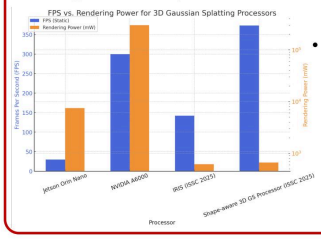
Mapping 3D Gaussian Splatting to a HW Architecture

Siddhant Gupta
Department of Electrical Engineering, Stanford University

Motivation

- 3DGS is a state of the art method of rendering 3D scenes (represented as Gaussians) to 2D camera views
- As shown in the chart, new accelerators are orders of magnitude more power efficient than GPUs for 3DGS render

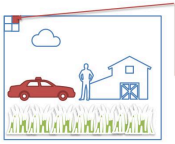


- In order to target a novel accelerator for 3DGS later, this project aims to explore quantization and search for bottlenecks in the rendering algorithm

Main Idea

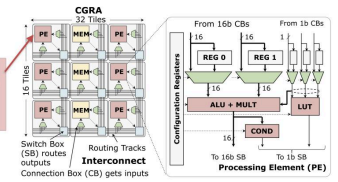
- Architectural exploration: compute-wise, can we map 3DGS to a CGRA SoC (CPU+CGRA)
- Identify bottleneck in algorithm, and experiment with 16 by 16 loop in C++ for it
- Simulate performance benefits from parallelism, and SNR at different quantizations

1. Preprocess image (CPU), divide image into tiles and associated gaussians for each tile



2. For each pixel in tile [16 x 16 pixels] in parallel:

Quantize gaussian, params to [bfloat16, E4M3, E5M2]
Compute quantized power, alpha, transparency update
Blend gaussians at quantized precision
Convert RGB for the pixel back to full precision float



CGRA SoC has a CPU, which can do preprocessing with global memory, and a 16 by 16 area of 16 bit processing elements [3]

Related Work

- Most works prior to ISSCC 2025 focused on improvements for 3DGS at an algorithmic level, to optimize it running for CUDA on an NVIDIA GPU
- The Shape Aware 3D GS Processor accelerator[1] and IRIS[2] are custom ASIC utilizing HW-SW co-design to achieve very low mW/ Frame, with dedicated units for 3DGS acceleration
- In this work, I am to conduct some architecture exploration with various quantizations and simulating 16x16 parallelism to guide in potentially accelerating 3DGS in a CGRA-like architecture

References

[1] Feng, Wang, "A 1.78mJ/Frame 373fps 3D GS Processor…", ISSCC, 2025
[2] Song, Kim, Park et al., "IRIS: A 8.55 mJ/frame Spatial Computing SoC…", ISSCC, 2025
[3] Koul, Melchert,Sreedhar, et al., "AHA: An Agile Approach…" , ACM Transactions on Embedded Computing Systems, Volume 22, Issue 2, 2023

Experimental Results

Baseline

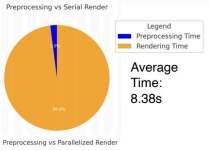
Playroom Dataset
SSIM : 0.8103411
PSNR : 24.4170532
LPIPS: 0.4125682

Flowers Dataset
SSIM : 0.4580899
PSNR : 18.8644238
LPIPS: 0.4646910

Quantized, bfloat16

Playroom Dataset
SSIM : 0.7620919
PSNR : 22.2413616
LPIPS: 0.4173917

Flowers Dataset
SSIM : 0.43133744
PSNR : 16.6820946
LPIPS: 0.54300000



Average Time: 8.38s

Average Time: 0.38s

- Quantization to 8 bit (E4M3 and E5M2) is excluded due to unrecognizable image quality (exp saturation)
- Using bfloat16 creates small artifacts, reasonable SNR, that future work can potentially correct with filtering
- Limitation: actual time is not helpful, since it's algorithm simulation on a desktop, not a cycle-accurate simulator
- However, timing charts display with 16x16 pixel-level parallelism per tile, the render bottleneck is almost eliminated