Semantic Segmentation and Depth Estimation for Real-Time Lunar Surface Mapping Using 3D Gaussian Splatting

Guillem Casadesus Vila, Adam Dai, and Grace Gao
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA, United States
{guillemc,adai,gracegao}@stanford.edu

***Abstract*— Navigation and mapping on the lunar surface require robust perception under challenging conditions, including poorly textured environments, high-contrast lighting, and limited computational resources. This paper presents a real-time mapping framework that integrates dense perception models with a 3D Gaussian Splatting (3DGS) representation. We first benchmark a suite of models on the synthetic LuSNAR dataset, selecting RAFT-Stereo for its balance of speed and accuracy in depth estimation, and U-Net++ for its superior performance in detecting hazardous rocks. Using ground truth poses, our pipeline reconstructs a 500-meter traverse with a geometric height accuracy of approximately 10 cm, achieving a quality comparable to a traditional point cloud baseline. The resulting 3DGS map enables high-fidelity novel view synthesis and serves as a foundation for a full SLAM system, where its capacity for joint map and pose optimization would offer significant advantages. Our results demonstrate that combining dense perception with explicit neural representations is an effective approach for creating detailed, large-scale maps to support future lunar rover missions.**

TABLE OF CONTENTS

1. INTRODUCTION

Motivation

The renewed focus on robotic exploration of the lunar surface brings new challenges for autonomous navigation and mapping [1]. Long-range traverses are required in poorly textured environments, and high-contrast and dynamic lighting conditions, as shown in Fig. 1. Furthermore, rovers are often limited to radiation-hardened hardware and low-power sensors, priorizing cameras over higher-resolution alternatives like LiDAR. These constraints require the development of computationally efficient, vision-based mapping and localization solutions.

Simultaneous Localization and Mapping (SLAM) is a fundamental technique for an agent to construct a map of an unknown environment while concurrently tracking its position within that map. Classical SLAM methods, such as those based on geometric features [2, 3], can operate in real time but exhibit reduced performance in poorly textured and high-contrast lighting conditions. While learning-based SLAM methods can improve robustness by learning feature representations from data, their generalization to novel environments like the lunar surface is not guaranteed. A common limitation of both classical and early learning-based methods is their reliance on discrete map representations (e.g., point clouds, voxels), which can be memory-intensive and may not capture fine surface detail.

Recent advances in neural scene representations have produced methods capable of high-fidelity 3D reconstruction from images. For terrestrial applications, recent work has proposed a variety of approaches using different sensor inputs (e.g., monocular, stereo, depth, IMU) and scene encodings such as Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) [5]. These systems often rely on perception models for tasks like feature extraction, depth estimation, or semantic understanding. However, since these perception components are typically trained on terrestrial data, it is not clear whether their performance will generalize to the unique visual characteristics of extraterrestrial environments.

Related Work

Semantic characterization of extraterrestrial scenes is an active area of research. A systematic literature review on semantic terrain segmentation for planetary rovers [6] highlights a gap in existing solutions, noting that no current method simultaneously achieves pixel-level accuracy, real-time inference, and compatibility with onboard hardware. Specific models have shown promise for feature identification, such as detecting impact craters from digital elevation models [7]. Despite these advances in perception, the integration of such semantic information into modern neural mapping frameworks for planetary surfaces has not been thoroughly investigated.

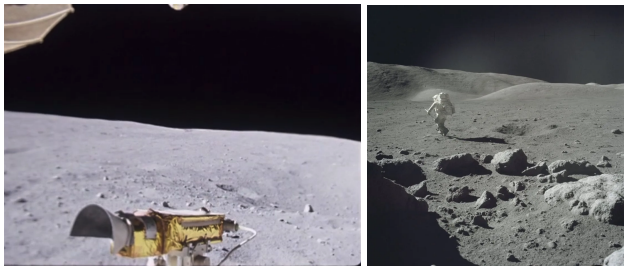


Figure 1: **Images from the Apollo missions [4].**

979-8-3315-7360-7/26/$31.00 ©2026 IEEE

Several works have applied neural scene representations to model the lunar surface. Some have focused on surface reconstruction from orbital imagery to generate digital elevation models, often to handle challenging illumination in permanently shadowed regions [8, 9]. More relevant to rover navigation, a few recent studies have explored using NeRFs with surface-level imagery for localization, mapping, and path planning [10–13].

While these approaches demonstrate the potential of neural representations for capturing lunar terrain, they mostly rely on NeRF [14], which has several limitations for real-time mapping. First, NeRF's rendering process is computationally expensive due to its reliance on volumetric sampling, making it ill-suited for real-time applications on resource-constrained hardware. Second, the underlying multi-layer perceptron (MLP) represents a fixed volume, which is difficult to extend as the rover explores new areas and is not easily deformable to accommodate loop closures. This necessitates stitching multiple models, which can introduce inconsistencies. Finally, the implicit nature of NeRF makes it less interpretable and difficult to integrate with or query for explicit semantic information. Many of these works also assume diverse camera viewpoints during training, a condition not met by typical rover trajectories.

Contributions

This work addresses the aforementioned limitations by developing a framework for building semantic, large-scale maps of the lunar surface in real time. Our contributions are:

- We propose a real-time mapping framework for lunar surface environments based on 3DGS, which offers fast rendering and a flexible, explicit scene representation suitable for incremental updates.
- We integrate and benchmark a several perception models, including semantic segmentation and both stereo and monocular depth estimation networks. We analyze the direct impact of these perception inputs on the geometric and semantic quality of the final 3D reconstruction.

Paper Organization

The rest of the paper is organized as follows. Section 2 describes the selected dataset and details the perception models used for monocular depth estimation, stereo depth estimation, and semantic segmentation, as well as the fundamentals of 3D Gaussian Splatting. Section 3 presents our proposed real-time mapping framework, outlining the perception front-end, the incremental mapping process, and the optimization back-end, including the loss function. Section 4 defines the evaluation metrics and presents a quantitative and qualitative analysis of the dense depth estimation, semantic segmentation, and final surface reconstruction results. Finally, Section 5 concludes the paper with a summary of our findings and a discussion of future work.

2. Datasets and Models

This section details the datasets and models that form the basis of our mapping framework. We first introduce the simulated lunar dataset used for evaluation. Following this, we provide a comprehensive overview of various methods for two key perception tasks: dense depth estimation—-from both monocular and stereo inputs—-and semantic segmentation. The primary objective is to benchmark these models to characterize their performance within a lunar context and determine their suitability for the downstream task of real-time 3D reconstruction. Finally, we describe the fundamentals of 3D Gaussian Splatting, the scene representation used in our mapping pipeline.

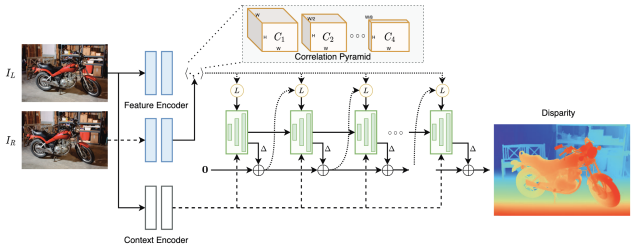


Figure 2: **RAFT-Stereo architecture (adapted from [17]).**

Dataset

We use the Lunar Segmentation, Navigation, and Reconstruction (LuSNAR) dataset, a comprehensive lunar benchmark dataset designed for evaluating autonomous perception and navigation systems [15]. It contains high-resolution stereo image pairs, semantic labels, dense depth maps, LiDAR point clouds, and rover position data. The dataset consists of 9 simulated lunar scenes created using Unreal Engine, each varying in terrain features and object density. These scenes are designed to represent different lunar surface conditions, from flat plains to cratered regions with varying rock distributions. The dataset enables the evaluation of key perception tasks, including semantic segmentation, 3D reconstruction, and autonomous navigation, making it a valuable resource for testing and validating lunar surface algorithms. However, it has some limitations: it does not provide full 3D geometry information, lacks closed-loop simulation capabilities, and has limited control over illumination conditions. We are currently working on extending this study to include other datasets.

Stereo Depth Estimation

We evaluate the following stereo depth estimation models:

- **Block Matching (BM)**: Computes disparity by comparing fixed-size image patches along epipolar lines using a cost function, such as Sum of Absolute Differences (SAD). Its performance is limited in textureless regions and under non-ideal illumination.
- **Semi-Global Matching (SGM)** [16]: Aggregates matching costs along multiple 1D paths across the image to approximate a 2D global smoothness constraint. This approach combines the efficiency of local methods with improved performance in low-texture areas.
- **RAFT-Stereo** [17]: Adapts the RAFT architecture from optical flow by constructing a 3D correlation volume of all disparities at all pixels. A recurrent, GRU-based unit then iteratively updates a high-resolution disparity field from this volume.
- **CREStereo** [18]: A cascaded recurrent network that operates in a coarse-to-fine manner. It uses recurrent refinement units at each stage and an adaptive group correlation layer to handle large displacements between stereo images.

Monocular Depth Estimation

We evaluate the following monocular dense depth estimation models:

• **Depth Anything V2** [19]: A transformer-based model trained on a dataset of over 62 million synthetic and pseudo-labeled real images. It is released in variants from 25M to 1.3B parameters for zero-shot relative depth estimation.

• **GLPN** [20]: An architecture that uses a transformer-based encoder to capture global context and a lightweight decoder. A selective feature fusion module combines features from different stages of the encoder.

• **DPT** [21]: Uses a Vision Transformer (ViT) as its backbone for dense prediction tasks. It processes feature maps at a constant resolution, providing global receptive fields at every stage of the network.

• **Depth Pro** [22]: A model for zero-shot metric depth estimation that uses a multi-scale vision transformer and is designed to operate on high-resolution inputs (e.g., 1536x1536).

Semantic Segmentation

We evaluate the following semantic segmentation models:

• **U-Net** [23]: An encoder-decoder architecture with skip connections that concatenate features from the downsampling path to the upsampling path to retain spatial information.

• **U-Net++** [24]: Modifies the U-Net skip pathways with nested and dense convolutional blocks to reduce the semantic gap between encoder and decoder features.

• **MA-Net** [25]: Augments an encoder-decoder network with a module that combines position-wise and channel-wise attention to capture contextual dependencies.

• **LinkNet** [26]: A lightweight encoder-decoder network for real-time applications that passes encoder features at each level directly to the corresponding decoder level.

• **FPN** [27]: Constructs a multi-scale feature pyramid using a top-down pathway and lateral connections to merge semantic information from deep layers with spatial information from shallow layers.

• **PSPNet** [28]: Introduces a pyramid pooling module that applies pooling operations at multiple scales to aggregate global context information.

• **PAN (Path Aggregation Network)** [29]: Augments the top-down feature pyramid with an additional bottom-up pathway to shorten the information path for low-level features.

• **DeepLabV3** [30]: Uses atrous (dilated) convolutions to control the spatial resolution of feature maps and an Atrous Spatial Pyramid Pooling (ASPP) module to probe features at multiple scales.

• **DeepLabV3+** [31]: Extends DeepLabV3 by adding a decoder module to refine object boundaries and uses depthwise separable convolutions for computational efficiency.

• **UPerNet** [32]: A unified framework that combines a Feature Pyramid Network (FPN) backbone with a pyramid pooling module to parse features at various scales simultaneously.

• **Segformer** [33]: A transformer-based model using a hierarchical transformer encoder to produce multi-scale features without positional encodings, combined with a lightweight multilayer perceptron (MLP) decoder.

• **DPT** [21]: Uses a Vision Transformer (ViT) as its backbone, providing global receptive fields at every stage of the feature extraction process for dense prediction tasks.

3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [34] is a rasterization-based method for novel view synthesis that represents a 3D scene with a collection of explicit, optimizable primitives. Unlike implicit representations like NeRF [14], 3DGS uses thousands to millions of 3D Gaussians to explicitly model the scene's geometry and appearance, as illustrated in Fig. 3.

Each Gaussian is defined by a set of learnable parameters:

• **Mean:** $\boldsymbol{\mu} \in \mathbb{R}^3$ determines its location in 3D space.

• **Covariance:** $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ defines its shape and orientation. To ensure $\boldsymbol{\Sigma}$ is always a valid positive semi-definite matrix and to allow for intuitive optimization, it is parameterized by a scaling vector $\mathbf{s} \in \mathbb{R}^3$ and a rotation quaternion $\mathbf{q} \in \mathbb{R}^4$.

$$\boldsymbol{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top \tag{1}$$

where $\mathbf{R}$ is the rotation matrix derived from $\mathbf{q}$ and $\mathbf{S}$ is a diagonal scaling matrix derived from $\mathbf{s}$.

• **Opacity:** $\alpha \in [0, 1]$ controls the transparency of the Gaussian.

• **Color:** View-dependent color is modeled using Spherical Harmonics (SH) coefficients.

Differentiable Rendering—To synthesize outputs from a novel viewpoint, the 3D Gaussians are projected onto the 2D image plane, forming elliptical splats. These splats are sorted by depth and composited in a front-to-back order using alpha blending. This differentiable rendering process produces not only the final RGB image ($\hat{I}$), but also a depth map and an accumulation (opacity) map. The rendered image can be directly compared to a ground truth image for gradient-based optimization, while the depth and accumulation maps provide additional supervision or regularization signals as needed.

Adaptive Densification Strategy— A key component of the 3DGS training process is the strategy for adaptive densification, which dynamically adjusts the set of Gaussians to efficiently represent the scene. This process is governed by periodic checks that add or remove primitives based on three main heuristics: the magnitude of the positional image-plane gradient, the 3D scale, and the opacity of each Gaussian. The strategy consists of three primary operations: growing, pruning, and opacity reset.

• **Growing (Densification)**: To represent complex regions that are not yet well-reconstructed, new Gaussians are introduced where the image-plane gradients exceed a threshold. This indicates that the optimizer is struggling to place the existing Gaussians correctly. Two methods are used:

– Duplication: If a Gaussian has a high gradient and small 3D scale, it is duplicated to add detail in under-reconstructed regions.

– Splitting: If a Gaussian has a high gradient and large 3D scale, it is split into two smaller Gaussians to better capture complex geometry.

• **Pruning**: To maintain a compact representation and remove artifacts, unnecessary Gaussians are pruned. A Gaussian is removed if it meets certain criteria, such as:

– Its opacity α falls below a minimum threshold, rendering it effectively invisible.

– Its 3D scale grows excessively large, which can cause blurry or hazy artifacts.

Figure 3: **3D Gaussian Splatting (adapted from [35]).**

• **Opacity Reset**: Periodically during training, the opacities of all Gaussians are reset to a low value. This acts as a regularizer, forcing the model to re-evaluate the importance of each Gaussian. Primitives that are essential to the reconstruction will quickly regain high opacity, while transient or unnecessary ones will fail to do so and be removed in a subsequent pruning phase.

Loss Function—The total loss $\mathcal{L}_{\text{total}}$ used to train the standard 3DGS model is a weighted sum of a reconstruction loss and a scale regularization term.

The reconstruction loss $\mathcal{L}_{\text{recon}}$ combines the L1 and D-SSIM losses, balanced by a hyperparameter λ_{SSIM}:

$$\mathcal{L}_{\text{recon}} = (1 - \lambda_{\text{SSIM}}) \cdot \|I - \hat{I}\|_1 + \lambda_{\text{SSIM}} \cdot \left(1 - \text{SSIM}(I, \hat{I})\right) \tag{2}$$

where I is the ground truth image and $\hat{I}$ is the rendered image.

To prevent the Gaussians from becoming overly stretched, a scale regularization loss $\mathcal{L}_{\text{scale}}$ is applied. For each Gaussian g with a scale vector $\mathbf{s}_g$ out of N_g total Gaussians, this loss penalizes the ratio of its largest to smallest scale component if it exceeds a threshold τ_{ratio}:

$$\mathcal{L}_{\text{scale}} = \frac{1}{N_g} \sum_{g=1}^{N_g} \max\left(0, \frac{\max(\mathbf{s}_g)}{\min(\mathbf{s}_g)} - \tau_{\text{ratio}}\right) \tag{3}$$

The final loss is the sum of these two components, with a weighting factor λ_{scale} for the regularization term:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{scale}} \mathcal{L}_{\text{scale}} \tag{4}$$

3. METHODOLOGY

Our primary contribution is a real-time pipeline for the semantic 3D reconstruction of the lunar surface using a 3D Gaussian Splatting (3DGS) representation. The architecture is composed of three principal stages: a perception frontend, an incremental mapping backend, and a keyframe-based optimization engine.

Perception and Tracking Frontend

The frontend processes raw sensor data into a structured format. We assume that rover pose estimates are continuously available from an external tracking system, such as visual-inertial odometry; the development of this tracking component is outside the scope of this work. For each incoming camera frame, the frontend computes a dense depth map using either traditional stereo algorithms or monocular depth estimation networks, demonstrating modular support for various sensor configurations. Concurrently, a semantic segmentation network classifies each pixel into relevant lunar categories (e.g., regolith, rock), providing critical context. Pixels classified as sky are explicitly masked to prevent the erroneous creation of 3D points.

For monocular depth estimation models, we use a scaling strategy using triangulated points from matched features between consecutive frames. We detect and match keypoints using SuperPoint [36] and SuperGlue [37], as shown in Fig. 5, then triangulate these points to obtain metric depth estimates. We use these triangulated depths to scale the output of the monocular models. To ensure robust scaling, we apply depth masking to filter out unreliable matches and use random sample consensus (RANSAC) to remove outliers from the triangulated points. Specifically, we solve for the scale parameter θ and offset γ by minimizing:

$$\min_{\theta,\gamma} \sum_{p \in M_{\text{sparse}}} \|\theta \hat{D}(p) + \gamma - \hat{D}_s(p)\|_2^2, \tag{5}$$

where $\hat{D}$ represents the estimated dense depth from the monocular model, $\hat{D}_s$ is the sparse depth from triangulation, and M_{sparse} is the mask of sparse depth estimates.

Incremental Mapping

The backend receives the processed frames from the frontend and is responsible for incrementally building the global 3D map. For each new frame, the estimated pose, dense depth map, and semantic labels are fused to generate a registered, per-frame 3D semantic point cloud. This step transforms the 2D perception outputs into a 3D context where every point is associated with a color and a semantic class. This point cloud is then used to add new Gaussians to the global 3DGS model. Each new Gaussian's size is set proportional to the average distance to its nearest neighbors, ensuring adaptive coverage based on local point density. To maintain real-time performance and prevent unbounded map growth over long trajectories, we use a voxel-based filtering strategy. Only 3D points that fall into previously unobserved regions of space are used to initialize new Gaussians in order to maintain real-time performance and prevent unbounded map growth. A key aspect of our approach is that each new Gaussian is initialized with the semantic label from its corresponding point.

Optimization Backend

The optimization engine runs as a background process, continuously refining all Gaussian parameters and camera poses to improve global consistency.

Keyframe-Based Asynchronous Optimization— A keyframe buffer stores a history of processed frames, serving as a long-term memory to prevent catastrophic forgetting. The optimization engine runs asynchronously, periodically sampling batches of past views from this buffer to enforce global consistency. This process is decoupled from the frame rate and triggered by time intervals, allowing for the efficient use of idle compute time during rover operations. While this work focuses on optimizing the Gaussian parameters, the keyframe-based approach provides a foundation for future extensions such as camera pose refinement and loop closure integration.

Densification Strategy—To manage the set of Gaussians, we modify the standard 3DGS densification strategy. The data structure for each Gaussian is augmented to include a semantic class identifier. The periodic, global reset of Gaussian opacities is removed; this is critical for long-term mapping as our optimization prioritizes recent keyframes, and a global reset could cause stable, older portions of the map to be

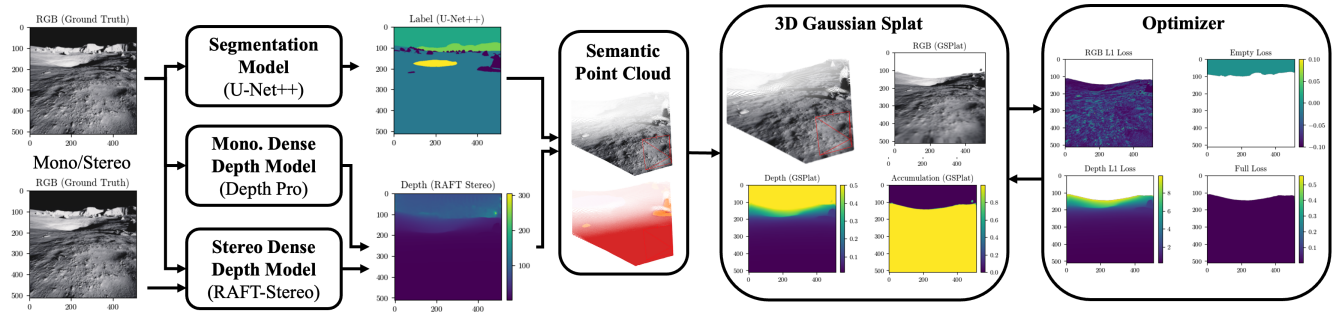


Figure 4: **Diagram of the proposed real-time 3DGS mapping pipeline.**

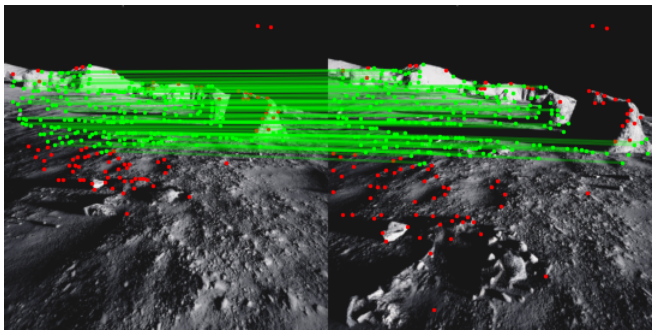


Figure 5: **Matches between consecutive frames used for monocular depth estimation scaling.**

incorrectly pruned, leading to catastrophic forgetting. Furthermore, our implementation omits the standard splitting and cloning operations. Since the backend incrementally adds geometry from dense depth maps with each new keyframe, the large, under-reconstructed regions that typically necessitate splitting are less prevalent. Instead, we manage map size with the voxel-based filtering strategy and by pruning Gaussians that are inconsistent with a known prior Digital Elevation Model (DEM).

Loss Function—The optimization process minimizes a total loss function, $\mathcal{L}_{\text{total}}$, which extends the standard 3DGS objective ($\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{scale}}$). The unique characteristics of lunar environments—namely high-contrast lighting and deep, hard-edged shadows with no atmospheric scattering—make relying on a simple background color insufficient. A deep shadow on the regolith can be photometrically indistinguishable from the black sky, providing ambiguous signals to the optimizer. To address this, we introduce explicit geometric and semantic supervision through additional loss components that use masks derived from sensor data.

These losses are applied using two masks. The surface mask, M_{surface}, identifies pixels corresponding to valid geometry. A pixel is included in this mask only if it is not classified as sky, its estimated depth value is finite, and, very importantly, it is within a pre-specified depth range for optimization. In this sense, a pixel that corresponds to a rock that is far from the current camera pose is not included in the surface mask. The empty space mask, M_{empty}, identifies pixels known to contain no geometry and is primarily composed of pixels identified as sky. Our additional loss components are:

- **Supervised Depth Loss**: To leverage the dense depth maps from our perception frontend, we add a supervised depth loss, $\mathcal{L}_{\text{depth}}$. This term enforces geometric accuracy by penalizing the L1 difference between the rendered depth, $\hat{D}$, and the input depth map, D, only on the M_{surface} mask:

$$\mathcal{L}_{\text{depth}} = \frac{1}{|M_{\text{surface}}|} \sum_{p \in M_{\text{surface}}} \frac{|D(p) - \hat{D}(p)|}{\max_{q \in M_{\text{surface}}} D(q)} \tag{6}$$

- **Volumetric Regularization Losses**: To regularize the density distribution and remove artifacts, we use two losses based on the rendered alpha accumulation, $\hat{\alpha}$. An *empty loss*, $\mathcal{L}_{\text{empty}}$, penalizes density in regions defined by M_{empty}. A complementary *full loss*, $\mathcal{L}_{\text{surface}}$, encourages surfaces within M_{surface} to be opaque:

$$\mathcal{L}_{\text{empty}} = \frac{1}{|M_{\text{empty}}|} \sum_{p \in M_{\text{empty}}} \hat{\alpha}(p) \tag{7}$$

$$\mathcal{L}_{\text{surface}} = \frac{1}{|M_{\text{surface}}|} \sum_{p \in M_{\text{surface}}} (1 - \hat{\alpha}(p)) \tag{8}$$

The final objective function is a weighted summation of the baseline and our new components:

$$\begin{aligned} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{scale}}\mathcal{L}_{\text{scale}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} \\ + \lambda_{\text{empty}}\mathcal{L}_{\text{empty}} + \lambda_{\text{surface}}\mathcal{L}_{\text{surface}} \end{aligned} \tag{9}$$

where the λ terms are the corresponding weighting coefficients. We also consider incorporating prior Digital Elevation Model (DEM) data into the loss function as a geometric constraint. This is implemented by adding a loss term that penalizes differences between the reconstructed geometry and the DEM in regions where the DEM is available and reliable.

4. Results

This section evaluates the key components of our mapping pipeline, beginning with the perception front-end. We first present a quantitative benchmark of various dense depth estimation and semantic segmentation models to assess their performance in the synthetic lunar environment. Subsequently, we analyze the final 3D reconstruction quality, evaluating how the selected perception models impact the geometric accuracy and semantic fidelity of the map. Both quantitative metrics and qualitative results are provided to support the analysis. The experiments were performed on a system equipped with an Intel i9-14900K CPU (32 cores, 5.7 GHz), 128 GB RAM, and an NVIDIA GeForce RTX 4090 GPU with 24 GB VRAM. All results related to computational requirements (such as memory usage and runtime) are only indicative, as the current implementation, methods, and models have not been optimized.

Evaluation Metrics

This subsection defines the evaluation metrics used to assess the performance of the proposed framework.

Depth Estimation— All depth estimation metrics are computed only over the valid pixels defined by the surface mask, M_{full}. For a rendered depth map $\hat{D}$ and a ground truth depth map D:

- **Mean Absolute Error (MAE) [m]** $\downarrow$: The average L1 distance between the rendered and ground truth depth.

$$\text{MAE} = \frac{1}{|M_{\text{full}}|} \sum_{p \in M_{\text{full}}} |\hat{D}(p) - D(p)| \tag{10}$$

- **Absolute Relative Error (AbsRel)** $\downarrow$: The scale-invariant average of the absolute relative difference.

$$\text{AbsRel} = \frac{1}{|M_{\text{full}}|} \sum_{p \in M_{\text{full}}} \frac{|\hat{D}(p) - D(p)|}{D(p)} \tag{11}$$

- **Threshold Accuracy** ($\delta_{25\%}$) $\uparrow$: The percentage of pixels where the ratio between the rendered and ground truth depth is within a factor of 25%.

$$\delta_{25\%} = \frac{1}{|M_{\text{full}}|} \sum_{p \in M_{\text{full}}} \mathbb{1} \left\{ \max \left(\frac{\hat{D}(p)}{D(p)}, \frac{D(p)}{\hat{D}(p)} \right) < 1.25 \right\} \tag{12}$$

- **Frames Per Second (FPS)** $\uparrow$: The processing throughput of the depth estimation model.

Semantic Segmentation—We compute the following metrics over all the pixels in the image for the semantic segmentation models:

- **Intersection over Union (mIoU)** $\uparrow$: The overlap between the predicted and ground truth masks for a single class or label.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{13}$$

- **Accuracy (Acc.)** $\uparrow$: The percentage of all pixels in the image that are correctly classified.

$$\text{Acc.} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{14}$$

Surface Reconstruction—To evaluate the final 3D map, we extract a point cloud, $\hat{P}$, by taking the means of the Gaussians in the dense representation and compare it to the ground truth point cloud, P. While a more accurate surface reconstruction could be achieved by leveraging the full density field of the Gaussians, this is beyond the scope of our current work. However, since our map is sufficiently dense, using the means of the Gaussians provides a good approximation for evaluation.

- **Accuracy (Chamfer-L_2) [cm]** $\downarrow$: The average distance from each point in the reconstructed point cloud to its nearest neighbor in the ground truth point cloud. This measures the correctness of the reconstructed surface.

$$\text{Accuracy} = \frac{1}{|\hat{P}|} \sum_{\hat{p} \in \hat{P}} \min_{p \in P} \|\hat{p} - p\|_2 \tag{15}$$

- **Completeness (Chamfer-L_2) [cm]** $\downarrow$: The average distance from each point in the ground truth to its nearest neighbor in the reconstruction. This measures how well the reconstruction covers the ground truth surface.

$$\text{Completeness} = \frac{1}{|P|} \sum_{p \in P} \min_{p \in \hat{P}} \|p - \hat{p}\|_2 \tag{16}$$

- **Precision** (d) **[%]** $\uparrow$: The percentage of reconstructed points within a distance threshold d of the ground truth, measuring correctness.

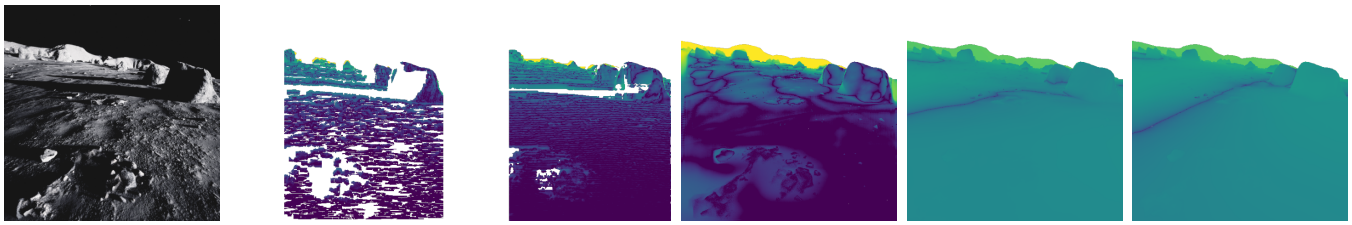
$$\text{Precision}(d) = \frac{1}{|\hat{P}|} \sum_{p \in \hat{P}} \mathbb{1} \left\{ \min_{p \in P} \|p - \hat{p}\|_2 < d \right\} \tag{17}$$

- **Recall** (d) **[%]** $\uparrow$: The percentage of ground truth points that have a reconstructed point within a distance threshold d, measuring completeness.

$$\text{Recall}(d) = \frac{1}{|P|} \sum_{p \in P} \mathbb{1} \left\{ \min_{p \in \hat{P}} \|p - \hat{p}\|_2 < d \right\} \tag{18}$$

- **F1-Score** (d) **[%]** $\uparrow$: The harmonic mean of Precision and Recall, providing a single metric that balances correctness and completeness.

$$F_1(d) = 2 \cdot \frac{\text{Precision}(d) \cdot \text{Recall}(d)}{\text{Precision}(d) + \text{Recall}(d)} \tag{19}$$



(a) **Input** (b) **BM** (c) **SGBM** (d) **RAFT-Stereo** (e) **Depth Anything** (f) **DPT**

Figure 6: **Input image and depth estimation errors for a sample of the LuSNAR dataset [15]. The other image used for stereo methods is not shown. The errors are in logarithmic scale to qualitatively characterize the performance of the models.**

Table 1: **Comparison of depth estimation models. Monocular depth is scaled using sparse depth estimates from matched features.**

Model	Param	MAE↓	AbsRel↓	$\delta_{25\%}$↑	FPS↑
Stereo					
BM	–	**1.6**	0.02	0.30	**24.5**
SGBM	–	**1.6**	0.02	0.41	14.9
RAFT-Stereo	11M	7.3	0.05	0.72	4.5
CREStereo	5M	3.5	0.04	**0.73**	1.4
Monocular					
Depth Anything V2					
Small	25M	11.1	0.21	0.53	13.2
Base	97M	10.8	0.19	0.56	13.2
Large	335M	10.7	0.18	0.59	9.8
GLPN	61M	15.1	1.82	0.07	9.9
DPT	343M	11.2	0.19	0.55	13.6
Depth Pro	952M	11.4	0.20	0.55	1.7

Depth Estimation

This section evaluates the depth estimation models on the LuSNAR dataset, combining qualitative analysis with quantitative metrics. Figure 6 provides a visual comparison of model outputs, while Table 1 lists the numerical results.

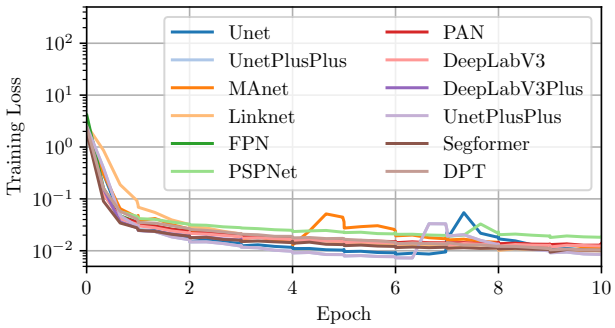
Traditional stereo methods, such as Block Matching (BM) and Semi-Global Matching (SGBM), produce sparse depth estimates, struggling in regions with challenging illumination or low texture, as shown qualitatively in Figure 6. The quantitative results in Table 1 reflect this; while their Mean Absolute Error (MAE) is low, this is a consequence of providing depth only in high-confidence areas, as indicated by their low $\delta_{25\%}$.

In contrast, learning-based stereo models like RAFT-Stereo and CREStereo demonstrate superior performance, generating dense depth maps with high completion rates and accuracy (Figure 6). While their MAE is higher than traditional methods due to denser predictions at long ranges, their low Absolute Relative Error (AbsRel) and significantly higher $\delta_{25\%}$ in Table 1 confirm better overall geometric coverage. Although CREStereo achieves slightly higher accuracy, its processing speed is substantially lower, as reported in Table 1. Consequently, we select RAFT-Stereo for our mapping pipeline as it provides the best trade-off between strong accuracy and a frame rate more suitable for real-time operation.

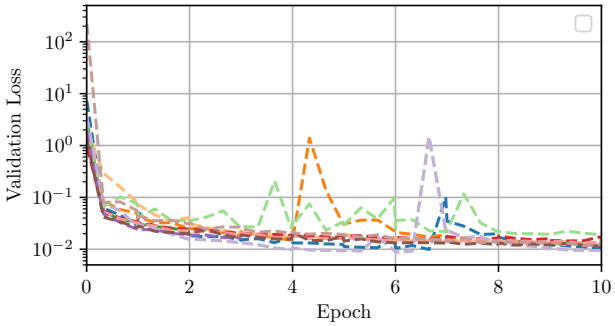
Monocular depth estimation models underperform across all metrics in Table 1, even after being scaled with sparse feature matches. Their performance is likely constrained by the domain shift from the terrestrial datasets they were trained on. Despite their lower accuracy, they remain a potentially valuable source of geometric information when stereo data is unavailable.

Semantic Segmentation

We benchmarked a suite of semantic segmentation architectures on the LuSNAR dataset to evaluate their effectiveness for lunar terrain classification. All models were trained to classify five semantic categories present in the dataset: regolith, craters, rocks, mountains, and sky [15]. The dataset was split into approximately 8,000 images for training, 1,000 for validation, and 2,000 for testing. For training, we used the Adam optimizer over 10 epochs with a batch size of 24.



(a) **Training losses.**



(b) **Validation losses.**

Figure 7: **Training and validation losses for the considered semantic segmentation models.**

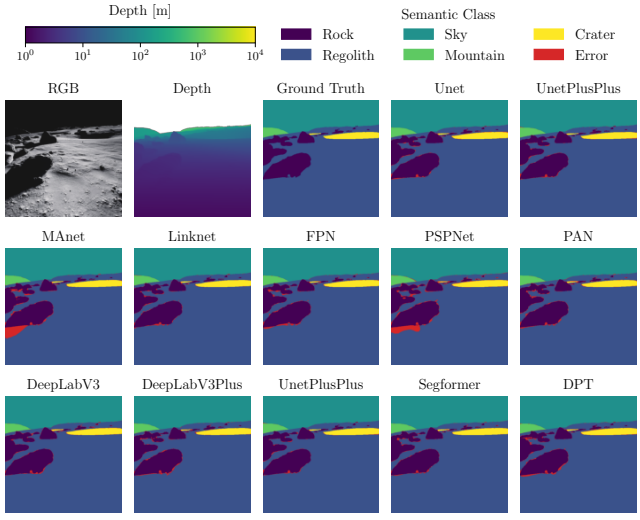


Figure 8: **Test sample of semantic segmentation predictions for the LuSNAR dataset. Misclassied pixels are highlighted in red.**

The initial learning rate was 1×10^{-3}, which was adjusted by a scheduler that reduced the rate by a factor of 0.75 with a patience of 10 epochs, down to a minimum of 1×10^{-7}.

As shown by the training and validation loss curves in Figures 7a and 7b, all models converged successfully. Qualitatively, the majority of architectures are able to accurately segment the primary terrain features, with representative

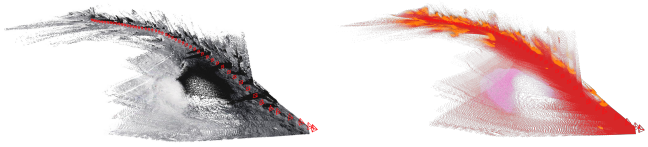
Table 2: **Semantic segmentation results on the LuSNAR dataset.**

Model	Size [MB]	Params	FPS ↑	IoU [%] ↑					Mean	Mean
				Regolith	Crater	Rock	Mountain	Sky	IoU ↑	Accuracy ↑
U-Net	55.0	14.3M	114.4	**99.5**	95.5	89.2	96.5	**99.8**	96.1	99.5
U-Net++	61.0	16.0M	78.0	**99.5**	91.1	**91.6**	**98.5**	**99.8**	96.1	**99.6**
MA-Net	83.0	21.7M	71.7	99.2	70.9	90.1	97.5	**99.8**	91.5	99.3
Linknet	44.0	11.7M	104.5	99.2	96.4	91.1	97.0	**99.8**	**96.7**	99.3
FPN	50.0	13.0M	108.7	98.9	**97.7**	86.8	97.8	99.7	96.2	99.2
PSPNet	**3.0**	**0.9M**	**203.2**	98.8	92.7	79.7	94.3	99.6	93.0	99.0
PAN	43.0	11.4M	92.8	99.0	94.6	84.3	97.4	99.7	95.0	99.2
DeepLabV3	61.0	15.9M	131.5	99.0	96.7	86.6	98.3	99.7	96.1	99.2
DeepLabV3Plus	47.0	12.3M	118.8	99.2	96.4	89.2	98.0	**99.8**	96.5	99.4
Segformer	45.0	11.8M	140.5	99.2	95.5	88.9	97.7	**99.8**	96.2	99.4
DPT	159.0	41.6M	79.6	99.1	94.1	87.4	95.7	99.7	95.2	99.3

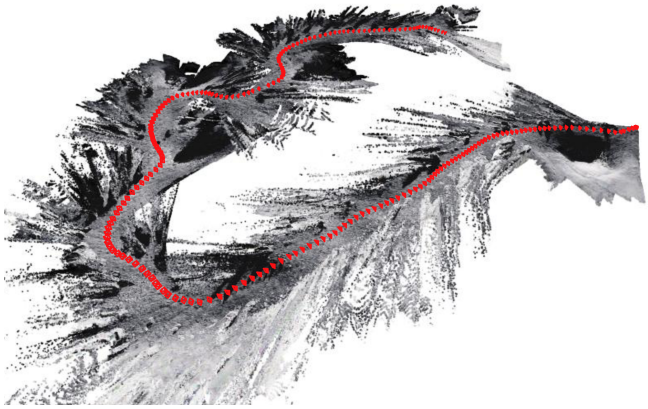
examples shown in Figure 8. The quantitative results in Table 2 reveal key performance trade-offs. While models like LinkNet and DeepLabV3+ achieve high overall mean IoU, other models excel at specific classes critical for navigation; for instance, FPN is the most effective at crater detection. Notably, PSPNet offers a highly efficient solution, achieving the highest FPS with by far the smallest model size, making it a strong candidate for resource-constrained hardware. For our downstream mapping task, we selected U-Net++ due to having the best overall accuracy and IoU for rocks (91.6% IoU), a critical capability for ensuring rover safety, while being computationally comparable to the other models.

Surface Reconstruction

We evaluate our real-time mapping framework on a 500-meter traverse within a scene from the LuSNAR dataset. The input is sourced from a single front-facing stereo camera pair (512x512 pixel resolution) at a rate of 1 Hz. The pipeline uses U-Net++ for semantic segmentation and RAFT-Stereo for dense depth estimation. We use a ground truth map sampled of 1 cm resolution. To benchmark our method and selection of perception models, we compare the reconstruction accuracy of this full pipeline against a configuration that uses ground truth depth and segmentation. For this evaluation, we assume ground truth poses are provided, as our primary focus is to analyze the mapping feasibility and performance independent of a front-end tracking system.



(a) **Initial map (RGB).** (b) **Partial map (semantic).**



(c) **Final map (RGB).**

Figure 9: **Partial and final 3DGS maps.**

For this preliminary analysis, it is important to note how the surface reconstruction metrics are computed. We extract a point cloud from the 3DGS map by using only the position of each Gaussian. This approach is a simplification, as the center of a Gaussian is not constrained to lie directly on the physical surface. By treating the Gaussians as discrete points, we do not yet leverage the continuous, density-based surface representation that is a key advantage of the method [38]. We are currently developing a more accurate surface reconstruction pipeline that will use this density information to provide a better evaluation of the map's quality.

Figure 9 shows the output of the mapping process, displaying a partial 3DGS map that visualizes the semantic information embedded within each Gaussian, as well as the final, complete map for the entire traverse. The quality of the scene representation is further demonstrated in Figure 10, which presents novel view synthesis results. These images are rendered from viewpoints that were not seen during the mapping process, highlighting the framework's ability to build a coherent and renderable 3D model of the environment.

The quantitative results for the map reconstruction are presented in Tables 3 and 6. The first table provides a detailed breakdown of performance for each semantic class, while the second summarizes the average results across all classes. From these aggregate results, our full pipeline achieves a mean height error of 10 cm. With a 10 cm evaluation threshold, the precision and recall metrics are over 70%, indicating that the majority of the reconstructed surface is geometrically consistent with the ground truth.

A key finding is the minimal impact of the segmentation model's accuracy on the final geometry; using ground truth semantic labels resulted in negligible changes to the aggregate metrics, demonstrating the high performance of the U-Net++ model. The per-class breakdown in Table 6 reveals that reconstruction errors are largest for the crater category. This is an expected outcome, as craters in the trajectory are typically distant from the rover and are often poorly illumi-

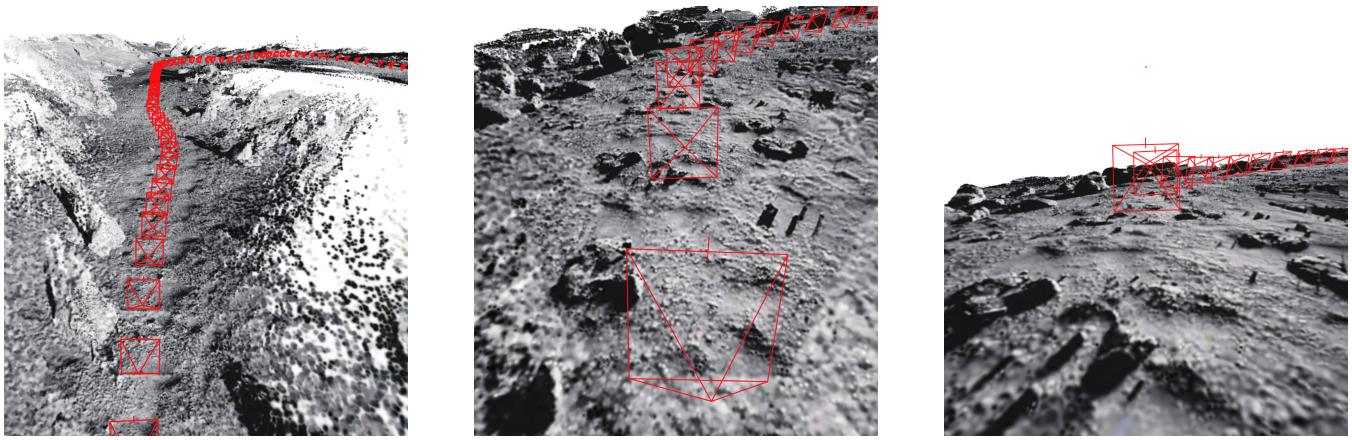


Figure 10: **Qualitative novel view synthesis results rendered from the 3DGS map.**

Table 3: **Summary of surface reconstruction results. Precision and recall using a threshold of 10 cm.**

Metric	Accuracy [cm] ↓	Completion [cm] ↓	Precision [%] ↑	Recall [%] ↑	F-Score [%] ↑	Height Error [cm] ↓
3DGS	19.4	25.8	73.1	78.9	75.9	10.8
+ GT Segmentation	19.4	25.8	73.1	78.8	75.9	10.8
+ GT Depth	13.3	29.4	82.3	81.1	81.7	6.0
+ GT Depth + GT Segmentation	13.3	29.4	82.3	81.1	81.7	6.0
Point Cloud	19.2	26.0	77.1	78.0	77.5	9.8
+ GT Segmentation	19.2	26.0	77.1	78.0	77.5	9.8
+ GT Depth	12.5	29.8	89.5	81.0	85.1	4.2
+ GT Depth + GT Segmentation	12.5	29.8	89.5	81.0	85.1	4.2

nated within deep shadows, making accurate depth estimation challenging. A qualitative analysis of the reconstruction shows that the largest geometric errors are concentrated along the dark, high-contrast edges of rocks. These regions represent a common failure mode for both the depth estimation and semantic segmentation models, which struggle to distinguish shadowed rock boundaries from the dark sky or shadowed regolith.

When using estimated depth under ideal pose conditions, the 3DGS pipeline's performance is quantitatively similar to the point cloud baseline, with the cm-level difference in average height error falling close to the 1 cm resolution of the ground truth map. Our next analysis will explore more challenging scenarios, such as re-mapping the same area to evaluate consistency or introducing pose noise. We hypothesize that in such cases, the 3DGS framework's ability to jointly optimize both the map and camera poses to achieve multi-view consistency would demonstrate a significant advantage.

Table 4 and Table 5 show the memory requirements for the 500-meter traverse, resulting in a map of approximately 15 million Gaussians and 12 million points. It is important to note that the current memory usage can be significantly reduced, as techniques for pruning and compressing the 3DGS map, keyframe buffer, and baseline point cloud are left as future work [39]. The 3DGS map representation itself (775 MB) is more compact than the sum of the images taken RGB (2x1200 MB), indicating that the 3DGS map can become a memory efficient alternative that offers rendering capabilities.

Compared to a traditional semantic point cloud of similar scale, the 3DGS map requires about 2.5 times more memory (336 MB for the point cloud). However, the memory footprint of our framework is highly dependent on the configuration. Only a small subset of the keyframe buffer storing RGB images, depths, and masks is necessary for global optimization tasks like loop closure or refining past poses. Similarly, the memory for strategy variables is only required for visible Gaussians and can be pruned for areas of the map no longer in view, substantially reducing the active memory load. Further analysis and optimization of the framework's memory and computational requirements is an area of ongoing work. We are investigating compression techniques for both the 3D Gaussian representation and the keyframe buffer to substantially reduce the memory footprint for long-range traverses.

5. Conclusions

In this work, we presented and evaluated a real-time framework for creating dense, semantic 3D maps of the lunar surface. Our approach integrates dense perception networks with a 3D Gaussian Splatting (3DGS) representation to address the unique challenges of lunar environments. The primary contributions are the benchmarking of modern perception models for this domain and the demonstration of their integration into an incremental, high-fidelity mapping pipeline.

Our experimental evaluation on the LuSNAR dataset provided several key findings. We identified RAFT-Stereo as a suitable depth estimation model, offering a robust balance between accuracy and real-time processing speed. For semantic segmentation, U-Net++ was selected for its superior performance in detecting rocks, a critical capability for hazard avoidance; its effectiveness was confirmed by the fact

Table 4: 3DGS memory usage.

GPU VRAM			995 MB	**System RAM**	1900 MB
3DGS	830 MB	**Strategy**	165 MB	**Buffer**	1900 MB
Rotations	222 MB	Radii	55 MB	RGBs	1200 MB
Means	166 MB	Gradients	55 MB	Depths	400 MB
Scales	166 MB	Counts	55 MB	Masks	200 MB
Features	166 MB			Labels	100 MB
Opacities	55 MB				
Labels	55 MB				

Table 5: Point Cloud memory usage.

System RAM	336 MB
Points	144 MB
Color	144 MB
Label	48 MB

Table 6: **Surface reconstruction results. Precision and recall using a threshold of 10 cm.**

Metric	**Accuracy [cm] ↓**	**Completion [cm] ↓**	**Precision [%] ↑**	**Recall [%] ↑**	**F-Score [%] ↑**	**Height Error [cm] ↓**
3DGS						
Rock	64.6	27.0	22.9	56.9	32.6	16.2
Regolith	14.5	28.0	81.8	82.0	81.9	8.5
Crater	944.7	27.1	13.2	41.5	20.0	75.3
All	19.4	25.8	73.1	78.9	75.9	10.8
+ GT Segmentation						
Rock	64.7	26.5	23.0	56.8	32.7	16.3
Regolith	14.4	28.0	81.8	82.0	81.9	8.5
Crater	936.2	27.6	13.4	41.4	20.2	75.1
All	19.4	25.8	73.1	78.8	75.9	10.8
+ GT Depth						
Rock	29.5	34.4	51.6	55.3	53.4	7.5
Regolith	10.2	30.0	87.3	85.2	86.2	5.3
Crater	384.3	47.5	42.0	73.3	53.4	42.0
All	13.3	29.4	82.3	81.1	81.7	6.0
+ GT Depth + GT Segmentation						
Rock	27.7	35.5	52.1	55.5	53.8	7.4
Regolith	10.0	30.5	87.5	85.4	86.4	5.2
Crater	344.5	48.8	43.2	73.4	54.4	40.8
All	13.3	29.4	82.3	81.1	81.7	6.0
Point Cloud						
Rock	64.3	27.9	25.7	55.2	35.1	15.9
Regolith	13.6	28.1	87.2	81.6	84.3	7.0
Crater	975.5	27.1	15.1	44.9	22.6	74.7
All	19.2	26.0	77.1	78.0	77.5	9.8
+ GT Segmentation						
Rock	64.4	27.5	25.8	55.4	35.2	16.0
Regolith	13.6	28.1	87.2	81.7	84.3	7.0
Crater	967.1	27.6	15.4	44.8	22.9	74.4
All	19.2	26.0	77.1	78.0	77.5	9.8
+ GT Depth						
Rock	29.9	35.9	66.0	55.1	60.1	6.4
Regolith	8.7	30.1	93.5	85.0	89.0	3.3
Crater	406.2	47.7	47.9	77.9	59.3	42.5
All	12.5	29.8	89.5	81.0	85.1	4.2
+ GT Depth + GT Segmentation						
Rock	27.8	37.0	67.2	55.1	60.6	6.3
Regolith	8.4	30.6	93.7	85.2	89.3	3.1
Crater	365.6	49.2	49.4	77.9	60.4	41.0
All	12.5	29.8	89.5	81.0	85.1	4.2

that using ground truth segmentation provided no quantitative improvement to the final map. When using these models, our pipeline achieved a geometric height accuracy of approximately 10 cm. Under the assumption of perfect poses, the reconstruction quality was comparable to a traditional point cloud baseline, with the largest errors occurring in distant, poorly-lit craters and along the dark edges of rocks where perception models are most likely to fail.

This work establishes a foundation for several avenues of future research. The most critical next step is to integrate a tracking front-end to create a complete Simultaneous Localization and Mapping (SLAM) system. This will remove the reliance on ground truth poses and test the 3DGS representation's ability to jointly refine the map and camera trajectory, which we hypothesize will show a significant advantage over simpler aggregation methods. Future work will also focus on more sophisticated, density-based techniques for surface extraction from the Gaussians to better capture the continuous geometry. Finally, a more rigorous characterization of the system—including its performance with varied camera configurations, additional datasets, and the incorporation of perception uncertainty—is necessary to develop advanced, map-aware path planning algorithms that can directly enhance rover autonomy.

ACKNOWLEDGEMENTS

We gratefully acknowledge Blue Origin for funding this project and for valuable discussions that contributed to its development. This manuscript benefited from the use of AI-based assistants, including Claude Sonnet 4.5 and Gemini 2.5 Pro, which were employed for coding assistance and revising. All final content was reviewed and edited by the authors.

REFERENCES

[1] Euroconsult, "Prospects for space exploration, 4th edition."

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multimap SLAM," vol. 37, no. 6, pp. 1874–1890.

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136.

[4] NASA. Apollo lunar surface journal.

[5] F. Tosi, Y. Zhang, Z. Gong, E. Sandström, S. Mattoccia, M. R. Oswald, and M. Poggi, "How NeRFs and 3d gaussian splatting are reshaping SLAM: a survey."

[6] B. Kuang, C. Gu, Z. A. Rana, Y. Zhao, S. Sun, and S. G. Nnabuife, "Semantic terrain segmentation in the navigation vision of planetary rovers—a systematic literature review," vol. 22, no. 21, p. 8393, publisher: Multidisciplinary Digital Publishing Institute.

[7] Y. Jia, L. Liu, and C. Zhang, "Moon impact crater detection using nested attention mechanism based UNet++," vol. 9, pp. 44 107–44 116.

[8] E. Van Kints, A. Hammond, C. Adams, and I. G. Lopez-Francos, "Neural radiance methods for lunar terrain modeling," in *2025 IEEE Aerospace Conference*, pp. 1–17, ISSN: 2996-2358.

[9] C. Adams, I. Lopez-Francos, E. V. Kints, and A. Hammond, "A summary of neural radiance fields for shadow removal and relighting of satellite imagery."

[10] R. Huang, C. Liu, H. Xie, J. Yu, T. Tao, Y. Xu, Z. Ye, and X. Tong, "Monocular visual SLAM with adjusting neural radiance fields for 3-d reconstruction in planetary environments," vol. 63, pp. 1–19.

[11] M. Hansen, C. Adams, T. Fong, and D. Wettergreen, "Analyzing the effectiveness of neural radiance fields for geometric modeling of lunar terrain," in *2024 IEEE Aerospace Conference*, pp. 1–12, ISSN: 1095-323X.

[12] A. Dai, S. Gupta, and G. Gao, "Neural radiance maps for extraterrestrial navigation and path planning," pp. 1606–1620, ISSN: 2331-5954.

[13] X. Zhang, L. Cui, and J. Yin, "Neural radiance fields for unbounded lunar surface scene," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 16 858–16 864.

[14] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: representing scenes as neural radiance fields for view synthesis," vol. 65, no. 1, pp. 99–106.

[15] J. Liu, Q. Zhang, X. Wan, S. Zhang, Y. Tian, H. Han, Y. Zhao, B. Liu, Z. Zhao, and X. Luo. LuSNAR:a lunar segmentation, navigation and reconstruction dataset based on muti-sensor for autonomous exploration.

[16] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," vol. 30, no. 2, pp. 328–341.

[17] L. Lipson, Z. Teed, and J. Deng, "RAFT-stereo: Multilevel recurrent field transforms for stereo matching," in *2021 International Conference on 3D Vision (3DV)*, pp. 218–227, ISSN: 2475-7888.

[18] J. Li, P. Wang, P. Xiong, T. Cai, Z. Yan, L. Yang, J. Liu, H. Fan, and S. Liu, "Practical stereo matching via cascaded recurrent network with adaptive correlation."

[19] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," vol. 37, pp. 21 875–21 911.

[20] D. Kim, W. Ka, P. Ahn, D. Joo, S. Chun, and J. Kim, "Global-local path networks for monocular depth estimation with vertical CutDepth."

[21] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction."

[22] A. Bochkovskii, A. Delaunoy, H. Germain, M. Santos, Y. Zhou, S. R. Richter, and V. Koltun, "Depth pro: Sharp monocular metric depth in less than a second."

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation."

[24] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested u-net architecture for medical image segmentation."

[25] T. Fan, G. Wang, Y. Li, and H. Wang, "MA-net: A multi-scale attention network for liver and tumor segmentation," vol. 8, pp. 179 656–179 665.

[26] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmen-

tation," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4.

[27] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," pp. 2117–2125.

[28] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network."

[29] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation."

[30] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation."

[31] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation."

[32] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding."

[33] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers."

[34] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." vol. 42, no. 4, pp. 139–1.

[35] A. Dalal, D. Hagen, K. G. Robbersmyr, and K. M. Knausgård, "Gaussian splatting: 3d reconstruction and novel view synthesis: A review," vol. 12, pp. 96 797–96 820.

[36] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description."

[37] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks."

[38] Y. Wolf, A. Bracha, and R. Kimmel, "GS2mesh: Surface reconstruction from gaussian splatting via novel stereo views," in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Springer Nature Switzerland, pp. 207–224.

[39] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3dgs.zip: A survey on 3d gaussian splatting compression methods," vol. 44, no. 2, p. e70078, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70078.

BIOGRAPHY

Guillem Casadesus Vila is a Ph.D. candidate in the Department of Aeronautics and Astronautics at Stanford University. He received his B.Sc. degree in Aerospace and Telecommunications Engineering in 2022 from the Universitat Politecnica de Catalunya (UPC) under the CFIS program. His research interests include robotic space exploration and space navigation and communication.

Adam Dai is a Ph.D. candidate in the Department of Electrical Engineering at Stanford University. He received his B.Sc. degree in Electrical Engineering with a minor in Computer Science in 2019 from the California Institute of Technology. His research interests include navigation, mapping, and planning in unstructured 3D environments.

Grace Gao is an assistant professor in the Department of Aeronautics and Astronautics at Stanford University. Before joining Stanford University, she was an assistant professor at University of Illinois at Urbana-Champaign. She obtained her Ph.D. degree at Stanford University. Her research is on robust and secure positioning, navigation, and timing with applications to manned and unmanned aerial vehicles, autonomous driving cars, as well as space robotics.