POCKETSPLAT: MOBILE GAUSSIAN RECONSTRUCTION VIA WORLD-SPACE LATENT ALLOCATION

Anonymous authors
Paper under double-blind review



(a) Mobile Inference of Gaussian Scene (b) Quality vs. Count

Figure 1: **PocketSplat for mobile Gaussian reconstruction.** (a) Multi-view images are converted into a budgeted Gaussian representation and rendered on a mobile device. (b) The DL3DV quality–budget curve shows the reconstruction trade-off across representation budgets; bubble area denotes peak GPU memory.

ABSTRACT

Mobile Gaussian reconstruction must satisfy two requirements: the reconstruction model must execute within a device resource envelope, and the resulting Gaussian asset must expose a representation size suited to downstream mobile use. Existing feed-forward Gaussian reconstructors commonly decode dense, image-aligned candidates whose final cardinality is implicitly determined by the input resolution and number of views. We present PocketSplat, a feed-forward framework for budgeted mobile Gaussian asset construction. Given a prescribed output budget, PocketSplat organizes dense geometry-aware latent candidates in predicted world space, allocates exact integer capacity across local latent cells, and decodes complete Gaussian attributes only for retained candidates. Cell-conditioned latent fusion aggregates repeated multi-view evidence before decoding, while spatial responsibility decoding adapts Gaussian support after local sparsification. Experiments on DL3DV and out-of-distribution benchmarks establish a strong quality–budget trade-off against feed-forward Gaussian reconstruction baselines. On Mip-NeRF 360, PocketSplat executes directly on a target iPhone and constructs compact, higher-quality Gaussian assets substantially faster than a deployable streamed MVSplat variant; native MVSplat and DepthSplat exceed the device memory budget.

1 INTRODUCTION

Mobile 3D reconstruction would allow users to capture, revisit, and share everyday environments from a small set of images. Its practical value depends on two coupled requirements: the network must execute within device resources, and its Gaussian asset must remain usable after construction. Asset cardinality affects local storage, transmission, rasterization, and interactive processing, whereas the execution graph determines whether reconstruction can run on-device at all. Recent mobile systems address compact Gaussian rendering and representation processing (Du et al., 2026), while on-device optimization approaches expose coupled time and memory constraints (Guo et al., 2026). We therefore treat representation size as a first-class deployment variable. This paper studies an explicit *output* budget: the prescribed number of Gaussian primitives made available to a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

downstream mobile application. It lets an application select a representation-size operating point for its storage, transfer, and rendering requirements, rather than accepting an asset size fixed implicitly by reconstruction inputs. This formulation makes representation capacity visible to the mobile application rather than an incidental by-product of input sampling.

Feed-forward Gaussian reconstruction is a promising route to this setting. 3D Gaussian Splatting represents a scene with anisotropic, differentiably rasterized primitives (Kerbl et al., 2023); generalizable predictors amortize reconstruction across scenes rather than optimizing a representation from scratch for every capture. pixelSplat infers Gaussians from image pairs (Charatan et al., 2024), MVSplat incorporates multi-view stereo (Chen et al., 2024c), and DepthSplat couples Gaussian prediction with learned depth (Xu et al., 2025). However, dense image-aligned prediction commonly ties the final Gaussian population to image resolution and input views. Repeated observations of one local region can thus create many latent candidates before a representation decision is made. Dense multi-view intermediates can also be difficult to execute directly within a mobile memory envelope. Applying a budget after independently decoding every candidate leaves capacity allocation and asset size as secondary decisions. It also prevents an application from selecting its desired quality–asset-size operating point before scene construction begins.

Controllable-count and adaptive-density methods have begun to address representation size. SparseSplat predicts a sparse set of pixel-unaligned primitives but does not enforce an exact output budget or adapt support to local compression (Zhang et al., 2026b); EcoSplat adapts density in image space without associating candidates across views (Park et al., 2025); F^4Splat controls count through threshold-based pruning after decoding, without fusing grouped observations or adapting primitive support to local compression (Kim et al., 2026). Under a prescribed budget, three decisions remain coupled. First, image-space candidates must be associated in a common world-space structure despite viewpoint changes and imperfect geometry. Second, a retained candidate should incorporate information from repeated observations that will not become output primitives. Third, reducing local density changes how much of the scene a surviving primitive represents. Pixel ranking and post-hoc pruning do not coordinate these decisions. We instead formulate budgeted reconstruction as *world-space latent allocation*: distribute a prescribed population across predicted world-space latent cells, aggregate cell observations, and adapt retained support to local compression.

We present PocketSplat, a budget-constrained feed-forward framework for mobile Gaussian asset construction. A frozen multi-view geometry backbone supplies geometric priors and intermediate features, from which a trainable head produces dense geometry-aware latent candidates. We lift candidates to predicted world space and group them into latent cells, so repeated observations participate in a common allocation decision. The allocator assigns an exact feasible budget using local image detail, view-mean weighting that discounts repeated evidence, a support floor for zero-detail cells, and capacity completion. For each retained candidate, hidden fusion aggregates weighted cell information before complete Gaussian attributes are decoded. Spatial responsibility decoding (SRD) then modifies Gaussian scale through an analytic local-compression prior and a learned residual. Every operating point returns exactly its requested number of Gaussians while allocating capacity across predicted scene structure. The resulting asset cardinality is therefore independent of the dense candidate field used for geometric reasoning. It decouples dense geometric inference from the compact representation delivered to the mobile application after construction.

Our contributions are:

- The first feed-forward 3D Gaussian reconstruction framework for mobile asset construction, with an explicit output budget that exposes controllable representation cardinality for downstream mobile use.
- A world-space exact-budget allocator that combines local detail, cross-view redundancy weighting, zero-detail support, and capacity completion before attribute decoding.
- Cross-view hidden fusion and scale-only spatial responsibility decoding that adapt retained primitives to local sparsification, with in-domain, out-of-domain, and on-device evaluation.

2 Related Work

Gaussian representations and feed-forward reconstruction. Novel-view synthesis progressed from neural radiance fields and efficient grid or hash encodings (Mildenhall et al., 2021; Müller

et al., 2022) to 3D Gaussian Splatting, which enables real-time rasterization after scene-specific optimization (Kerbl et al., 2023). Subsequent work improves Gaussian fidelity and structure through anti-aliasing, anchors, and surface-oriented representations (Yu et al., 2024a; Lu et al., 2024; Huang et al., 2024; Guédon & Lepetit, 2024; Chen et al., 2024a; Yu et al., 2024b), while pruning, quantization, coding, and constrained optimization compact optimized scenes (Fan et al., 2024; Niedermayr et al., 2024; Navaneet et al., 2024; Chen et al., 2024b; Fang & Wang, 2024; Mallick et al., 2024). Generalizable reconstruction instead predicts renderable Gaussians without per-scene optimization, spanning monocular, image-pair, multi-view, variational, transformer, depth-guided, pose-free, and unconstrained settings (Szymanowicz et al., 2024; 2025; Charatan et al., 2024; Chen et al., 2024c; Wewer et al., 2024; Zhang et al., 2024; Xu et al., 2025; Smart et al., 2024; Ye et al., 2025; Zhang et al., 2025b; Jiang et al., 2025). Sparse and controllable-count approaches further adapt primitive placement, alignment, anchors, or density (Zhang et al., 2026b; Moreau et al., 2026; Zhang et al., 2026a; Park et al., 2025; Kim et al., 2026). PocketSplat complements these approaches by allocating an exact output population over grouped world-space latent candidates before attribute decoding.

Geometry priors and mobile deployment. Geometry foundation models provide depth, pointmap, matching, and camera priors (Wang et al., 2024; Leroy et al., 2024; Zhang et al., 2025a; Yang et al., 2025; Yin et al., 2023; Piccinelli et al., 2024; Yang et al., 2024; Wang et al., 2025; Lin et al., 2025). Efficiency work makes neural representations practical through baked grids, memory-efficient rendering, mobile rasterization, optimized Gaussian rasterizers, pruning, and localization (Hedman et al., 2021; Reiser et al., 2023; Chen et al., 2023; Feng et al., 2025; Hanson et al., 2025a;b); Mobile-GS and PocketGS address mobile Gaussian inference and on-device optimization (Du et al., 2026; Guo et al., 2026). PocketSplat uses frozen geometry priors to construct a budgeted Gaussian asset, combining direct mobile execution with explicit control of downstream representation cardinality across device-specific storage, transfer, rendering, and interactive processing requirements.

3 METHOD

PocketSplat is organized around an image-conditioned world-space latent field and a budget-conditioned Gaussian decoder. Given V multi-view images with intrinsics and camera-to-world poses, denoted by $\mathcal{X}_c$, and a feasible Gaussian budget B, it constructs

$$\mathcal{G}_B = f_\theta(\mathcal{X}_c, B) = \{\mu_i, \Sigma_i, c_i, o_i\}_{i \in \mathcal{S}_B}, \qquad |\mathcal{S}_B| = B. \tag{1}$$

Here $\mu_i \in \mathbb{R}^3$, $\Sigma_i \in \mathbb{R}^{3\times3}$, c_i, and o_i denote a world-space mean, covariance, degree-2 RGB spherical-harmonic coefficients, and opacity logit. The latent field consolidates dense multi-view geometric evidence before materialization; the decoder converts it into a mobile Gaussian asset whose cardinality is specified by B. Figure 2 summarizes this representation-to-asset construction.

3.1 WORLD-SPACE LATENT FIELD

A frozen multi-view transformer, camera modules, and a depth head produce four feature levels together with predicted depth, intrinsics, and world-to-camera transforms. A trainable GS head converts these outputs to a 32-channel hidden map and predicts geometry-related reliability and cross-view support. We instantiate the dense image-conditioned field as

$$\mathcal{Z} = \{(x_i, h_i, r_i, t_i)\}_{i=1}^{N}, \tag{2}$$

where x_i is a predicted world-space anchor and (h_i, r_i, t_i) are the candidate feature, reliability, and support. This is an allocation-time representation rather than a renderable Gaussian set: complete Gaussian attributes are produced only for the active support selected by the decoder. At runtime, $V = 4$, $H = 252$, and $W = 448$, giving $N = VHW = 451{,}584$ dense candidates. The same field supports every requested representation budget.

Latent structure. For a pixel $p = (u, v, 1)^\top$ in multi-view image j, let $\hat{D}_j$, $\hat{K}_j$, and $\hat{E}_j$ denote predicted depth, intrinsics, and world-to-camera pose. We first unproject the pixel in camera coordinates and transform it to world coordinates:

$$x_i^c = \hat{D}_j(u, v)\hat{K}_j^{-1}p, \qquad \begin{bmatrix} x_i \\ 1 \end{bmatrix} = \hat{E}_j^{-1} \begin{bmatrix} x_i^c \\ 1 \end{bmatrix}. \tag{3}$$

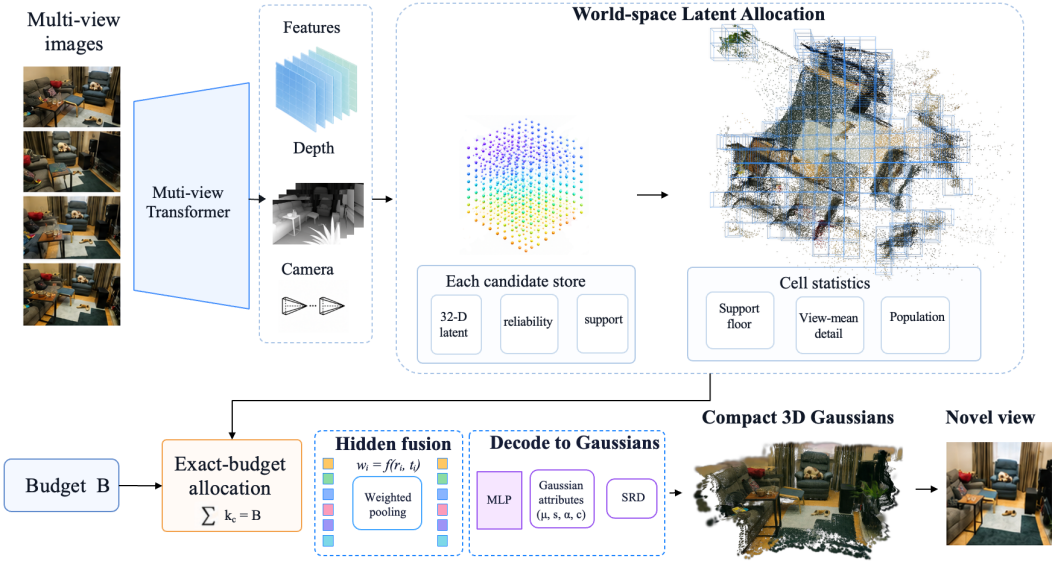

Figure 2: **PocketSplat overview.** Multi-view images enter a frozen multi-view transformer, yielding features, depth, and cameras. Each dense candidate stores a latent, reliability, and support. World-space anchors form cells with population, support-floor, and view-mean-detail statistics. Given B, exact-budget allocation returns quotas satisfying $\sum_c k_c = B$. Weighted hidden fusion pools each cell, after which retained candidates are decoded and adapted by SRD to form compact Gaussians for novel-view rendering.

The median finite positive multi-view depth d_{scene} defines a scene-relative cell width $v_s = 0.02 d_{\text{scene}}$. Candidates are assigned by $b_i = \lfloor x_i / v_s \rfloor$, yielding world-space cells $\mathcal{C}_c = \{i : b_i = c\}$ with populations $n_c = |\mathcal{C}_c|$. These cells provide the common unit for allocation and feature aggregation, while all decoded means μ_i remain continuous. By grouping in world space, observations that project to distant image coordinates but correspond to a common local region can participate in the same representation decision.

Cell statistics. The allocation policy gives capacity to cells according to local visual detail and cross-view evidence. Candidate detail e_i is the entropy of a 7×7 multi-view RGB neighborhood. Image preprocessing follows the fixed convention in Appendix C. Given local histogram probability $p_{i,b}$, detail is

$$e_i = -\sum_b p_{i,b} \log p_{i,b}, \tag{4}$$

with zero terms omitted. Let $\bar{e}_c$ be the mean detail in cell c. We define a zero-detail support floor, positive-detail capacity, detail mass, view count, and view-mean weight as

$$l_c = \mathbf{1}[n_c > 0 \ \wedge \ \bar{e}_c = 0], \qquad m_c = n_c - l_c, \tag{5}$$

$$W_{\text{sum}}(c) = m_c \bar{e}_c, \qquad V_c = \left| \left\{ j : \sum_{i \in \mathcal{C}_c \cap \text{view}(j)} e_i > 0 \right\} \right|, \tag{6}$$

$$W_{\text{vm}}(c) = \begin{cases} W_{\text{sum}}(c)/V_c, & W_{\text{sum}}(c) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The support floor retains one representative for an occupied zero-detail cell. The view-mean term discounts detail evidence replicated in multiple multi-view images, so capacity is assigned according to scene content rather than repeated image-plane evidence alone.

3.2 BUDGET-CONDITIONED GAUSSIAN ASSET DECODING

Exact-budget support. Let $L = \sum_c l_c$ and $R = B - L$. For feasible support constraints ($L \leq B \leq N$), residual quotas for positive-detail cells are

$$q_{\mathrm{res},c} = \min(m_c, \lambda W_{\mathrm{vm}}(c)), \qquad q_c = l_c + q_{\mathrm{res},c}, \tag{8}$$

where active-set water filling sets λ such that residual quotas sum to R. When positive-detail cells reach capacity, V2 capacity completion distributes remaining slots over the residual capacity of zero-detail cells. Corrected stochastic rounding then returns integer cell quotas

$$l_c \leq k_c \leq n_c, \qquad \sum_c k_c = B. \tag{9}$$

Thus, every feasible operating point produces the exact requested asset cardinality without leaving budget slots unused.

Within each cell, reproducible keys select k_c distinct candidates. At the dense operating point $B = N$, the allocator returns $\mathcal{S}_N = \mathrm{arange}(N)$ and $k_c = n_c$ for every cell; Appendix C gives the key construction.

Cell-conditioned Gaussian decoding and SRD. After allocation, we aggregate evidence within each cell before decoding its retained candidates. Normalized reliability r_i and support $t_i \in [0, 1]$ define pooling weights

$$w_i = (0.25 + 0.75 r_i)(0.5 + 0.5 t_i), \tag{10}$$

The pooled latent and selected feature update are

$$\bar{h}_c = \frac{\sum_{i \in \mathcal{C}_c} w_i h_i}{\sum_{i \in \mathcal{C}_c} w_i}, \qquad h'_i = h_i + g \odot (\bar{h}_c - h_i), \quad g = 0.5 \tanh(\theta_g). \tag{11}$$

The learned channel-wise gate g transfers cell-level evidence to selected candidates. Complete Gaussian attributes are projected from h'_i, so unselected candidates remain latent and do not enter the output asset.

Reducing a cell from n_c candidates to k_c changes the local support assigned to every retained primitive. For $k_c > 0$, we define the local compression ratio and the SRD parameters as

$$\rho_c = \log \frac{n_c}{k_c}, \qquad a_i = 1 + \tanh(u_i), \qquad \delta_i = 0.25 \tanh(v_i), \tag{12}$$

where a learned SRD head predicts u_i and v_i. After decoding a physical base scale, SRD applies

$$\gamma_i = \exp\left(\frac{a_i}{2}\rho_c + \delta_i\right), \qquad s'_i = \gamma_i s_i, \qquad \Sigma'_i = \gamma_i^2 \Sigma_i. \tag{13}$$

The factor $\sqrt{n_c/k_c}$ embedded in Eq. 13 allocates a larger local area responsibility to each retained Gaussian as the cell becomes sparse. The learned gate and residual adapt this geometric prior to the local scene. SRD modifies covariance after selection and leaves centers, rotations, opacity, spherical-harmonic appearance, image-plane offsets, and depth offsets unchanged.

3.3 TRAINING

We train on DL3DV chunks for 50,000 steps with batch size two, AdamW (2×10^{-4} learning rate and 0.01 weight decay), gradient clipping at 0.5, and VGG LPIPS. The multi-view geometry backbone remains frozen; the GS head, allocation-related modules, fusion module, and SRD head are optimized with RGB reconstruction, perceptual, and budget-selection supervision:

$$\mathcal{L} = \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{LPIPS}} \mathcal{L}_{\mathrm{LPIPS}} + \lambda_{\mathrm{sel}} \mathcal{L}_{\mathrm{selection}}. \tag{14}$$

Training uses the Full budget for the first 10,000 steps and then cycles through descending budgets with 100-step holds. The geometry grid is 252×448 after processing the configured 256×448 bucket. The iOS fixed-shape decoder is described separately in Appendix D.

4 EXPERIMENTS

We evaluate the evidence chain behind budgeted mobile Gaussian asset construction. We first measure reconstruction quality at exact Gaussian budgets, then test transfer beyond the training distribution, profile construction on GPU, and evaluate direct deployment on a target iPhone. Ablations isolate the contribution of world-space allocation, spatial responsibility decoding, and allocation before complete attribute projection. This organization connects the evaluation to controllable assets, reconstruction quality, and feasible mobile execution. The resulting evaluation directly links each empirical result to a component of the proposed budgeted construction pipeline.

4.1 EXPERIMENTAL SETUP

Datasets and protocol. The in-domain benchmark uses DL3DV-Benchmark, the official test set of the DL3DV dataset. Each scene uses four multi-view images and 50 evaluation frames at 252×448. We evaluate frozen checkpoints out of distribution on Tanks & Temples and MegaDepth. Section 4.3 reports a separate on-device evaluation on Mip-NeRF 360. Together, these settings test in-distribution reconstruction, transfer to unfamiliar scenes, and deployed mobile behavior. They also expose the same model family to both standard benchmark scenes and real mobile capture sequences.

Baselines, budgets, and metrics. We compare pixelSplat, MVSplat, DepthSplat, TranSplat, F^{4}Splat, and PocketSplat. The first four use their native Gaussian counts and F^{4}Splat uses its native threshold-based count mechanism. One PocketSplat checkpoint is evaluated at $B \in \{113{,}000, 226{,}000, 338{,}000, 451{,}584\}$. We report PSNR after clipping RGB to $[0, 1]$, SSIM with an 11-pixel Gaussian window, and VGG LPIPS. Metrics are averaged within each scene and then over scene means. SRD confidence intervals use paired scene bootstrapping with 10,000 resamples. Training settings appear in Section 3.3. All evaluations use these definitions to report reconstruction quality across the stated operating configurations. This protocol makes the reported reconstruction measures directly comparable across the budget settings of the proposed method.

4.2 RECONSTRUCTION QUALITY ACROSS DATASETS

In-domain quality and exact budget control. Table 1 establishes the quality–budget operating curve on DL3DV. At 226K, PocketSplat uses 16.4% of pixelSplat's reported Gaussian count while reaching 22.790 dB, a 1.559 dB PSNR improvement. Increasing the budget from 226K to 338K adds 0.332 dB PSNR and reduces LPIPS from 0.2142 to 0.1887. The 338K configuration uses 25.2% fewer Gaussians than Full while changing PSNR by only $+0.0526$ dB and SSIM by -0.0034; Full provides the lowest LPIPS at 0.1776. Even Light at 113K reaches 21.018 dB, exceeding pixelSplat while using 8.2% of its reported count. This establishes high-quality operating points for compact mobile Gaussian assets.

The comparison also separates exact allocation from threshold-driven count control. At similar 113K, 226K, and 338K populations, PocketSplat exceeds F^{4}Splat by 2.477, 4.196, and 4.518 dB, respectively. At 338K, it improves over pixelSplat, MVSplat, DepthSplat, and TranSplat by 1.891, 2.467, 5.305, and 2.454 dB. Figure 3 visualizes this progression: the 226K and 338K assets preserve scene layout and appearance while exposing materially smaller representation sizes. Together, the table and figure show that the requested budget selects a usable quality–asset-size operating point rather than merely discarding a fixed dense output.

Cross-dataset generalization. Frozen-checkpoint evaluation on Tanks & Temples and MegaDepth retains the budget-controlled behavior beyond DL3DV. PocketSplat PSNR peaks at 226K, reaching 17.965 dB on Tanks & Temples and 17.178 dB on MegaDepth; Full gives the lowest LPIPS, 0.3335 and 0.3450. The 226K configuration improves on the strongest listed baseline PSNR by 2.126 dB on Tanks & Temples and 0.965 dB on MegaDepth. At 338K, PocketSplat maintains comparable PSNR while reducing LPIPS to 0.3476 and 0.3562. Figure 4 shows the resulting reconstructions on Meetingroom, Truck, Church, and Caterpillar, and Table 1 gives the complete measurements. These results show that the same allocation rule transfers a family of representation budgets across datasets. The budget-control behavior thus extends beyond the scenes used to train the reconstruction model. This consistency supports the use of one checkpoint across the reported

Table 1: Reconstruction quality across in-domain and out-of-distribution datasets. Higher PSNR/SSIM and lower LPIPS are preferable.

		DL3DV			Tanks & Temples			MegaDepth		
Method	Gaussians	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelSplat	1,376,256	21.231	0.7119	0.3463	15.839	0.3968	0.5665	16.213	0.5036	0.5079
MVSplat	458,752	20.655	0.6646	0.2694	14.783	0.3471	0.4971	14.779	0.4218	0.4671
DepthSplat	458,752	17.817	0.5370	0.3508	13.722	0.3224	0.5581	14.450	0.4079	0.4869
TranSplat	458,752	20.668	0.6856	0.2630	14.692	0.3423	0.4987	14.880	0.4441	0.4525
F^4Splat (113K)	111,234	18.541	0.5230	0.3927	12.910	0.3181	0.6146	15.359	0.4230	0.5247
F^4Splat (226K)	222,468	18.594	0.5275	0.3741	12.911	0.3177	0.6031	15.351	0.4226	0.5092
F^4Splat (338K)	332,718	18.604	0.5292	0.3645	12.911	0.3183	0.5962	15.365	0.4227	0.5013
F^4Splat (Full)	451,584	18.604	0.5288	0.3619	12.876	0.3197	0.5913	15.337	0.4211	0.4985
PocketSplat(113K)	113,000	21.018	0.6690	0.2952	17.371	0.4621	0.4285	16.550	0.4952	0.4382
PocketSplat(226K)	226,000	22.790	0.7399	0.2142	17.965	0.4947	0.3711	17.178	0.5335	0.3806
PocketSplat(338K)	338,000	23.122	0.7561	0.1887	17.956	0.4982	0.3476	17.144	0.5396	0.3562
PocketSplat(Full)	451,584	23.070	0.7595	0.1776	17.712	0.4925	0.3335	16.626	0.5274	0.3450

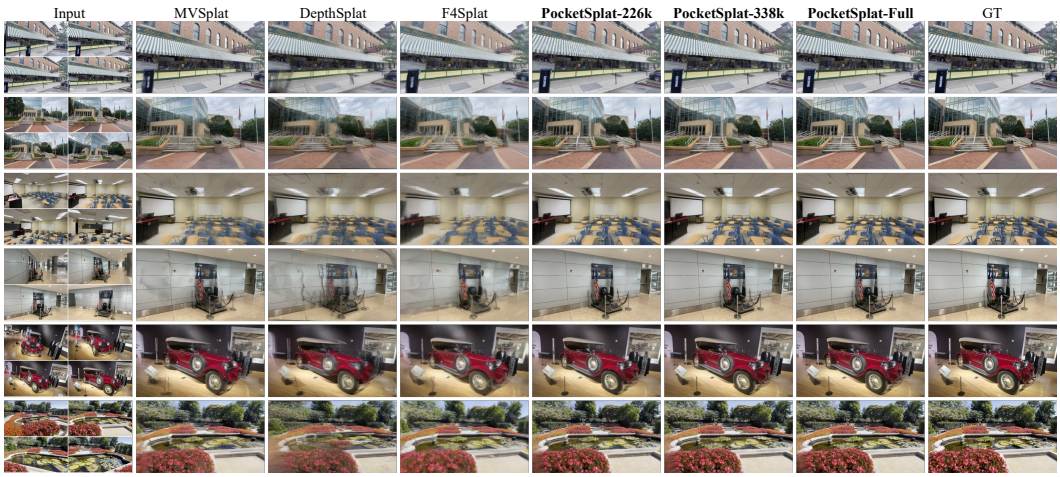

Figure 3: Qualitative DL3DV comparison with six examples. Columns show context inputs, MVSplat, DepthSplat, F^4Splat, PocketSplat at 226K, 338K, and Full, and ground truth.

representation-size operating points. It shows that the allocation mechanism remains useful when scene content and capture conditions differ from the training distribution.

4.3 Mobile Evaluation

Protocol and deployment path. We deploy on Mip-NeRF 360 (Barron et al., 2022). Frames 0, 2, 4, and 6 provide four context inputs for each sequence; evaluation uses the first 20 remaining frames. All methods receive 256×448 RGB inputs with shared resize, crop, and camera preprocessing. The four PocketSplat user settings—Light, Balanced, Quality, and Dense—correspond to increasing output budgets. We use a shared FP32 hybrid execution path with Core ML visual backbones and ONNX Runtime for remaining computation. This protocol fixes the available multi-view evidence and output resolution for every reported mobile result.

PocketSplat executes directly through its standard Core ML and ONNX Runtime path. Native MVSplat and DepthSplat exhaust memory on the target iPhone because their multi-view cost volumes materialize depth candidates. We therefore report MVSplat-Stream: an engineering-modified MVSplat path that streams eight depth candidates at a time, reduces each chunk to channel correlations, writes it to the final cost-volume buffer, and releases its transient storage before processing the next chunk. Its Core ML backbone runs on CPU and remaining stages use ONNX Runtime. MVSplat-Stream completes deployment, whereas DepthSplat remains OOM after streamed cost-volume and staged execution. Appendix D and Table 4 detail these direct-deployment outcomes.

Quality, time, and asset size. Table 2 jointly measures direct deployability, representation size, construction time, inference time, peak footprint, and reconstruction quality. PocketSplat runs at

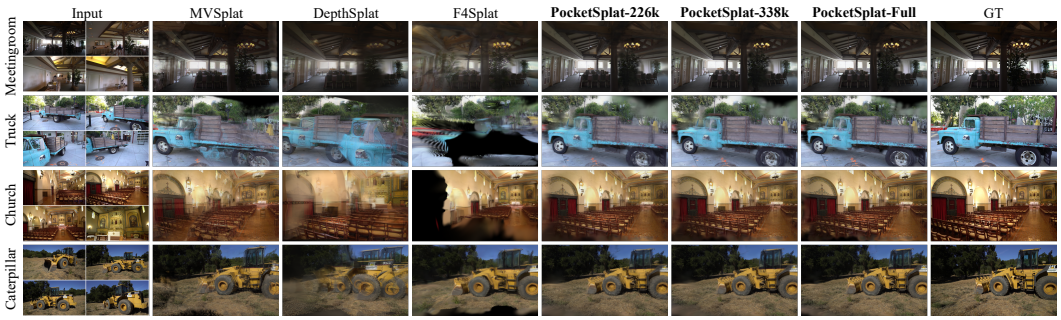

Figure 4: Tanks & Temples qualitative comparison on Meetingroom, Truck, Church, and Caterpillar. Columns show context inputs, MVSplat, DepthSplat, PocketSplat at 226K, 338K, and Full, and ground truth.

Table 2: On-device Mip-NeRF 360 evaluation. Times are seconds, peak footprint is reported in GB, and higher PSNR/SSIM and lower LPIPS are preferable.

Method / setting	Status	Gaussians	Reconstruction	Inference	Peak	PSNR	SSIM	LPIPS
MVSplat	OOM	458,752	–	–	–	–	–	–
DepthSplat	OOM	458,752	–	–	–	–	–	–
MVSplat-Stream	Deployable	458,752	42.00	40.16	1.99	17.875	0.5793	0.3165
PocketSplat Light	Deployable	112,896	10.67	10.00	2.68	20.018	0.6690	0.2952
PocketSplat Balanced	Deployable	225,792	10.20	9.24	2.89	21.790	0.6343	0.2142
PocketSplat Quality	Deployable	338,000	10.95	9.59	2.64	22.132	0.6503	0.1923
PocketSplat Dense	Deployable	451,584	10.50	9.27	2.66	22.034	0.6491	0.1897

every budget, constructing assets in 10.20–10.95 s and completing inference in 9.24–10.00 s. Balanced uses half the Dense primitive population, gives the lowest reconstruction and inference times, and reaches 21.790 dB. Quality reaches the strongest PSNR and SSIM, 22.132 dB and 0.6503, while Dense gives the lowest LPIPS, 0.1897. These settings let a mobile application select between compactness and reconstruction fidelity without changing the deployed model.

MVSplat-Stream uses 458,752 Gaussians, takes 42.00 s for reconstruction and 40.16 s for inference, and reaches 17.875 dB. In contrast, PocketSplat Light uses 75.4% fewer Gaussians while improving PSNR by 2.143 dB, SSIM by 0.0897, and LPIPS by 0.0213. PocketSplat Quality improves PSNR by 4.257 dB and SSIM by 0.0710 over MVSplat-Stream, with 338,000 output primitives. PocketSplat Dense reduces construction and inference time by 75.0% and 76.9%, respectively, relative to MVSplat-Stream. Its peak footprint ranges from 2.64 to 2.89 GB across user budgets, while the unmodified MVSplat and DepthSplat paths do not complete within the target-device memory envelope. Figure 5 shows the corresponding progression on Room, Bonsai, and Counter.

4.4 Ablation Studies

We isolate the components that turn the dense candidate field into a budgeted Gaussian asset on DL3DV. The following studies identify the contributions of allocation, support adaptation, and decode ordering. Each ablation holds the requested budget fixed while changing the corresponding construction mechanism or adaptation component.

World-space allocation. Uniform random candidate selection preserves the requested output count but ignores scene structure. World-space allocation improves PSNR by 3.0049 dB at 113K, 3.6915 dB at 226K, and 2.1916 dB at 338K (Table 3). The corresponding Random-to-PocketSplat changes are 18.0134 to 21.0183 dB, 19.0984 to 22.7899 dB, and 20.9305 to 23.1221 dB. These results isolate the benefit of assigning fixed capacity across world-space latent cells instead of selecting a count-matched random subset. The gain at every sparse operating point shows that count alone does not determine representation quality. Allocating the same primitive population according to predicted world-space structure is therefore essential at tight budgets.

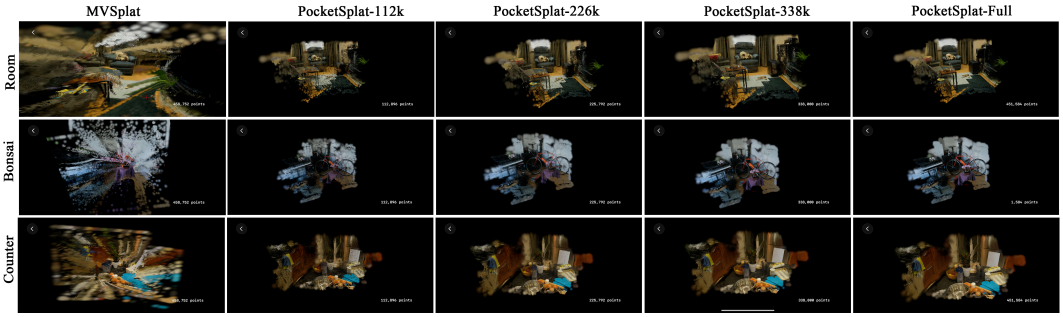

Figure 5: Mobile visualizations on Mip-NeRF 360: Room, Bonsai, and Counter (rows), comparing MVSplat-Stream and PocketSplat at Light, Balanced, Quality, and Dense budgets (columns). PocketSplat budgets are 112,896, 225,792, 338,000, and 451,584, as reported in Table 2.

Table 3: Quality ablations on DL3DV (PSNR, dB).

Budget	Random	World-space + SRD	No SRD	SRD gain [95% CI]
113,000	18.0134	21.0183	17.6617	3.3567 [3.1792, 3.5340]
226,000	19.0984	22.7899	21.9609	0.8290 [0.7672, 0.8925]
338,000	20.9305	23.1221	22.9514	0.1707 [0.1492, 0.1932]
Full	identity	23.0695	22.9190	0.1505 [0.1163, 0.1847]

Spatial responsibility decoding. For SRD, all branches start from a common dense-10K parent and train for a further 40,000 steps with the same seed, differing only in SRD enablement. SRD improves PSNR by 3.3567 dB at 113K, 0.8290 dB at 226K, and 0.1707 dB at 338K. It also improves every one of the 140 scenes at 113K and 226K. The paired 95% bootstrap intervals remain positive across all reported budgets (Table 3), linking the largest gain to the most severe local compression and demonstrating the value of adapting Gaussian support after allocation.

Allocation before complete decoding. We compare decode-before-allocate (DBA) and budget-before-decoding (BBD) using identical selected candidates. At 113K, BBD reduces attribute projection from 1.308 to 0.411 ms and construction from 166.089 to 163.216 ms. Attribute-projection reductions are 68.5%, 44.9%, and 21.3% at 113K, 226K, and 338K, respectively, because only selected candidates undergo complete attribute projection. The corresponding construction reductions are 1.73%, 1.09%, and 0.66%. At Full, DBA and BBD require 96.75 and 96.65 ms because every candidate is retained. Appendix F reports the full budget sweep.

Conclusion

Mobile Gaussian reconstruction must construct an asset that a downstream device can store, transmit, render, and manipulate, rather than predict a dense primitive set at a fixed input-dependent size. We introduce PocketSplat, a feed-forward framework that constructs a Gaussian asset at an explicit representation budget. Its image-conditioned world-space latent field associates repeated multi-view observations in latent cells, whose local detail, view-mean evidence, and population determine exact integer capacity. A budget-conditioned decoder fuses cell information into retained latents, projects complete Gaussian attributes only for the selected support, and applies spatial responsibility decoding to adapt covariance under local sparsification. Results on DL3DV, Tanks & Temples, MegaDepth, and on-device Mip-NeRF 360 demonstrate quality–asset-size operating points and direct mobile execution across user-selected budgets. These results establish world-space latent allocation as an effective formulation for feed-forward mobile Gaussian reconstruction.

References

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469. IEEE, 2022.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19457–19467. IEEE, 2024.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 31 (9):6100–6111, 2024a.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pp. 422–438. Springer, 2024b.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pp. 370–386. Springer, 2024c.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16569–16578. IEEE, 2023.

Xiaobiao Du, Yida Wang, Kun Zhan, and Xin Yu. Mobile-gs: Real-time gaussian splatting for mobile devices. *arXiv preprint arXiv:2603.11531*, 2026.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European conference on computer vision*, pp. 165–181. Springer, 2024.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, et al. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26652–26662, 2025.

Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5354–5363. IEEE, 2024.

Wenzhi Guo, Guangchi Fang, Shu Yang, and Bing Wang. Pocketgs: On-device training of 3d gaussian splatting for high perceptual modeling. *arXiv preprint arXiv:2601.17354*, 2026.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546. IEEE, 2025a.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5949–5958. IEEE, 2025b.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5875–5884, 2021.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Injae Kim, Chaehyeon Kim, Minseong Bae, Minseok Joo, and Hyunwoo J. Kim. F 4 splat: Feed-forward predictive densification for feed-forward 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2026.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20654–20664. IEEE, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Arthur Moreau, Richard Shaw, Michal Nazarczuk, Jisu Shin, Thomas Tanay, Zhensong Zhang, Songcen Xu, and Eduardo Pérez-Pellitero. Off the grid: Detection of primitives for feed-forward 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11756–11766, 2026.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pp. 330–349. Springer, 2024.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10349–10358. IEEE, 2024.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. Ecosplat: Efficiency-controllable feed-forward 3d gaussian splatting from multi-view images. *arXiv preprint arXiv:2512.18692*, 2025.

Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10106–10116. IEEE, 2024.

Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (ToG)*, 42(4):1–12, 2023.

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10208–10217, 2024.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, Joao F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. In *2025 International Conference on 3D Vision (3DV)*, pp. 670–681. IEEE, 2025.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *European conference on computer vision*, pp. 456–473. Springer, 2024.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16453–16463. IEEE, 2025.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21924–21935. IEEE, 2025.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in neural information processing systems*, 37:21875–21911, 2024.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *International Conference on Learning Representations*, volume 2025, pp. 54009–54033, 2025.

Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9009–9019. IEEE, 2023.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456. IEEE, 2024a.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (ToG)*, 43(6):1–13, 2024b.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Forrester Cole, Deqing Sun, Ming-Hsuan Yang, et al. Monst3r: A simple approach for estimating geometry in the presence of motion. In *International Conference on Learning Representations*, volume 2025, pp. 82863–82886, 2025a.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2024.

Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21936–21947. IEEE, 2025b.

Xiaoxue Zhang, Xiaoxu Zheng, Yixuan Yin, Tiao Zhao, Kaihua Tang, Michael Bi Mi, Zhan Xu, and Dave Zhenyu Chen. Anchorsplat: Feed-forward 3d gaussian splatting with 3d geometric priors. *arXiv preprint arXiv:2604.07053*, 2026a.

Table 4: Mobile deployment outcomes on the target iPhone under the four-view 256×448 FP32 protocol.

Method	Direct path	Final status	Mobile configuration
PocketSplat	Deployable	Deployable	Core ML + ONNX Runtime
MVSplat	OOM	OOM	Direct cost-volume path
MVSplat-Stream	–	Deployable	CPU Core ML + streamed cost volume ($D = 8$)
DepthSplat	OOM	OOM	Streamed cost volume + staged execution

Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. Sparsesplat: Towards applicable feed-forward 3d gaussian splatting with pixel-unaligned prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5049–5058, 2026b.

AI USE STATEMENT

Generative AI tools were primarily used to polish the manuscript's language and presentation. The authors are responsible for all technical content, experimental results, citations, and conclusions.

A MOBILE EXPORT AND PARITY REQUIREMENTS

Export artifacts comprise preprocessing and post-allocation ONNX graphs, an FP32 pre-allocator Core ML package, and six DA3 Core ML stage packages. Export manifests should retain checkpoint/config hashes, exporter revision, precision, input shapes, model hashes, and interface names. Equivalent Core ML rewrites include reshape/expand in place of fixed meshgrids, elementwise products or matrix multiplication in place of some einsums, non-in-place pose decoding, and rank-5 cross-view aggregation. Their numerical behavior must be checked, especially near voxel boundaries.

B REPORT CONSISTENCY AND ADDITIONAL EXPERIMENTAL DETAILS

This appendix provides supplementary experimental details and implementation information.

C ALLOCATION IMPLEMENTATION DETAILS

Candidate detail is computed after restoring context RGB from ImageNet normalization, converting to luma $0.299R + 0.587G + 0.114B$, and quantizing to 256 levels. Within each cell, SHA256/splitmix64-derived pseudorandom keys use scene identity, a fixed seed, budget, and cell identity to select the quota. Reliability is normalized and clipped between its 20th and 95th percentiles before reliability–support weighted pooling.

D HYBRID IOS DEPLOYMENT AND IMPLEMENTATION DETAILS

This section describes the on-device execution of geometry, allocation, fusion, decoding, and rendering.

Runtime partition. The PocketSplat deployment uses ONNX Runtime for preprocessing and post-allocation decoding, six Core ML stages for the backbone, Core ML FP32 for depth, cameras, and GS pre-allocation, Swift for voxel grouping, V2 allocation and hidden fusion, and Metal for viewing. Table 5 summarizes the tensor interfaces. For the cost-volume baselines, the common direct hybrid path exhausts memory. MVSplat becomes deployable after streaming its 32 depth candidates in chunks of eight and placing the Core ML backbone on CPU. DepthSplat remains out of memory after streamed cost-volume construction and staged execution.

Output budget versus decoding work. The post graph retains N output rows at all budgets. Unselected entries receive opacity logit -20, and the PLY writer filters values ≤ -19. The selected

Table 5: Fixed mobile interfaces; batch size is one. $V = 4$, $H = 252$, $W = 448$, $N = 451{,}584$. All listed shapes include the batch dimension.

Boundary / tensor	Shape
Preprocess image / intrinsics / C2W	$[1, 4, 3, H, W]$ / $[1, 4, 3, 3]$ / $[1, 4, 4, 4]$
Camera token	$[1, 4, 1024]$
Backbone state / raw feature	$[1, 4, 577, 1024]$ / $[1, 4, 577, 2048]$
Pre-allocator hidden / depth	$[1, 4, 32, H, W]$ / $[1, 4, H, W]$
Fused hidden to post graph	$[1, N, 32]$
Post means / covariance / opacity	$[1, N, 3]$ / $[1, N, 3, 3]$ / $[1, N]$

count is exactly B under the allocation constraints. Covariance is decomposed into scale and rotation for export. Current UI budgets are $451{,}584$, $338{,}000$, $225{,}792$, and $112{,}896$.

Camera contract. ARKit provides OpenGL camera-to-world poses. Conversion uses $C2W_{\mathrm{cv}} = C2W_{\mathrm{gl}}\,\mathrm{diag}(1, -1, -1, 1)$. Images and intrinsics follow the GPU resize/crop transform. Preprocessing constructs cameras relative to the first view and normalizes translation using the median multi-view relative translation scale. Before export, similarity alignment maps predicted context cameras to the capture coordinate system. This contract preserves geometric consistency between captured multi-view images and the exported world-space Gaussian asset.

E Cross-Dataset Results and Protocol

Table 1 reports frozen-checkpoint results on Tanks & Temples and MegaDepth. Within PocketSplat, 226K gives the largest PSNR on both datasets (17.965 and 17.178 dB), Full gives the lowest LPIPS (0.3335 and 0.3450), and 338K gives the highest SSIM. These results summarize cross-dataset quality across all reported budgets.

At 338K, PocketSplat records 17.956 dB on Tanks & Temples and 17.144 dB on MegaDepth. Table 1 reports the complete cross-dataset results. The intermediate budgets remain useful choices outside the training distribution.

F Efficiency Protocols and Complete Measurements

Measurement protocol. We use the deployment efficiency round in the combined report on an NVIDIA GeForce RTX 4080 SUPER. Eight fixed scenes receive three warm-up and ten measured repeats per scene in a fresh process per arm. T_{INFER50} measures the GPU-resident interval from four input images and cameras to 50 rendered RGBs; loading, disk I/O, image decoding, CPU preprocessing, host-to-device copies, metric computation, and saving are excluded. We report the source medians. Peak memory is measured after load and warm-up in a separate process; values are reported in GB. The protocol fixes the timing interval and aggregation procedure for every evaluated GPU configuration. These fixed conditions isolate runtime variation from scene sampling and transient initialization effects. They also make the construction-time and peak-memory summaries reproducible under a single clearly defined profiling path.

Runtime and memory trade-offs. At Full, the deployment profiling path reports 322.3 ms and a 2.705 GB peak (Table 6). Its total is close to MVSplat's 322.8 ms, while its reported peak is lower than the other methods in this table. This profile records the complete GPU construction path at the dense operating point.

The four PocketSplat rows in Table 6 show that reducing the output from Full to 113K increases T_{INFER50} from 322.3 to 601.8 ms while leaving peak memory at 2.705 GB. Dense features and allocation overhead dominate the savings from fewer rendered primitives. The allocation-order ablation isolates the budget-dependent decoding effect.

Construction effect of selecting before decoding. The paired experiment compares decode-before-allocate (DBA) and budget-before-decoding (BBD) with identical allocation and selected candidates, eight scenes, and 20 measured repeats. Table 7 shows that skipping attribute projection

Table 6: GPU efficiency for the deployment path. Times are median T_{INFER50} in milliseconds and peak memory is reported in GB.

Method	T_{INFER50} (ms)↓	Peak (GB)↓
pixelSplat	837.7	8.483
MVSplat	322.8	13.885
DepthSplat	306.2	7.519
TranSplat	537.8	3.552
F^{4}Splat (113K)	301.8	17.796
F^{4}Splat (226K)	306.4	17.796
F^{4}Splat (338K)	314.9	17.796
F^{4}Splat (Full)	334.0	17.796
PocketSplat(113K)	601.8	2.705
PocketSplat(226K)	593.2	2.705
PocketSplat(338K)	466.3	2.705
PocketSplat(Full)	322.3	2.705

Table 7: Execution-order ablation. Values are median milliseconds per allocation call.

	Attribute projection		Construction		Construction
Budget	DBA	BBD	DBA	BBD	reduction
113,000	1.308	0.411	166.089	163.216	1.730%
226,000	1.289	0.710	174.178	172.276	1.092%
338,000	1.308	1.029	180.567	179.378	0.659%
Full	1.29	1.32	96.75	96.65	0.096%

on unselected rows reduces projection time by 68.5% at 113K and construction time by 1.73%. At that budget, allocation itself takes approximately 127 ms, versus 1.308 ms for dense projection.

G Archived OOD Qualitative Panels

The following OOD panels are reproduced from the supplied experiment directory without altering individual renders. They preserve the original method labels. Their image-level provenance and budgets are not specified in the supplied summaries. They are included as archived illustrations, not controlled matched-budget evidence. The panels nevertheless provide a visual record of the supplied out-of-distribution reconstructions. They complement the quantitative transfer results with examples from the archived evaluation package. Their consistent method ordering offers an additional qualitative view of the representation budgets reported in the quantitative transfer evaluation.



Figure 6: MegaDepth qualitative comparison with five examples. Columns show context inputs, MVSplat, DepthSplat, F^{4}Splat, PocketSplat at 226K, 338K, and Full, and ground truth.