Splatwizard: A Benchmark Toolkit for 3D Gaussian Splatting Compression

Supplementary Material

1. Implementation Details

This section summarizes only the benchmark protocol and measurement details that are not fully spelled out in the main paper.

1.1. Shared Experimental Protocol

All methods are implemented or ported into Splatwizard and evaluated with the same data loading pipeline, camera split definitions, and rendering interface. For photometric evaluation, we follow the standard train/test split of each dataset and report image quality on held-out views. For bitrate evaluation, each method is required to encode all information needed for reconstruction into a single compressed binary, and decoding starts from a newly initialized model instance without access to training-time states. This avoids leakage from intermediate checkpoints or cached tensors and makes the reported size directly comparable across methods.

Unless a method intrinsically requires a different procedure, reproduced models retain their original training schedules, pruning rules, quantization operations, and entropy coding logic after being mapped into the scheduler abstraction of Splatwizard. The framework therefore standardizes the surrounding software stack and evaluation protocol, while preserving the method-specific optimization behavior as much as possible.

1.2. Runtime and Memory Measurement

We evaluate deployment-oriented efficiency from three complementary aspects: runtime, GPU memory footprint, and structural compactness. Runtime is measured as the wall-clock latency of the profiled inference region, while GPU memory is reported using both peak allocated memory and peak reserved memory. We keep both memory statistics because methods with customized rasterizers or entropy modules may exhibit similar active usage but different allocator behavior. Structural compactness is reported separately through Gaussian count.

To make timing comparable across methods, all measurements are collected with the same profiling wrapper and on the same hardware platform. Since CUDA execution is asynchronous, we synchronize the device before and after the profiled region so that the measured duration reflects actual GPU completion time rather than CPU-side dispatch time. Peak memory statistics are reset before profiling and queried after profiling, which restricts the reported memory footprint to the measured region instead of the entire program lifecycle.

Unless otherwise noted, runtime and memory are measured during inference under the same testing protocol used for image-quality evaluation. For anchor-based methods such as HAC and CAT-3DGS, the reported primitive count corresponds to anchors rather than expanded Gaussians, so these curves are marked separately in the figures.

1.3. Geometric Evaluation Protocol

Geometric evaluation is performed on DTU to complement the standard photometric benchmark. Unlike novel-view-synthesis evaluation, geometric evaluation targets reconstruction fidelity rather than held-out view generalization. We therefore train on the full set of input views and then reconstruct explicit geometry from the optimized Gaussian representation.

Following the reconstruction module described in the main paper, we render depth maps from the optimized Gaussians and fuse them with a TSDF-based pipeline implemented with Open3D. For methods that do not natively expose depth through their default renderer, we replace the rendering backend with depth-enabled variants during the reconstruction stage while keeping the optimized Gaussian parameters unchanged. We then report Chamfer distance on both point clouds and reconstructed meshes. These metrics should be interpreted as scene fitting quality under a unified reconstruction protocol.

2. Scheduler Configurations of Reproduced Methods

This section lists scheduler configurations for reproduced methods. We use Python-style `range` expressions to indicate the iterations at which each task is executed.

Stage	Task	Plan
Pre	update lr	range(30000)
	increase SH degree	range(0, 30000, 1000)
Post	collect gradient statistics	range(15000)
	prune and densify	range(500, 15000, 100)
	reset opacity	range(0, 15000, 3000)

Table 1. Schedule plan for vanilla 3DGS.

Stage	Task	Plan
Pre	update lr	range(30000)
Post	gradient aware prune	range(100, 5000, 500)

Table 2. Schedule plan for Trimming the Fat.

Stage	Task	Plan
Pre	update lr	range(30000)
Post	gradient aware prune	range(100, 5000, 500)

Table 3. Schedule plan for PUP-3DGS.

Stage	Task	Plan
Pre	update lr	range(30000)
	increase SH degree	range(0, 30000, 1000)
Post	collect gradient statistics	range(15000)
	prune and densify	range(500, 15000, 100)
	reset opacity	range(0, 15000, 3000)
	soft prune	range(6000, 15000, 3000)
	hard prune	range(15000, 30000, 3000)

Table 4. Schedule plan for Speedy-Splat.

Stage	Task	Plan
PRUNE stage Post	prune	1
DISTILL stage Post	calculate importance score	5000

Table 5. Schedule plan for LightGaussian. Note we separate prune and distillation to different stages.

Stage	Task	Plan
Pre	update lr	range(3000)
	calculate importance score	1
	create octree	1

Table 6. Schedule plan for MesonGS.

Stage	Task	Plan
Pre	update lr	range(30000)
	VQ compress	1

Table 7. Schedule plan for C3DGS.

Stage	Task	Plan
Pre	update lr	range(30000)
	increase SH degree	range(0, 30000, 1000)
	switch to RVQ training	29000
Post	collect gradient statistics	range(15000)
	prune and densify	range(500, 15000, 100)
	reset opacity	range(0, 15000, 3000)
	mask prune	range(15000, 30000, 1000)

Table 8. Schedule plan for Compact3DGS.

Stage	Task	Plan
Pre	update lr	range(30000)
	update anchor bound	1
Post	collect gradient statistics	range(15000)
	adjust anchor	range(1500, 3000, 100)
	adjust anchor	range(4000, 15000, 100)
	switch to quantized training	3000
	switch to entropy training	10001

Table 9. Schedule plan for HAC.

Stage	Task	Plan
Pre	update lr	range(30000)
	update cam mask	range(30000)
	update anchor bound	1
	setup triplane	10000
	switch to quantized training	3000
	switch to entropy training	10001
Post	collect gradient statistics	range(15000)
	adjust anchor	range(1500, 3000, 100)
	adjust anchor	range(4000, 15000, 100)

Table 10. Schedule plan for CAT-3DGS.

3. Additional Benchmark Results on Selected Datasets

This section provides additional benchmark plots for datasets not shown in the main paper, namely Deep Blending, NeRF Synthetic, BungeeNeRF, and Tanks & Temples. All training and testing were completed on a single RTX 3090. For HAC and CAT-3DGS, the reported primitive count corresponds to anchors rather than expanded Gaussians, so these curves are drawn with dashed lines.

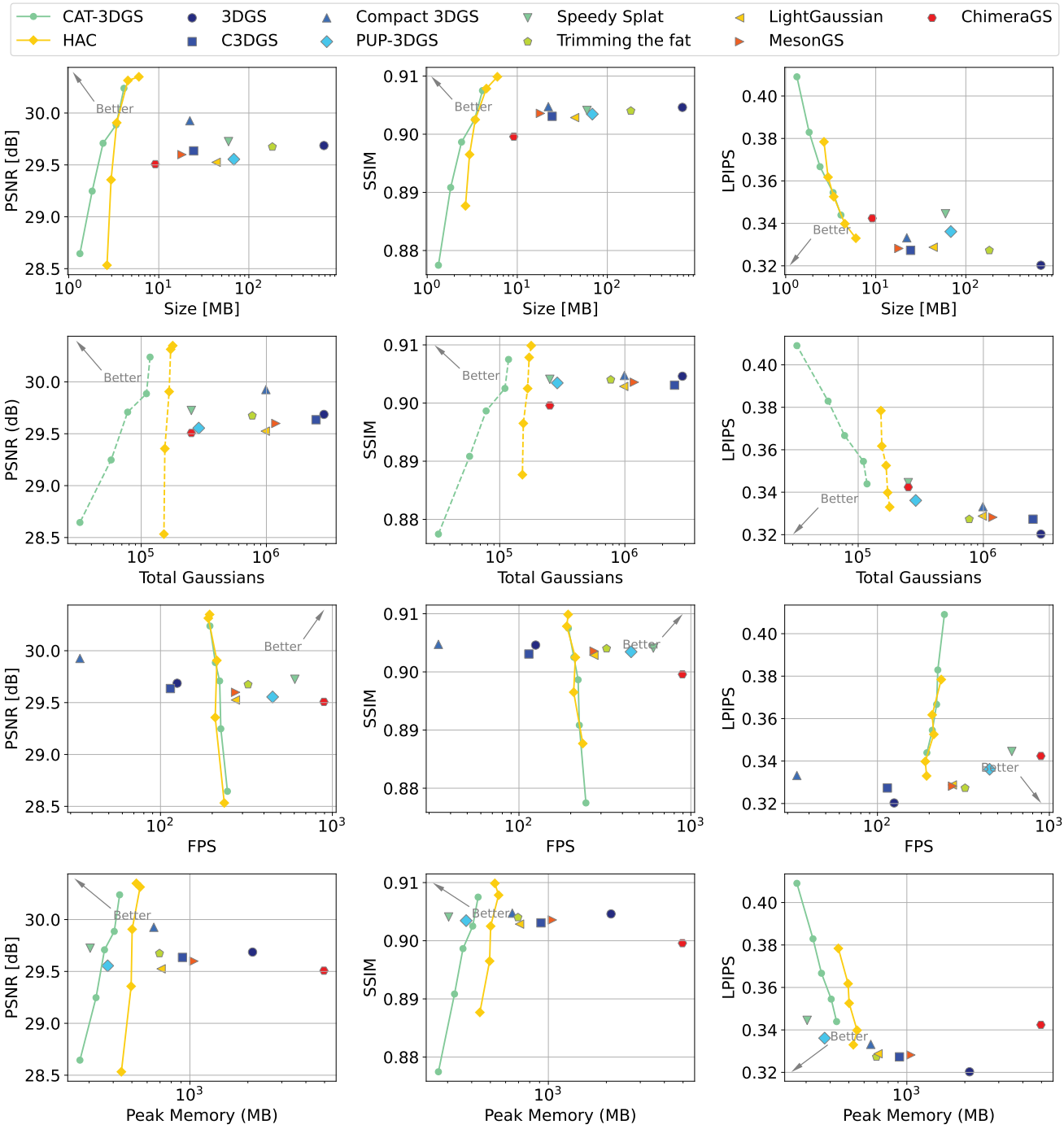

Figure 1. Results of DeepBlending.

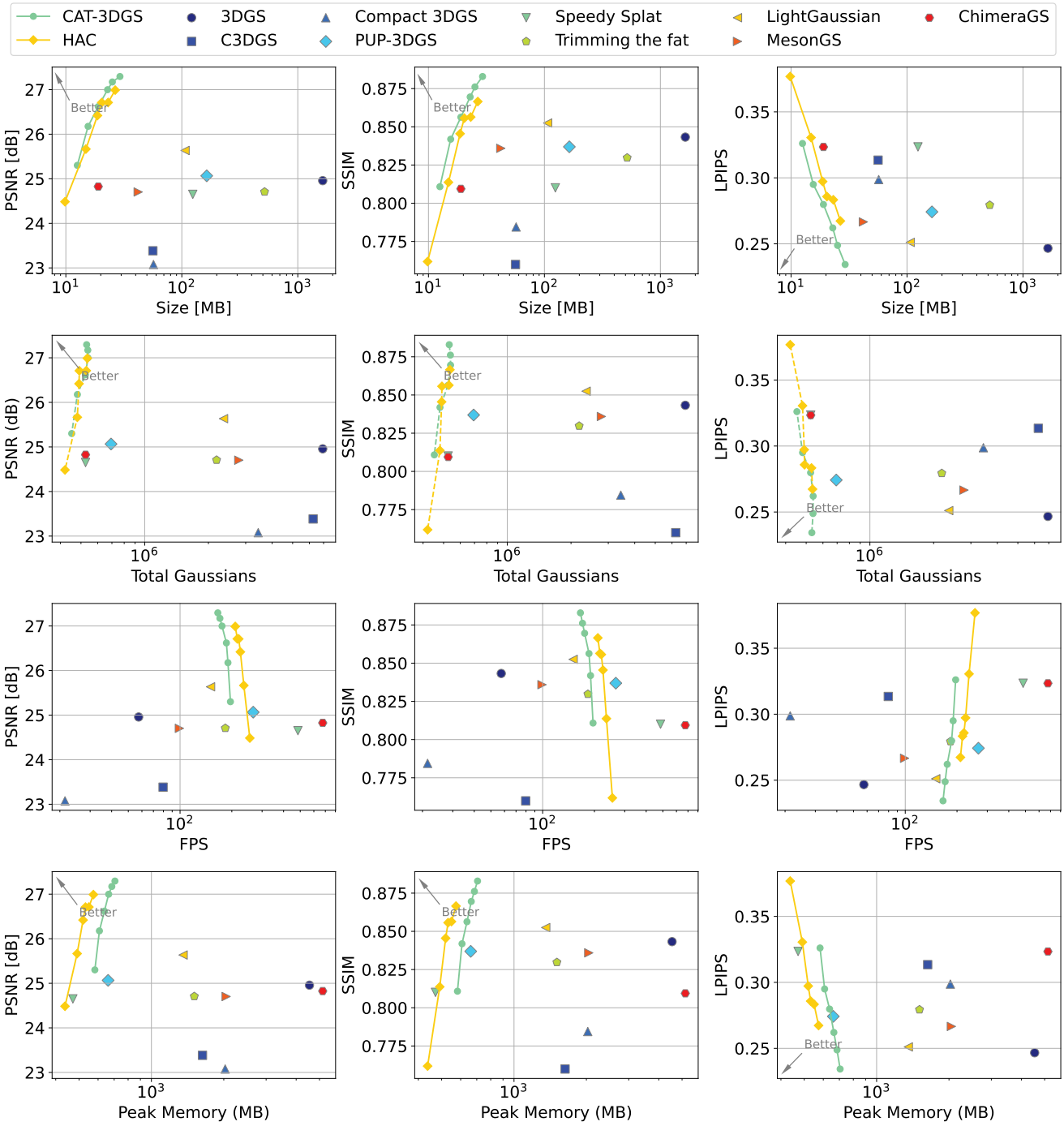

Figure 2. Results of BungeeNeRF.

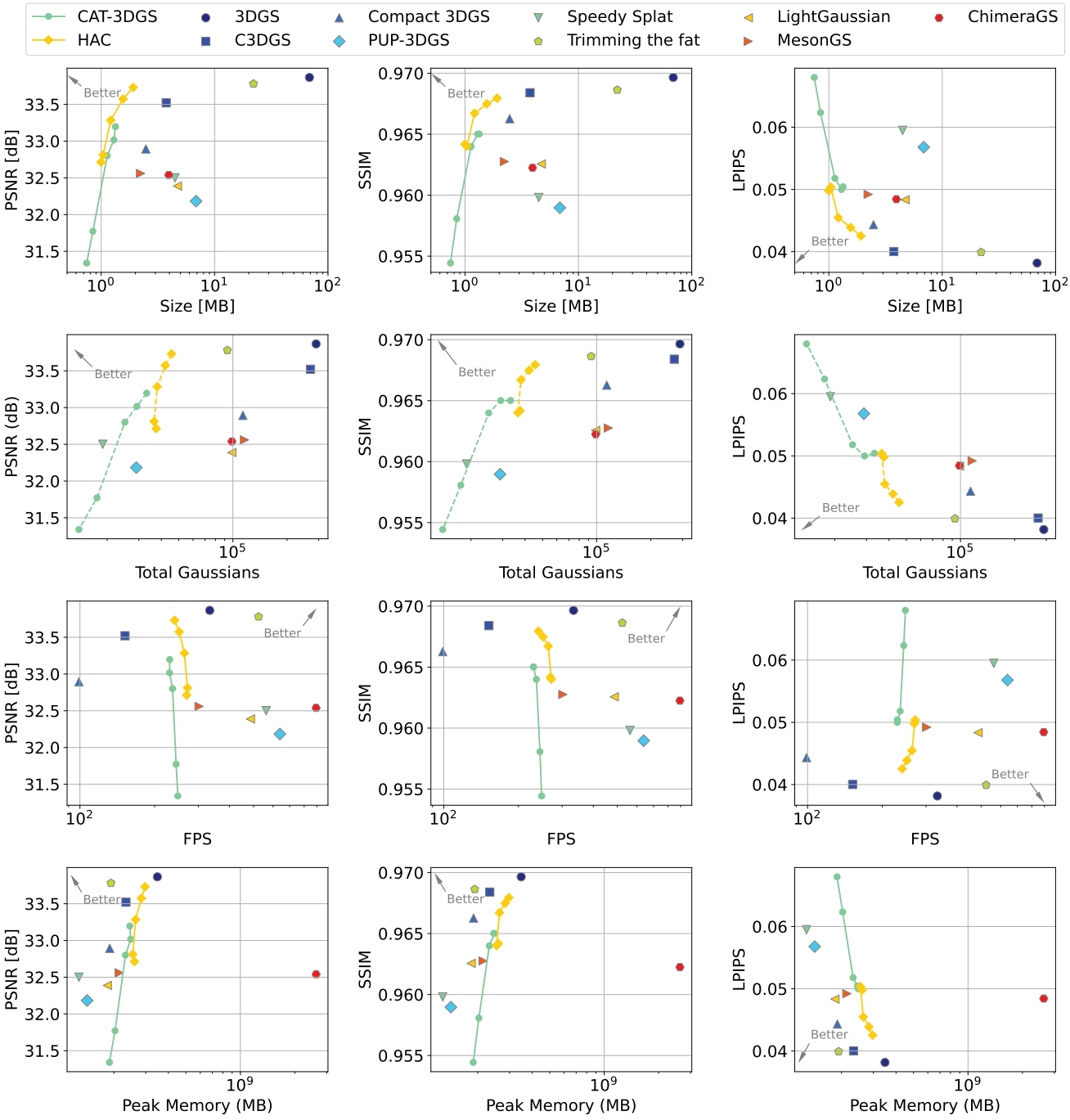

Figure 3. Results of NeRF Synthetic.

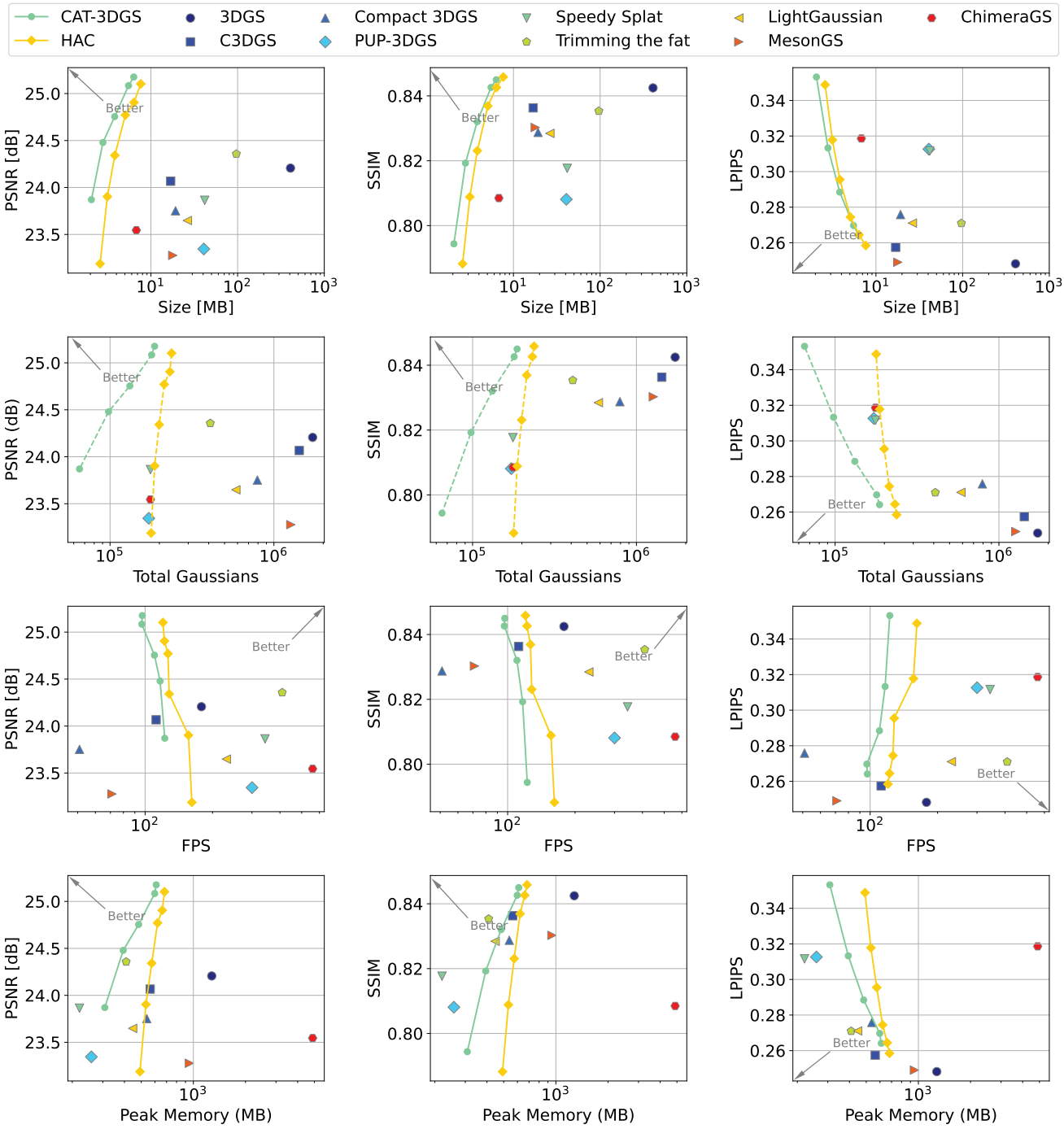

Figure 4. Results of Tanks & Temples.