Depth-Variance Guided Next-Best-View Planning for Active 3D Gaussian Splatting Reconstruction

***Abstract*— We present an autonomous system for 3D reconstruction of bounded objects using a UAV equipped with an RGB-D camera. The system combines incremental 3D Gaussian Splatting (3DGS) with a novel depth-variance acquisition function for next-best-view (NBV) planning. Unlike alpha-based coverage metrics that are fooled by hallucinated renders, our scorer measures per-pixel geometric uncertainty via alpha-blended depth second moments in a single render pass. We demonstrate the system on a PX4 drone in Gazebo simulation, achieving $70{\pm}4\%$ surface coverage compared to $40{\pm}3\%$ for fixed orbit patterns across 30 trials, with comparable reconstruction quality. The depth-variance signal maintains viewpoint differentiation throughout the mission while conventional alpha coverage saturates, enabling informed viewpoint selection even with mature models.**

I. Introduction

Recent advances in Gaussian-based rendering, such as 3DGS [1] and ActiveSplat [2], have enabled photorealistic rendering with fast training and real-time display. These methods present a compelling opportunity when integrated with mission planning for autonomous systems operating under battery or onboard computation constraints, for terrain mapping and surveying applications.

In this paper, we present an autonomous PX4-based drone performing online Gaussian Splatting object reconstruction within a Gazebo [3] simulation environment using our proposed active mapping method for online object reconstruction, subject to onboard computation constraints.

II. Related Work

A. Surface Splatting

Surface Splatting [4] is a point-based rendering algorithm that represents surfaces using splat primitives via a screen-space Elliptical Weighted Average (EWA) filter. It takes irregularly spaced point samples as input and produces high-quality texture filtering without global parameterization. While effective for real-time point rendering, it is limited to surface representations. NeRF [5] later bridged this gap through learned volumetric representations.

B. Neural Radiance Fields

NeRF [5] renders novel views via a learned volumetric representation. Given calibrated camera poses from COLMAP [6] and RGB images, a multi-layer perceptron (MLP) outputs color and density at sampled 3D points, accumulated along camera rays via volume rendering. However, NeRF's reliance on a MLP results in long training and rendering times, making it unsuitable for real-time applications.

C. 3D Gaussian Splatting

3DGS [1] replaces the implicit MLP with explicit 3D Gaussian primitives, achieving real-time rendering and fast training. A sparse SfM point cloud [6] initializes Gaussian primitives, which are refined through adaptive density control and differentiable tile-based rasterization with alpha compositing. Extensions such as SuGaR [7] extract meshes from the Gaussian representation, and PGSR [8] enforces planar constraints for surface reconstruction. Dense SLAM systems including GS-SLAM [9] and RTG-SLAM [10] extend 3DGS to incremental mapping with real-time camera tracking. However, these methods process sequential input and do not actively select viewpoints.

D. Active Gaussian Splatting

ActiveSplat [2] is an active mapping system that combines a Gaussian map with a Voronoi graph workspace abstraction for exploration. It uses accumulated opacity to identify incomplete regions and a hierarchical planner to select viewpoints. By leveraging explicit Gaussian primitives rather than implicit neural representations, it enables online reconstruction at reduced computational cost.

E. Active Mapping

Active mapping plans sensor trajectories to maximize information gain. Sampling-based methods such as [11] plan paths over volumetric representations to explore unknown spaces online. FisherRF [12] computes uncertainty via the diagonal Fisher Information of radiance field parameters, providing principled view selection but requiring per-candidate Jacobian computation. RT-GuIDE [13] approximates uncertainty through Gaussian mean displacement and avoids rendering at candidate poses entirely, achieving real-time operation on a ground robot. However, existing methods either target room-scale exploration rather than bounded object reconstruction, or impose computational demands that preclude deployment on resource-constrained platforms. Our method addresses this gap with a lightweight depth-variance acquisition function operating within 420 MB on a UAV.

III. Problem Statement

We consider the task of autonomously reconstructing a bounded object using a UAV with an onboard RGB-D camera and limited computational resources. The system must determine viewpoints during flight, build a 3D model incrementally, and maximize surface coverage. An approximate object location is assumed from a preliminary survey;

the system then estimates a conservative cuboidal prior from two onboard depth observations before reconstruction begins.

Fixed flight patterns such as circular orbits capture dense views from a narrow elevation band but leave significant portions of the object unobserved. In applications such as planetary sample inspection or post-disaster structural assessment, communication latency precludes operator-guided view selection, requiring fully autonomous operation under onboard compute constraints.

A. Contributions

- A depth-variance acquisition function for NBV planning with 3DGS that computes per-pixel geometric uncertainty via alpha-blended depth second moments in a single render pass. The metric maintains $3\times$ the dynamic range of alpha-based coverage throughout the mission.
- An integrated autonomous pipeline comprising PX4 flight control, incremental 3DGS training, real-time NBV scoring, and collision-aware path planning, operating within 420 MB peak GPU memory, deployable on the NVIDIA Jetson Orin Nano (4 GB shared, 7–15 W).
- Experimental evaluation over 30 trials demonstrating $70\pm4\%$ surface coverage compared to $40\pm3\%$ for fixed orbit patterns, with the captured viewpoints producing 35 dB PSNR under offline reconstruction.

IV. Method

A. System Overview

The system operates three concurrent threads on a single GPU: a ROS 2 flight controller at 20 Hz, a CUDA-based 3DGS optimizer, and a real-time splat viewer. The drone captures RGB-D images from viewpoints selected by an acquisition function and incrementally trains a Gaussian Splatting model during flight.

The mission has three phases (Fig. 1). First, the system estimates a cuboidal prior around the target. Second, a *seed orbit* captures 4 keyframes at 90° intervals to initialize the model. Third, the system iterates through *NBV rounds*: score 95 candidate viewpoints, select the top 4, fly to capture, and retrain. This repeats for 11 rounds (44 adaptive keyframes, 48 total). The full procedure is detailed in Algorithm 1.

B. Cuboidal Prior Estimation

Given an approximate object location from a preliminary survey, the drone captures two depth images: one from directly overhead and one from the side. The overhead view provides the horizontal center and extent via ground-plane subtraction (the 85th depth percentile identifies the ground; pixels significantly closer are segmented as the object). The lateral view provides the vertical extent. The bounding box center is placed at the median of the segmented points in each axis, and the box size is set to the maximum extent plus a 0.5 m margin, clamped to $[1,4]$ m. This prior defines the candidate hemisphere, voxel grid, and Gaussian clipping volume for the remainder of the mission.

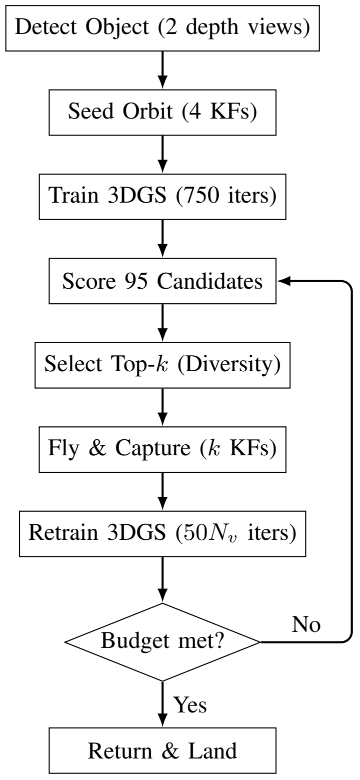


Fig. 1: System pipeline. The drone first estimates a cuboidal prior from two depth views (overhead + lateral), then captures seed views to initialize the model. The system iterates: score candidates via depth-variance, select diverse viewpoints, fly and capture, retrain until budget is met.

C. Candidate Generation

Candidate viewpoints are distributed on a Fibonacci hemisphere of radius $R = B + 0.5$ m centered on the target object, where B is the detected bounding box size (Fig. 2). From 200 initial points, filtering for minimum altitude (0.3 m) and bounding box clearance (0.5 m margin) yields 95 valid candidates spanning the upper hemisphere. This provides uniform angular coverage without axis-aligned clustering.

D. Depth-Variance Acquisition Function

For each candidate viewpoint c, we render per-pixel depth statistics using gsplat's arbitrary-channel rasterization with two channels per Gaussian: depth d_i and squared depth d_i^2. Alpha-blended accumulation yields:

$$\bar{D}(p) = \textstyle\sum_i d_i\, w_i \qquad \text{(expected depth)} \qquad (1)$$

$$\overline{D^2}(p) = \textstyle\sum_i d_i^2\, w_i \qquad \text{(second moment)} \qquad (2)$$

where $w_i = T_i \alpha_i$ are standard alpha-blending weights. The per-pixel variance $\mathrm{Var}(p) = \overline{D^2}(p) - \bar{D}(p)^2$ measures depth consistency along each ray. Well-reconstructed surfaces produce consistent depths, yielding low variance. Poorly modeled regions exhibit conflicting depth values, yielding high variance. Transparent pixels ($\alpha < 0.1$) receive maximum variance, correctly identifying unobserved geometry.

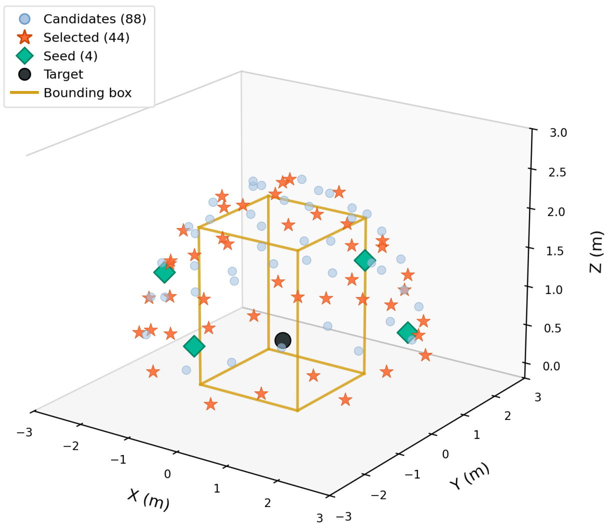


Fig. 2: Fibonacci hemisphere candidates (blue) around the target bounding box (gold). Red stars indicate viewpoints selected by the acquisition function. Green diamonds are seed orbit positions.

The geometric uncertainty for candidate c is:

$$u_c = \text{mean}(\text{Var}) \times \frac{A_{\text{bbox}}}{A_{\text{img}}} \tag{3}$$

where A_{bbox} is the projected area of the target bounding box in the rendered image and A_{img} is the total image area. This area fraction prevents overhead views with small projections from receiving inflated scores.

E. Scoring and Batch Selection

Each candidate receives a combined score:

$$s_c = 0.6\,\widetilde{u}_c + 0.4\,\widetilde{a}_c \tag{4}$$

where $\widetilde{u}_c = \text{clip}(u_c/P_{95}(u), 0, 1)$ uses robust percentile normalization and $\widetilde{a}_c$ is the minimum angular distance to visited viewpoints normalized by the grid scale $\Delta\theta = \sqrt{2\pi/N}$, where N is the total number of candidate viewpoints. The 0.6/0.4 weighting prioritizes geometric uncertainty while ensuring sufficient angular diversity to prevent view clustering; lower angular weight led to directional bias in early experiments.

Batch selection applies multiplicative diversity to prevent clustering. The first pick maximizes s_c; subsequent picks are penalized by proximity to prior selections:

$$s_c^{\text{eff}} = s_c \times \text{clamp}\left(\frac{\min_j \angle(c, c_j^*)}{\pi/2},\ 0,\ 1\right) \tag{5}$$

where c_j^* is the j-th viewpoint already picked in the current batch. A candidate at 0° from an existing pick scores zero; at 90° or more, it receives its full score. Between viewpoints, the drone climbs to a safe altitude above the bounding box before transiting horizontally, avoiding collision with the target object.

F. Incremental 3DGS Training

Each captured depth image is backprojected to 3D using the known camera intrinsics and pose, and the resulting points are added as new isotropic Gaussians. We define a *training round* as one complete optimization cycle triggered when new keyframes are queued. A *pass* is one gradient update on a single view. The number of iterations per round scales with the view count:

$$N_{\text{iter}} = \min(2500,\ \max(750,\ 50 \times N_v)) \tag{6}$$

This maintains approximately 50 passes per view regardless of the current view count N_v. A *training window* restricts each round to a subset of views. The orbit baseline uses a window of 10 spatially nearest views. For NBV, we set the window to include all captured views (i.e., no windowing), because NBV viewpoints are spread across the full hemisphere and training on a subset causes quality degradation on excluded views. The loss function combines photometric [14], depth, and isotropic regularization terms:

$$\mathcal{L} = \underbrace{0.8\|I - \hat{I}\|_1 + 0.2(1 - \text{SSIM})}_{\text{color}} + 0.5\,\mathcal{L}_{\text{depth}} + 0.1\,\mathcal{L}_{\text{iso}} \tag{7}$$

V. Experiments

A. Setup

Experiments use Gazebo Harmonic [3] with a PX4 X500 quadcopter carrying a 640×480 RGB-D gimbal camera (HFOV 1.204 rad, 30 fps). The target is an Apollo lunar sample on a textured lunar surface. All computation runs on one NVIDIA RTX 4070 Ti SUPER (16 GB). The 3DGS model is capped at 500K Gaussians, trained at 320×240. All reported metrics are averaged over 30 trials. Surface coverage is measured using a voxel grid (5 cm resolution) aligned with the cuboidal prior: a voxel is *occupied* if any depth backprojection falls within it, and *covered* if observed from at least two distinct viewpoints. Coverage is the ratio of covered to occupied voxels.

The **orbit baseline** flies a predetermined dual-altitude pattern: 24 waypoints at 1.0 m (−15° pitch) and 2.5 m (−30° pitch). It uses a spatial window of 10 nearest views per training round with 500 iterations, exploiting the spatial coherence of sequential orbit captures.

NBV uses 4 seed views followed by 44 adaptive views selected by the depth-variance scorer. It trains on all views each round (no windowing) with iterations scaling from 750 to 2400 as views accumulate. The higher iteration count is a consequence of the training strategy (all views per round vs. windowed), not an independent advantage; the offline comparison at equal 7K iterations isolates the viewpoint selection contribution.

Algorithm 1: Depth-Variance Guided NBV Planning

```
Input: Rock position $\mathbf{r}$, radius $R$, budget $B$, batch
       size $k$
Output: $B$ keyframe images with camera poses
$\mathcal{G} \leftarrow$ InitGaussianModel();
$\mathcal{C} \leftarrow$ FibonacciHemisphere($\mathbf{r}$, $R$) ;            // 95
  candidates
$\mathcal{V} \leftarrow$ SeedOrbit($\mathbf{r}$, $R$, 4) ;        // 4 seed views
foreach $\mathbf{v} \in \mathcal{V}$ do
    FlyTo($\mathbf{v}$);
    $I, D \leftarrow$ Capture();
    $\mathcal{G}$.AddPoints(Backproject($D$));
end
$\mathcal{G}$.Train($\mathcal{V}$, iters $= 750$);
while $|\mathcal{V}| < B$ do
    foreach $c \in \mathcal{C}$ do
        $\bar{D}, \overline{D^2} \leftarrow \mathcal{G}$.Render($c$, $[d, d^2]$);
        Var($c$) $\leftarrow \overline{D^2} - \bar{D}^2$;
        $u_c \leftarrow \mathrm{mean}(\mathrm{Var}) \cdot A_{\mathrm{bbox}}/A_{\mathrm{img}}$;
        $a_c \leftarrow \min_{j \in \mathcal{V}} \arccos(\hat{v}_c \cdot \hat{v}_j)/\Delta\theta$;
        $s_c \leftarrow 0.6\,\tilde{u}_c + 0.4\,\tilde{a}_c$;
    end
    $\mathcal{B} \leftarrow$ GreedyDiverseSelect($s$, $k$) ;        // Top-$k$
    foreach $\mathbf{v} \in \mathcal{B}$ do
        FlyTo($\mathbf{v}$); $I, D \leftarrow$ Capture();
        $\mathcal{G}$.AddPoints(Backproject($D$));
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{v}\}$;
    end
    $\mathcal{G}$.Train($\mathcal{V}$, iters $= \min(2500, \max(750, 50|\mathcal{V}|))$);
end
return $\mathcal{V}, \mathcal{G}$
```

TABLE I: Reconstruction comparison (mean ± std over 30 trials). Live PSNR is the incremental model at mission end. Offline uses Kerbl et al.'s 3DGS [1] on captured images with COLMAP [6] poses. The 7K column reflects ~90 s of GPU time.

	Live (dB)	Off. 7K (dB)	Off. 30K (dB)	Cov. (%)	VRAM (MB)	Time (s)
Orbit	18.6±1.8	26.5±0.8	34.8±0.3	40±3	180	120
NBV (ours)	21.9±1.6	31.6±1.2	35.1±0.4	70±4	420	394

B. Quantitative Results

Table I summarizes the results. At equal compute budget (7K iterations, ~90 s GPU time), NBV achieves 31.6 dB compared to 26.5 dB for the orbit, a **5.1 dB improvement**. The orbit requires the full 30K iterations to reach 34.8 dB, while NBV reaches comparable quality (35.1 dB) at the same budget. Diverse viewpoints contribute unique geometric constraints, enabling faster convergence.

The live PSNR (18–22 dB) is bounded by the incremental training regime and serves as a decision engine for viewpoint selection, not as the final reconstruction.

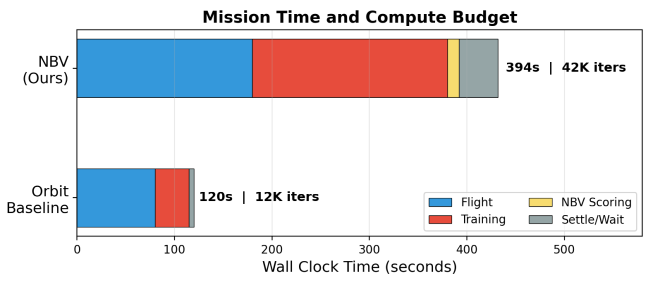
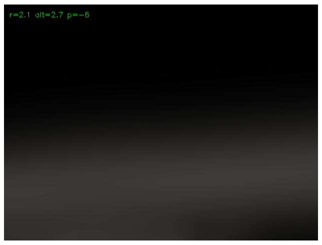
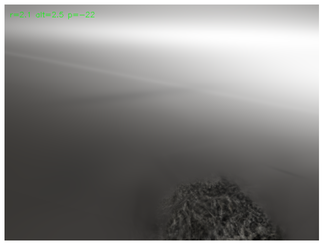


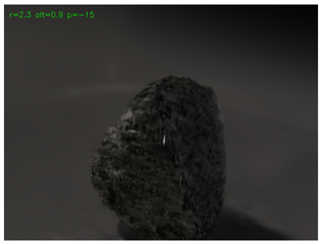
Fig. 3: NBV trains on all 48 views per round (no windowing) to prevent quality degradation on excluded views, requiring more iterations. The orbit trains on a window of 10 correlated views. NBV scoring adds only 12 s (2.5%) to the total mission.



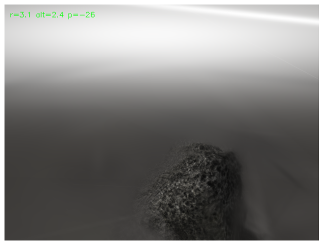
(a) Orbit: overhead



(b) NBV: overhead



(c) Orbit: side



(d) NBV: side

Fig. 4: Novel view renders from the live model. The orbit (a) produces no content from overhead as it never observes the object from above. NBV (b) captures elevated viewpoints, producing visible geometry from all angles. Both methods produce comparable side-view quality (c, d).

C. Training Budget

Fig. 3 explains the computational difference. The orbit can use spatial windowing (10 views/round) because adjacent orbit views share geometry-training on nearby views does not degrade distant ones. NBV viewpoints span the hemisphere; windowing causes quality degradation on excluded views as the model overwrites their regions. Training on all views each round requires more iterations to maintain 50 passes per view, increasing total compute from ~12K to ~42K iterations.

The NBV scoring itself is lightweight: 95 candidates scored in ~1 s per round (12 s total), consuming only 2.5% of mission time. The dominant overhead is flight time between diverse viewpoints.

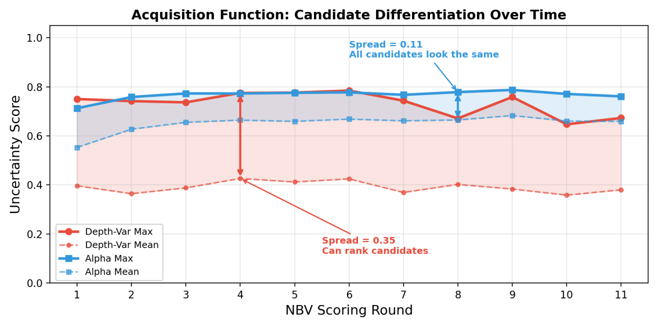
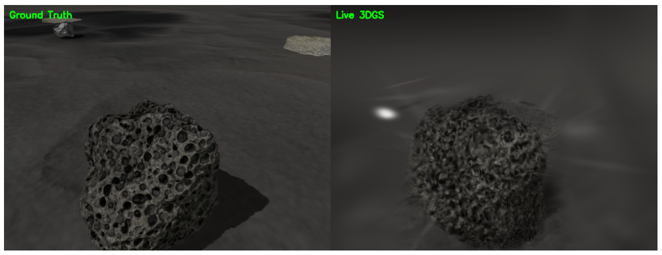
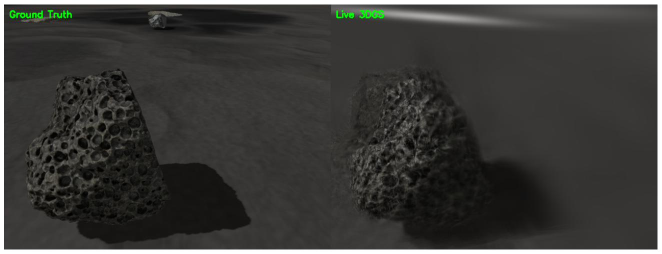


Fig. 5: Depth-variance uncertainty vs. alpha-based coverage across NBV rounds. Alpha coverage saturates near 0.66 by round 3, reducing candidate differentiation. Depth-variance maintains 3× greater dynamic range throughout all 11 rounds.



(a) Orbit: Ground truth (left) vs. live 3DGS render (right)



(b) NBV: Ground truth (left) vs. live 3DGS render (right)

Fig. 6: Live model quality for keyframe 37. Both methods reconstruct the rock surface. The orbit produces sharper results at orbit-ring elevations due to denser view overlap. The NBV model distributes capacity across the full hemisphere.

D. Coverage and Novel View Quality

Fig. 4 shows the coverage difference. From an overhead viewpoint, the orbit model renders nothing as it never observed the object from above. The NBV model, guided by the depth-variance scorer, selected elevated viewpoints that cover the top surface. From side views visited by both methods, quality is comparable.

E. Acquisition Function Comparison

Fig. 5 compares the depth-variance scorer against alpha-based coverage (mean opacity), both computed on every round. Alpha coverage saturates as the model fills in surfaces with plausible-looking Gaussians, even from unobserved angles. The depth-variance metric detects these hallucinated surfaces because the underlying Gaussians have inconsistent depths along viewing rays, maintaining a mean spread of 0.33 across all 11 rounds versus 0.11 for alpha coverage.

F. Reconstruction Quality

Fig. 6 compares ground truth captures against live model renders for both methods. The orbit achieves sharper detail at elevations within its orbit rings, while NBV distributes reconstruction quality across more diverse viewing angles.

G. Computational Feasibility

The system peaks at 420 MB, occupying 10% of the 4 GB shared LPDDR5 on an NVIDIA Jetson Orin Nano, a 512-CUDA-core edge module operating at 7–15 W, widely used in UAV applications. NBV scoring adds only 12 s total (95 candidates per round, 11 rounds). The primary constraint is the serial flight–train–score pipeline: the GPU is idle during flight, limiting total training to ∼42K iterations. Offline reconstruction on the same GPU achieves 31–35 dB in 6.5 minutes, confirming that viewpoint quality, not compute, determines final reconstruction fidelity.

H. Discussion

The results highlight two properties of the depth-variance acquisition function. First, it maintains candidate differentiation throughout the mission (Fig. 5), whereas alpha-based coverage saturates once the model learns to render plausible surfaces from unobserved angles. This saturation is a fundamental limitation of opacity-based metrics: a well-optimized 3DGS model produces high alpha everywhere, regardless of whether the underlying geometry is correct. In early experiments, using alpha coverage as the acquisition function produced viewpoint distributions indistinguishable from random selection after round 3, as all candidates received near-identical scores and selection was dominated by the angular diversity term. Depth-variance sidesteps this by measuring consistency rather than coverage.

Second, the offline reconstruction results in Table I show that the primary value of active viewpoint selection lies in the captured images, not the live model. Both methods reach similar PSNR at 30K offline iterations (35.1 vs. 34.8 dB), but at 7K iterations the NBV views converge 5.1 dB faster. This suggests that diverse viewpoints provide stronger geometric constraints that accelerate optimization.

The live PSNR gap between methods (21.9 vs. 18.6 dB) is modest compared to the offline gap at 7K. This is expected: incremental training with 48 views distributed across the hemisphere produces gradient interference between distant views, limiting online quality regardless of the acquisition function. The forgetting analysis in Section VI-A examines this further.

Compared to room-scale exploration systems such as RT-GuIDE [13] (16–20 dB on indoor scenes) and ActiveSplat [2] (21.7 dB online), our system achieves comparable live PSNR (21.9 dB) while operating on a UAV within 420 MB, an order of magnitude less memory than typical SplaTAM-based pipelines. FisherRF [12] provides principled uncertainty via Fisher Information but requires 11 ms per candidate with a custom CUDA kernel; our depth-variance scores 95 candi-

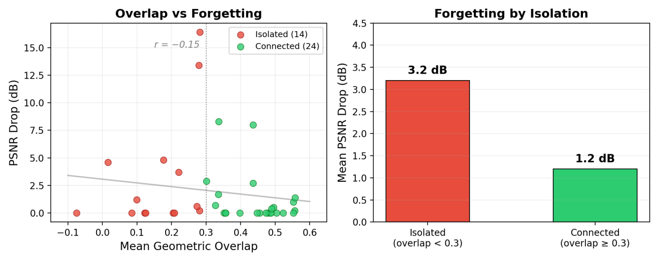


Fig. 7: Left: Per-keyframe PSNR drop vs. mean geometric overlap with other views. Views with low overlap (isolated on the hemisphere) suffer greater quality degradation during incremental training. Right: Isolated views lose 3× more PSNR than connected views on average.

dates in $\sim$1 s using standard rasterization, with no additional implementation beyond two render channels.

VI. Conclusion and Future Work

We presented a depth-variance guided NBV planning system for autonomous 3D reconstruction using incremental Gaussian Splatting on a PX4 UAV. The acquisition function computes per-pixel geometric uncertainty via alpha-blended depth second moments, maintaining 3× the dynamic range of alpha-based coverage metrics throughout the mission. The system achieves 70±4% surface coverage within 420 MB VRAM, compared to 40±3% for fixed orbit patterns.

A. Forgetting Analysis

The primary limitation of live incremental training is gradient interference: when views from opposite sides of the object train on shared Gaussians, their gradients conflict, degrading previously converged regions. We quantify this by measuring each keyframe's geometric overlap (cosine similarity of view directions) against its PSNR degradation across training rounds.

Fig. 7 shows that views with low geometric overlap (isolated on the viewing hemisphere) suffer 3× more quality degradation than well-connected views (3.2 dB vs. 1.2 dB mean drop, $r = -0.15$). This suggests that training view order and grouping, not iteration count, is the primary lever for improving live reconstruction quality.

B. Future Directions

Two directions follow from this analysis:

Visibility-aware training. Rather than shuffling views uniformly each epoch, views sharing high Gaussian visibility overlap could be grouped and trained consecutively. This would align gradient directions within each training batch, reducing interference on shared parameters. The overlap can be computed at negligible cost from view direction dot products.

Elastic weight consolidation. Following Kirkpatrick et al. [15], parameters important for well-converged views (measured via diagonal Fisher information) could be regularized against large updates when training on new views:

$$\mathcal{L}_{\text{EWC}} = \mathcal{L} + \lambda \sum_i F_i(\theta_i - \theta_i^*)^2 \tag{8}$$

where F_i is the per-parameter importance and θ_i^* are the parameters after convergence on earlier views. This would allow the model to incorporate new viewpoints without degrading existing reconstructions.

Deployment on physical hardware with onboard compute (e.g., NVIDIA Jetson Orin Nano, 4 GB shared memory, 15 W) and real depth sensors is the natural next step.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023.

[2] Y. Li, Z. Kuang, T. Li, Q. Hao, Z. Yan, G. Zhou, and S. Zhang, "ActiveSplat: High-Fidelity Scene Reconstruction Through Active Gaussian Splatting," *IEEE Robotics and Automation Letters*, vol. 10, no. 8, pp. 8099–8106, 2025.

[3] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2149–2154.

[4] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface Splatting," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. Association for Computing Machinery, 2001, pp. 371–378.

[5] 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," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 405–421.

[6] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[7] A. Guédon and V. Lepetit, "SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 5354–5363.

[8] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang, "PGSR: Planar-Based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 9, pp. 6100–6111, Sep. 2025.

[9] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 19 595–19 604.

[10] Z. Peng, T. Shao, Y. Liu, J. Zhou, Y. Yang, J. Wang, and K. Zhou, "RTG-SLAM: Real-time 3D Reconstruction at Scale using Gaussian Splatting," in *ACM SIGGRAPH 2024 Conference Papers*, ser. SIGGRAPH '24. Association for Computing Machinery, 2024.

[11] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.

[12] W. Jiang, B. Lei, and K. Daniilidis, "FisherRF: Active View Selection and Mapping with Radiance Fields Using Fisher Information," in *European Conference on Computer Vision (ECCV)*. Springer, 2024, pp. 413–430.

[13] Y. Tao, D. Ong, V. Murali, I. Spasojevic, P. Chaudhari, and V. Kumar, "RT-GuIDE: Real-Time Gaussian Splatting for Information-Driven Exploration," *IEEE Robotics and Automation Letters*, vol. 10, no. 11, pp. 11 594–11 601, Nov. 2025.

[14] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming Catastrophic Forgetting in Neural Networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.