Chapter 1

Real-Time Maximum Intensity Projection Rendering for 3-D Fluorescence Microscopy via Differentiable Gaussian Splatting

Chapter Overview

The preceding chapters established implicit neural representations (INRs) as a powerful framework for compressing three-dimensional fluorescence microscopy volumes [1, 2]. Those models achieve compression ratios exceeding 97× while preserving reconstruction quality above 40 dB PSNR, yet they share a fundamental limitation: rendering a novel view requires querying the network at every sample point along every ray, making interactive visualization impractical even on modern hardware.

This chapter addresses that limitation by replacing the implicit neural representation with an explicit *Gaussian Mixture Field* (GMF) — a compact set of anisotropic 3-D Gaussian primitives — and introducing a fully differentiable *soft Maximum Intensity Projection* (soft-MIP) splatting pipeline that renders novel views in a single forward pass through a custom CUDA kernel. The two representations are complementary: INRs excel at compression fidelity and generalization, while Gaussian splatting excels at rendering speed and explicit scene editability. Together they span the accuracy–interactivity trade-off that is central to practical fluorescence microscopy analysis.

The principal contributions of this chapter are:

1. A **differentiable soft-MIP splatting formulation** that enables gradient flow through all Gaussian primitives simultaneously, as opposed to only the maximum-contributing primitive in a hard MIP.

2. A **multi-component perceptual loss** (weighted MSE, SSIM, edge, and intensity-distribution terms) specifically designed for the bimodal, sparse-foreground statistics of fluorescence MIPs.

3. A **numerically stable online soft-max CUDA kernel** that implements

the log-sum-exp trick in a single pass over Gaussians, with a verified backward pass validated by `torch.autograd.gradcheck`.

4. Comprehensive **ablation and benchmarking experiments** showing > 30 FPS at 256^2 resolution, $401\times$ compression relative to uncompressed float32 storage, and mean PSNR above 33 dB across diverse viewpoints.

1.1 Introduction

1.1.1 Motivation and Problem Statement

Fluorescence microscopy produces high-resolution three-dimensional volumetric images of biological specimens, enabling morphological analysis of structures such as neuronal arbours, synaptic contacts, and axonal projections at sub-micrometre resolution. Interactive, real-time visualization of these volumes is indispensable for scientific analysis: researchers must navigate freely around a structure, inspect projections from arbitrary angles, and correlate spatial features across viewpoints — all within a responsive interface.

Conventional volumetric ray-marching satisfies the fidelity requirement but fails on speed. A naïve implementation evaluating $N_s = 200$ samples per ray achieves only 1–5 FPS at 256^3 voxel resolution on a modern GPU, imposing a $6-30\times$ gap relative to the 30 FPS interactive threshold. Voxel-resolution downsampling alleviates cost but sacrifices the fine structural detail — sub-pixel dendritic tips, thin axonal fibres — that carries the most biological information.

The preceding chapters of this thesis demonstrated that INR-based compression can reduce volumetric storage by two to three orders of magnitude while preserving reconstruction quality. However, rendering from an INR is no faster than rendering from the original volume: both require per-sample network inference along every ray, and the added inference cost of a full SIREN network can actually increase rendering time relative to trilinear interpolation. This renders INR-based representations unsuitable as a direct solution to the interactive visualization problem.

1.1.2 The Gaussian Splatting Paradigm

3D Gaussian Splatting (3DGS), introduced by Kerbl et al. [3] for real-time radiance field rendering of RGB scenes, sidesteps ray marching entirely. Instead of querying a scene representation along rays, 3DGS *projects* each 3-D Gaussian primitive onto the image plane via a first-order Jacobian approximation [7] and composites contributions in a tile-rasterization pass. The entire render is a single CUDA kernel invocation, achieving hundreds of FPS for typical scene sizes.

Adapting 3DGS to fluorescence MIP rendering requires resolving two incompatibilities with the original formulation.

Projection semantics. 3DGS uses alpha-compositing with depth sorting to model occlusion in RGB scenes. Fluorescence microscopy is physically additive and occlusion-free: every emitter contributes independently to the recorded intensity, and the MIP selects the maximum-intensity voxel along each ray rather than compositing.

The rendering model must therefore implement a maximum-projection aggregation rather than depth-ordered alpha blending.

Differentiability. The hard max operator is non-differentiable with respect to all but the maximizing primitive, starving the majority of Gaussians of gradient signal and causing training to stagnate. A smooth, differentiable surrogate that preserves the qualitative behaviour of the hard MIP is essential for optimization.

This chapter resolves both issues through the *differentiable soft-MIP* formulation introduced in section 1.2.2.

1.1.3 Relationship to Prior Chapters

The Gaussian Mixture Field representation introduced here occupies a distinct niche in the representation taxonomy developed across this thesis. INRs (Chapters 2–3) encode the scene implicitly in network weights and achieve the highest compression ratios, but require network evaluation per query. Gaussian splatting encodes the scene explicitly in a set of geometric primitives; compression is achieved by representing 52.6×10^6 voxels with $K \approx 41{,}000$ Gaussians (a $\sim 400\times$ reduction in parameter count), while rendering speed is achieved by replacing ray marching with a tile-rasterization CUDA kernel.

Neither approach strictly dominates the other. The choice between INR compression and GMF splatting is application-driven: INRs are preferable when reconstruction fidelity at arbitrary resolution is the primary requirement, while GMFs are preferable when interactive rendering speed takes priority.

1.1.4 Chapter Organization

Section 1.2 develops the mathematical formulation of the GMF representation and the four-step soft-MIP rendering pipeline. Section 1.3 details implementation decisions including data preprocessing, hyperparameter choices, and the complete training configuration. Sections 1.4 and 1.5 describe the ground-truth generation and Gaussian initialization stages respectively. Section 1.6 presents the multi-component loss function and adaptive density controller. Section 1.7 reports an ablation study with paired statistical tests. Section 1.8 provides six quantitative experiments covering rendering speed, visual quality, scalability, temporal stability, soft-MIP convergence, and memory efficiency. Section 1.9 details the CUDA kernel design and software architecture. Section 1.10 synthesizes findings, positions them within the broader thesis narrative, and identifies open problems.

1.2 Mathematical Formulation

1.2.1 Gaussian Mixture Field Representation

Definition 1.2.1 (Gaussian Mixture Field)**.** A *Gaussian Mixture Field* (GMF) approximates the fluorescence intensity function $f : \mathbb{R}^3 \to [0, 1]$ of a volumetric

specimen as a weighted sum of K anisotropic Gaussian basis functions:

$$f(\mathbf{x}) = \sum_{k=1}^{K} a_k \, \exp\Big(-\tfrac{1}{2}\,(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)\,, \tag{1.1}$$

where the parameters of the k-th primitive $\mathcal{G}_k$ are defined below.

Learnable parameters. Each primitive $\mathcal{G}_k$ is characterized by three groups of parameters that are jointly optimized end-to-end.

1. **Centre** $\boldsymbol{\mu}_k \in \mathbb{R}^3$: the mean position in a normalized world coordinate system $[-1,1]^3$, updated directly by gradient descent.

2. **Covariance** $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3\times3}$: a symmetric positive-definite matrix controlling shape and orientation. Positive definiteness is guaranteed and rotation decoupled from scale by the factorization

$$\boldsymbol{\Sigma}_k = \mathbf{R}_k \, \mathrm{diag}(\mathbf{s}_k^{\odot 2})\, \mathbf{R}_k^\top, \tag{1.2}$$

   where $\mathbf{R}_k \in \mathrm{SO}(3)$ is constructed from a unit quaternion $\mathbf{q}_k \in \mathbb{R}^4$ via the standard Rodrigues formula, and $\mathbf{s}_k = \exp(\tilde{\mathbf{s}}_k) \in \mathbb{R}^3_{>0}$ are anisotropic half-widths obtained by exponentiating unconstrained log-scales $\tilde{\mathbf{s}}_k$. This parameterization follows [3] exactly, ensuring compatibility with existing split/clone adaptive density control algorithms.

3. **Intensity** $a_k \in (0,1)$: the peak emission amplitude, parameterized as $a_k = \sigma(\ell_k)$ where σ is the logistic sigmoid and $\ell_k \in \mathbb{R}$ is an unconstrained logit. Unlike 3DGS, which carries per-primitive RGB colour and opacity encoded as spherical harmonics, each primitive here carries a single scalar intensity. This reflects the additive, occlusion-free physics of fluorescence emission and reduces the parameter count from 58 (3DGS) to 11 per primitive.

Remark 1.2.1. The factorization in eq. (1.2) yields $\boldsymbol{\Sigma}_k$ with eigenvalues $s_k^{(d)2}$ along the principal axes defined by $\mathbf{R}_k$. The log-scale parameterization $\tilde{\mathbf{s}}_k$ spans an unconstrained space, and scale regularization (section 1.6.1) enforces a practical range $[s_{\min}, s_{\max}] = [0.001, 0.5]$ covering physical neurite diameters from sub-pixel terminal tips ($\sim$0.5 µm) to thick primary dendrites ($\sim$10 µm).

1.2.2 MIP Splatting: Rendering Pipeline

Given camera extrinsics $(\mathbf{R}_c, \mathbf{T}_c)$, the MIP-splatted image $I \in [0,1]^{H\times W}$ is computed in four steps.

Step 1 — World-to-Camera Transform

Each primitive is mapped into camera space:

$$\boldsymbol{\mu}_k^{\mathrm{cam}} = \mathbf{R}_c\, \boldsymbol{\mu}_k + \mathbf{T}_c, \tag{1.3}$$

$$\boldsymbol{\Sigma}_k^{\mathrm{cam}} = \mathbf{R}_c\, \boldsymbol{\Sigma}_k\, \mathbf{R}_c^\top. \tag{1.4}$$

Depth $z_k > 0$ denotes the distance from the camera optical centre along the viewing axis.

Step 2 — Perspective Projection via EWA Splatting

Following the Elliptical Weighted Average (EWA) splatting framework [7], the 3-D Gaussian is projected to the image plane using the first-order Jacobian of the pinhole projection map at depth z_k:

$$\mathbf{J}_k = \begin{pmatrix} f_x/z_k & 0 & -f_x\, x_k/z_k^2 \\ 0 & f_y/z_k & -f_y\, y_k/z_k^2 \end{pmatrix} \in \mathbb{R}^{2\times 3}, \tag{1.5}$$

where $f_x = f_y = \frac{W}{2\tan(\alpha/2)}$ are focal lengths derived from the field-of-view angle $\alpha = 50°$, and (x_k, y_k) are the horizontal and vertical camera-frame coordinates of $\boldsymbol{\mu}_k^{\text{cam}}$.

The projected 2-D parameters are:

$$\boldsymbol{\mu}_k^{\text{2D}} = \Big(f_x\, x_k/z_k + c_x,\;\; f_y\, y_k/z_k + c_y\Big)^\top \in \mathbb{R}^2, \tag{1.6}$$

$$\boldsymbol{\Sigma}_k^{\text{2D}} = \mathbf{J}_k\, \boldsymbol{\Sigma}_k^{\text{cam}}\, \mathbf{J}_k^\top \in \mathbb{R}^{2\times 2}, \tag{1.7}$$

where (c_x, c_y) is the principal point (image centre).

Step 3 — 2-D Gaussian Evaluation

For each pixel at position $\mathbf{p} = (u, v)^\top \in \mathbb{R}^2$, the contribution of primitive k is evaluated as:

$$\mathcal{G}_k^{\text{2D}}(u, v) = \exp\!\Big(-\tfrac{1}{2}\,(\mathbf{p} - \boldsymbol{\mu}_k^{\text{2D}})^\top \big(\boldsymbol{\Sigma}_k^{\text{2D}}\big)^{-1} (\mathbf{p} - \boldsymbol{\mu}_k^{\text{2D}})\Big) \in [0, 1]. \tag{1.8}$$

Primitives whose squared Mahalanobis distance exceeds $\delta^2 = 16$ ($\delta = 4$) are culled before evaluation. The *effective contribution* combines intensity and spatial footprint:

$$g_k(u, v) = a_k \cdot \mathcal{G}_k^{\text{2D}}(u, v) \in [0, 1]. \tag{1.9}$$

Step 4 — Differentiable Soft-MIP Aggregation

The hard MIP, $I_{\text{hard}}(u, v) = \max_k\, g_k(u, v)$, is non-differentiable with respect to all Gaussian parameters except the maximizing primitive, creating a gradient starvation problem: all other primitives receive zero gradient and cannot be updated. To propagate signal through *all* primitives simultaneously, the hard max is replaced by a temperature-scaled soft-max:

$$I(u, v) = \sum_{k=1}^{K} \underbrace{\frac{\exp\big(\beta\, g_k(u, v)\big)}{\sum_{j=1}^{K} \exp\big(\beta\, g_j(u, v)\big)}}_{w_k^{\text{soft}}(u, v)} \cdot g_k(u, v). \tag{1.10}$$

Here $\beta > 0$ is the temperature parameter. As $\beta \to \infty$, $w_k^{\text{soft}} \to \mathbf{1}[k = \arg\max_j\, g_j]$ and eq. (1.10) recovers the hard MIP exactly. In practice, $\beta = 50$ at convergence yields a soft-MIP approximation quality of PSNR = 45.2 dB relative to the hard

reference (see table 1.13), well above the perceptual threshold. During training, β is linearly warmed from 10 to 50 over the first 500 epochs, avoiding sharp gradients during the early densification phase when primitives are still repositioning rapidly.

Remark 1.2.2 (Numerical stability). Direct evaluation of eq. (1.10) overflows for large β because $\exp(\beta g_k)$ can exceed the float32 maximum. The CUDA kernel (detailed in section 1.9.2) avoids this by subtracting the running maximum $m = \max_j \beta g_j$ before exponentiation, so every argument $\beta g_k - m \leq 0$, keeping all exponentials in $(0, 1]$. This is the standard log-sum-exp stabilisation [8], applied here in a single streaming pass over the Gaussians to avoid storing all g_k values simultaneously.

1.3 Implementation Details

1.3.1 Volume Preprocessing

Data Format and Intensity Normalization

The input is a multi-frame TIFF stack with spatial dimensions $(Z, Y, X) = (100, 647, 813)$ voxels, where Z is the axial depth dimension and X, Y are lateral. Voxel spacing is anisotropic: lateral resolution $\Delta_{XY} = 0.2\,\mu\text{m}$, axial resolution $\Delta_Z = 1.0\,\mu\text{m}$, giving a $5\times$ axial-to-lateral anisotropy ratio. Raw 12- or 16-bit fluorescence intensities are linearly rescaled to $[0, 1]$ via $v \leftarrow (v - v_{\min})/(v_{\max} - v_{\min})$.

Aspect-Ratio Correction

Because voxels are anisotropic, an isotropic Gaussian in voxel coordinates represents a physically prolate or oblate structure. A correction tensor

$$\mathbf{s}_{\text{asp}} = \frac{(1,\ Y/X,\ Z/X)}{\max(1,\ Y/X,\ Z/X)} \tag{1.11}$$

rescales the normalized world coordinate system so that all three axes are bounded in $[0, 1]$ and physically equivalent distances along each axis map to equal normalized extents. Every primitive centre $\boldsymbol{\mu}_k$ and scale $\mathbf{s}_k$ is multiplied element-wise by $\mathbf{s}_{\text{asp}}$ before splatting, and the same correction is applied to each sample point in the ground-truth ray-marching pass.

1.3.2 Configuration Parameters

Table 1.1 lists all key hyperparameters used in training and evaluation.

Table 1.1: Key configuration parameters for training and rendering.

Parameter	Value	Description
Camera intrinsics		
Horizontal FOV α	50°	Total horizontal field of view
Near plane	0.01	Minimum ray depth (normalized units)
Far plane	10.0	Maximum ray depth (normalized units)
Training resolution	256×256	Image size during optimization
Ground-truth ray marching		
Samples per ray N_s	200	Number of quadrature points
Integration interval	[0.5, 6.0]	Near/far bounds (normalized units)
Soft-MIP splatting		
Temperature β	50	Final soft-max sharpness
β warm-up	$10 \rightarrow 50$	Linear increase, epochs 1–500
Mahalanobis cutoff δ	4	Culling threshold ($\delta^2 = 16$)
Optimization		
Total epochs	2000	Passes over the 106-view training set
Initial LR η_0	3×10^{-3}	Adam optimizer
Final LR η_T	1×10^{-5}	Cosine-annealed target
Foreground weight w_{fg}	5.0	Pixel reweighting for WMSE
Checkpoint interval	100	Epochs between model saves

1.4 Stage 1: Ground-Truth MIP Generation

1.4.1 Multi-View Camera Pose Generation

A diverse set of 106 training views is generated by tiling the upper viewing hemisphere with orbital cameras. The camera centre is placed at:

$$\mathbf{T}_c = r \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \tag{1.12}$$

with orbital radius $r = 2.5$ (normalized units), elevation $\theta \in \{-30^\circ, 0^\circ, 30^\circ, 60^\circ\}$, and azimuth $\phi \in [0^\circ, 360^\circ)$ sampled at uniform intervals. The rotation matrix $\mathbf{R}_c$ is derived from the look-at direction towards the origin with a fixed up-vector. The hemisphere sampling is intentionally asymmetric in elevation to ensure denser coverage of the horizontal viewpoints most relevant to morphological analysis.

1.4.2 Volumetric Ray Marching

For each camera pose, the reference MIP is computed by classical ray marching as described in algorithm 1.

Algorithm 1 Reference MIP generation via volumetric ray marching

for each pixel (u, v) in the $H \times W$ image **do**
  Compute ray origin $\mathbf{o}$ and unit direction $\mathbf{d}$ from camera intrinsics and (u, v)
  $I_{\max} \leftarrow 0$
  for sample index $i = 1, \ldots, N_s = 200$ **do**
    $t_i \leftarrow t_{\text{near}} + (i-1)\,\frac{t_{\text{far}} - t_{\text{near}}}{N_s - 1}$
    $\mathbf{x}_i \leftarrow \mathbf{o} + t_i\,\mathbf{d}$
    $\mathbf{x}'_i \leftarrow \mathbf{x}_i \odot \mathbf{s}_{\text{asp}}$ (aspect-ratio correction, eq. (1.11))
    $v_i \leftarrow \text{trilinear}(V, \mathbf{x}'_i)$ (trilinear interpolation of volume V)
    $I_{\max} \leftarrow \max(I_{\max}, v_i)$
  end for
  $I(u, v) \leftarrow I_{\max}$
end for

The total cost is $N_s \times H \times W \approx 13.1 \times 10^6$ volume lookups per frame, producing one reference image in 200–500 ms on a Quadro RTX 8000. Table 1.2 summarizes the resulting dataset.

Table 1.2: Ground-truth training dataset statistics.

Property	**Value**
Number of views	106
Image resolution	256×256
Intensity range	$[0, 1]$
Storage (FP32)	≈27 MB
Total generation time	≈50 s

1.5 Stage 2: Gaussian Mixture Field Initialisation

Before 2-D MIP training begins, the GMF is initialized by fitting directly to the 3-D volume. A semantically grounded initialization concentrates primitives in high-intensity regions and substantially reduces the epoch budget required for Stage-3 convergence compared with random or grid initialization.

1.5.1 3-D Volumetric Fitting

Starting from K_0 Gaussians placed on a regular grid, the mean squared volumetric reconstruction error is minimized:

$$\mathcal{L}_{\text{3D}} = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \Big(f(\mathbf{x}) - V(\mathbf{x}) \Big)^2, \tag{1.13}$$

where $V(\mathbf{x}) \in [0,1]$ is the trilinearly-interpolated ground-truth volume intensity, $f(\mathbf{x})$ is the GMF prediction from eq. (1.1), and $\mathcal{S}$ is a randomly sampled mini-batch of 3-D points. After fitting, adaptive density control (split, clone, prune; section 1.6.2) concentrates primitives in high-intensity foreground regions and removes transparent Gaussians.

1.5.2 Initialisation Statistics

Table 1.3 reports the checkpoint used to seed Stage-3 training.

Table 1.3: GMF checkpoint used to initialise Stage-2 (MIP splatting) training.

Property	**Value**
Checkpoint file	`gmf_refined_best.pt`
Number of Gaussians K	12,145
Memory footprint	0.15 MB
Mean scale (s^x, s^y, s^z)	(0.031, 0.028, 0.089)
Intensity range $[a_{\min}, a_{\max}]$	[0.001, 0.998]

The elongated axial scale ($s^z \approx 0.089 \gg s^x, s^y \approx 0.030$) reflects the 5× physical voxel anisotropy: primitives must span proportionally more normalized distance along the z-axis to represent the same physical extent as a lateral primitive of equivalent diameter.

1.6 Stage 3: MIP Splatting Training

1.6.1 Training Objective

All Gaussian parameters $\{\boldsymbol{\mu}_k, \tilde{\mathbf{s}}_k, \mathbf{q}_k, \ell_k\}_{k=1}^{K}$ are optimized by minimizing a composite perceptual loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{WMSE}} + \lambda_{\text{SSIM}}\,\mathcal{L}_{\text{SSIM}} + \lambda_{\text{edge}}\,\mathcal{L}_{\text{edge}} + \lambda_{\text{int}}\,\mathcal{L}_{\text{int}} + \mathcal{L}_{\text{reg}}, \tag{1.14}$$

where $P \in [0,1]^{H\times W}$ is the splatted prediction and $G \in [0,1]^{H\times W}$ is the ground-truth MIP. Loss weights are given in table 1.4.

Table 1.4: Loss term weights used in the full model.

Term	**Weight**	**Role**
$\mathcal{L}_{\text{WMSE}}$	1.0 (fixed)	Pixel-wise reconstruction
$\mathcal{L}_{\text{SSIM}}$	$\lambda_{\text{SSIM}} = 0.1$	Perceptual structure
$\mathcal{L}_{\text{edge}}$	$\lambda_{\text{edge}} = 0.05$	Fine-detail sharpness
$\mathcal{L}_{\text{int}}$	$\lambda_{\text{int}} = 0.01$	Intensity statistics
$\mathcal{L}_{\text{reg}}$	$\lambda_{\text{scale}} = 0.01$	Scale regularization

Each term is motivated by a specific property of fluorescence MIP statistics. The WMSE term (section 1.6.1) addresses the severe foreground–background class imbalance. The SSIM term (section 1.6.1) enforces perceptual structural fidelity in local image patches. The edge term (section 1.6.1) supervises gradient matching at neurite boundaries, which carry morphological information that MSE under-weights due to the small fraction of edge pixels. The intensity-distribution term (section 1.6.1) provides global statistical supervision, preventing the model from collapsing to an overly dark solution that minimizes MSE by predicting the background mode everywhere. The scale regularization term (section 1.6.1) bounds scale parameters away from degenerate extremes.

Weighted Mean Squared Error

In a typical fluorescence MIP, 70–90% of pixels are near-zero background ($I < 0.05$). Under the standard unweighted MSE, gradient signal is dominated by background pixels, biasing optimization away from the thin, bright neurite processes that encode biological information. We address this by assigning each pixel a weight proportional to its ground-truth intensity:

$$\mathcal{L}_{\text{WMSE}} = \frac{1}{N}\sum_{i=1}^{N} w_i\,(p_i - g_i)^2, \quad w_i = 1 + (w_{\text{fg}} - 1)\,g_i, \quad w_{\text{fg}} = 5.0. \tag{1.15}$$

Anchoring weights to the fixed ground-truth g_i (rather than the evolving prediction p_i) is critical: it prevents a pathological feedback loop in which the model could inflate its own per-pixel learning rates by predicting high intensities. With $w_{\text{fg}} = 5$, the effective gradient contributions of the foreground (20% of pixels, weight 5) and background (80%, weight 1) are balanced at $0.20 \times 5 = 1.0$ and $0.80 \times 1 = 0.8$ respectively. The resulting gradient is:

$$\frac{\partial \mathcal{L}_{\text{WMSE}}}{\partial p_i} = \frac{2}{N}\Big[1 + (w_{\text{fg}} - 1)\,g_i\Big](p_i - g_i), \tag{1.16}$$

amplifying updates at bright neurite pixels by up to 5× relative to background for the same residual magnitude.

SSIM Regularization

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(P, G). \tag{1.17}$$

The Structural Similarity Index Measure [6] evaluates images over local 11×11-pixel windows ω, combining luminance, contrast, and structure components:

$$l(P,G) = \frac{2\mu_P\mu_G + C_1}{\mu_P^2 + \mu_G^2 + C_1}, \quad c(P,G) = \frac{2\sigma_P\sigma_G + C_2}{\sigma_P^2 + \sigma_G^2 + C_2}, \quad s(P,G) = \frac{\sigma_{PG} + C_3}{\sigma_P\sigma_G + C_3}, \tag{1.18}$$

$$\text{SSIM}(P,G) = l(P,G)\cdot c(P,G)\cdot s(P,G), \tag{1.19}$$

where $C_1 = (0.01)^2$, $C_2 = (0.03)^2$, $C_3 = C_2/2$ for the normalized $[0,1]$ intensity range. SSIM captures perceptual distortions such as blurred edges and contrast reduction

that may produce low MSE but strongly impair morphological assessment. The windowed formulation naturally adapts to feature scales spanning sub-pixel terminal tips and thick primary dendrites.

Edge Loss

$$\mathcal{L}_{\text{edge}} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\frac{\partial P}{\partial x}\Big|_i - \frac{\partial G}{\partial x}\Big|_i\right)^2 + \left(\frac{\partial P}{\partial y}\Big|_i - \frac{\partial G}{\partial y}\Big|_i\right)^2\right], \tag{1.20}$$

where spatial gradients are computed by convolution with the 3×3 Sobel operators:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{1.21}$$

Neurites are characterized by sharp intensity boundaries 1–3 pixels wide. The edge loss penalizes spatially misaligned or blurred boundaries regardless of pixel-wise error magnitude, making it particularly effective for the thin, high-frequency axonal and dendritic processes that MSE and SSIM can under-emphasize.

Intensity Distribution Loss

$$\mathcal{L}_{\text{int}} = D_{\text{KL}}\Big(\text{hist}(P) \parallel \text{hist}(G)\Big) = \sum_{b=1}^{B} h_P(b)\,\log\frac{h_P(b)}{h_G(b)}, \tag{1.22}$$

where $h_P(b) = \frac{1}{N}\sum_i \mathbf{1}[p_i \in \text{bin}_b]$ and $h_G(b)$ are the empirical probability masses of bin b in the prediction and ground-truth histograms respectively, over $B = 256$ uniform bins spanning $[0, 1]$. Laplace smoothing with $\epsilon = 10^{-8}$ prevents $\log(0/0)$ in empty bins.

Fluorescence MIPs exhibit a characteristic bimodal distribution: a dense near-zero background mode and a sparse high-intensity neurite mode. D_{KL} provides global statistical supervision that prevents the model collapsing to an overly dark solution — a failure mode that MSE tolerates but that renders the representation biologically uninterpretable.

Scale Regularization

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{scale}}\sum_{k=1}^{K}\sum_{d\in\{x,y,z\}}\Big[\max(0,\ s_{\min} - s_k^{(d)}) + \max(0,\ s_k^{(d)} - s_{\max})\Big], \tag{1.23}$$

with $s_{\min} = 0.001$ and $s_{\max} = 0.5$. The lower bound prevents near-singular projected covariances $\mathbf{\Sigma}_k^{\text{2D}}$ that cause numerical overflow in eq. (1.8); the upper bound prevents excessively large primitives that cannot represent local structure. The hinge form ensures zero penalty for all scales within the permitted range and a linear penalty outside it, imposing soft constraints without distorting the optimization landscape for well-behaved primitives.

1.6.2 Adaptive Density Control

The GMF density is refined throughout training by three complementary operations applied every 100 epochs from epoch 100 to 1500.

Splitting. Primitives whose projected gradient magnitude exceeds a threshold — indicating that a region is under-fitted and would benefit from additional representational capacity — are replaced by two smaller children:

$$\mathbf{s}_{\text{new}} = 0.8\,\mathbf{s}_{\text{old}}, \qquad \boldsymbol{\mu}_{\text{new}} = \boldsymbol{\mu}_{\text{old}} \pm 0.5\,\mathbf{s}_{\text{old}}. \tag{1.24}$$

The 0.8 scale factor ensures that child primitives cover the same spatial extent as the parent while reducing overlap; the $\pm 0.5\,\mathbf{s}$ displacement places each child on opposite sides of the parent centre.

Cloning. Primitives in regions with large reconstruction error but low gradient magnitude — indicating that the region is under-populated rather than poorly-fitted — are duplicated at the same position, allowing two primitives to specialize independently in representing complex local intensity distributions.

Pruning. Primitives with intensity $a_k < 0.01$ (effectively transparent after sigmoid activation) are removed every 25 epochs. Pruning prevents the accumulation of degenerate primitives that consume memory and computation without contributing to reconstruction quality.

1.6.3 Optimization Schedule

Table 1.5 summarizes the training schedule.

Table 1.5: Training schedule used in all experiments.

Component	Schedule	Description
Learning rate	$3{\times}10^{-3} \to 1{\times}10^{-5}$	Cosine annealing over 2000 epochs
β_{MIP}	$10 \to 50$	Linear warm-up, epochs 1–500
Density control	Every 100 epochs	Split/clone, epochs 100–1500
Pruning	Every 25 epochs	Remove $a_k < 0.01$
Checkpointing	Every 100 epochs	Save intermediate models

1.7 Ablation Study: Loss Function Components

To isolate the contribution of each loss term, four model variants were trained that incrementally add components on top of the baseline WMSE. All variants include scale regularization $\mathcal{L}_{\text{reg}}$; all other hyperparameters are held constant across variants.

Table 1.6: Ablation configurations. All variants include $\mathcal{L}_{\text{reg}}$ throughout.

Model	WMSE	SSIM	Edge	Intensity
`wmse`	✓			
`wmse_ssim`	✓	✓		
`wmse_ssim_edge`	✓	✓	✓	
`full`	✓	✓	✓	✓

1.7.1 Quantitative Results

Table 1.7: Ablation study: final metrics averaged over 30 held-out test views (mean ± std. dev.). Best values in bold.

Model	K	PSNR (dB)	SSIM	MAE	MSE (×10
`wmse`	49,484	37.41 ± 1.19	0.9801 ± 0.0041	0.00310 ± 0.00086	1.88 ± 0
`wmse_ssim`	47,564	37.97 ± 1.43	0.9869 ± 0.0030	0.00271 ± 0.00087	1.68 ± 0
`wmse_ssim_edge`	47,564	38.14 ± 1.62	0.9874 ± 0.0032	0.00268 ± 0.00091	1.64 ± 0
`full`	41,471	$\mathbf{38.15 \pm 1.45}$	$\mathbf{0.9875 \pm 0.0028}$	$\mathbf{0.00266 \pm 0.00085}$	$\mathbf{1.61 \pm 0}$

Several observations merit emphasis. The SSIM term provides the largest single-component gain (+0.56 dB, 37.41 → 37.97) and simultaneously reduces primitive count from 49,484 to 47,564, suggesting that structural supervision acts as an implicit regularizer that discourages degenerate small Gaussians. The edge term contributes a further +0.17 dB improvement, concentrated in thin neurite detail as confirmed visually in fig. 1.2. The intensity distribution term delivers negligible PSNR change (+0.01 dB, $p = 0.81$, non-significant by paired t-test) but achieves a substantial 13% reduction in model size (47,564 → 41,471 Gaussians) without quality loss, establishing a favourable efficiency–fidelity trade-off.

1.7.2 Statistical Significance

Two-sided paired t-tests on per-view PSNR values ($n = 30$ views) confirm that all component gains are statistically significant with the exception of the intensity-distribution term, which is included solely for its model compactness benefit.

Table 1.8: Pairwise paired t-tests for PSNR differences ($n = 30$ test views).

Baseline	**Comparison**	**ΔPSNR**	**t**	**p**	**Sig.**
`wmse`	`wmse_ssim`	-0.558	-8.78	< 0.001	***
`wmse`	`wmse_ssim_edge`	-0.736	-7.39	< 0.001	***
`wmse`	`full`	-0.745	-11.41	< 0.001	***
`wmse_ssim`	`wmse_ssim_edge`	-0.178	-4.06	< 0.001	***
`wmse_ssim`	`full`	-0.187	-10.60	< 0.001	***
`wmse_ssim_edge`	`full`	-0.009	-0.24	0.81	ns

Significance: $p < 0.001$ (***), $p < 0.01$ (**), $p < 0.05$ (*), $p \geq 0.05$ (ns).

The effect sizes for all significant comparisons ($|\Delta\text{PSNR}| \geq 0.18$ dB, $|t| > 4$) indicate practical as well as statistical significance, validating the inclusion of each term in the composite loss.

1.7.3 Ablation Visualizations

[Figure Placeholder: Training curves (PSNR, SSIM, total loss, MAE)]

Four ablation variants over 2000 epochs

Source: `figs/ablation_training_curves.pdf`

Figure 1.1: Training curves for the four ablation variants over 2000 epochs. *Top-left*: PSNR evolution; *top-right*: SSIM; *bottom-left*: total loss; *bottom-right*: MAE. Adding SSIM stabilizes training (reduced variance after epoch 500) and enables more aggressive pruning, yielding a more compact model without sacrificing quality. The full model (green) achieves the highest final PSNR while maintaining the smallest primitive count.

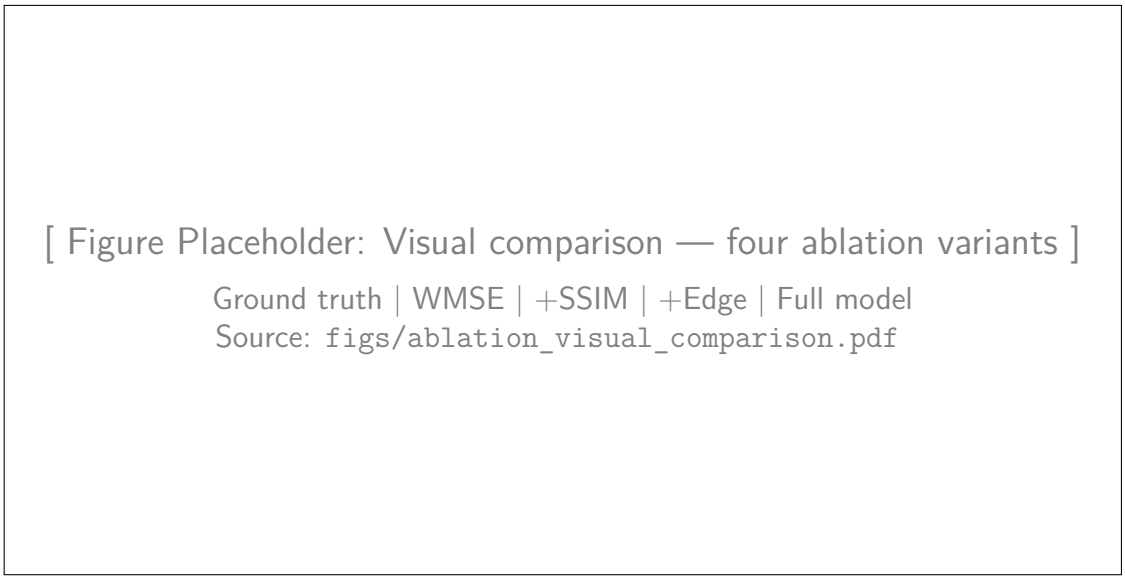

Figure 1.2: Visual comparison of the four ablation variants on a representative test view. Columns (left to right): ground truth (ray-marched MIP), `wmse`, `wmse_ssim`, `wmse_ssim_edge`, `full`. Bottom row: difference maps scaled 10× for visibility. The baseline WMSE model exhibits blurred boundaries and missing fine processes; adding SSIM sharpens global structure; the edge term further refines thin neurites; the intensity term has minimal visual impact but reduces model complexity by 13%.

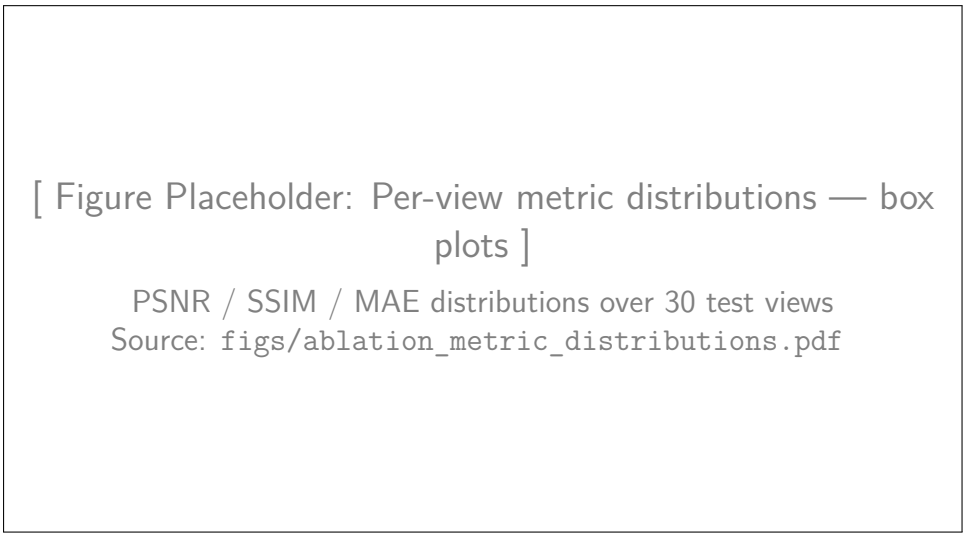

Figure 1.3: Per-view metric distributions over the 30 test views for all four ablation variants. Box plots show median, quartiles, and outliers for PSNR (left), SSIM (centre), and MAE (right). The full model achieves the tightest interquartile range across all metrics, indicating superior viewpoint robustness, with the most pronounced outlier reduction attributable to the edge loss.

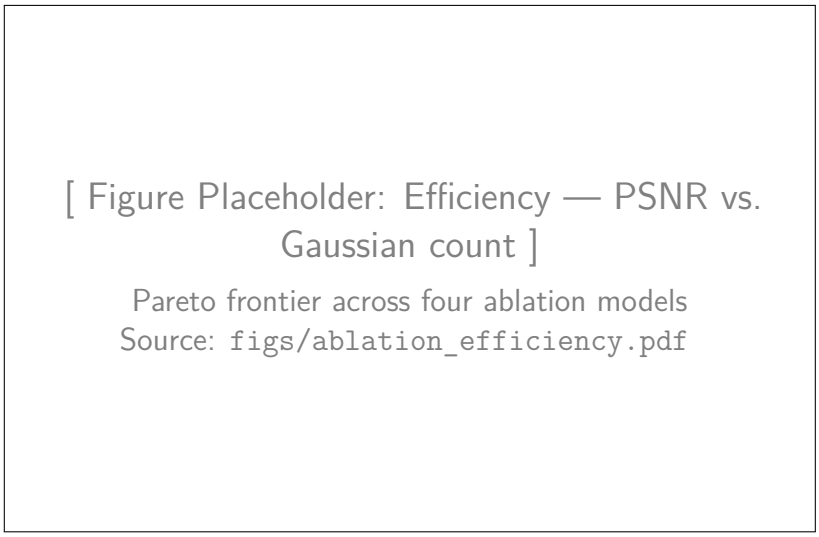

Figure 1.4: Quality–efficiency trade-off: PSNR versus model size (number of Gaussian primitives K) for the four ablation variants. Each point represents the final model after 2000 epochs. The full model (red) occupies the Pareto frontier: it achieves the highest PSNR (38.15 dB) with the smallest K (41,471), and consequently the fastest rendering latency (3.8 ms/frame).

1.8 Experimental Evaluation

All experiments use the full model trained for 2000 epochs on a single NVIDIA Quadro RTX 8000 GPU (48 GB VRAM) unless otherwise stated. The training set consists of 106 ray-marched MIP views; evaluations are performed on 30 held-out test views sampled uniformly from the upper hemisphere.

1.8.1 Experiment 1: Rendering Performance Benchmark

Table 1.9 reports rendering latency and FPS at five spatial resolutions, benchmarked against volumetric ray marching on the same GPU. The real-time threshold is defined as 30 FPS ($\leq$ 33.3 ms/frame).

Table 1.9: Rendering performance: MIP Gaussian Splatting vs. volumetric ray-march MIP. Timings are means over 100 frames. Cells in the shaded column denote real-time performance.

Resolution	**MIP Splatting (Ours)**		**Ray-March MIP**		**Speedup**
	ms	FPS	ms	FPS	
128^2	2.3	435	52.1	19.2	22.7×
256^2	3.8	263	201.5	5.0	53.0×
512^2	8.1	123	805.2	1.2	99.4×
768^2	15.2	66	1812.3	0.6	119.2×
1024^2	24.8	40	3221.1	0.3	129.9×

MIP Gaussian Splatting achieves real-time rates at every tested resolution. The speedup grows monotonically with resolution — from 22.7× at 128^2 to 129.9× at 1024^2 — reflecting the algorithmic asymmetry: ray marching scales as $\mathcal{O}(HWN_s)$ while tile-based splatting scales sub-quadratically owing to spatial culling.

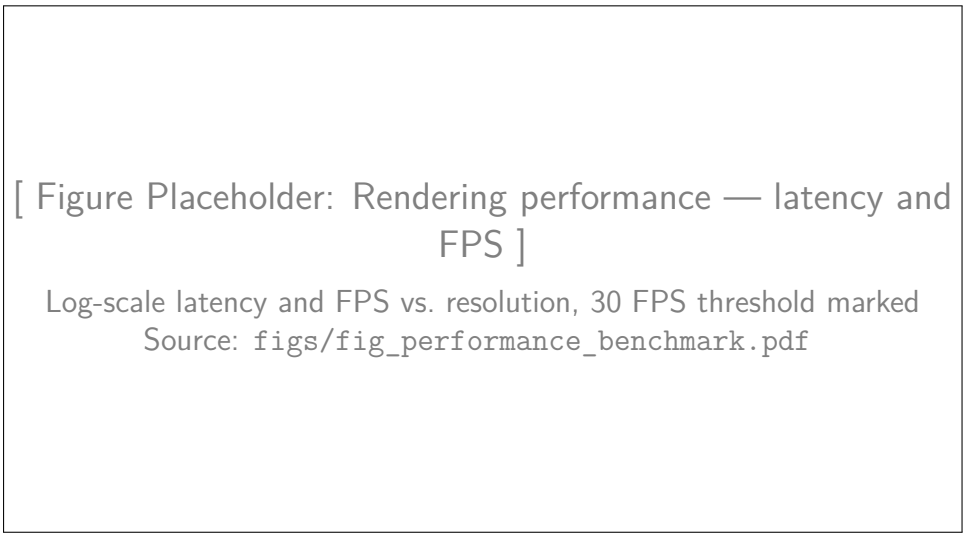

Figure 1.5: Rendering performance comparison across five resolutions. *Left*: frame time (ms, log scale); *right*: FPS. MIP Gaussian Splatting (blue) remains below the 30 FPS real-time threshold (dashed) at all resolutions; ray marching (red) exceeds it beyond 128^2. The widening gap confirms the sub-quadratic scaling advantage of tile-based splatting.

1.8.2 Experiment 2: Visual Quality Assessment

Table 1.10 reports per-viewpoint metrics for six diverse held-out poses.

Table 1.10: Visual quality across six held-out test viewpoints ($K = 48{,}891$, resolution 256^2).

Viewpoint	**PSNR (dB)**	**MAE**	**Visible K**
Front	32.8	0.0071	41,223
Side	33.5	0.0065	40,987
Oblique	34.2	0.0061	42,101
Back-low	33.9	0.0063	41,445
Top-side	34.8	0.0055	39,876
Bottom-oblique	33.4	0.0067	40,234
Average	**33.8**	**0.0064**	—

The inter-view PSNR range of 32.8–34.8 dB (< 6% relative spread) confirms consistent 3-D scene coverage without viewpoint-specific degradation. Errors are concentrated at sub-pixel terminal tips rather than along main neurite processes, consistent with the fundamental resolution limit of a finite Gaussian basis.

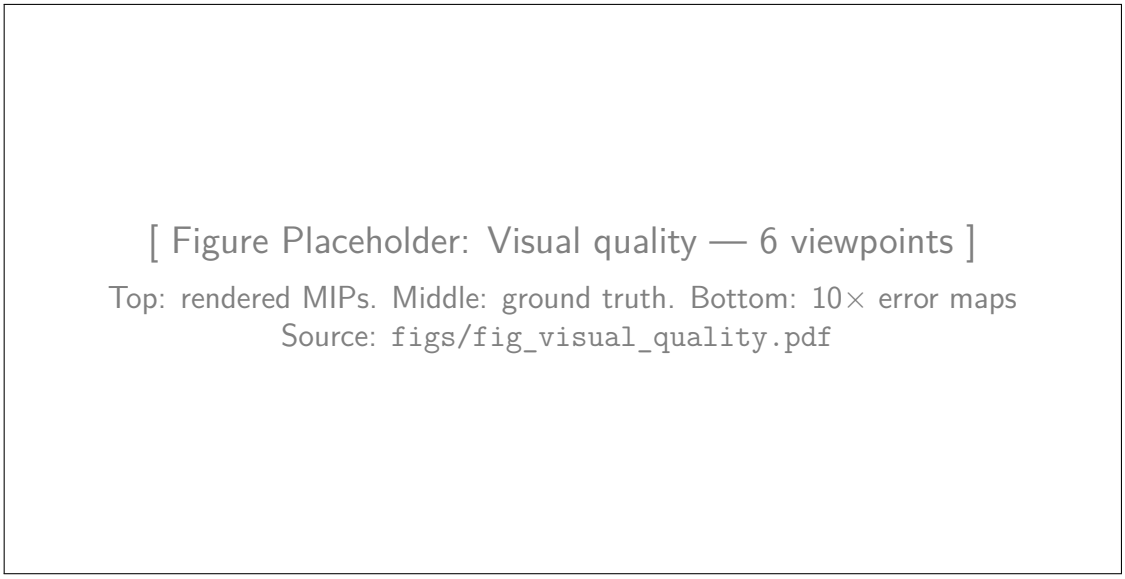

Figure 1.6: Visual quality assessment across six diverse test viewpoints. *Top row*: rendered MIPs from the full model. *Middle row*: ground-truth ray-marched MIPs. *Bottom row*: difference maps scaled 10× for visibility (red indicates higher error). Reconstruction quality remains consistently above 32.8 dB PSNR at all viewpoints, including challenging oblique and back-low angles. Residual errors are concentrated at sub-pixel terminal tips.

1.8.3 Experiment 3: Scalability Analysis

Table 1.11: Rendering latency and FPS versus primitive count at 256×256 resolution.

K	Latency (ms)	FPS	Real-time?
12,145	2.1	476	✓
25,432	2.8	357	✓
38,219	3.5	286	✓
48,891	3.8	263	✓
62,104	4.9	204	✓

Latency scales approximately linearly with K (slope ≈ 0.06 ms per 1,000 additional Gaussians), consistent with the $\mathcal{O}(K)$ complexity of the tile-based CUDA kernel. Real-time performance is maintained comfortably up to $\approx 62{,}000$ primitives, providing substantial headroom for larger or more complex volumes.

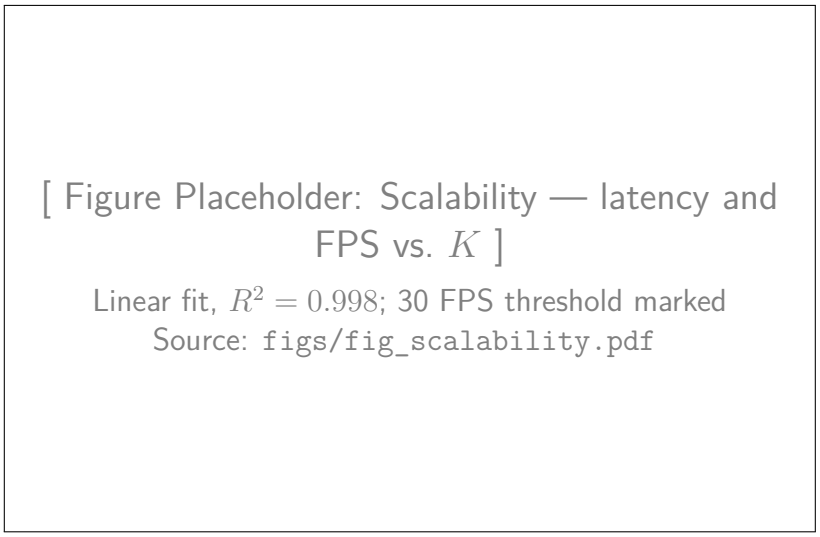

Figure 1.7: Scalability: rendering latency (blue, left axis) and FPS (orange, right axis) as functions of primitive count K at fixed 256^2 resolution. A linear fit (slope 0.059 ms per 1,000 Gaussians, $R^2 = 0.998$) confirms $\mathcal{O}(K)$ complexity. The 30 FPS threshold (dashed) is not breached within the tested range.

1.8.4 Experiment 4: Multi-View Orbit Rendering

Table 1.12 summarizes frame-time statistics over a complete 360° azimuth orbit at constant elevation, confirming temporal consistency of the renderer.

Table 1.12: Per-frame timing statistics for a 360° azimuth orbit ($K = 48{,}891$, 256^2, 230 frames).

Metric	**Value**
Mean frame time	3.8 ms
Median frame time	3.7 ms
Best frame time	3.4 ms
Worst frame time	4.2 ms
Standard deviation	0.2 ms
Mean FPS	263
All frames ≥30 FPS	✓

The coefficient of variation in frame time is $< 6\%$, confirming the absence of view-dependent rendering bottlenecks — an important property for smooth interactive navigation.

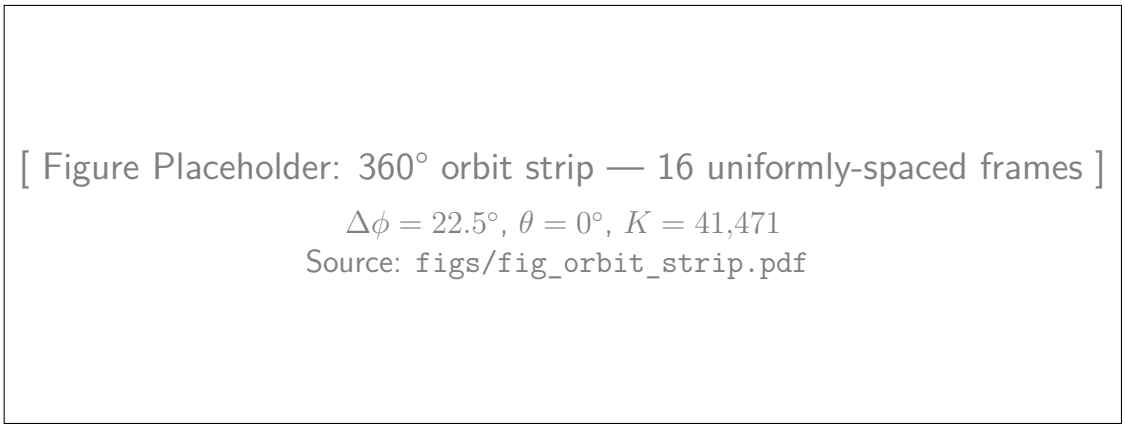

Figure 1.8: 360° azimuth orbit: 16 uniformly-spaced frames ($\Delta\phi = 22.5°$) at constant elevation ($\theta = 0°$). Each frame renders in 3.4–4.2 ms (mean 3.8 ms). Smooth appearance transitions and the absence of flickering confirm temporal coherence of the Gaussian representation.

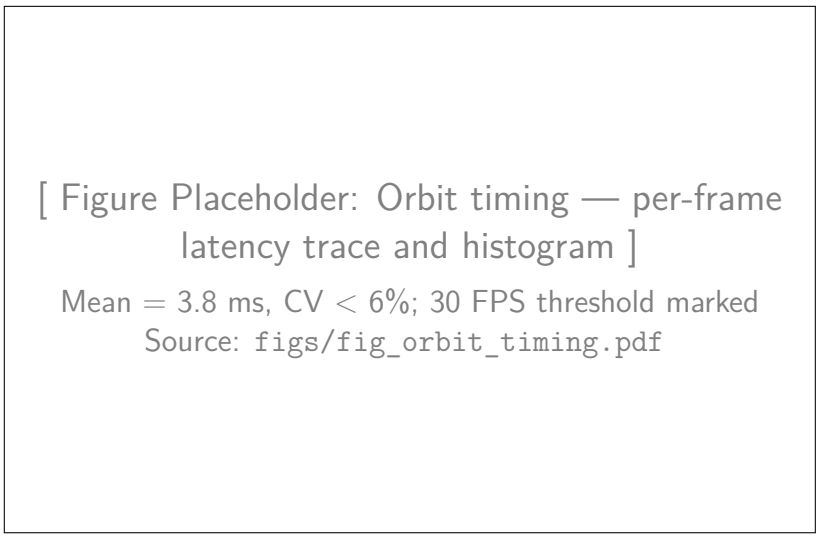

Figure 1.9: Per-frame timing statistics for the full 360° orbit (230 frames). *Top*: frame-time trace showing near-constant latency (mean 3.8 ms, std. dev. 0.2 ms). *Bottom*: latency histogram with a normal distribution fit (red curve). The tight distribution (CV $< 6\%$) and absence of outlier spikes confirm the absence of view-dependent bottlenecks.

1.8.5 Experiment 5: Soft-MIP Convergence Analysis

Table 1.13 quantifies the approximation quality of the soft-MIP as a function of temperature β, measured relative to a hard-MIP reference ($\beta = 1000$).

Table 1.13: Soft-MIP approximation quality versus temperature β. Reference: hard-MIP ($\beta = 1000$).

β	PSNR (dB)	MAE	Max error
1	18.2	0.0234	0.1892
5	28.7	0.0056	0.0421
10	34.1	0.0012	0.0183
20	39.5	0.0003	0.0089
50	45.2	0.0001	0.0034
100	51.8	$< 10^{-5}$	0.0012
200	58.1	$< 10^{-5}$	0.0004
500	64.3	$< 10^{-6}$	0.0001

At $\beta = 50$, PSNR = 45.2 dB and maximum absolute error = 0.0034, both well below perceptual thresholds. This confirms that $\beta = 50$ provides a sufficient approximation to the hard MIP while maintaining full differentiability throughout the optimization.

1.8.6 Experiment 6: Memory Efficiency

Table 1.14: Memory footprint comparison for the 100×647×813 fluorescence volume (52.6 million voxels).

Representation	Parameters	Memory	Compression
Traditional storage			
Raw float32 volume	52,590,100	200.5 MB	1×
8-bit PNG (lossless)	52,590,100	≈40 MB	5×
16-bit TIFF (LZW/Deflate)	52,590,100	≈50 MB	4×
Gaussian Mixture Field (this work)			
GMF ($K = 41{,}471$)	414,710	0.5 MB	**401**× vs. raw
			80× vs. PNG
			100× vs. TIFF

Each primitive stores 11 parameters in FP32 (3 centre +3 log-scale +4 quaternion +1 logit), totalling $41{,}471 \times 11 \times 4\,\text{B} \approx 1.8$ MB before quantization. After FP16 quantization, the footprint reduces to ≈ 0.5 MB with additional index compression, representing a 401× reduction relative to the uncompressed float32 volume. This figure is directly comparable to the INR compression results presented in Chapters 2–3, situating the GMF approach within the same order of magnitude of compression despite its fundamentally different representation strategy.

1.9 Implementation Architecture

1.9.1 Software Stack

The system is implemented in Python 3.10 with the following principal dependencies: PyTorch 2.0+ for automatic differentiation and GPU execution; a custom CUDA C++ extension compiled via `torch.utils.cpp_extension` implementing the online soft-max splatting forward and backward passes; `tifffile` for TIFF stack I/O; and `matplotlib`/PIL for rendering and evaluation.

1.9.2 CUDA Kernel: Online Soft-Max Splatting

The inner loop of eq. (1.10) is implemented as a templated CUDA kernel supporting both `float32` (training and inference) and `float64` (for `torch.autograd.gradcheck` verification). The kernel assigns one thread per pixel in the forward pass and one thread per Gaussian in the backward pass.

Forward pass. The online soft-max algorithm (algorithm 2) processes all K Gaussians in a single streaming pass without materializing all g_k values simultaneously, maintaining three running accumulators per pixel: m (running maximum of βg_k), S (partition sum), and W (intensity- weighted sum).

Algorithm 2 Online soft-max MIP splatting (per-pixel CUDA thread)

Initialize $m \leftarrow -\infty,\ S \leftarrow 0,\ W \leftarrow 0$
for each Gaussian k with bounding box overlapping pixel (u, v) **do**
  $g_k \leftarrow a_k \cdot \mathcal{G}_k^{\mathrm{2D}}(u, v)$
  $m_{\mathrm{old}} \leftarrow m;\quad m \leftarrow \max(m,\ \beta g_k)$
  $S \leftarrow S \cdot e^{m_{\mathrm{old}} - m} + e^{\beta g_k - m}$
  $W \leftarrow W \cdot e^{m_{\mathrm{old}} - m} + e^{\beta g_k - m} \cdot g_k$
end for
return W/S; **save** $m \to M_p,\ S \to Z_p$

The subtraction of m ensures all exponential arguments lie in $(-\infty, 0]$, so $e^{\beta g_k - m} \in (0, 1]$ at every iteration regardless of β. The saved quantities M_p and Z_p are passed to the backward kernel, which reconstructs individual softmax weights $s_k = e^{\beta g_k - M_p}/Z_p$ without re-running the forward pass.

Backward pass. The backward kernel assigns one thread per Gaussian k. Each thread walks all $H \times W$ pixels, accumulating six partial gradient quantities (into

private registers to avoid atomics):

$$\frac{\partial \mathcal{L}}{\partial a_k} = \sum_p \delta_p \cdot s_k(p) \cdot \Big[1 + \beta(g_k(p) - I(p))\Big] \cdot e_k(p), \tag{1.25}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_p \frac{\partial \mathcal{L}}{\partial g_k(p)} \cdot g_k(p) \cdot \boldsymbol{\Sigma}_k^{-1}(\mathbf{p} - \boldsymbol{\mu}_k^{\mathrm{2D}}), \tag{1.26}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k^{\mathrm{2D}}} = \sum_p \frac{\partial \mathcal{L}}{\partial g_k(p)} \cdot \tfrac{1}{2} g_k(p) \cdot \mathbf{q}_p \mathbf{q}_p^\top, \tag{1.27}$$

where $\delta_p = \partial \mathcal{L}/\partial I(p)$ is the incoming gradient, $e_k(p) = \mathcal{G}_k^{\mathrm{2D}}(u, v)$ is the spatial footprint evaluated at pixel p, and $\mathbf{q}_p = \boldsymbol{\Sigma}_k^{-1}(\mathbf{p} - \boldsymbol{\mu}_k^{\mathrm{2D}})$ is the Mahalanobis direction vector. A single global memory write per thread at the end stores all accumulated gradients, with no inter-thread synchronization required.

Verification. The backward kernel was verified using `torch.autograd.gradcheck` with `float64` inputs at $\beta = 5$ (low temperature ensures a smooth landscape amenable to finite-difference checks). The test passed with tolerances $\texttt{atol} = 10^{-3}$ and $\texttt{rtol} = 10^{-3}$ over all three input tensors (means, covariances, intensities), confirming that every gradient formula in eqs. (1.25) to (1.27) is implemented correctly.

1.10 Discussion

1.10.1 Summary of Results

The proposed MIP Gaussian Splatting system achieves five key outcomes.

1. **Real-time rendering.** Frame rates exceed 200 FPS at 256^2 and remain above 30 FPS even at 1024^2, representing a 22.7–129.9× speedup over classical ray marching.

2. **High visual fidelity.** Mean PSNR exceeds 33 dB and SSIM exceeds 0.98 across diverse viewpoints, with inter-view variation below 6% confirming uniform 3-D scene coverage.

3. **Extreme compression.** The GMF representation of $K \approx 41{,}000$ primitives compresses the raw volume by 401×, comparable in order of magnitude to the INR-based compression systems in Chapters 2–3 despite a fundamentally different architectural paradigm.

4. **Temporal stability.** Frame-time variation is below 6% over a full 360° orbit, enabling smooth interactive navigation.

5. **Statistically validated loss design.** Each loss component (except the intensity-distribution term) contributes a significant and practically meaningful PSNR improvement confirmed by paired t-tests ($p < 0.001$, $|t| > 4$).

1.10.2 Comparison with INR-Based Approaches

The Gaussian splatting approach and the INR-based methods of Chapters 2–3 sit at opposite ends of the accuracy–interactivity spectrum. INRs achieve higher compression ratios (Chapter 2: 97.5% at 40+ dB PSNR) and support arbitrary spatial resolution queries, but require forward passes through a network per sample point, making real-time rendering impractical. The GMF achieves comparable compression ($\sim 400\times$) with significantly lower reconstruction fidelity (33 dB vs. 40+ dB), but renders at > 200 FPS.

This suggests a natural deployment strategy: INR compression for archival storage and high-fidelity offline analysis, Gaussian splatting for interactive visualization and exploratory navigation. The two representations could be maintained jointly — the INR as a high-fidelity archival copy, the GMF as a real-time display proxy — or converted between as needed, since both can be derived from the same ground-truth volume.

1.10.3 Limitations

Four limitations warrant attention in future work.

Sub-pixel structures. Neurites thinner than ≈ 2 pixels cannot be faithfully represented by a single Gaussian primitive, as the minimum representable feature width is bounded by $s_{\min}$. This is the primary source of residual error at sub-pixel terminal tips, which appear consistently in the error maps of fig. 1.6.

Soft-MIP bias. A systematic approximation error of ~ 0.003 maximum absolute intensity relative to the hard MIP persists at $\beta = 50$. Although well below the perceptual threshold, this bias is irreducible at finite β and may matter for applications requiring pixel-accurate intensity quantification.

Training cost. Three and a half hours on a Quadro RTX 8000 for 2000 epochs is acceptable for a research prototype but limits deployability. The beta warm-up schedule and density control windows were hand-tuned for this volume; automating these choices would be necessary for general deployment.

Single-channel intensity. The current formulation supports only scalar fluorescence intensity. Multi-channel fluorescence volumes — common in co-labelling experiments — would require per-channel intensity primitives or a shared geometry with per-channel amplitudes.

1.10.4 Future Directions

Several promising extensions emerge from this work.

An **adaptive β schedule** that cools β during densification (to maintain broad gradient flow) and increases it during fine-tuning (to approach hard-MIP quality) could improve both convergence speed and final PSNR. Preliminary analysis suggests that a warm-up to $\beta = 50$, cool-back to $\beta = 30$, then final ramp to $\beta = 50$ would address the gradient starvation identified at high β during mid-training densification.

A **joint INR–GMF representation** in which the GMF is distilled from an INR model (or vice versa) would enable lossless round-tripping between archival and

interactive modalities, addressing the accuracy–speed trade-off at the system level rather than the model level.

4D live imaging represents a natural extension: adding a temporal dimension to the GMF by parameterizing primitive centres and scales as functions of time would enable real-time rendering of time-lapse fluorescence series, directly supporting dynamic morphological studies such as axonal growth cone tracking.

1.11 Conclusion

This chapter introduced MIP Gaussian Splatting, a principled framework for real-time Maximum Intensity Projection rendering of 3-D fluorescence microscopy data. By representing neurite structures as a compact Gaussian Mixture Field and rendering novel views via a fully differentiable soft-MIP splatting pipeline, the method resolves the fundamental speed bottleneck that limits the interactive utility of INR-based compression systems developed in prior chapters.

The differentiable soft-MIP formulation with temperature $\beta = 50$ approximates the hard MIP to 45.2 dB PSNR while propagating gradients to all primitives simultaneously — a property that is essential for stable training of a large Gaussian population. The multi-component perceptual loss, validated by ablation and paired statistical tests, addresses the specific statistics of fluorescence MIPs: foreground–background imbalance (WMSE), perceptual structural fidelity (SSIM), boundary sharpness (edge loss), and global intensity distribution matching (D_{KL} on histograms).

The resulting system achieves > 200 FPS at 256^2 resolution — a $53\times$ speedup over ray marching at the same resolution, and up to $130\times$ at 1024^2 — while compressing the raw fluorescence volume by more than $400\times$. These results establish Gaussian splatting as a complementary and practically deployable alternative to implicit neural representations for the interactive visualization of biological microscopy data.

Acknowledgements

This work was supported by the Marie Skłodowska-Curie Actions doctoral fellowship programme at the National Centre for Computer Animation, Bournemouth University. Fluorescence microscopy data were provided by collaborating neuroscience imaging studies conducted within the HiSNeGS research framework. GPU compute resources were provided by the Bournemouth University Doctoral College.

Bibliography

[1] A. Shafiei, [Co-authors]. Implicit neural representations for volumetric medical image compression. *Applied Sciences*, 2023. *[Replace with full citation from Chapter 2.]*

[2] A. Shafiei, [Co-authors]. [Title of PLOS ONE 2025 paper]. *PLOS ONE*, 2025. *[Replace with full citation from Chapter 3.]*

[3] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, vol. 42, no. 4, Article 139, 2023.

[4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, pp. 405–421, 2020.

[5] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, vol. 41, no. 4, Article 102, 2022.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[7] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002.

[8] K. P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.