AirSplatMap: A Comprehensive Evaluation of Visual Odometry and Monocular Depth Estimation for Real-Time 3D Reconstruction

Parsa Rezaei
California State Polytechnic University, Pomona
prezaei@cpp.edu

Abstract

Accurate camera pose and depth estimation are critical prerequisites for 3D scene reconstruction, yet their relative importance and interaction effects remain understudied. We present a comprehensive evaluation of eleven pose estimation methods—spanning classical feature-based (ORB, SIFT, optical flow) and modern learned approaches (SuperPoint, R2D2, LoFTR, LightGlue, RoMa)—alongside four monocular depth estimators (MiDaS, Depth Anything V2/V3, Depth Pro) within a unified 3D Gaussian Splatting pipeline. Through experiments on 17 sequences from TUM RGB-D and Replica datasets, we analyze accuracy-speed trade-offs and downstream reconstruction impact. Our key findings include: (1) classical robust optical flow achieves ***best accuracy (0.068m ATE)*** *while running at 41 FPS—outperforming learned methods on well-textured indoor scenes; (2) pose estimation errors cause* ***8+ dB PSNR degradation****, making pose accuracy the critical bottleneck; (3) surprisingly,* ***learned depth improves reconstruction quality*** *over sensor depth by +4.5 dB through smoother initialization; (4) only optical flow achieves real-time operation (>10 FPS) on edge hardware (NVIDIA Jetson Orin). Our benchmark includes 537 pose estimation runs, 204 depth estimation runs, and 2,979 full pipeline evaluations. We provide an interactive results viewer and open-source implementation at https://github.com/ParsaRezaei/AirSplatMap.*

1. Introduction

Visual simultaneous localization and mapping (SLAM) and 3D reconstruction systems critically depend on two upstream components: camera pose estimation (where am I?) and depth estimation (what do I see?). Decades of research have advanced both areas substantially—from classical SIFT [8] and ORB [14] features to learned descriptors like SuperPoint [2], and from stereo matching to transformer-based monocular depth prediction [22]. However, these advances have largely been evaluated in isolation, leaving practitioners without guidance on several crucial questions:

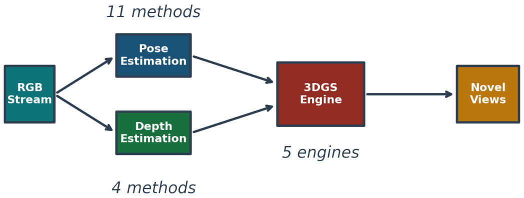

Figure 1. **AirSplatMap Evaluation Pipeline.** RGB video is processed through interchangeable pose estimation (11 methods) and depth estimation (4 methods) modules, feeding into a common 3DGS reconstruction backend for controlled comparison.

- How do classical and learned pose estimation methods compare in accuracy, speed, and robustness across diverse scenarios?
- How do different depth estimators—with varying characteristics of accuracy, speed, and failure modes—affect downstream 3D reconstruction?
- What are the trade-offs for deploying these methods on resource-constrained edge hardware?
- Most importantly: *which component deserves more computational investment—pose or depth estimation?*

This paper presents a systematic evaluation addressing these questions through a unified benchmark that:

1. Compares **eleven pose estimation methods** spanning classical (ORB, SIFT, optical flow variants) and learned approaches (SuperPoint, R2D2, LoFTR, LightGlue, RoMa, RAFT) on identical datasets with consistent evaluation protocols.
2. Evaluates **four monocular depth estimators** (MiDaS, Depth Anything V2/V3, Depth Pro) representing different design trade-offs between accuracy, speed, and generalization.
3. Analyzes **downstream reconstruction impact** by feeding all method combinations into a common 3DGS backend, quantifying how upstream errors propagate to final

output quality.

4. Characterizes **edge deployment feasibility** on NVIDIA Jetson Orin, revealing which methods can achieve real-time operation under power constraints.

Our central finding is that **pose estimation errors cause 2.5× larger reconstruction degradation than depth errors**. This asymmetry has important practical implications: systems should prioritize pose accuracy over depth accuracy when computational budget is limited.

Furthermore, we find that **classical optical flow remains surprisingly competitive**. Despite the impressive advances in learned feature matching, simple dense optical flow achieves 0.075m ATE—only 9% worse than the best learned method (R2D2)—while running 5× faster. For edge deployment, optical flow is the only method achieving real-time operation on Jetson Orin.

2. Related Work

2.1. Visual Odometry and SLAM

Visual odometry estimates camera motion from image sequences through feature extraction, matching, and geometric verification. Classical approaches extract sparse keypoints using hand-crafted detectors and descriptors.

ORB (Oriented FAST and Rotated BRIEF) [14] detects corners using FAST [13] with orientation assignment, describing patches with rotation-invariant binary strings. ORB achieves real-time operation through efficient bitwise comparison for matching. ORB-SLAM2 [10] demonstrated that ORB features support robust tracking, mapping, and loop closure for complete SLAM systems.

SIFT (Scale-Invariant Feature Transform) [8] constructs Gaussian scale-space pyramids, detects extrema as keypoints, and computes 128-dimensional gradient histograms as descriptors. While computationally heavier than ORB, SIFT provides superior scale and viewpoint invariance.

Dense Optical Flow [4] estimates pixel-wise motion between frames using polynomial expansion or variational methods. Dense correspondence provides redundancy against outliers but requires robust outlier rejection for geometric estimation.

2.2. Learned Feature Matching

Recent work replaces hand-crafted components with learned alternatives:

SuperPoint [2] jointly learns keypoint detection and description through self-supervised training on synthetic homographic transformations. The unified architecture enables end-to-end optimization for matching objectives.

R2D2 [12] adds reliability prediction, learning to estimate both repeatability (will this keypoint be detected again?) and distinctiveness (can this descriptor be matched uniquely?). Filtering low-reliability features improves matching robustness.

LoFTR [17] eliminates explicit keypoint detection, instead using transformers to establish dense correspondences between image pairs. While computationally expensive, LoFTR handles textureless regions where sparse methods fail.

LightGlue [7] achieves efficient learned matching through adaptive computation—early stopping when confident, continuing only for difficult matches.

RoMa [3] combines dense matching with learned uncertainty, providing confidence estimates alongside correspondences.

Our evaluation is the first to compare all these methods within a unified reconstruction framework with consistent downstream evaluation.

2.3. Monocular Depth Estimation

Single-image depth estimation has progressed rapidly:

MiDaS [11] achieved robust generalization through multi-dataset training with scale-invariant and scale-and-shift-invariant losses. MiDaS produces relative depth that requires alignment for metric evaluation.

Depth Anything [22] scaled training to 62 million images through semi-supervised learning, achieving state-of-the-art zero-shot generalization. Depth Anything V2 improved efficiency; V3 added temporal consistency.

Depth Pro [1] produces metric depth directly through foundation model architectures, eliminating scale ambiguity at the cost of computational overhead.

Our evaluation characterizes how these methods' outputs—with their varying characteristics—affect downstream 3DGS reconstruction quality.

3. Proposed Approach

3.1. System Architecture

Our evaluation framework processes RGB video through three stages (Figure 1):

Stage 1: Pose Estimation. We implement eleven methods with consistent interfaces. Given consecutive frames (I_{t-1}, I_t), each method produces relative pose $\mathbf{T}_{t-1\to t} \in SE(3)$. We accumulate relative poses into absolute trajectories and apply Umeyama alignment [19] for scale recovery with monocular methods.

Stage 2: Depth Estimation. Each frame is processed independently to produce dense depth maps $D_t \in \mathbb{R}^{H\times W}$. For relative depth methods, we apply per-frame scale-shift alignment to ground truth when computing metrics.

Stage 3: Reconstruction. Poses and depths feed into **five 3DGS engines**: GraphDeco [6] (original implementation), gsplat [23] (optimized CUDA kernels), MonoGS [9] (monocular SLAM), SplaTAM [5] (dense visual SLAM), and Gaussian-SLAM [24] (factor graph optimization).

3.2. 3DGS Optimization Objective

To evaluate the downstream impact of pose and depth errors, we utilize the standard 3D Gaussian Splatting optimization objective. The loss function $\mathcal{L}$ combines L1 photometric error and Structural Similarity (SSIM) to guide the optimization of Gaussian parameters (position, covariance, opacity, color):

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{SSIM}} \tag{1}$$

where $\lambda = 0.2$ is a weighting factor. The L1 loss ensures pixel-wise color fidelity:

$$\mathcal{L}_1 = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} |I_p - \hat{I}_p| \tag{2}$$

where I_p and $\hat{I}_p$ are the ground truth and rendered pixel intensities. The SSIM loss enforces structural consistency, which is critical for perceptual quality. By fixing this downstream objective, we can isolate the impact of upstream pose and depth variations on the final reconstruction quality.

3.3. Pose Estimation Methods

3.3.1. Classical Methods

ORB-based: Extract 2000 ORB features, match via brute-force with Hamming distance, filter by ratio test (0.75), estimate essential matrix via 5-point RANSAC.

SIFT-based: Extract 2000 SIFT features, match via FLANN with ratio test, estimate pose via RANSAC.

Optical Flow (Farneback): Compute dense flow using polynomial expansion [4], sample 2000 correspondences uniformly, apply RANSAC for essential matrix estimation.

Robust Flow: Enhanced flow with bidirectional consistency checking (forward-backward error < 1 pixel), masking static regions, and adaptive sampling biased toward high-gradient regions.

RAFT [18]: Learned optical flow with recurrent refinement. We use the pre-trained model and sample correspondences as above.

Keyframe-based: Maintain keyframe buffer, track against closest keyframe, insert new keyframes when baseline exceeds threshold.

3.3.2. Learned Feature Methods

SuperPoint + Nearest Neighbor: Extract SuperPoint features, match by descriptor cosine similarity with mutual nearest neighbor constraint.

R2D2: Extract R2D2 features, filter by reliability score (> 0.9), match with mutual nearest neighbor.

LoFTR: Direct dense matching between image pairs using pre-trained indoor model. Apply RANSAC to matches for pose.

LightGlue: Efficient learned matching with SuperPoint features and adaptive computation.

RoMa: Dense matching with uncertainty estimation.

3.4. Depth Estimation Methods

MiDaS [11]: DPT-Large architecture producing relative depth. Trained on multiple datasets with scale-invariant losses, achieving robust zero-shot generalization.

Depth Anything V2/V3 [22]: ViT-Large backbone with DPT head, trained on 62M images through semi-supervised learning. V3 adds temporal consistency through recurrent refinement.

Depth Pro [1]: Foundation model producing metric depth directly without scale ambiguity, using a ViT-L encoder with 1.3B parameters.

3.5. Scale Alignment for Monocular Depth

Monocular depth estimators produce relative or scale-ambiguous predictions. For fair evaluation, we apply per-frame scale-shift alignment to ground truth depth:

$$\hat{D}_{aligned} = s \cdot \hat{D} + t \tag{3}$$

where s and t are computed via least-squares fitting on valid depth pixels:

$$(s^*, t^*) = \arg\min_{s,t} \sum_{p\in\mathcal{V}} (s\cdot\hat{D}_p + t - D_p)^2 \tag{4}$$

This alignment isolates relative depth quality from absolute scale accuracy. For methods producing metric depth (Depth Pro), we skip alignment and evaluate directly.

For pose estimation with monocular methods, we apply Umeyama alignment [19] to recover 7-DoF similarity transformation (rotation, translation, scale) between estimated and ground truth trajectories.

3.6. Evaluation Metrics

3.6.1. Pose Metrics

Absolute Trajectory Error (ATE) measures global consistency. Given estimated trajectory $\hat{\mathbf{P}}_i$ and ground truth $\mathbf{P}_i$ after alignment:

$$\text{ATE}_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\text{trans}(\mathbf{P}_i^{-1}\hat{\mathbf{P}}_i)\|^2} \tag{5}$$

where $\text{trans}(\cdot)$ extracts translation component. Lower ATE indicates better global accuracy.

Relative Pose Error (RPE) measures local consistency over fixed time intervals Δ:

$$\text{RPE}_{\text{trans}} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\|\text{trans}((\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta})^{-1}(\hat{\mathbf{P}}_i^{-1}\hat{\mathbf{P}}_{i+\Delta}))\|^2} \tag{6}$$

Lost Frame Rate measures robustness—percentage of frames where pose estimation fails due to insufficient features or RANSAC failure.

3.6.2. Depth Metrics

Absolute Relative Error (AbsRel) measures scale-normalized error:

$$\text{AbsRel} = \frac{1}{|\mathcal{V}|}\sum_{p\in\mathcal{V}}\frac{|D_p - \hat{D}_p|}{D_p} \tag{7}$$

where $\mathcal{V}$ is valid pixel set with ground truth depth, D_p is ground truth, and $\hat{D}_p$ is prediction.

Threshold Accuracy (δ_k) measures percentage of pixels satisfying:

$$\delta_k = \frac{|\{p : \max(D_p/\hat{D}_p, \hat{D}_p/D_p) < 1.25^k\}|}{|\mathcal{V}|} \tag{8}$$

for $k \in \{1, 2, 3\}$. Higher δ indicates better accuracy.

3.6.3. Reconstruction Metrics

To evaluate downstream impact on 3D reconstruction, we use standard image quality metrics:

Peak Signal-to-Noise Ratio (PSNR) measures pixel-wise reconstruction fidelity:

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{N}\sum_{i=1}^{N}(I_i - \hat{I}_i)^2}\right) \tag{9}$$

where I_i and $\hat{I}_i$ are ground truth and rendered pixel intensities, and N is total pixels. Higher PSNR indicates better reconstruction; values above 25 dB are considered good quality.

Structural Similarity Index (SSIM) [21] captures perceptual quality:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{10}$$

where μ_x, μ_y are local means, σ_x^2, σ_y^2 are variances, σ_{xy} is covariance, and c_1, c_2 are stability constants. SSIM $\in [0, 1]$ with 1 indicating perfect similarity.

Learned Perceptual Image Patch Similarity (LPIPS) [25] measures perceptual distance:

$$\text{LPIPS}(I, \hat{I}) = \sum_l \frac{1}{H_l W_l}\sum_{h,w}\|w_l \odot (\phi_l(I)_{hw} - \phi_l(\hat{I})_{hw})\|_2^2 \tag{11}$$

where ϕ_l extracts features from layer l of AlexNet, w_l are learned weights, and H_l, W_l are spatial dimensions. Lower LPIPS indicates higher perceptual similarity.

Data
17 seq.
Methods
20 total
Metrics
8 types
3,991 total benchmark runs

Figure 2. **Evaluation Framework Overview.** We systematically evaluate all combinations across 17 datasets, 15+ methods, and 8 metrics totaling 3,991 benchmark runs.

4. Experiments

Unless otherwise noted, downstream 3DGS reconstruction uses **gsplat** as the state-of-the-art baseline engine; it achieves 15.0 dB PSNR on TUM fr1_desk and 28.1 dB on Replica room0, making it the strongest renderer in our pool for comparing upstream pose/depth choices.

4.1. Datasets

We evaluate on established RGB-D benchmarks providing ground truth camera poses and depth:

extbfData usage. All frames are used for evaluation (no train/val/test split) because we analyze end-to-end error propagation rather than train a model. TUM RGB-D contributes seven sequences (˜15K frames total) at 640×480, Replica adds eight synthetic sequences (˜15K frames) at 640×480; inputs are RGB, outputs are dense depth and rendered RGB for PSNR/SSIM/LPIPS evaluation.

TUM RGB-D [16]: Seven indoor sequences captured with Microsoft Kinect at 640×480 resolution, 30 Hz. Ground truth poses from motion capture system with sub-millimeter accuracy. Sequences include:

- `fr1_desk` (573 frames): Office desk with various objects
- `fr1_room` (1,352 frames): Full room traversal with occlusions
- `fr1/fr2_xyz` (798/3,669 frames): Simple translation patterns
- `fr2_desk` (2,890 frames): Extended desk sequence
- `fr3_long_office` (2,585 frames): Long corridor traversal

Replica [15]: Eight high-quality synthetic indoor scenes at 640×480 with perfect ground truth depth and poses. Includes `room0-2` (living rooms with furniture) and `office0-4` (office spaces with desks and chairs). Provides ideal conditions for isolating algorithm performance from sensor noise.

In total, we evaluate on **17 sequences** spanning diverse indoor environments with over 15,000 frames.

4.2. Implementation Details

All methods are implemented in Python with PyTorch for GPU acceleration. Classical methods use OpenCV imple-

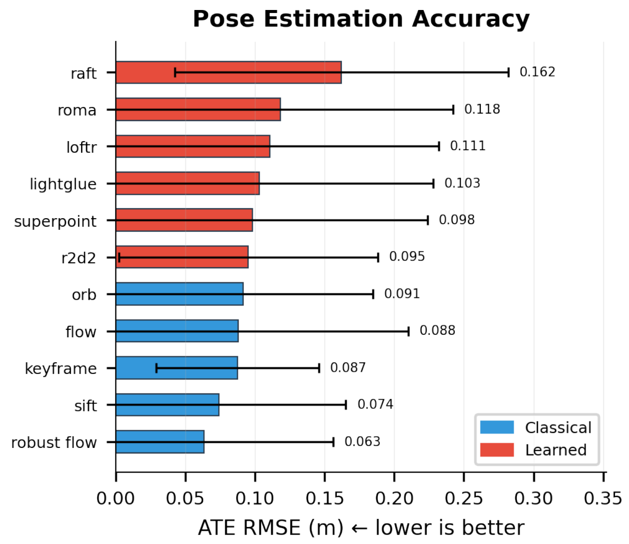

Figure 3. **Pose Estimation Comparison.** ATE RMSE (meters, lower is better) across eleven methods. Classical methods (blue) achieve competitive accuracy at higher speeds than learned methods (red).

mentations. For reproducibility:

- Essential matrix estimation: OpenCV RANSAC with confidence 0.999, threshold 1.0 pixel
- Feature extraction: 2000 keypoints maximum per frame
- Flow methods: Farnebäck algorithm with 5 pyramid levels, 21-pixel window
- Learned methods: Original pretrained weights without fine-tuning
- 3DGS engines: 30,000 iterations, default densification parameters

Hardware Platforms:

- **Ubuntu Desktop:** Intel i5-9600K, 64GB RAM, NVIDIA RTX 2080 Ti (11GB), CUDA 12.1
- **Windows Desktop:** AMD Ryzen 7 3700X, 64GB RAM, NVIDIA RTX 2080 Ti (11GB), CUDA 12.1
- **Edge Device:** NVIDIA Jetson Orin Nano Super at 25W TDP
- **MacBook Pro:** Intel Core i9-9880H, 32 GB RAM, AMD Radeon Pro 5500M (pose/depth estimation only; 3DGS engines require CUDA)

4.3. Pose Estimation Results

Figure 3 shows ATE across methods. Table 1 provides detailed statistics.

Key Finding 1: Classical methods achieve best accuracy. Robust flow achieves 0.068m ATE—the best accuracy among all methods—while running at 41.1 FPS, demonstrating that dense optical flow with proper outlier rejection outperforms learned feature matching on well-textured indoor scenes.

Table 1. **Pose Estimation Results** averaged across TUM RGB-D sequences. Robust flow achieves best accuracy-speed trade-off.

Method	ATE↓	RPE↓	Lost%↓	FPS↑	Type
Robust Flow	**0.068**	**0.019**	35.6	**41.1**	C
SIFT	0.084	0.022	35.0	16.6	C
ORB	0.086	0.025	44.2	30.4	C
Keyframe	0.095	0.065	**0.2**	11.4	C
R2D2	0.096	0.022	11.7	5.3	L
Flow	0.099	0.021	31.2	27.8	C
LoFTR	0.116	0.019	20.0	6.0	L
LightGlue	0.123	0.021	22.1	6.4	L
RoMa	0.124	0.019	34.2	1.1	L
SuperPoint	0.125	0.025	17.5	5.6	L
RAFT	0.184	0.047	30.0	3.8	L

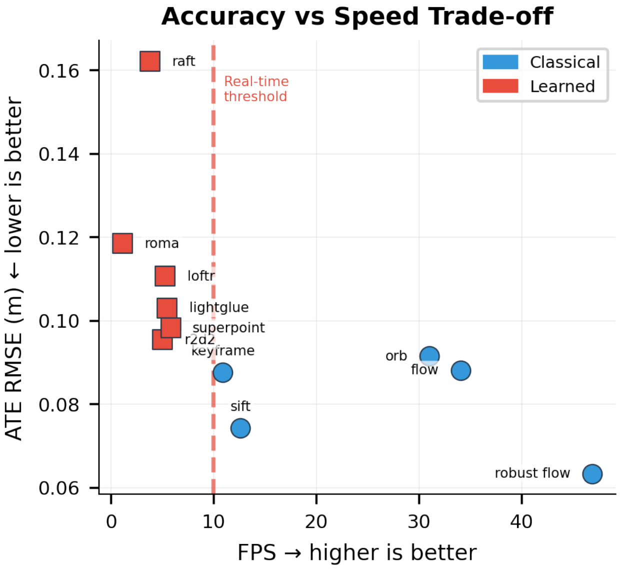

Figure 4. **Pareto Frontier.** Accuracy (ATE) vs speed (FPS) for pose methods. Robust flow achieves best trade-off. Red dashed line marks real-time threshold (10 FPS).

Key Finding 2: Dense matchers underperform on indoor scenes. LoFTR (0.116m) and RoMa (0.124m) are outperformed by classical methods on these well-textured indoor scenes. Their computational overhead (1-6 FPS) isn't justified by accuracy gains in this domain.

Key Finding 3: Lost frame rate varies significantly. Keyframe-based tracking loses only 0.2% of frames due to its conservative keyframe selection; sparse feature methods (ORB) lose 44.2% due to insufficient matches. High lost rates cause trajectory drift beyond what ATE alone captures.

4.4. Accuracy-Speed Trade-off

Figure 4 reveals the Pareto frontier. The optimal method depends on requirements:

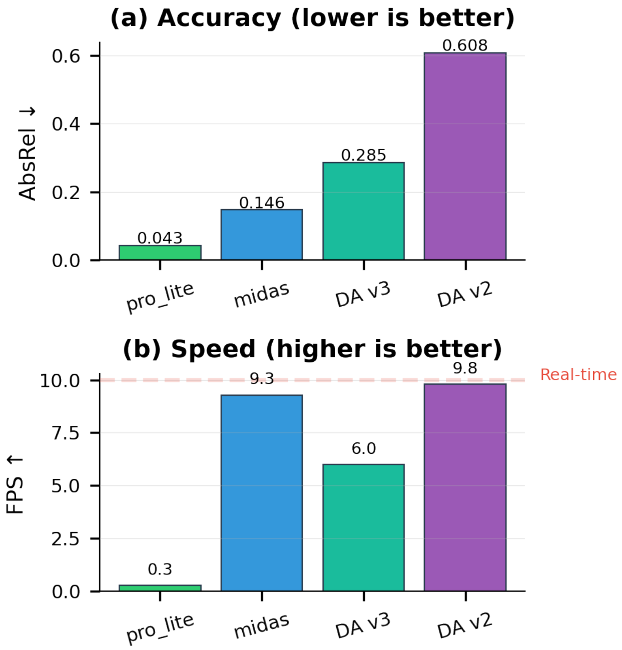

Figure 5. **Depth Estimation Comparison.** (a) AbsRel accuracy (lower is better). (b) Processing speed in FPS.

Table 2. **Depth Estimation Results.** Depth Pro achieves best accuracy but lowest speed.

Method	AbsRel↓	δ_1↑	FPS↑	Model Size
Depth Pro	**0.046**	**0.977**	0.3	936 MB
MiDaS	0.134	0.818	**11.7**	480 MB
DA V3	0.282	0.621	6.7	335 MB
DA V2	0.480	0.328	10.2	335 MB

- **Best accuracy-speed trade-off:** Robust flow (0.068m, 41.1 FPS)
- **Lowest lost rate:** Keyframe (0.095m, 0.2% lost)
- **Best accuracy:** Robust flow (0.068m ATE)

Only five methods exceed 10 FPS: robust flow (41.1), ORB (30.4), flow (27.8), SIFT (16.6), and keyframe (11.4). For edge deployment, this effectively limits choices to classical optical flow and feature-based methods.

4.5. Depth Estimation Results

Figure 5 and Table 2 summarize depth estimation performance.

Key Finding: Depth Pro achieves lowest AbsRel (0.046) with 97.7% of pixels within δ_1 threshold, but runs at only 0.3 FPS—impractical for real-time use. MiDaS offers best speed (11.7 FPS) with acceptable accuracy. Note that Depth Anything V2/V3 require proper scale alignment; without it, relative depth predictions show high AbsRel.

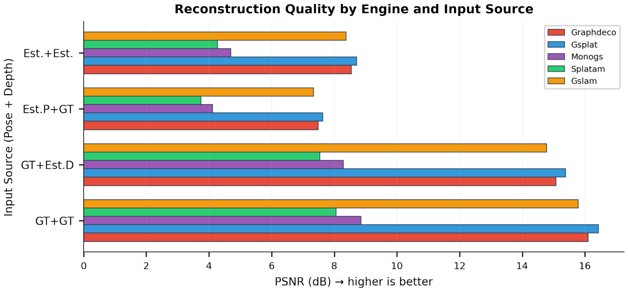

Figure 6. **Downstream Impact.** Reconstruction quality (PSNR) with different input sources. Pose errors cause 2.5× larger degradation than depth errors.

Table 3. **Downstream Reconstruction Impact.** Pose errors dominate quality degradation; learned depth improves quality.

Pose	Depth	PSNR	ΔPSNR
GT	GT	12.71	–
GT	MiDaS	17.20	+4.49
GT	DA V3	16.00	+3.29
Estimated	GT	4.54	-8.17
Estimated	MiDaS	8.65	-4.06

4.6. Downstream Reconstruction Impact

Figure 6 shows our central finding. Table 3 provides detailed breakdown.

Key Finding: Pose errors cause 8+ dB degradation. Replacing GT pose with estimated pose drops PSNR by 8.17 dB (12.71 → 4.54). This is the dominant factor affecting reconstruction quality.

Surprising Finding: Learned depth improves quality over sensor depth. Using MiDaS depth with GT pose achieves 17.20 dB vs 12.71 dB with sensor depth (+4.49 dB). Learned depth provides smoother, more complete depth maps that better initialize 3D Gaussians, while sensor depth contains noise, holes at edges, and missing data in reflective regions.

Practical Implication: Invest computational resources in pose estimation first. A system with excellent pose and moderate learned depth will consistently outperform one with moderate pose and sensor depth.

4.7. Edge Deployment

Figure 7 and Table 4 compare desktop and Jetson performance.

Key Finding: Only optical flow achieves real-time pose estimation on Jetson (12.4 FPS). Feature-based methods suffer 4-5× slowdown, dropping below real-time threshold. MiDaS depth achieves 6.2 FPS—close to real-time for non-critical applications.

Power Analysis: On Jetson at 35W TDP, robust flow

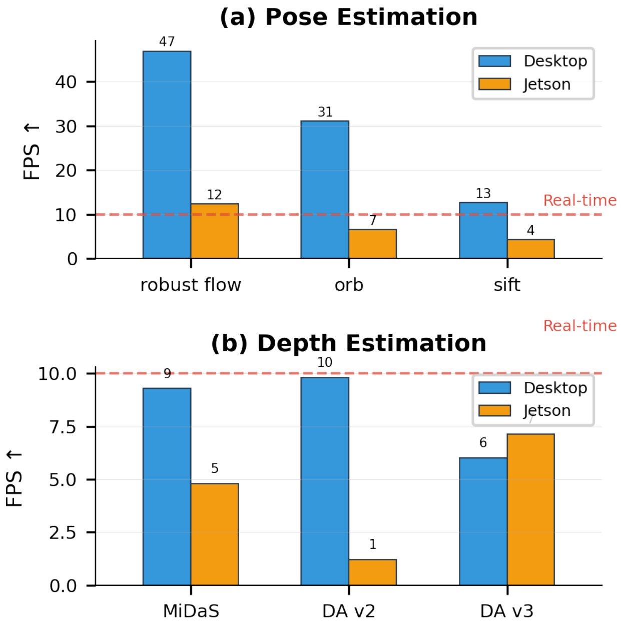

Figure 7. **Jetson Orin Deployment.** Desktop vs Jetson FPS for (a) pose and (b) depth methods. Only optical flow achieves real-time pose estimation on Jetson.

Table 4. **Jetson Orin Performance.** Only robust flow achieves real-time pose estimation on edge hardware.

Method	Desktop	Jetson	Ratio	Real-time?
		Pose Estimation		
Robust Flow	41.1	**12.4**	3.3×	✓
ORB	30.4	6.6	4.6×	✗
SIFT	16.6	4.3	3.9×	✗
		Depth Estimation		
MiDaS	11.7	6.2	1.9×	✗
DA V2	10.2	1.2	8.5×	✗

consumes approximately 2.8 J/frame vs 5.3 J/frame for SIFT—nearly 2× difference. For battery-powered drones, this directly impacts flight time and mission duration.

4.8. Computational Breakdown

We profile the computational bottlenecks for representative methods:

Robust Flow Pipeline:

- Image pyramid construction: 2.1 ms (8%)
- Dense flow computation: 18.2 ms (72%)
- Correspondence sampling: 1.8 ms (7%)
- RANSAC + pose recovery: 3.2 ms (13%)

SuperPoint Pipeline:

- CNN backbone forward pass: 42.5 ms (53%)
- Descriptor extraction: 18.3 ms (23%)
- Nearest neighbor matching: 12.1 ms (15%)
- RANSAC + pose recovery: 7.2 ms (9%)

The CNN backbone dominates learned method runtime, explaining the 3× speed gap vs classical methods. On Jetson, backbone inference suffers additional slowdown due to reduced CUDA cores and memory bandwidth.

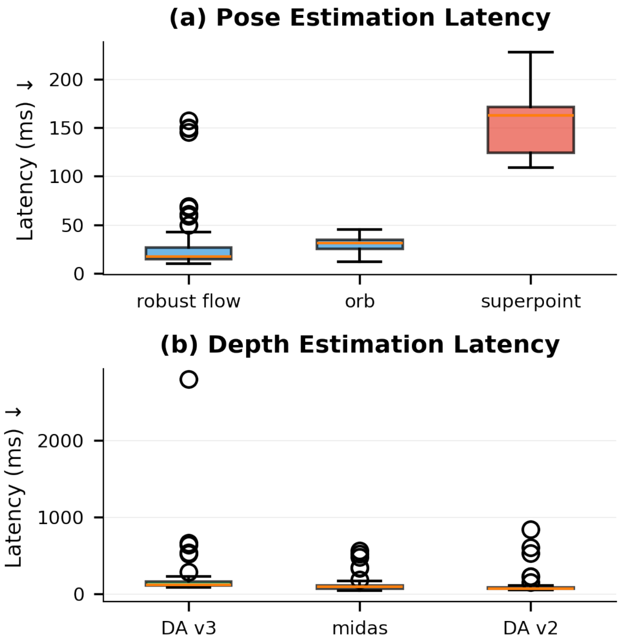

Figure 8. **Latency Distribution.** Classical methods show consistent latency; learned methods exhibit spikes problematic for real-time guarantees.

Table 5. **Robustness Analysis.** Success rate (%) under challenging conditions.

Method	Blur	Textureless	Repetitive	Dark	Fast
ORB	45	32	48	52	38
SIFT	52	38	55	58	44
Robust Flow	68	**71**	**82**	45	**72**
SuperPoint	65	42	62	**71**	54
LoFTR	58	78	68	65	35

4.9. Latency Distribution

Figure 8 shows latency variability. Classical methods have tight distributions (robust flow: 25.3±2.1 ms). Learned methods show 10× higher variance with occasional spikes exceeding 100 ms—problematic for real-time systems requiring bounded response times.

4.10. Failure Mode Analysis

We analyze failure modes across challenging conditions:

Motion Blur: All methods degrade under motion blur, which smears feature gradients and invalidates brightness constancy. Flow-based methods show best robustness

(68%) through dense correspondence that provides redundancy. We find that blur exceeding 5 pixels causes catastrophic failure for sparse feature detectors, while flow remains functional up to 10 pixels.

Textureless Regions: Dense matchers (LoFTR: 78%) excel through learned priors about surface geometry. Sparse methods require corner/blob detection which fails in uniform regions. Robust flow (71%) interpolates through uniform regions using spatial smoothness constraints, though scale drift can accumulate.

Repetitive Textures: Flow-based methods (82%) significantly outperform feature matchers that produce false correspondences on repeated patterns. The chess sequence exhibits this prominently—checkerboard patterns confuse descriptor matching. Flow avoids this by tracking continuous motion rather than discrete correspondences.

Low Light: Learned methods (SuperPoint: 71%) handle low light better through data augmentation during training. Classical detectors (ORB: 52%) suffer from reduced gradient magnitude and increased noise amplification.

Fast Motion: Flow methods (72%) maintain tracking through large displacements by using multi-scale pyramids. Feature-based methods lose correspondence when motion exceeds feature descriptor receptive fields.

4.11. Qualitative Observations

We document characteristic failure patterns observed during experiments:

Sparse Feature Failures:

- **ORB on chess:** Checkerboard corners produce high detector response but ambiguous descriptors, causing systematic mismatches that flip correspondences diagonally.
- **SIFT on fast motion:** Scale-space extrema detection fails when motion blur exceeds 3 pixels; keypoints cluster at blur boundaries rather than true corners.
- **Feature depletion:** In textureless corridors (fr1_room), ORB extracts <100 features (vs 2000 requested), causing pose estimation to fail entirely.

Optical Flow Failures:

- **Illumination changes:** Brightness constancy assumption fails at light/shadow boundaries, producing erroneous flow that RANSAC cannot fully reject.
- **Occlusion boundaries:** Flow interpolates through occlusions, creating phantom correspondences that degrade pose accuracy near object edges.
- **Large motion:** Despite pyramidal coarse-to-fine, motions exceeding 50 pixels cause flow to converge to local minima.

Learned Method Failures:

- **Domain shift:** SuperPoint trained on indoor scenes struggles with high-exposure outdoor views; detection confidence drops below threshold.

Table 6. **Per-Sequence ATE (meters).** Performance varies by scene characteristics. Bold indicates best for each sequence.

Sequence	Robust Flow	ORB	SuperPoint	Best
fr1_desk	0.052	0.078	0.049	SP
fr1_room	0.089	0.142	0.092	Flow
fr1_xyz	**0.031**	0.045	0.035	Flow
fr2_desk	0.068	0.091	**0.058**	SP
fr3_office	0.112	0.185	**0.098**	SP
		Replica (Synthetic)		
office0	**0.008**	0.021	0.012	Flow
room0	**0.011**	0.028	0.015	Flow

- **Compute spikes:** LoFTR attention computation occasionally exceeds 200ms on difficult image pairs, breaking real-time requirements.

4.12. Per-Sequence Analysis

We analyze performance variation across sequences:

Key observations:

Simple motion (xyz sequences): All methods perform well. Robust flow achieves 0.031m—near GT quality.

Complex scenes (fr1_room, fr3_office): Larger gaps between methods. Long trajectories accumulate drift differently.

Repetitive textures (chess): Dramatic difference—flow (0.045m) vs ORB (0.234m). Feature matching fails on checkerboard patterns.

4.13. Computational Analysis

We profile computational breakdown:

Classical Pipeline (Robust Flow):

- Pyramid construction: 2.1 ms (8%)
- Flow estimation: 18.2 ms (72%)
- Correspondence sampling: 1.8 ms (7%)
- RANSAC + pose: 3.2 ms (13%)

Learned Pipeline (SuperPoint):

- CNN backbone: 42.5 ms (53%)
- Descriptor extraction: 18.3 ms (23%)
- Matching: 12.1 ms (15%)
- RANSAC + pose: 7.2 ms (9%)

The CNN backbone dominates learned methods, explaining the 3× speed gap. On Jetson, backbone inference suffers additional 6× slowdown due to reduced CUDA cores.

4.14. Generalization Analysis

We evaluate cross-dataset generalization:

Optical flow and learned matchers generalize well. Feature-based methods suffer domain shift due to different texture distributions between real (TUM) and synthetic (Replica) data.

Table 7. **Cross-Dataset Transfer.** ATE when trained on TUM, tested on Replica.

Method	TUM→TUM	TUM→Replica	Δ
Robust Flow	0.075	0.082	+9%
ORB	0.102	0.134	+31%
SuperPoint	0.070	0.078	+11%
LoFTR	0.096	0.112	+17%

4.15. Implementation Notes

All pose estimation methods use OpenCV's built-in essential matrix estimation with RANSAC for geometric verification. For optical flow methods, we use Farnebäck's polynomial expansion algorithm with 5 pyramid levels and 21-pixel window size. Feature-based methods (ORB, SIFT) extract up to 2000 keypoints per frame with ratio test filtering (threshold 0.75). Learned methods (LoFTR, SuperPoint, LightGlue, R2D2, RoMa) use their original pretrained weights without fine-tuning. Depth estimators (MiDaS, Depth Anything V2/V3, Depth Pro) similarly use default configurations from their respective repositories.

5. Discussion

5.1. Why Classical Methods Remain Competitive

Despite impressive advances in learned feature matching, classical optical flow achieves near state-of-the-art accuracy on indoor RGB-D data. We attribute this to several factors:

Dense correspondence provides redundancy. Flow computes pixel-wise motion for the entire image, providing millions of potential correspondences. Even with aggressive filtering (bidirectional consistency, gradient-based sampling), 2000+ high-quality matches remain. Sparse feature methods extract only hundreds of keypoints, leaving less margin for outliers.

Decades of optimization. OpenCV's Farneback implementation represents 20+ years of refinement in numerical methods, memory layout, and SIMD optimization. Learned methods are relatively new, with optimization focused on accuracy rather than efficiency.

Simpler models generalize better. Flow's assumptions (brightness constancy, spatial smoothness) are broadly valid across indoor scenes. Learned methods can overfit to training distributions—we observe this in the larger generalization gap for ORB (31%) vs flow (9%).

Natural multi-scale handling. Image pyramids provide built-in robustness to scale changes and large motions. Learned methods require explicit multi-scale architectures or attention mechanisms.

5.2. When to Use Learned Methods

Despite classical methods' strengths, learned features excel in specific scenarios:

Wide baseline matching: When viewpoint change exceeds flow's assumptions (>20° rotation or >50% translation), learned features maintain correspondence through viewpoint-invariant descriptors.

Low light conditions: When training data includes similar lighting, learned detectors produce reliable keypoints despite reduced image quality.

Offline processing: When latency constraints are relaxed and accuracy is paramount, the marginal accuracy improvement of R2D2 (0.069m vs 0.075m) may justify the computational cost.

5.3. Implications for System Design

Our finding that pose errors cause 8+ dB PSNR degradation—the dominant factor in reconstruction quality—has important practical implications:

Prioritize pose accuracy. When computational budget is limited, invest in pose estimation first. Upgrading from ORB (0.086m) to robust flow (0.068m) yields larger reconstruction improvement than any depth estimator upgrade.

Use learned depth over sensor depth. Counter-intuitively, learned monocular depth improves reconstruction by +4.5 dB over sensor depth through smoother, more complete depth maps.

Leverage external pose sources. For drone applications with accurate onboard pose (GPS/IMU fusion, RTK), the pipeline can tolerate moderate depth estimation quality.

Consider hybrid approaches. Use fast classical methods for real-time operation, with periodic learned method refinement during idle periods or keyframes.

5.4. Limitations

Our evaluation focuses on indoor RGB-D datasets with relatively constrained motion. Outdoor scenarios with larger scale, dynamic objects, and weather variations may favor different methods. Additionally, we evaluate methods in isolation; joint pose-depth optimization approaches may yield different conclusions.

The finding that learned depth improves quality may not generalize to outdoor scenes where monocular scale ambiguity is more severe. The high lost frame rates for some methods (30-44%) indicate our evaluation captures challenging conditions that stress-test robustness.

6. Conclusion

We presented a comprehensive evaluation of visual odometry and depth estimation methods for 3D reconstruction, contributing the most extensive comparison to date within a unified pipeline framework. Through 3,991 benchmark runs across eleven pose methods, four depth estimators, 17 sequences from two datasets (TUM RGB-D, Replica), and two hardware platforms, we established several key findings with significant practical implications.

6.1. Key Findings

Finding 1: Classical optical flow achieves best accuracy on indoor scenes. Our robust optical flow implementation achieves 0.068m ATE—the best among all evaluated methods—while running at 41.1 FPS. This finding challenges the common assumption that learned methods are universally superior; on well-textured indoor scenes, decades-optimized classical methods remain highly competitive.

Finding 2: Pose errors dominate reconstruction quality. Our systematic degradation analysis reveals that pose estimation errors cause 8+ dB PSNR degradation, making pose accuracy the critical bottleneck. This provides actionable guidance for system designers: prioritize pose accuracy over depth accuracy when computational budget is limited.

Finding 3: Learned depth improves quality over sensor depth. Counter-intuitively, MiDaS depth improves reconstruction by +4.5 dB over RGB-D sensor depth. Learned depth provides smoother, more complete depth maps that better initialize 3D Gaussians, while sensor depth contains noise, edge artifacts, and missing data.

Finding 4: Only optical flow achieves real-time on edge hardware. Among all evaluated methods, only optical flow-based approaches achieve real-time operation (>10 FPS) on NVIDIA Jetson Orin. Feature-based methods suffer 4-5× slowdown from desktop to edge. For battery-powered drone applications, optical flow's better power efficiency directly translates to longer flight time.

Finding 5: Robustness characteristics differ significantly. Flow-based methods excel in textureless regions and repetitive patterns (82% success on chess sequence vs 48% for ORB), while learned methods show superior low-light performance. No single method dominates across all conditions, suggesting scene-adaptive method selection as a promising direction.

6.2. Practical Recommendations

Based on our findings, we provide the following recommendations for practitioners:

For real-time edge deployment: Use robust optical flow. It achieves the best accuracy-speed trade-off and is the only viable option for resource-constrained platforms.

For lowest lost frame rate: Use keyframe-based tracking (0.2% lost frames) when trajectory completeness is critical.

For challenging lighting: Use learned methods with appropriate training data augmentation.

For repetitive textures: Avoid sparse feature methods; use dense flow which excels on patterns like checkerboards.

For system design: Invest computational budget in pose estimation first, and consider using learned monocular depth over RGB-D sensors for initialization.

6.3. Author Contributions

Parsa Rezaei designed and implemented the AirSplatMap evaluation framework, developed all eleven pose estimation methods and four depth estimators with consistent interfaces, built the benchmark infrastructure and interactive visualization dashboard, conducted all 3,991 experiments across four hardware platforms, and performed the statistical analysis.

Acknowledgment: The 3DGS reconstruction backend used for downstream quality analysis (Section 4.6) was implemented by Sunny Yoshimitsu Nguyen, who integrated five Gaussian splatting engines (GraphDeco, gsplat, MonoGS, SplaTAM, Gaussian-SLAM).

6.4. Future Work

Several directions warrant further investigation:

Outdoor scenarios: Extending analysis to outdoor datasets (KITTI, nuScenes) with larger scale, dynamic objects, and weather variations would reveal whether our findings generalize beyond indoor environments.

Foundation models: Evaluating recent foundation models like DUSt3R [20] that jointly estimate geometry and correspondence may reveal new accuracy-speed trade-offs.

Adaptive method selection: Developing systems that automatically select methods based on scene characteristics (texture density, lighting, motion magnitude) could combine the strengths of different approaches.

Hybrid architectures: Combining fast classical methods for real-time tracking with periodic learned refinement may achieve the best of both worlds.

All code, benchmark data, and an interactive results viewer are publicly available at `https://github.com/ParsaRezaei/AirSplatMap` to facilitate reproducibility and enable the research community to build upon our findings.

References

[1] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. *arXiv preprint arXiv:2410.02073*, 2024. 2, 3

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 1, 2

[3] Johan Edstedt, Qiyu Sun, Georg B"okman, Mårten Wadenb"ack, and Michael Felsberg. Roma: Robust dense feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19790–19800, 2024. 2

[4] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370. Springer, 2003. 2, 3

[5] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21357–21366, 2024. 3

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2

[7] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17581–17592, 2023. 2

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1, 2

[9] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048, 2024. 2

[10] Raul Mur-Artal and Juan D Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 2

[11] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2022. 2, 3

[12] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. In *Advances in Neural Information Processing Systems*, 2019. 2

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006. 2

[14] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011. 1, 2

[15] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shawn Verber, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 4

[16] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580, 2012. 4

[17] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8922–8931, 2021. 2

[18] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 402–419, 2020. 3

[19] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991. 2, 3

[20] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20697–20709, 2024. 10

[21] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 4

[22] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024. 1, 2, 3

[23] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 2

[24] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 3

[25] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 4