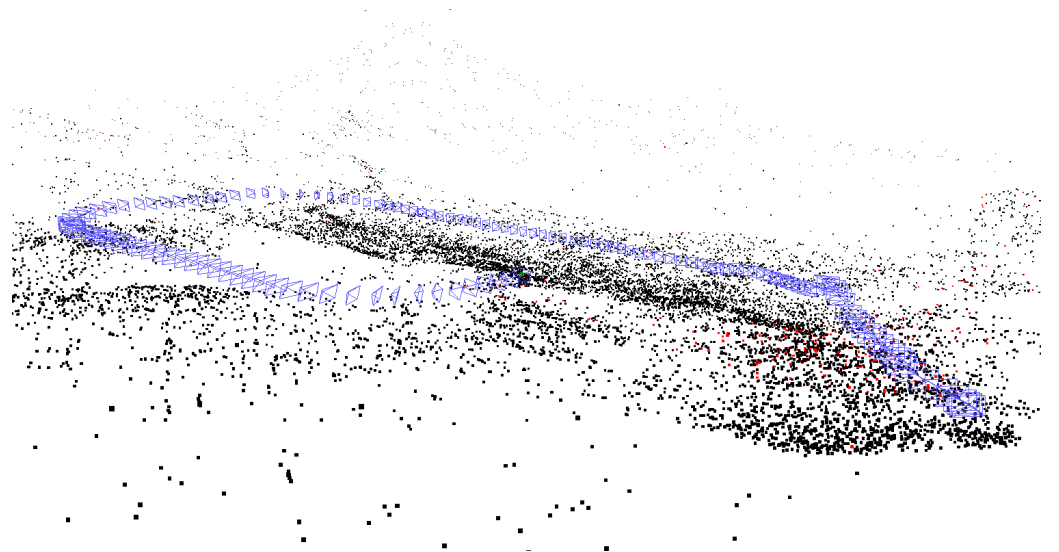
Table 2. Comparison of model accuracy depending on the training and test data set used.

Train Dataset	Test Dataset	F1-Score
ALLD (synthetic)	ALLD (real)	0.431
ALLD (synthetic)	ALLD (real) + our (real)	0.428
Ours (synthetic)	ALLD (real)	0.449
Ours (synthetic)	ALLD (real) + our (real)	0.537

4.3. Visual Odometry—x64-Based Mini-PC

Accurate localisation of mobile robots (rovers) is one of the most challenging tasks—especially when the target environment is unknown to a large extent. In the case of space applications, it is even more challenging, as the appearance of exoplanetary environments is notably different to Earth environments, making appearance-based navigation implementation and testing significantly more difficult. Visual odometry (VO) enables self-localisation based on a series of images captured by the robot’s onboard camera. It is a relatively low-cost solution compared to other localisation techniques that rely on expensive sensors or external infrastructure. It is lightweight and computationally efficient, making it suitable for real-time applications on mobile robots with limited processing power.

To demonstrate the cooperation between our simulator and visual odometry, the Open-VSLAM framework was used [58]. This framework is compatible with ROS 2, utilises the fast ORB feature extraction, and supports mono, stereo, and depth cameras. In our experiments, a stereovision-based approach was selected (/lunarsim/left_camera/raw and /lunarsim/right_camera/raw topics). The trajectory of movement was generated using a visual odometry algorithm and was compared to the ground-truth trajectory (/lunarsim/gt/pose topic) provided by the simulator (Figure 7). The calculated trajectory reached a mean error of 0.55 m and a maximal error of 1.26 m at the testing distance of 106.31 m. This outcome signified the ability of the VO algorithm to accurately reconstruct the trajectory, even in the face of challenging environmental conditions and a limited number of features, due to the significant self-similarity of the environment. Furthermore, the level of error implied that the simulation environment was just as demanding as real-world outdoor scenarios [58]. This simple proof of concept illustrates the simulator’s capability for running vision-based navigation algorithms and using the generated ground truth information to evaluate their accuracy.

**Figure 7.** Cont.

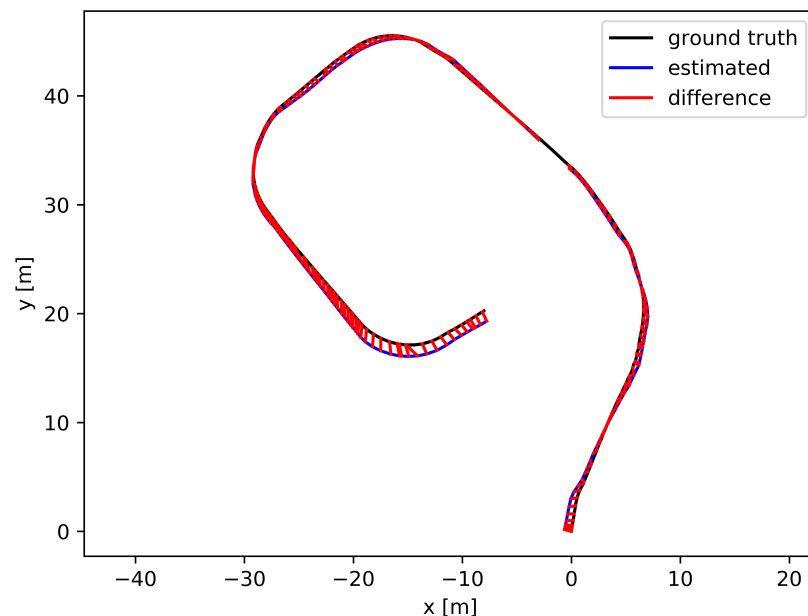


Figure 7. Trajectory of rover calculated by visual odometry (**top**) and its visual comparison to ground truth (**bottom**).

5. Conclusions

In this article, we presented a new space rover simulator and toolkit, achieving a high degree of rendered scene realism, ease of use through the adoption of a popular game engine, and familiar, industry-standard ROS interface compatibility, with integrated support for a wide range of heterogeneous computational hardware. The presented solution allows researchers to develop and test perception and control algorithms for autonomous lunar and exoplanetary rovers based on vision sensors. At its core, it is integrated with ROS 2 while providing superior visual fidelity when compared to other existing approaches. Two modes of operation allow for synthetic data generation and software integration testing. The configuration file can be used to modify the simulation parameters to meet specific usage scenarios, enabling automated testing across a range of parameters, along with easy experiment versioning and replication. Integration with ROS 2 allows for testing a rich set of existing algorithms in exploration scenarios involving broadly understood perception, navigation, and motion planning, using the newest methods based on deep learning.

The simulator displayed favourable performance, in terms of generated image fidelity. It achieved results that surpassed those achieved with the most popular synthetic lunar rock dataset in the transfer learning experiment, in which the test set consisted of real lunar and analogue mission data, confirming its usefulness using objective evaluation. The simulator-generated data were also successfully used for evaluating the object detection and visual odometry algorithms. All the experiments were performed using a range of various computational platforms, including conventional microcomputers and embedded devices with GPU AI accelerators, as well as complex system-on-chip devices with programmable hardware. This confirmed the simulator's usefulness, in terms of testing using real mission hardware. The simulator is flexible enough to accommodate other devices.

The solution consists of modules based on standardised components and, as such, has very good potential for expansion, in terms of the range of simulated sensors, environments, and other functions or integration as a part of more complex test setups. Overall, the presented simulator significantly extends the capabilities for replicating critical components of space robotics applications in perception and autonomy. The simulator can easily be extended, due to its modular nature, providing support for other hardware, sensors,

and terrestrial and extraterrestrial environments. The visual fidelity can also further be extended by using more realistic textures and object models, as well as by the application of domain transfer methods [59]. The authors hope that the presented suite of software will pave the way for further advancements in the field of space robotics and beyond.

Author Contributions: Conceptualisation, D.P., B.P. and M.K.; methodology, D.P., B.P. and M.K.; software, D.P. and B.P.; validation, D.P. and B.P.; formal analysis, D.P. and B.P.; investigation, D.P. and B.P.; writing—original draft preparation, D.P., B.P., M.K. and P.D.; writing—review and editing, D.P., B.P., M.K. and P.D.; visualisation, D.P. and B.P.; supervision, M.K. and P.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Polish Ministry of Education and Science under Grant 0214/SBAD/0244.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <https://github.com/PUTvision/LunarSim> (accessed on 15 November 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jones, H. The recent large reduction in space launch cost. In Proceedings of the 48th International Conference on Environmental Systems, Albuquerque, NM, USA, 8–12 July 2018.
2. Denis, G.; Alary, D.; Pasco, X.; Pisot, N.; Texier, D.; Toulza, S. From new space to big space: How commercial space dream is becoming a reality. *Acta Astronaut.* **2020**, *166*, 431–443. [\[CrossRef\]](#)
3. Furano, G.; Tavoularis, A.; Rovatti, M. AI in space: Applications examples and challenges. In Proceedings of the 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Frascati, Italy, 19–21 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
4. Ghiglione, M.; Serra, V. Opportunities and challenges of AI on satellite processing units. In Proceedings of the 19th ACM International Conference on Computing Frontiers, Turin, Italy, 17–22 May 2022; pp. 221–224.
5. Li, D.; Zhong, L.; Zhu, W.; Xu, Z.; Tang, Q.; Zhan, W. A survey of space robotic technologies for on-orbit assembly. *Space Sci. Technol.* **2022**, *2022*, 9849170. [\[CrossRef\]](#)
6. Nesnas, I.A.; Fesq, L.M.; Volpe, R.A. Autonomy for space robots: Past, present, and future. *Curr. Robot. Rep.* **2021**, *2*, 251–263. [\[CrossRef\]](#)
7. Almeshini, R.; Garg, P.; Wali, M. Thermal Management System Design and Analysis of Rashid Rover—Emirates Lunar Mission. In Proceedings of the International Astronautical Congress (IAC), Dubai, United Arab Emirates, 25–29 October 2021; pp. 25–29.
8. Creech, S.; Guidi, J.; Elburn, D. Artemis: An overview of NASA's activities to return humans to the moon. In Proceedings of the 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–7.
9. Ding, L.; Zhou, R.; Yuan, Y.; Yang, H.; Li, J.; Yu, T.; Liu, C.; Wang, J.; Li, S.; Gao, H.; et al. A 2-year locomotive exploration and scientific investigation of the lunar farside by the Yutu-2 rover. *Sci. Robot.* **2022**, *7*, eabj6660. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Karanam, D.P.; Bhatt, M.; Sathyan, S.; Misra, D.; Srivastava, N.; Bhardwaj, A. Contextual Characterisation Study of Chandrayaan-3 Primary Landing Site. *Mon. Not. R. Astron. Soc. Lett.* **2023**, *526*, L116–L123. [\[CrossRef\]](#)
11. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Kothari, V.; Liberis, E.; Lane, N.D. The final frontier: Deep learning in space. In Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications, Austin, TX, USA, 3 March 2020; pp. 45–49.
13. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
14. Badger, J.; Gooding, D.; Ensley, K.; Hambuchen, K.; Thackston, A. ROS in space: A case study on Robonaut 2. In *Robot Operating System (ROS) The Complete Reference (Volume 1)*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 343–373.
15. Probe, A.; Oyake, A.; Chambers, S.W.; Deans, M.; Brat, G.; Cramer, N.B.; Roberts, B.; Hambuchen, K. Space ROS: An Open-Source Framework for Space Robotics and Flight Software. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA, 23–27 January 2023; p. 2709.
16. Peterson, M.; Du, M.; Springle, B.; Black, J. Comprehensive Assessment of Orbital Robotics, Space Application Simulation/Machine Learning, and Methods of Hardware in the Loop Validation. In Proceedings of the 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–13.
17. Open Robotics. Robotic Operating System. Available online: <https://www.ros.org> (accessed on 13 November 2022).

18. Belsare, K.; Rodriguez, A.C.; Sánchez, P.G.; Hierro, J.; Kołcon, T.; Lange, R.; Lütkebohle, I.; Malki, A.; Losa, J.M.; Melendez, F.; et al. Micro-ROS. In *Robot Operating System (ROS) The Complete Reference (Volume 7)*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 3–55.
19. Jones, D.; Snider, C.; Nassehi, A.; Yon, J.; Hicks, B. Characterising the Digital Twin: A systematic literature review. *CIRP J. Manuf. Sci. Technol.* **2020**, *29*, 36–52. [\[CrossRef\]](#)
20. Yoshida, K. The SpaceDyn: A MATLAB toolbox for space and mobile robots. In Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289), Kyongju, Republic of Korea, 17–21 October 1999; IEEE: Piscataway, NJ, USA, 1999; Volume 3, pp. 1633–1638.
21. Jain, A.; Guineau, J.; Lim, C.; Lincoln, W.; Pomerantz, M.; Sohl, G.T.; Steele, R. *ROAMS: Planetary Surface Rover Simulation Environment*; California Institute of Technology: Pasadena, CA, USA, 2003.
22. Poulakis, P.; Joudrier, L.; Wailliez, S.; Kappellos, K. 3DROV: A planetary rover system design, simulation and verification tool. In Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08), Los Angeles, CA, USA, 26–29 February 2008.
23. Ding, L.; Gao, H.; Deng, Z.; Li, W. Advances in simulation of planetary wheeled mobile robots. In *Mobile Robots-Current Trends; Books on Demand: Schleswig-Holstein, Germany*, 2011; pp. 375–402.
24. Li, W.; Ding, L.; Gao, H.; Deng, Z.; Li, N. ROSTDyn: Rover simulation based on terramechanics and dynamics. *J. Terramech.* **2013**, *50*, 199–210. [\[CrossRef\]](#)
25. Rulka, W. SIMPACK—A computer program for simulation of large-motion multibody systems. In *Multibody Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 265–284.
26. Schäfer, B.; Gibbesch, A.; Krenn, R.; Rebele, B. Planetary rover mobility simulation on soft and uneven terrain. *Veh. Syst. Dyn.* **2010**, *48*, 149–169. [\[CrossRef\]](#)
27. Woods, M.; Shaw, A.; Tidey, E.; Van Pham, B.; Simon, L.; Mukherji, R.; Maddison, B.; Cross, G.; Kisdi, A.; Tubby, W.; et al. SEEKER—autonomous long-range rover navigation for remote exploration. *J. Field Robot.* **2014**, *31*, 940–968. [\[CrossRef\]](#)
28. Geromichalos, D.; Azkarate, M.; Tsardoulas, E.; Gerdes, L.; Petrou, L.; Del Pulgar, C.P. SLAM for autonomous planetary rovers with global localization. *J. Field Robot.* **2020**, *37*, 830–847. [\[CrossRef\]](#)
29. Meyer, L.; Smíšek, M.; Fontan Villacampa, A.; Oliva Maza, L.; Medina, D.; Schuster, M.J.; Steidle, F.; Vayugundla, M.; Müller, M.G.; Rebele, B.; et al. The MADMAX data set for visual-inertial rover navigation on Mars. *J. Field Robot.* **2021**, *38*, 833–853. [\[CrossRef\]](#)
30. Bingham, L.; Kincaid, J.; Weno, B.; Davis, N.; Paddock, E.; Foreman, C. Digital Lunar Exploration Sites Unreal Simulation Tool (DUST). In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2023; IEEE: Piscataway, NJ, USA, 2023.
31. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 3, pp. 2149–2154.
32. Allan, M.; Wong, U.; Furlong, P.M.; Rogg, A.; McMichael, S.; Welsh, T.; Chen, I.; Peters, S.; Gerkey, B.; Quigley, M.; et al. Planetary rover simulation for lunar exploration missions. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–19.
33. Toupet, O.; Del Sesto, T.; Ono, M.; Myint, S.; Vander Hook, J.; McHenry, M. A ROS-based simulator for testing the enhanced autonomous navigation of the mars 2020 rover. In Proceedings of the 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–11.
34. Zhou, R.; Feng, W.; Ding, L.; Yang, H.; Gao, H.; Liu, G.; Deng, Z. MarsSim: A high-fidelity physical and visual simulation for Mars rovers. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *59*, 1879–1892. [\[CrossRef\]](#)
35. Nvidia. Isaac Sim. Available online: <https://developer.nvidia.com/isaac-sim> (accessed on 21 February 2023).
36. Pessia, R.; Ishigami, G. Artificial Lunar Landscape Dataset. Kaggle. 2019. Available online: <https://www.kaggle.com/datasets/romainpessia/artificial-lunar-rocky-landscape-dataset> (accessed on 13 November 2023).
37. NVIDIA. PhysX 5.0 SDK. Available online: <https://developer.nvidia.com/physx-sdk> (accessed on 28 August 2023).
38. Le Cras, J.; Paxman, J.; Saracik, B. Improving robustness of vision based localization under dynamic illumination. In *Recent Advances in Robotics and Automation*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 155–170.
39. Murmann, L.; Gharbi, M.; Aittala, M.; Durand, F. A dataset of multi-illumination images in the wild. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4080–4089.
40. Ge, Y.; Xiao, Y.; Xu, Z.; Wang, X.; Itti, L. Contributions of shape, texture, and color in visual recognition. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 369–386.
41. Matthies, L.; Maimone, M.; Johnson, A.; Cheng, Y.; Willson, R.; Villalpando, C.; Goldberg, S.; Huertas, A.; Stein, A.; Angelova, A. Computer vision on Mars. *Int. J. Comput. Vis.* **2007**, *75*, 67–92. [\[CrossRef\]](#)
42. Unity Technologies. Unity Perception Package. 2020. Available online: <https://docs.unity3d.com/Packages/com.unity.perception@1.0/manual/index.html> (accessed on 13 November 2023).
43. Li, H.; Sima, C.; Dai, J.; Wang, W.; Lu, L.; Wang, H.; Xie, E.; Li, Z.; Deng, H.; Tian, H.; et al. Delving into the Devils of Bird’s-eye-view Perception: A Review, Evaluation and Recipe. *arXiv* **2022**, arXiv:2209.05324.
44. RobotecAI. Ros2-for-unity: High-performance ROS2 solution for Unity3D. *Appl. Sci.* **2023**, *13*, 1292.

45. LaBel, K.A.; Sampson, M.J.; Pellish, J.A. Electrical, electronic and electromechanical (EEE) parts in the new space paradigm: When is better the enemy of good enough? In Proceedings of the International School on the Effects of Radiation on Embedded Systems for Space Applications (SERESSA 2018), Noordwijk, The Netherlands, 12–16 November 2018.
46. Mintus, A.; Orzechowski, L.; Ćwilichowska, N. LunAres Analog Research Station—Overview of updated design and research potential. *Acta Astronaut.* **2022**, *193*, 785–794. [[CrossRef](#)]
47. Hertel, L.; Barth, E.; Käster, T.; Martinetz, T. Deep convolutional neural networks as generic feature extractors. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–4.
48. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 19–23 June 2018; pp. 4510–4520.
49. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 248–255.
50. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [[CrossRef](#)]
51. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
52. Xilinx, Inc. *RT Kintex UltraScale FPGAs for Ultra High Throughput and High Bandwidth Applications*; Technical Report; Xilinx: San Jose, CA, USA, 2020;
53. Ono, M.; Fuchs, T.J.; Steffy, A.; Maimone, M.; Yen, J. Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In Proceedings of the 2015 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–10.
54. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
55. Wightman, R.; Touvron, H.; Jégou, H. Resnet strikes back: An improved training procedure in timm. *arXiv* **2021**, arXiv:2110.00476.
56. He, X.; Zhou, Y.; Zhao, J.; Zhang, D.; Yao, R.; Xue, Y. Swin transformer embedding UNet for remote sensing image semantic segmentation. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–15. [[CrossRef](#)]
57. Ibrahim, Y.; Chen, L.; Haonan, T. Deep Learning-based Ship Detection on FPGAs. In Proceedings of the 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 4–6 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 454–459.
58. Sumikura, S.; Shibuya, M.; Sakurada, K. OpenVSLAM: A Versatile Visual SLAM Framework. In Proceedings of the 27th ACM International Conference on Multimedia, ACM, 2019, MM '19, Nice, France, 21–25 October 2019; pp. 2292–2295.
59. Zhang, L.; Gao, X. Transfer adaptation learning: A decade survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.