

3D Superquadric Splatting (Supplementary Material)

Daniel MacSwayne Aleš Leonardis Jianbo Jiao

The MIX Group, School of Computer Science, University of Birmingham

dxm378@student.bham.ac.uk, {a.leonardis, j.jiao}@bham.ac.uk

1. Methodology Details

Superquadrics can also be expressed in polar form:

$$\begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = d \begin{bmatrix} \hat{x}_s \\ \hat{y}_s \\ \hat{z}_s \end{bmatrix} = d \begin{bmatrix} a_x \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ a_y \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ a_z \sin^{\epsilon_1} \eta \end{bmatrix} \quad (1)$$

where η and ω are latitude and longitude, respectively. In practice, all polar equations with an exponent are computed as follows:

$$x^y = \text{sign}(x)|x|^y \quad (2)$$

1.1. Degenerate solution for the rim constraint

The solution to the orthographic rim constraint given by Eq. 8 in the main paper) is in the explicit form $\eta = f(\omega)$. However, when a superquadric is viewed from the side ($r_{33} \rightarrow 0$), the solution becomes degenerate. In this edge case, the rim is sampled using:

$$\eta \in \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right] \quad (3)$$

$$\omega = -\arctan \left(\left[\frac{a_y}{a_x} \cdot \frac{r_{13}}{r_{23}} \right]^{\frac{1}{2-\epsilon_2}} \right) \quad (4)$$

In the double degenerate edge case where $a_x r_{23} \rightarrow 0$, ω is fixed as:

$$\omega \in \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right] \quad (5)$$

2. Algorithmic Analysis

2.1. Complexity analysis

In terms of tile-based parallelization, our approach follows the same tile-based parallelization as the 3D Gaussian Splatting (GS) renderer.

1. Split the canvas into tiles.
2. Identify which splats are visible in each tile.
3. Load batches of depth-ordered splats.

4. Sequentially update the opacity weighted colour accumulation until saturation.

For an image with size H, W , broken into tiles of size T, h, w . N splats are visible but are sequentially loaded in depth-ordered batches of size L . Since the tile processing is parallelised, the memory complexity is $O(ThwL)$ and the time complexity is $O(N/L)$. This matches the complexity of GS, differing only in coefficients.

2.2. Inference speed

To isolate the coefficients of our new superquadric projection from the framework overhead, we also implemented a GS renderer in PyTorch. The results in Table A indicate that the majority of the runtime is attributable to PyTorch overhead rather than the superquadric projection itself. This suggests that with a dedicated CUDA implementation, our method could reach real-time rendering performance (~ 30 FPS), similar to the original GS framework, indicating that the method is practically usable.

Table A. The relative inference speed analysis of GS and SQS in Pytorch and CUDA implementations.

Method	Pytorch	CUDA	real-time
GS	2.7s	0.01s	✓
SQS	10s	0.037s (estimated)	✓

2.3. Scene storage costs

Splats consist of many parameters: 3 for position, 3 for size, 4 for orientation, typically 27 for colour harmonics, and 1 for opacity. Our implementation only adds 3 additional exponent parameters per splat. At most, our implementation uses 8% more hard storage space per splat.

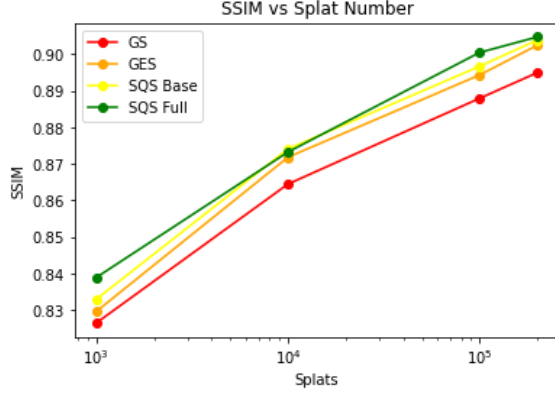


Figure A. How SSIM performance varies with number of splats for different splatting methods.

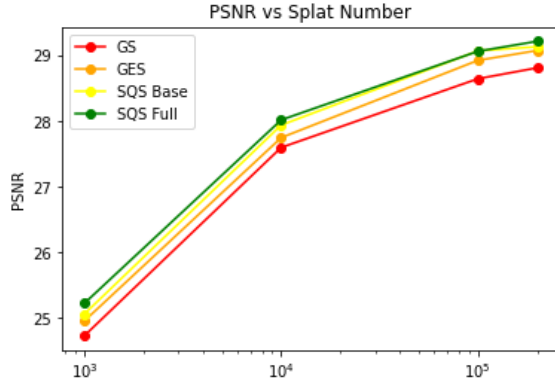


Figure B. How PSNR performance varies with number of splats for different splatting methods.

3. Performance Comparison

Our aim is to ensure a fair comparison. It is hard to directly compare the quoted performance of GS and our SQS methods. Our ablation study was intended to isolate the effect of the extra exponential parameters whilst eliminating all other experimental differences.

3.1. Experiment size

For example, we disabled the adaptive density control mechanism from GS because we want all methods to maintain a fixed number of splats throughout training. The typical scenes for GS use **millions** of splats and trains for **30k** iterations, while all of our experiments train **400k** splats for **1k** iterations. This was done to reduce computing resources. This could partially explain the difference in absolute performance. As evidence, we show in Figs. A, B, C that the number of splats greatly affects the performance. For an equal number of splats, our performance is consistently greater than GS.

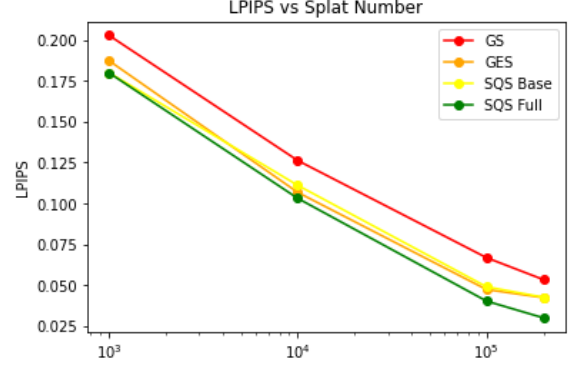


Figure C. How LPIPS performance varies with number of splats for different splatting methods.

3.2. Inconsistent datasets

Also, many papers across the 3D reconstruction community report results on *varying subsets of scenes* from these benchmark datasets, making it inappropriate to quote their original numbers for comparison directly. Since there can be quite a large variation in performance between scenes, it is important to ensure the exact same scenes and views are used for training and testing, to ensure fairness.

3.3. Initialisation

Finally, we start all experiments from the same point cloud initialisation, to ensure that the differences between COLMAP and DUS3R have no effect.

By doing this, the variation in performance can be more confidently attributed to the extra shape components. In all of the standardised experiments, our SQS outperformed the GS method.

Table B. Learnable parameters comparison. L: Learnable

Method	ϵ_1	ϵ_2	ϵ_3
GS	1	1	1
GES	1	1	L
SQS (Base)	L	L	1
SQS (Full)	L	L	L

3.4. SQS base vs SQS full

The results in Table 1 (in the original paper) include multiple scenes for each dataset, while Table 2 (in the original paper) includes only 1 scene. It is likely that statistically significant differences between the methods are only reliably revealed across many scenes. Compared to SQS Base, the class of functions is extended and the size of the parameter space is increased for SQS Full (please see Table B

and Fig. D). The larger parameter space probably contains a more optimal solution.

Speculated example: if a splat is trying to fit a sharp edge, it may raise its ϵ_3 parameter. This makes the shape very sharp but may also attenuate the repositioning gradient signal, meaning the splat may be more prone to getting stuck in local optima. To investigate this further, we propose temporarily freezing the ϵ_3 parameter until later on in the training.

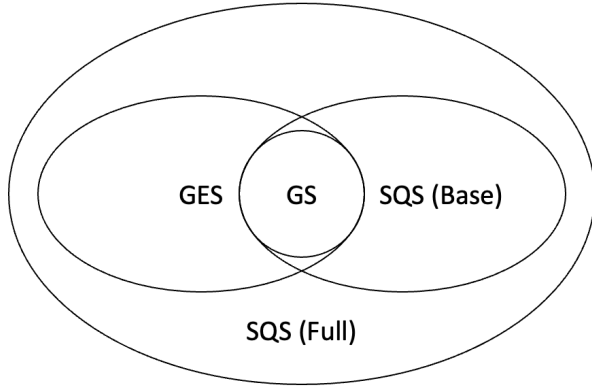


Figure D. **The relationship between different methods. SQS is a superset of Gaussian Splatting.**