INFORMATION-GEOMETRIC LATENT ENCODING FOR 3D GAUSSIAN SPLATTING

By

Lee Ostadi

July, 2026

Director of Thesis: David Hart, PhD

Major Department: Computer Science

ABSTRACT

3D Gaussian splatting represents a scene as an unordered set of anisotropic Gaussian primitives whose centers, covariances, colors, and opacities jointly determine rendered appearance. This thesis develops the Geometric Latent Encoding Autoencoder (GLEA), a class-agnostic model that maps a variable-cardinality splat set to a 512-dimensional code and reconstructs up to 3,200 Gaussian primitives without encoder skip connections. The central design question is whether primitive comparison should use covariance information directly rather than relying only on learned content features or center proximity. GLEA addresses that question with a gated attention branch derived from the Bhattacharyya coefficient, a closed-form measure of overlap between probability distributions. The coefficient is a positive-semidefinite kernel under the square-root density embedding, and its associated distance agrees locally, to second order, with Fisher–Rao geometry for nearby distributions.

The implemented system combines PCA pose alignment, cuboid normalization, multi-resolution hash features, learned grouping, gated geometric and content attention, a hierarchical intermediate representation, a deterministic latent-only decoder, sparse Bhattacharyya–Chamfer reconstruction, multi-view rendering supervision, and conditional flow matching. On a disjoint ShapeSplat split, evaluation over 128 held-out objects yields mean center Chamfer distance of 1.54×10^{-3}, PSNR of 25.57 ± 2.64 dB, SSIM of 0.956 ± 0.016, and LPIPS of 0.124 ± 0.032. Under a matched four-epoch, single-seed ablation budget, removing Bhattacharyya attention lowers mean held-out PSNR by 1.20 dB and worsens every reported

external reconstruction metric; removing PCA alignment lowers PSNR by 0.67 dB. These results support covariance-aware attention under the evaluated configuration. A controlled analytic battery independently verifies that the underlying kernel is covariance-sensitive in exactly the way a center-only score is not, matching its closed form to a relative error of 10^{-12}. Separately, relaxing the strict 512-dimensional bottleneck to decode from the full hierarchical latent raises reconstruction fidelity at the cost of the compact, analyzable code, making the fidelity–interpretability tradeoff explicit. They do not yet establish category-level transfer, unconditional generation, global preservation of Fisher–Rao distances in the learned code, or comparative state-of-the-art performance. Multi-seed replication and matched external baselines remain necessary.

INFORMATION-GEOMETRIC LATENT ENCODING FOR 3D GAUSSIAN SPLATTING

A Thesis

Presented to the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirement for the

Master of Science in Computer Science

By

Lee Ostadi

July, 2026

Director of Thesis: David Hart, PhD

Thesis Committee Members:

Nic Herndon, PhD

Moritz Dannhauer, PhD

DEDICATION

For those who taught me to look for the structure beneath the surface,

and for my family, whose support made the looking possible.

ACKNOWLEDGEMENTS

I thank my principal supervisor, David Hart, whose close reading and insistence on rigor shaped this thesis at every stage; its arguments are sharper for his questions. I am grateful to my committee members, Nic Herndon and Moritz Dannhauer, for their guidance and for the outside perspectives they brought to a problem I had been examining from only one direction.

I thank the Department of Computer Science at East Carolina University for the resources and computing support that made these experiments possible, and my fellow graduate students for the conversations that untangled more ideas than I can count.

This thesis grew out of a conviction that the right geometry, made explicit in the architecture, can do work that a larger model would otherwise be left to learn by force. I am grateful to the mathematicians and physicists whose ideas I have borrowed and reassembled here, and to everyone who tolerated my habit of taking a working system apart simply to understand why it worked.

Finally, I thank my family. Their steady support gave me the room to follow a difficult problem to its end, and their patience outlasted my own more than once.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Problem and Motivation

3D Gaussian splatting (3DGS) represents a scene as an unordered collection of anisotropic Gaussian primitives whose means, covariances, colors, and opacities jointly determine rendered appearance [Kerbl et al., 2023]. Existing efficiency work has largely focused on compressing the parameters of a fitted scene or reorganizing splats into structures that grid-based models can process [Youn et al., 2025, Niedermayr et al., 2024, Zhang et al., 2024]. Those directions address storage and downstream generation, but they leave a distinct representation-learning question open: how should an encoder compare and aggregate Gaussian primitives when each primitive has spatial extent and orientation rather than only a center point?

A 3D Gaussian primitive can be written as

$$G_i = \Big(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \mathbf{c}_i, \alpha_i\Big), \tag{1.1}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is its center, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times 3}$ is a positive-definite covariance matrix, $\mathbf{c}_i \in \mathbb{R}^3$ is color, and $\alpha_i \in [0, 1]$ is opacity. In practice, the covariance is parameterized through scale and rotation. A scene is therefore a variable-cardinality set of structured primitives rather than a conventional point cloud.

Point-set encoders can still process these parameters as feature channels, and a sufficiently expressive network may learn covariance-sensitive relationships. The narrower issue is that a center-only geometric score cannot express those relationships directly. Two primitives with identical means but different covariances have zero center distance even when their spatial supports differ substantially. Conversely, primitives with separated means may overlap

strongly when their covariances are broad. This motivates testing a comparison rule that uses the full Gaussian distributions.

1.2 Representation Learning Rather Than Scene Compression

The project is best understood as representation learning rather than only compression. A scene-specific codec asks how many bits are needed to preserve one fitted scene at a target rendering quality. GLEA instead asks whether a shared map learned across many objects can identify recurring structure and express each complete splat scene through a common code. The distinction is operational. The decoder receives no cached encoder features, so successful reconstruction requires the 512-dimensional code to carry every signal available to the reported decoder.

This bottleneck creates a useful test, but it does not automatically make the code semantic, disentangled, transferable, or generative. Those properties require separate measurements. Reconstruction establishes that the code is sufficient for the evaluated inverse problem; interpolation probes decoder behavior on selected paths; participation ratio describes variance concentration in sampled codes; and direct prior sampling tests whether a proposed generative path is actually wired to the decoder. The thesis keeps those evidentiary levels separate.

1.3 Covariance-Aware Comparison

Consider two zero-mean Gaussians,

$$G_1 = \mathcal{N}(\mathbf{0}, \mathrm{diag}(1, 1, 0.01)) \,, \tag{1.2}$$

$$G_2 = \mathcal{N}(\mathbf{0}, \mathrm{diag}(0.01, 0.01, 1)) \,. \tag{1.3}$$

Their centers coincide, so a center-distance score assigns distance zero. Their covariance

structures, however, describe different spatial supports. For these matrices, the Bhattacharyya coefficient is approximately 0.088, indicating limited distributional overlap despite the identical centers. The example does not imply that point-set networks cannot learn the distinction; it shows that a center-only affinity omits information that is available analytically.

This thesis studies the Bhattacharyya coefficient as a geometric affinity between Gaussian primitives. The coefficient compares complete probability distributions in closed form and is a positive-semidefinite kernel under the square-root density embedding. Its associated distance also agrees locally, to second order, with the Fisher-Rao metric for nearby distributions. These facts motivate the score, but they do not by themselves establish that it improves a learned representation. That question is empirical.

1.4 From Geometric Motivation to Testable Evidence

The argument of the thesis has three layers. First, the mathematics identifies information that a center-only score discards and supplies a tractable covariance-aware affinity. Second, the implementation specifies where that affinity enters the network, which optional paths are active, and what the decoder actually consumes. Third, the experiments determine whether the branch improves external reconstruction metrics under matched conditions. No layer substitutes for the next: a valid kernel is not evidence of better reconstruction, and better reconstruction is not proof that the final latent space is globally isometric to Fisher–Rao geometry.

EVIDENCE BOUNDARY

The strongest present claim is component-level and empirical: under the documented four-epoch v48 comparison, removing Bhattacharyya attention reduces held-out image and position fidelity. Broader claims about transfer, generation, topology, or universal superiority are hypotheses for dedicated experiments rather than conclusions of the current study.

1.5 Research Question and Scope

The central research question is:

> Does a covariance-aware geometric branch improve latent encoding and reconstruction of Gaussian-splat scenes relative to an otherwise matched content-attention model?

To study this question, the thesis introduces the Geometric Latent Encoding Autoencoder (GLEA). It maps a variable-length splat set to a 512-dimensional code and reconstructs at most 3,200 output primitives without encoder skip connections. The reported system combines a Bhattacharyya-based geometric score with learned content attention and uses both primitive-space and rendering-space supervision.

The evaluation is intentionally narrower than a complete generative model of 3D scenes. It measures reconstruction on a disjoint ShapeSplat split, compares selected components under a matched four-epoch budget, and reports descriptive analyses of the learned codes. It does not establish category-held-out transfer, cross-dataset generalization, unconditional sampling from a unit Gaussian, or state-of-the-art performance against matched external systems. Those boundaries are important because they separate what the current experiments show from longer-term uses of the representation.

1.6 System Overview

The reported computational path contains five stages:

1. **Canonicalization and spatial features.** PCA pose alignment and cuboid normalization reduce pose and scale variation. A multi-resolution hash encoding adds spatial features [Müller et al., 2022, Yao et al., 2025].
2. **Learned grouping.** Variable-length primitive sets are aggregated into 256 groups, reducing the sequence length used by the attention layers [Ma et al., 2025].
3. **Gated geometric and content attention.** Each reported attention layer mixes a Bhattacharyya-based geometric score with learned dot-product content scores before softmax normalization.

4. **Hierarchical intermediate representation.** Global, regional, and local summaries contain 128, 8×64, and $8 \times 8 \times 32$ dimensions, respectively. Their 2,688 concatenated dimensions are projected to the 512-dimensional decoder code.

5. **Latent-only reconstruction.** A deterministic decoder reconstructs up to 3,200 primitives using only the compact code. The reported runs disable variational output heads, learned point-count prediction, and noise injection. Conditional flow matching supplies an additional latent regularization objective along linear conditional paths [Lipman et al., 2023].

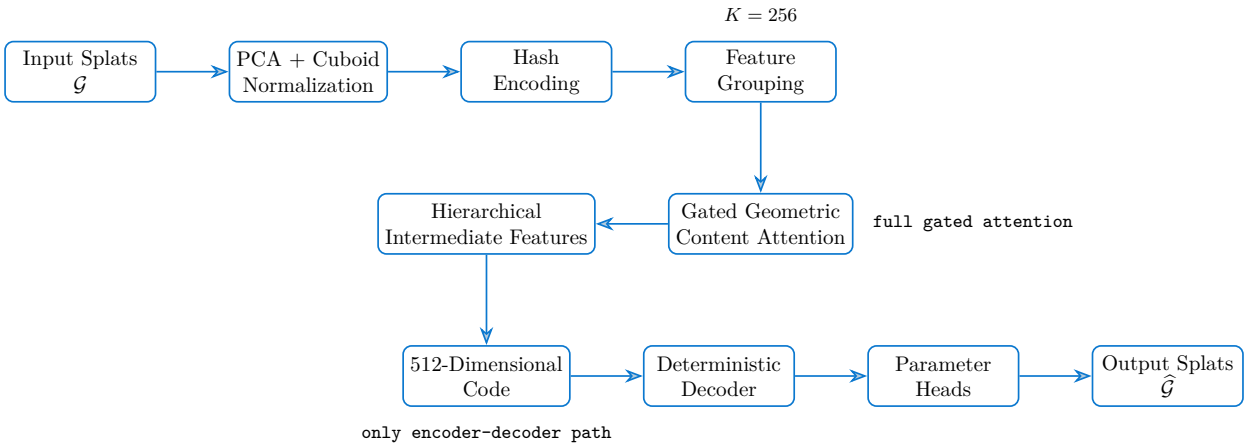

Figure 1.1: Overview of the reported GLEA configuration. The encoder compresses a variable-length splat set into a 512-dimensional code. No encoder features bypass this bottleneck. Optional variational and noise-injection paths are not active in the reported experiments.

The latent-only decoder is an architectural constraint, not evidence that the code is already suitable as a general-purpose "3D token." It makes such downstream tests possible because decoding does not require cached encoder features. Whether the code transfers to other tasks or representations remains a separate empirical question.

1.7 Contributions

The thesis makes three principal contributions:

1. **Distribution-aware comparison.** It derives and implements a Bhattacharyya-based affinity for anisotropic Gaussian primitives, explains its kernel interpretation, and states the local relationship to Fisher-Rao geometry.

2. **An across-scene latent autoencoder for native splats.** It combines canonicalization, hash features, learned grouping, gated geometric/content attention, a hierarchical

intermediate representation, and a strict 512-dimensional decoder bottleneck.

3. **A controlled empirical evaluation.** On 128 held-out ShapeSplat objects, the full model obtains mean PSNR of 25.57 dB, SSIM of 0.956, LPIPS of 0.124, and center Chamfer distance of 1.54×10^{-3}. In the matched four-epoch, single-seed ablation, removing geometric attention lowers mean held-out PSNR by 1.20 dB. These results support the utility of the branch under the evaluated configuration; longer, multi-seed experiments are needed to estimate stability.

4. **An auditable account of the reported system.** This thesis distinguishes the active v48 computational path from optional or historical paths, records the exact seed and split, separates training from evaluation rendering settings, and packages the metric summaries and training logs needed to trace the main numerical claims.

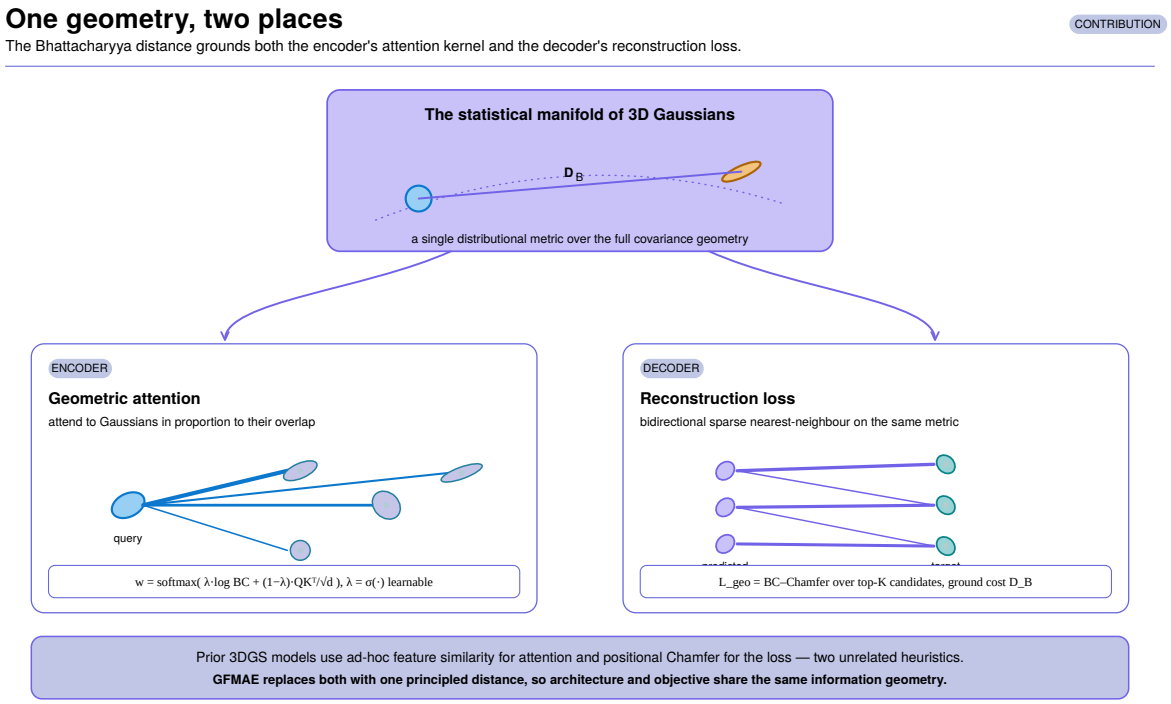

Figure 1.2: The organizing idea of this thesis. A single distributional comparison between anisotropic Gaussians—the Bhattacharyya affinity—appears in two otherwise unrelated roles: as the geometric term of the encoder's gated attention, and as the ground cost of the sparse reconstruction objective. Prior 3DGS pipelines typically pair a feature-similarity attention with a positional Chamfer loss; here both roles read the same covariance-aware quantity. The reconstruction aggregation is bidirectional sparse nearest-neighbour (BC–Chamfer), not optimal transport; see Chapter 4.

1.8 Thesis Organization

Chapter 2 positions the work relative to 3DGS compression, point-set representation learning, information geometry, and latent regularization. Chapter 3 develops the mathematical background for Gaussian distributions, Fisher information, and the Bhattacharyya coefficient. Chapter 4 presents the reported architecture and active training objectives. Chapter 5 describes the experimental protocol, reconstruction results, ablations, and latent diagnostics. Chapter 6 summarizes the evidence, limitations, and immediate next experiments.

CHAPTER 2: BACKGROUND AND RELATED WORK

This chapter positions GLEA relative to work on 3D Gaussian splatting, set-based representation learning, native-splat pretraining, compression and restructuring, information-geometric comparison, and latent regularization. The relevant distinction is not merely whether an earlier method uses Gaussian primitives. It is which learning problem the method solves, what information reaches its decoder or downstream task, and how similarity between primitives is represented. Some methods optimize or compress one fitted scene; some restructure splats into regular arrays; some learn discriminative features across a dataset; and some generate Gaussian parameters from an external conditioning signal. GLEA studies a narrower representation-learning problem: whether a complete splat scene can be encoded into a compact, self-contained latent code and reconstructed with an explicit covariance-aware comparison between primitives.

2.1 3D Gaussian Splatting as an Explicit Scene Representation

3D Gaussian splatting (3DGS) represents a scene as a collection of anisotropic Gaussian primitives with position, covariance, color, and opacity parameters [Kerbl et al., 2023]. In contrast with an implicit neural field, the scene state is explicit: each primitive has a location, an oriented spatial footprint, an appearance contribution, and an opacity. A differentiable rasterizer projects and composites these primitives to render images. The explicit form supports high rendering throughput and direct manipulation of primitive parameters.

The covariance is central to the representation rather than an auxiliary field. In common parameterizations,

$$\mathbf{\Sigma}_i = \mathbf{R}(\mathbf{q}_i)\,\mathrm{diag}(\mathbf{s}_i^2)\mathbf{R}(\mathbf{q}_i)^\top, \tag{2.1}$$

where the quaternion $\mathbf{q}_i$ determines orientation and the positive scale vector $\mathbf{s}_i$ determines the principal-axis lengths. Consequently, two primitives with the same center and appearance can contribute differently to a rendered image because their covariances place density in different spatial regions. This observation motivates the thesis's focus on covariance-aware primitive comparison.

The 3DGS representation has been extended beyond static novel-view synthesis. Dynamic 3D Gaussians track time-varying scenes [Luiten et al., 2024]; DreamGaussian uses Gaussian splats in a generative content-creation pipeline [Tang et al., 2024]; and a growing compression literature reduces the memory and transmission cost of fitted scenes [Niedermayr et al., 2024, Youn et al., 2025]. These developments establish the versatility of the representation, but they do not by themselves determine how to learn an across-scene latent code from native splat sets.

2.2 Compression, Restructuring, and Across-Scene Encoding

Youn et al. [Youn et al., 2025] organize efficient 3DGS methods around parameter compression and restructuring. Parameter-compression methods prune redundant primitives, quantize attributes, exploit entropy models, or otherwise reduce the storage cost of an already fitted scene [Niedermayr et al., 2024]. Their natural unit of optimization is often one scene, and their success criterion is a bitrate–quality tradeoff under novel-view rendering.

Restructuring methods impose a regular data layout so that architectures designed for grids or tensors can process Gaussian scenes. GaussianCube [Zhang et al., 2024], for example, maps Gaussian content into a structured representation suitable for downstream generative modeling. Other work seeks a unified representation of 3D Gaussian scenes across instances [Xin et al., 2025]. Restructuring can make convolutional or diffusion architectures applicable, but it changes the computational substrate: the model reasons over the imposed grid or tensor as well as over the original splats.

GLEA is related to both directions but solves a different objective. It learns one encoder and decoder across a dataset, accepts a variable-cardinality set of native Gaussian parameters, and reconstructs native Gaussian parameters from a compact code. Its primary target is therefore not minimum bitrate for an individual fitted scene, nor the construction of a regular grid for another model. The target is a reusable representation map

$$\mathcal{G} \xrightarrow{E_\phi} \mathbf{z} \xrightarrow{D_\theta} \widehat{\mathcal{G}}, \tag{2.2}$$

where $\mathbf{z}$ is self-contained at decode time.

This distinction matters when interpreting claims. A low-rate scene codec may preserve rendered views extremely well without learning a semantically organized cross-instance space. Conversely, an across-scene autoencoder may learn reusable regularities while using more bits than a scene-specific codec. The two systems answer different questions and should not be ranked by a single metric without matching the task and evaluation protocol.

2.3 Set and Point-Cloud Representation Learning

PointNet established a general recipe for permutation-invariant learning on unordered point sets [Qi et al., 2017]. A shared pointwise map followed by a symmetric aggregation operation allows the output to remain invariant to input ordering. FoldingNet [Yang et al., 2018] and AtlasNet [Groueix et al., 2018] provide important precedents for compact 3D shape codes and latent-only decoders that generate point sets or surface patches. These systems demonstrate that unordered geometric data can be compressed into fixed-dimensional representations without assigning a persistent input order.

A Gaussian-splat set inherits the permutation issue but contains more structure than a point set. Each primitive carries a covariance that determines spatial support and orientation, appearance variables that affect rendering, and an opacity that changes compositing. One

straightforward strategy is to concatenate these values as feature channels and let a sufficiently expressive set network discover useful relationships. That remains a valid baseline. GLEA asks whether one specific relationship – distributional overlap – should be supplied analytically as an inductive bias instead of being left entirely implicit.

The distinction can be expressed at the level of pairwise comparison. A center-only rule compares $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$. A learned content rule compares transformed attribute vectors, often through $\mathbf{q}_i^\top \mathbf{k}_j$. A covariance-aware rule additionally asks how the spatial densities overlap. These are not mutually exclusive: GLEA gates a geometric score with learned content attention rather than replacing learned content.

2.4 Native-Splat Pretraining and Cross-Modal Representation Learning

ShapeSplat provides a large collection of Gaussian-splat objects and a self-supervised pretraining framework for native splat parameters [Ma et al., 2025]. Its masked-autoencoding perspective is especially relevant because it treats Gaussian attributes as learnable 3D tokens rather than first converting the data into a point cloud, mesh, or voxel grid. GaussianCross similarly develops cross-modal self-supervision over Gaussian representations and uses spatial feature encoding to support downstream representation learning [Yao et al., 2025].

These works establish two points that GLEA adopts. First, native Gaussian parameters contain enough regularity for dataset-level representation learning. Second, masking, grouping, and spatial encoding are useful tools for making variable-length splat sets computationally manageable. GLEA differs in its principal evaluation target and architectural hypothesis. It reconstructs a complete native splat set through a strict fixed-dimensional bottleneck and tests an explicit Bhattacharyya-based attention branch through matched ablation.

The distinction between pretraining and autoencoding is consequential. A discriminative or masked-pretraining objective can produce features useful for classification even when a

full scene cannot be regenerated from one compact vector. A reconstruction bottleneck instead asks whether the code retains enough information to recover the original primitive configuration. Neither objective dominates the other; they measure different properties of the learned representation.

2.5 Multi-Resolution Spatial Features

Multi-resolution hash encoding was introduced as an efficient way to represent spatially varying signals with trainable feature tables queried at several grid scales [Müller et al., 2022]. Coarse levels capture broad structure while fine levels provide localized detail. Hash collisions make the representation compact relative to a dense high-resolution grid, with learned downstream layers resolving ambiguity using context. Later analysis has examined the behavior and interpretation of these encodings in neural fields [Luo, 2025].

GLEA uses the hash code as a spatial feature attached to each normalized primitive center. The hash representation does not replace the Gaussian parameters. It augments position, scale, rotation, color, and opacity before grouping. This is a practical mechanism for supplying multi-scale location information to a set encoder; it is not itself the thesis's geometric contribution. For that reason, a matched hash-encoding ablation remains an important future experiment.

2.6 Information Geometry and Distribution-Aware Comparison

Information geometry studies parametric probability distributions as points on differentiable manifolds equipped with metrics derived from statistical distinguishability. The Fisher information matrix defines the local Fisher–Rao metric under regularity conditions, and natural-gradient methods use this geometry to precondition optimization [Amari, 1998]. Riemannian optimization more broadly provides algorithms for parameters constrained to

nonlinear manifolds [Zhang et al., 2016].

GLEA uses a narrower connection than Riemannian optimization of the entire network. Each spatial primitive is interpreted as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The Bhattacharyya coefficient between two such distributions has a closed form, depends jointly on their means and covariances, and can be written as an inner product after mapping each density to its square root in L^2. It is therefore a positive-semidefinite kernel. The associated Bhattacharyya distance is locally related to Fisher–Rao geometry for nearby distributions, although it is not the exact global Fisher–Rao geodesic distance.

This construction supplies an analytic affinity at the primitive level. It does not force the final 512-dimensional code to be isometric to the statistical manifold. Learned grouping, nonlinear projection, attention, and reconstruction can all distort geometry. The empirical claim in this thesis is correspondingly limited: adding the covariance-aware branch improves reconstruction under the evaluated configuration.

Metric-learning research provides an additional caution. A theoretically appealing distance does not guarantee better downstream performance, and comparisons can be sensitive to sampling, losses, and evaluation choices [Musgrave et al., 2020]. GLEA therefore treats information geometry as a source of testable inductive bias, not as a substitute for ablation.

2.7 Latent Regularization, Wasserstein Autoencoding, and Flow Matching

Autoencoders can reconstruct well while organizing their codes poorly for interpolation or sampling. Variational autoencoders address this by regularizing each approximate posterior toward a prior, but strong regularization can conflict with reconstruction or contribute to posterior collapse in some regimes [Chen et al., 2018]. Wasserstein autoencoders instead match the aggregated posterior to a prior through a distributional penalty while retaining a reconstruction objective [Tolstikhin et al., 2018].

Flow matching learns a time-dependent velocity field that transports samples along probability paths using a regression objective rather than likelihood-based simulation of a stochastic process [Lipman et al., 2023]. Conditional Flow Matching constructs conditional paths between endpoint samples and regresses their target velocities. Related work has explored flow-based latent alignment and weighted flow-matching objectives [Li et al., 2025, Calvo-Ordoñez et al., 2025].

GLEA trains a velocity network jointly with the encoder along linear conditional paths between Gaussian prior samples and encoder outputs. This objective supplies a latent regularization signal, but the generative interpretation depends on wiring. Sampling from the prior is justified only when the transported endpoint is exactly the code consumed by the decoder and when direct samples are evaluated. The thesis therefore reports flow matching as an active regularizer while treating unconditional generation as unestablished.

2.8 Comparative Positioning

Table 2.1 summarizes the conceptual axes that separate GLEA from neighboring problem classes. The entries describe objectives rather than claiming that every paper in a category has exactly the same implementation.

Table 2.1: Conceptual positioning of GLEA relative to neighboring 3D representation-learning objectives. "Native splats" means that Gaussian parameters are the direct learned representation rather than only an intermediate rendering format.

Method class	**Across scenes**	**Native splats**	**Latent-only decode**	**Covariance-aware affinity by design**	**Primary objective**
Scene-specific compression	Often no	Yes	Not generally	Varies	Bitrate–rendering tradeoff
Grid/tensor restructuring	Often yes	Partly	Varies	Not required	Regular substrate for generation
Point/set autoencoding	Yes	No	Often yes	No	Compact shape representation
Native-splat pretraining	Yes	Yes	Not necessarily	Varies	Transferable features
GLEA	**Yes**	**Yes**	**Yes**	**Yes**	**Native-splat reconstruction**

The comparison should be read along these axes rather than reduced to a universal novelty statement. GLEA is not primarily a bitrate codec, a structured-grid generator, or a discriminative pretraining model. It is an across-scene autoencoder for native Gaussian parameters with three defining properties: a strict decoder bottleneck, a geometric attention branch based on distributional overlap, and joint supervision in primitive and rendering domains. Chapter 3 develops the mathematical motivation, Chapter 4 specifies the implemented model, and Chapter 5 tests the resulting claims.

CHAPTER 3: THEORETICAL FOUNDATIONS: INFORMATION GEOMETRY OF GAUSSIAN DISTRIBUTIONS

This chapter provides deeper mathematical foundations for our geometric attention mechanism, connecting the Bhattacharyya coefficient to the canonical structure of statistical manifolds.

3.1 Notation and Level of Analysis

A statistical model is a family $\{p(x \mid \theta) : \theta \in \Theta\}$ indexed by parameters θ. The same distribution may admit several coordinate descriptions, so information geometry distinguishes the distribution itself from a chosen parameterization. In this thesis, one spatial splat is associated with the Gaussian density $p_i(x) = \mathcal{N}(x; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Color and opacity affect appearance and reconstruction, but the geometric kernel is defined on the spatial density determined by $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$.

Three spaces should not be conflated. The primitive manifold contains individual Gaussian distributions. A scene is an unordered finite set of such primitives together with appearance attributes. The learned code is a vector in $\mathbb{R}^{512}$ produced by a nonlinear encoder. The theory in this chapter directly constrains only the first space. Scene-level and code-level geometry must be established by additional definitions and measurements.

3.2 The Statistical Manifold $\mathcal{M}$

The space of 3D multivariate Gaussian distributions forms a 9-dimensional Riemannian manifold $\mathcal{M}$. Each point on $\mathcal{M}$ corresponds to a unique Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The manifold has non-Euclidean geometry: while the mean $\boldsymbol{\mu}$ lives in $\mathbb{R}^3$, the covariance $\boldsymbol{\Sigma}$ lies in the open convex cone of symmetric positive-definite matrices $\mathcal{S}^3_{++}$. This cone is topologically simple

but is not a vector space, and Euclidean operations on matrix entries do not generally follow its natural Riemannian geometry.

DEFINITION: Positive Definite Cone

The space $\mathcal{S}^3_{++} = \{\mathbf{\Sigma} \in \mathbb{R}^{3\times3} : \mathbf{\Sigma} = \mathbf{\Sigma}^\top, \mathbf{x}^\top\mathbf{\Sigma}\mathbf{x} > 0 \ \forall \mathbf{x} \neq 0\}$ is an open convex cone. While convex combinations of positive definite matrices remain positive definite, such linear interpolation does not respect the Riemannian geometry of $\mathcal{S}^3_{++}$ – the Euclidean midpoint of two covariances is not the geodesic midpoint, leading to geometrically distorted intermediate representations.

The 3DGS parameterization $\mathbf{\Sigma} = \mathbf{R} \cdot \text{diag}(\mathbf{s}^2) \cdot \mathbf{R}^\top$ elegantly sidesteps this issue: for any rotation $\mathbf{R} \in SO(3)$ and positive scales $\mathbf{s} \in \mathbb{R}^3_{>0}$, the result is guaranteed positive definite. This is why Gaussian splatting uses quaternion + scale rather than raw covariance entries.

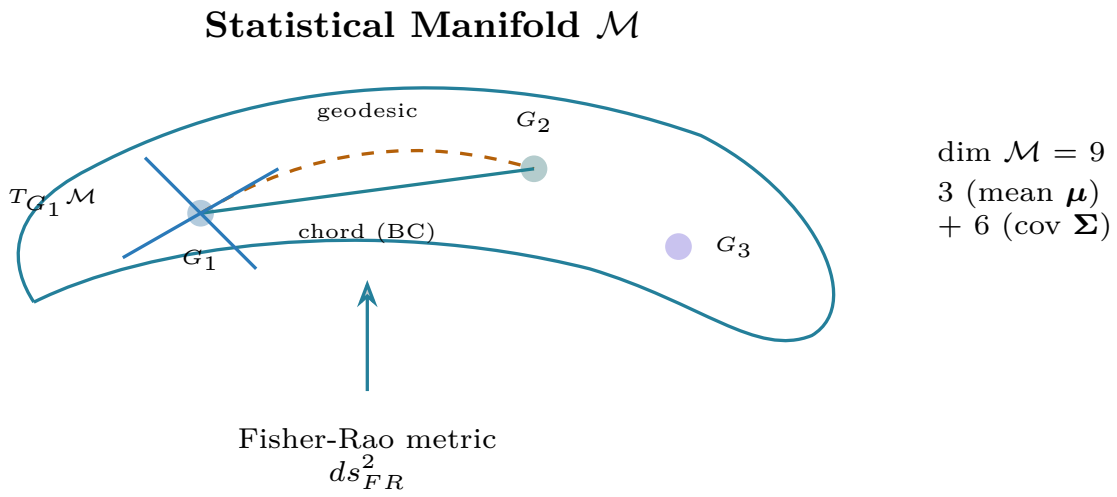

Figure 3.1: Schematic view of the statistical manifold of 3D Gaussians. Each point represents a distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$. The curved path denotes a Fisher-Rao geodesic, while the gray chord illustrates a local approximation rather than an exact global identity.

3.3 The Fisher-Rao Metric and Statistical Invariance

If the space of Gaussian distributions is a manifold, what is the *right* way to measure distances on it? This is not an idle question. Different choices of metric lead to different notions of similarity, different attention patterns, different learned representations. We could, in principle, use any Riemannian metric – the space of possible metrics is infinite. Yet one metric stands apart from all others, distinguished not by convenience but by deep theoretical properties. The Fisher-Rao metric arises from the Fisher information matrix, a concept that predates neural networks by half a century but remains central to modern statistical theory. The Fisher information measures, roughly speaking, how much information a random sample carries about the parameters that generated it. If small changes in the parameter produce large changes in the distribution of samples, the Fisher information is high; the parameter is "easy to estimate." If the distribution is insensitive to the parameter, the Fisher information is low. Formally, for a parametric family of distributions $p(x|\theta)$, the Fisher information matrix is:

$$I(\theta)_{ij} = \mathbb{E}\left[\frac{\partial \log p(x|\theta)}{\partial \theta_i} \cdot \frac{\partial \log p(x|\theta)}{\partial \theta_j}\right] \tag{3.1}$$

The matrix is positive semidefinite in general and positive definite for a regular, identifiable parameterization. Under those regularity conditions it defines an inner product on each tangent space and therefore a Riemannian metric: the Fisher-Rao metric. For multivariate Gaussians parameterized by mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the Fisher-Rao metric takes a particularly elegant form:

$$ds^2_{FR} = d\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1} d\boldsymbol{\mu} + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1} d\boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} d\boldsymbol{\Sigma}) \tag{3.2}$$

The first term is the Mahalanobis distance in the mean – changes in the mean are weighted by the inverse covariance. This captures an important intuition: if a Gaussian is elongated along one axis, moving the mean along that axis is a "small" change (the distribution still

covers roughly the same region of space), while moving perpendicular to it is a "large" change (the distribution shifts to a substantially different region). The second term measures changes in the covariance itself, with similar geometric intuition. But why should we prefer this metric over any other? The answer comes from a remarkable theorem due to Chentsov, which establishes the Fisher-Rao metric as *unique* among all Riemannian metrics satisfying certain natural invariance properties.

GEOMETRIC FOUNDATION

Uniqueness of Fisher-Rao (Chentsov's Theorem) The Fisher-Rao metric is the *unique* Riemannian metric (up to scaling) that satisfies two properties:

1. **Invariance under sufficient statistics**: If we reparameterize our distributions using sufficient statistics – those functions of the data that capture all relevant information about the parameters – the metric remains unchanged. This is a natural requirement: the "distance" between two distributions should not depend on how we choose to describe them.
2. **Monotonicity under Markov mappings**: If we process our data through a noisy channel – any stochastic transformation that discards information – the Fisher information can only decrease. This is the famous data processing inequality: you cannot create information by processing data; you can only destroy it.

These two properties, seemingly innocuous, are enough to determine the metric uniquely. Under these assumptions, the Fisher-Rao metric is canonical up to a constant scale. The conclusion is conditional on the monotonicity and invariance framework of the theorem.

Consider what it would mean if multiple metrics satisfied these invariance properties. We would face an arbitrary choice – which metric to use – and different choices would yield different attention patterns, different learned representations, different notions of similarity between Gaussians. The architecture would contain a free parameter with no principled way to set it. Chentsov's result removes much of this arbitrariness within its assumptions. It motivates Fisher-Rao geometry as the reference metric; the use of a Bhattacharyya-based approximation remains an architectural choice evaluated empirically.

3.4 Bhattacharyya in the α-Divergence Family

The Bhattacharyya coefficient is not an isolated construction – it belongs to a rich family of statistical divergences that includes many familiar quantities. Understanding this family illuminates why Bhattacharyya is particularly well-suited to our purpose. Shun-ichi Amari, the father of information geometry, introduced the α-divergence family as a parameterized class of divergences that interpolate between different notions of statistical distance:

$$D_\alpha(p\|q) = \frac{4}{1-\alpha^2}\left[1 - \int p(x)^{(1+\alpha)/2} q(x)^{(1-\alpha)/2} dx\right] \tag{3.3}$$

The parameter $\alpha \in [-1, 1]$ controls the character of the divergence. At the extremes, we recover the Kullback-Leibler divergences, workhorses of machine learning:

$$\alpha \to +1: \quad D_{KL}(p\|q), \tag{3.4}$$

$$\alpha \to -1: \quad D_{KL}(q\|p). \tag{3.5}$$

At $\alpha = 0$, we obtain a quantity directly related to the Bhattacharyya coefficient – and to the squared Hellinger distance:

$$D_0(p\|q) = 4(1 - \mathrm{BC}(p,q)) = 4H^2(p,q) = 4\left(1 - \int \sqrt{p(x)q(x)}\, dx\right) \tag{3.6}$$

where $H^2(p,q)$ denotes the squared Hellinger distance. The three quantities encode the same overlap information through fixed transformations: $D_0 = 4H^2$ and $H^2 = 1 - \mathrm{BC}$. They should not be treated as numerically identical, but optimizing one induces the same ordering as optimizing the corresponding transformed quantity. We use the symmetric $\alpha = 0$ member because overlap between two Gaussian primitives is naturally modeled as a symmetric relation. Attention does not generally require symmetric scores – standard query-key attention is

asymmetric – so symmetry is a modeling choice for the geometric branch. Other divergences remain possible alternatives.

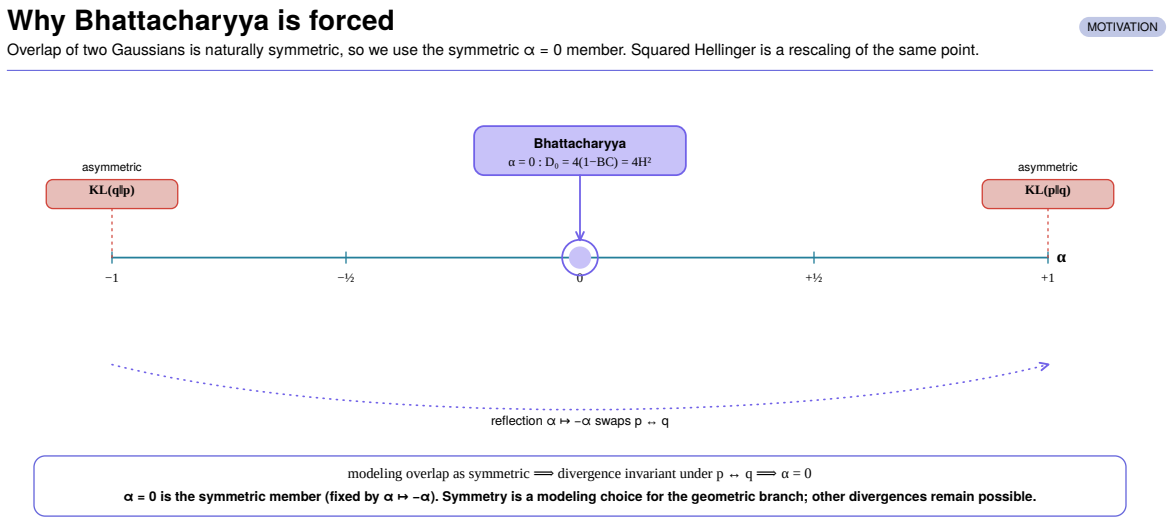

Figure 3.2: Amari's α-divergence family. The Kullback–Leibler divergences sit at the endpoints $\alpha \to \pm 1$ and are asymmetric. The Bhattacharyya quantity sits at $\alpha = 0$, the fixed point of the reflection $\alpha \mapsto -\alpha$ that exchanges p and q, and the squared Hellinger distance is a fixed rescaling of the same point ($D_0 = 4H^2$). Because primitive overlap is naturally symmetric, we use this member for the geometric attention branch; the choice is a modeling decision rather than a necessity.

3.5 Closed-Form Gaussian Overlap

For two d-dimensional Gaussian densities $G_1 = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $G_2 = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, define

$$\overline{\boldsymbol{\Sigma}} = \frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}, \qquad \boldsymbol{\delta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2. \tag{3.7}$$

The Bhattacharyya coefficient and distance are

$$\mathrm{BC}(G_1, G_2) = \frac{|\boldsymbol{\Sigma}_1|^{1/4}|\boldsymbol{\Sigma}_2|^{1/4}}{|\overline{\boldsymbol{\Sigma}}|^{1/2}} \exp\left[-\frac{1}{8}\boldsymbol{\delta}^\top \overline{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\delta}\right], \tag{3.8}$$

$$D_{\mathrm{B}}(G_1, G_2) = -\log \mathrm{BC}(G_1, G_2) \tag{3.9}$$

$$= \frac{1}{8}\boldsymbol{\delta}^\top \overline{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\delta} + \frac{1}{2}\log \frac{|\overline{\boldsymbol{\Sigma}}|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}. \tag{3.10}$$

The first term measures mean separation relative to average uncertainty. The second term penalizes covariance mismatch even when the means coincide. For the motivating covariances $\mathrm{diag}(1, 1, 0.01)$ and $\mathrm{diag}(0.01, 0.01, 1)$ with equal means, direct substitution gives $\mathrm{BC} \approx 0.0881$. Center distance is zero, whereas distributional overlap is limited.

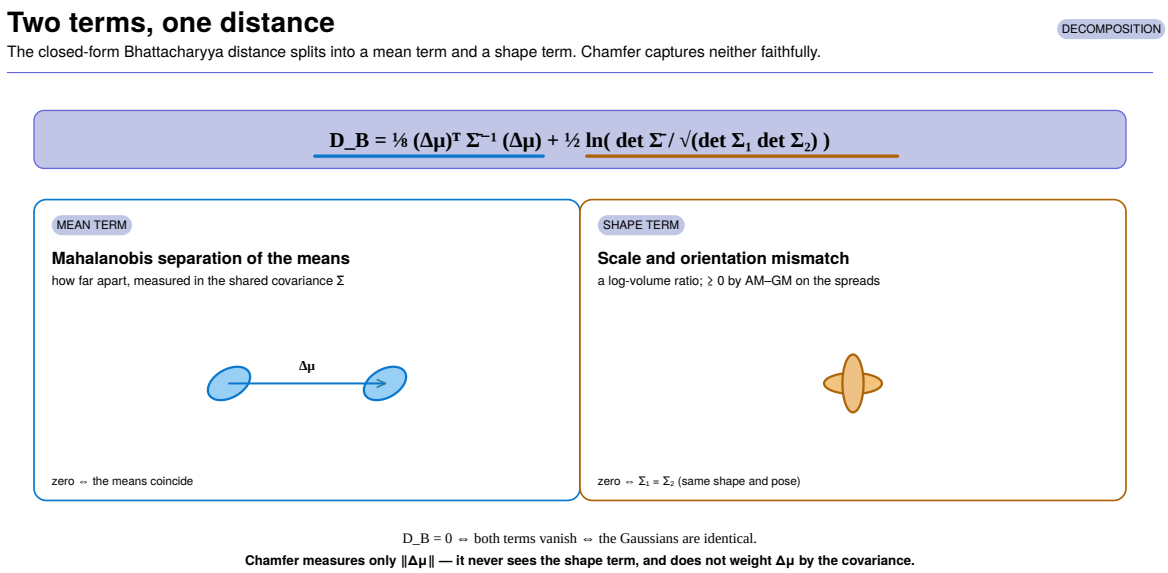

Figure 3.3: Geometric reading of the closed form. The mean term is a Mahalanobis separation of the centers in the shared covariance $\overline{\boldsymbol{\Sigma}}$; the shape term is a log-volume ratio that is nonnegative by the arithmetic–geometric-mean inequality and vanishes only when $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. A positional (Chamfer) comparison reads only $\|\boldsymbol{\delta}\|$: it never sees the shape term and does not weight the separation by covariance.

Figure 3.4 makes the same point through three cases. When two primitives share a center but differ in shape or orientation, a center-only comparison reports near-zero difference while

the Bhattacharyya distance remains large; the two measures agree only when the primitives differ in position. This is the specific blindness that a covariance-aware affinity removes.

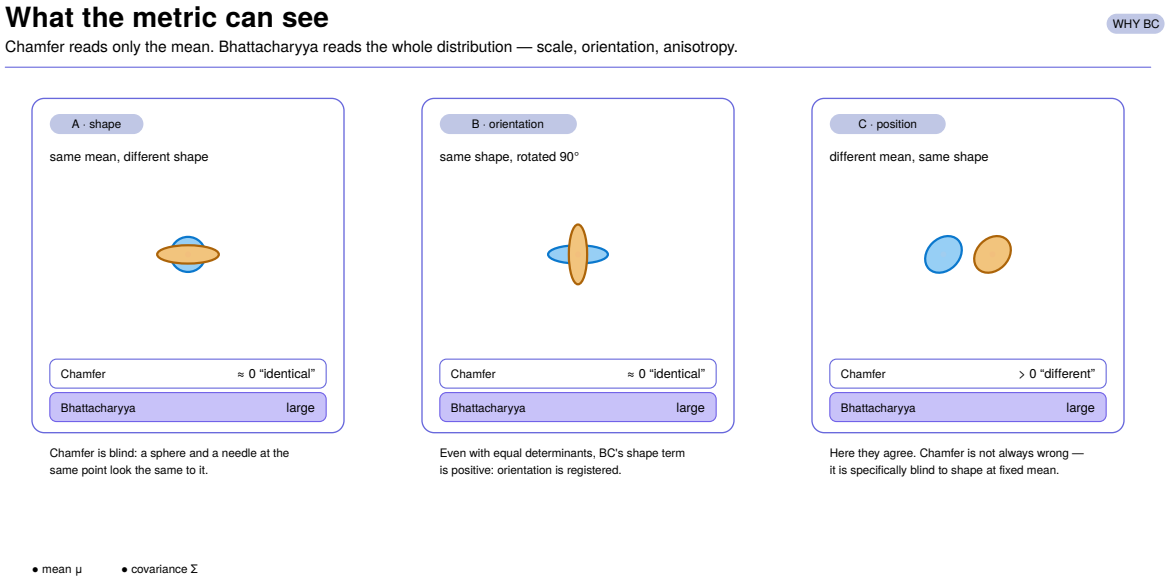

Figure 3.4: Three configurations of a primitive pair. **(A)** same center, different shape; **(B)** same shape, rotated 90° (case (B) is the $\mathrm{diag}(1, 1, 0.01)$ versus $\mathrm{diag}(0.01, 0.01, 1)$ example, $\mathrm{BC} \approx 0.0881$); **(C)** different center, same shape. A center-distance comparison is blind in (A) and (B) and only registers a difference in (C), whereas the Bhattacharyya distance responds in all three. Ellipses are shown in 2D for clarity; the affinity is computed on the full 3×3 covariance.

The square-root map $\Phi : p \mapsto \sqrt{p}$ places normalized densities on the unit sphere of L^2. Under this map,

$$\mathrm{BC}(p, q) = \int \sqrt{p(x)q(x)}\, dx = \langle \Phi(p), \Phi(q) \rangle_{L^2}. \tag{3.11}$$

Therefore, for any coefficients a_i and densities p_i,

$$\sum_{i,j} a_i a_j \,\mathrm{BC}(p_i, p_j) = \left\| \sum_i a_i \Phi(p_i) \right\|_{L^2}^2 \geq 0. \tag{3.12}$$

This proves positive semidefiniteness of the coefficient as a kernel. Positive semidefiniteness is useful for interpreting the score as an inner product, but softmax attention itself does not require a positive-semidefinite score matrix.

3.6 Geodesics on the Statistical Manifold

The geodesic (shortest path) between two Gaussians is *not* a straight line in parameter space. For the mean component with fixed covariance, geodesics are straight:

$$\boldsymbol{\mu}(t) = (1-t)\boldsymbol{\mu}_1 + t\boldsymbol{\mu}_2 \tag{3.13}$$

For the covariance component with fixed mean, geodesics follow the matrix geometric mean:

$$\boldsymbol{\Sigma}(t) = \boldsymbol{\Sigma}_1^{1/2}\left(\boldsymbol{\Sigma}_1^{-1/2}\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1/2}\right)^t\boldsymbol{\Sigma}_1^{1/2} \tag{3.14}$$

This ensures the interpolated covariance stays positive definite throughout the path. Bhattacharyya distance is computationally tractable and agrees locally with the Fisher-Rao metric to second order for nearby distributions. It is not the exact global Fisher-Rao geodesic distance.

3.7 Numerical Verification of the Closed-Form Overlap

Before the coefficient is placed inside a learned network, it is worth confirming that the closed-form Gaussian expression of Section 3.5 behaves exactly as the derivation predicts, and that its covariance-aware terms carry information a center-only comparison cannot. We evaluate the Bhattacharyya distance D_B in double precision on a battery of analytically solvable Gaussian pairs, using the same kernel primitive that the encoder attention and the reconstruction cost later share (Chapter 4).

Two counterfactuals are decisive. First, fix an axis-aligned covariance with major-to-minor scale ratio $\rho = \sigma_{\max}/\sigma_{\min}$ and displace one center by a fixed Euclidean magnitude, either along the broad axis or along a thin axis. The center distance is identical in the two cases,

but the mean term of D_B is a Mahalanobis form, so the closed form predicts that a thin-axis displacement costs a factor ρ^2 more than a broad-axis displacement of the same length. Across $\rho \in \{1, 2, 4, 8, 16\}$ the measured cost ratio reproduces the predicted ρ^2—exactly $1, 4, 16, 64, 256$—to a maximum relative error of 1.0×10^{-12} (Figure 3.5a). Second, hold two centers coincident and rotate one anisotropic covariance in place. The center distance is identically zero at every angle, whereas D_B rises monotonically from 0 to 1.40 between 0° and 90° (Figure 3.5b). A center-only affinity is, by construction, blind to this entire family of distinctions.

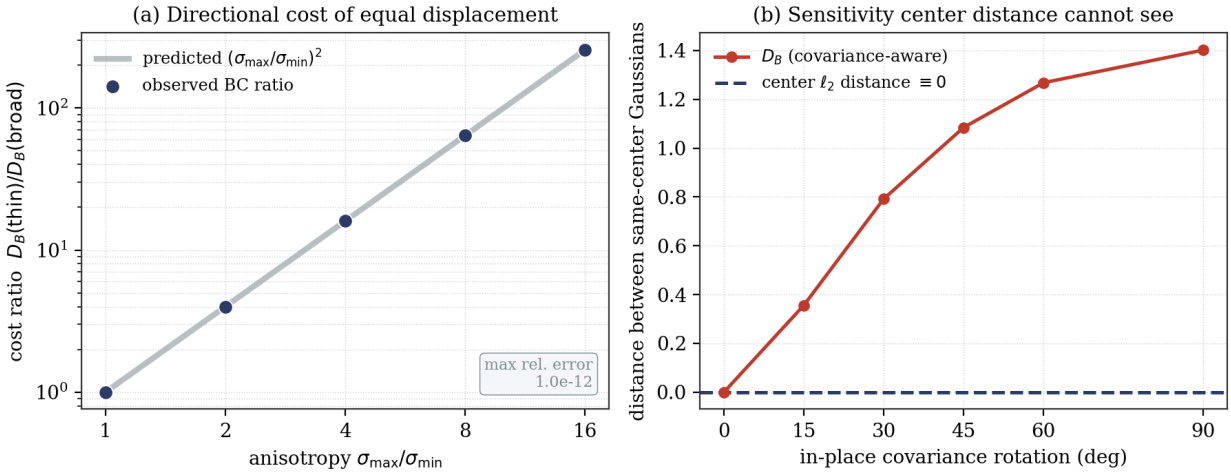

Figure 3.5: Analytic verification of the Bhattacharyya kernel on controlled Gaussian pairs, computed in double precision from the released experiment harness. (a) For a covariance with anisotropy $\rho = \sigma_{\max}/\sigma_{\min}$, an equal-magnitude center displacement along the thin axis incurs ρ^2 times the cost of the same displacement along the broad axis; the observed ratio matches the closed-form prediction to a maximum relative error of 1.0×10^{-12}. (b) For two Gaussians with coincident centers, rotating one covariance in place leaves the center distance identically zero while D_B grows monotonically—a distinction a center-only score cannot represent. The battery additionally confirms quaternion sign invariance ($D_B(\mathbf{q}, -\mathbf{q}) = 0$ to machine precision) and finiteness down to a minimum scale of 10^{-5}.

The same battery confirms the invariances the kernel must respect. The distance between a covariance built from a unit quaternion $\mathbf{q}$ and the same covariance built from $-\mathbf{q}$ is zero to machine precision, matching the double cover of $SO(3)$, and every reported quantity remains finite as the smallest scale is driven to 10^{-5} under the numerical hardening described in Chapter 4. These checks are analytic and deterministic: they do not depend on the

dataset, the trained weights, or the renderer, and they are reproducible from the released harness. They establish only that the primitive-level quantity is implemented correctly and is genuinely covariance-sensitive. Whether that sensitivity improves a *learned* representation is the separate empirical question of Chapter 5.

3.8 What the Primitive-Level Geometry Does and Does Not Imply

The preceding sections motivate a Bhattacharyya-based comparison at the level of individual Gaussian primitives. That choice supplies a covariance-aware affinity before learning, but it does not determine the metric of the final 512-dimensional code. Grouping, attention, nonlinear projection, pooling, and reconstruction training can all distort pairwise relationships.

Consequently, claims about the learned latent geometry require direct measurements. Useful tests include correlation between scene-level geometric distances and latent distances, neighborhood preservation, local distortion, and decoder sensitivity along controlled paths. Participation ratio can estimate effective linear dimension in a sampled code set, while interpolation can probe decoder behavior along selected line segments. Neither analysis alone establishes isometry, topology, curvature, or the intrinsic dimension of the space of all valid 3D scenes.

The appropriate inference is therefore layered. Fisher-Rao theory motivates the primitive-level reference geometry; the Bhattacharyya coefficient provides a tractable local surrogate and kernel; the architecture uses that quantity as one branch of a learned attention mechanism; and the experiments test whether the resulting model improves reconstruction. Any stronger claim that the latent code preserves information geometry remains a hypothesis for future measurement.

CHAPTER 4: METHODS: ARCHITECTURAL DESIGN

4.1 Problem Formulation

We establish the mathematical framework precisely. Let $\mathcal{G} = \{G_i\}_{i=1}^{N}$ denote a 3D Gaussian splat representation where each primitive comprises:

$$G_i = (\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{q}_i, \mathbf{c}_i, \alpha_i) \tag{4.1}$$

with position $\boldsymbol{\mu}_i \in \mathbb{R}^3$, scale $\mathbf{s}_i \in \mathbb{R}^3_{>0}$, rotation quaternion $\mathbf{q}_i \in \mathbb{H}$, color $\mathbf{c}_i \in \mathbb{R}^3$, and opacity $\alpha_i \in [0, 1]$ – totaling 14 parameters per Gaussian.

DEFINITION: Autoencoder Objective

We seek an encoder $E_\phi : \mathcal{G} \to \mathbf{z}$ mapping variable-cardinality splat collections ($N \in [500, 3200]$) to a fixed-dimensional latent code $\mathbf{z} \in \mathbb{R}^{512}$, paired with a decoder $D_\theta : \mathbf{z} \to \hat{\mathcal{G}}$ that reconstructs the input distribution.

The key architectural challenges distinguishing this from standard point cloud autoencoding:

1. **Permutation Invariance**: The encoding must be invariant to arbitrary splat ordering while preserving spatial relationships and geometric structure. We achieve this via attention-based aggregation.
2. **Variable Cardinality**: Scenes contain roughly 500–3,200 Gaussians depending on complexity. The encoder uses dynamic masks so padded input slots are not treated as content. In the reported configuration, count prediction is disabled and the decoder emits a fixed maximum of 3,200 primitive slots.
3. **Geometric Fidelity**: Reconstructions must preserve volumetric structure (not merely point positions). The covariance matrix $\boldsymbol{\Sigma} = \mathbf{R}\text{diag}(\mathbf{s}^2)\mathbf{R}^\top$ determines rendering appearance; we must preserve both orientation and anisotropic scale.
4. **Latent Regularity**: The manifold should support semantically meaningful interpolation and sampling. We use Conditional Flow Matching as the latent regularizer,

learning the transport path from prior to posterior jointly with the encoder rather than penalizing against a fixed distributional target.

Following the token/type framework established in Chapter 1, we explicitly design for latent codes that are holistic instance encodings (not semantic type embeddings). Each latent vector $\mathbf{z}_i$ represents one complete Gaussian-splat scene. The architecture is intended to make the code self-contained and regular enough for interpolation and sampling, but proximity, interpolation quality, and prior compatibility are empirical properties rather than consequences of the definition. The diagnostics in Chapter 5 probe those properties separately.

Figure 4.1 gives the end-to-end view before we develop each module in turn.

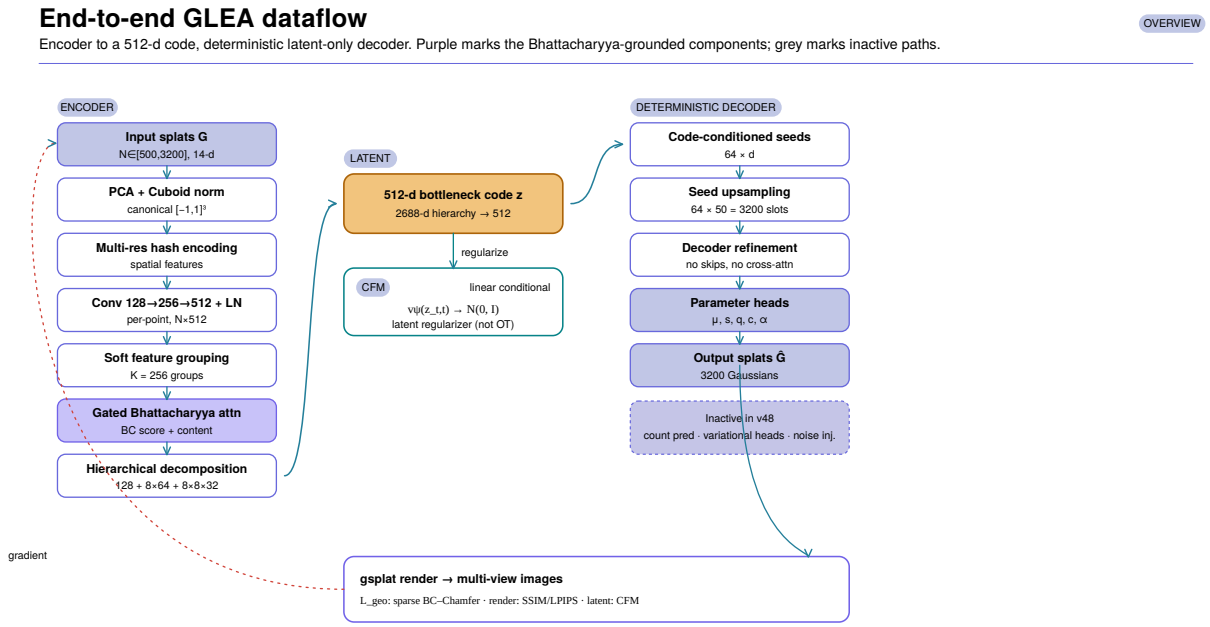

Figure 4.1: End-to-end GLEA dataflow. A variable-cardinality splat set is cuboid-normalized, hash-encoded, and softly grouped, then passed through gated geometric (Bhattacharyya) attention; a three-level hierarchy ($128 + 8{\times}64 + 8{\times}8{\times}32 = 2688$ dimensions) is compressed to the latent code $\mathbf{z} \in \mathbb{R}^{512}$. The deterministic decoder reconstructs the full primitive set from the code alone—64 code-conditioned seeds expanded by a factor of 50 into 3,200 slots, with no encoder skip connections—while Conditional Flow Matching shapes the latent geometry. Count prediction, variational heads, and noise injection are inactive in the reported configuration. The same Bhattacharyya comparison appears in the attention kernel and in the reconstruction ground cost.

4.2 Encoder Design

4.2.1 PCA Pose Alignment: Reducing Global Orientation Variation

The reported v48 configuration applies principal-component analysis (PCA) to the valid primitive centers before cuboid normalization. Let

$$\overline{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\mu}_i, \qquad \mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})^\top. \tag{4.2}$$

If $\mathbf{U}$ contains the eigenvectors of $\mathbf{C}$ ordered by decreasing eigenvalue, the aligned centers are

$$\boldsymbol{\mu}_i^{\mathrm{PCA}} = \mathbf{U}^\top(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}}), \tag{4.3}$$

and each spatial covariance is transformed by the same frame change,

$$\boldsymbol{\Sigma}_i^{\mathrm{PCA}} = \mathbf{U}^\top\boldsymbol{\Sigma}_i\mathbf{U}. \tag{4.4}$$

The implementation carries the equivalent rotation into the quaternion representation used by the network. This step removes a substantial portion of global pose variation so that the code can devote capacity to object structure rather than arbitrary orientation.

PCA alignment is an inductive bias, not a mathematically necessary preprocessing step. Eigenvectors have sign ambiguity, and nearly repeated eigenvalues make the frame sensitive to small perturbations. A deterministic convention must therefore be applied consistently by the data pipeline. Symmetric objects can remain ambiguous even with such a convention. The no-PCA ablation measures the practical contribution of the complete implemented alignment path rather than claiming that PCA defines a universal canonical pose.

4.2.2 Cuboid Normalization: Scale Invariance via Canonical Framing

Every architectural choice encodes an assumption about the structure of the data it processes. The first assumption we encode is that object identity is scale-invariant: a chair spanning 1 meter should produce the same latent code as an identical chair scaled to 10 meters. Without this invariance, the latent space fragments into disconnected regions based on arbitrary absolute scale. Following GaussianCross [Yao et al., 2025], we normalize inputs to a canonical $[-1, 1]^3$ coordinate frame before any learned processing. Given axis-aligned bounding box (AABB) corners $\mathbf{b}_{\min}, \mathbf{b}_{\max}$ computed from valid (unmasked) Gaussian positions:

$$\mathbf{c} = \frac{\mathbf{b}_{\min} + \mathbf{b}_{\max}}{2} \quad \text{(AABB center)} \tag{4.5}$$

$$d = \max(\mathbf{b}_{\max} - \mathbf{b}_{\min}) + \epsilon \quad \text{(max extent)} \tag{4.6}$$

$$\tilde{\boldsymbol{\mu}}_i = \frac{2(\boldsymbol{\mu}_i - \mathbf{c})}{d} \quad \text{(normalized position)} \tag{4.7}$$

$$\tilde{\mathbf{s}}_i = \frac{2\mathbf{s}_i}{d} \quad \text{(normalized scale)} \tag{4.8}$$

Positions map to $[-1, 1]^3$ and scales shrink proportionally. Rotations and colors pass through unchanged (they are already scale-invariant). The normalization parameters $(\mathbf{c}, d)$ are stored for denormalization during decoding, enabling reconstruction in the original coordinate frame. The ϵ term (typically 10^{-6}) prevents division by zero for degenerate cases and provides numerical stability for nearly-degenerate configurations where the AABB is very thin along one axis.

4.2.3 Multi-Resolution Hash Encoding: Spatial Feature Embedding

The choice of spatial feature representation encodes an implicit assumption about the structure of spatial information – whether it is smooth, local, multi-scale, or some combination. To

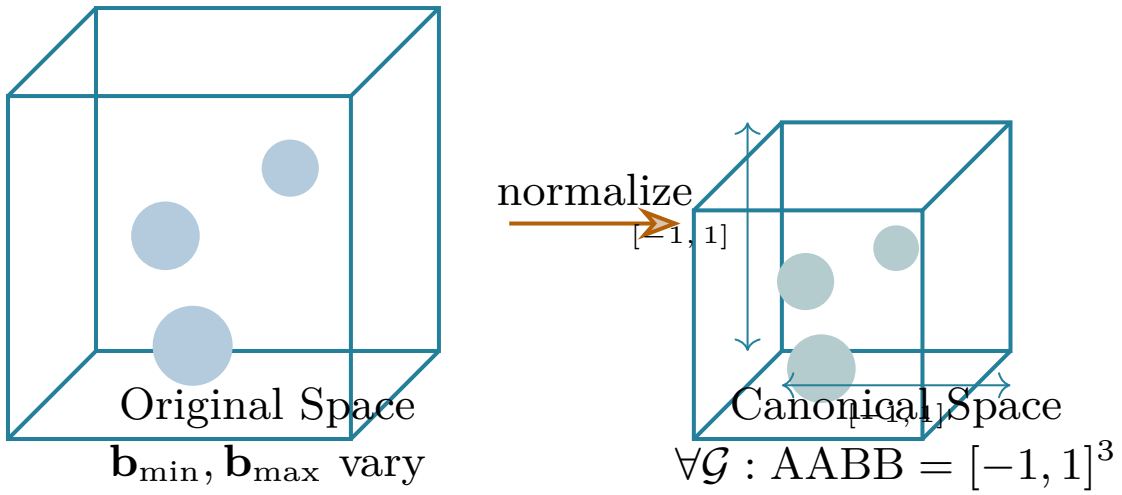

Figure 4.2: Cuboid normalization maps arbitrary AABBs to canonical $[-1, 1]^3$ cube, preserving relative geometric relationships while achieving scale invariance. Gaussian scales shrink proportionally.

capture fine-grained spatial structure without prohibitive network depth, we employ Instant-NGP-style [Müller et al., 2022] multi-resolution hash encoding. This maps each normalized position $\tilde{\boldsymbol{\mu}}_i$ to a learned feature vector by querying trainable hash tables at $L = 16$ resolution levels. For resolution level ℓ with grid size $N_\ell = \left\lfloor N_{\min} b^\ell \right\rfloor$, where $N_{\min} = 16$, $N_{\max} = 2048$, $b = \exp((\log N_{\max} - \log N_{\min})/(L-1))$, and $L = 16$:

$$\mathbf{h}_i^{(\ell)} = \text{TrilinearInterp}\left(\mathcal{H}^{(\ell)}, \tilde{\boldsymbol{\mu}}_i \cdot N_\ell\right) \in \mathbb{R}^F \tag{4.9}$$

Each hash table $\mathcal{H}^{(\ell)}$ maps $T = 2^{19} = 524288$ entries to $F = 2$-dimensional features using prime-based spatial hashing:

$$h(x, y, z) = (x \cdot 1 \oplus y \cdot 2654435761 \oplus z \cdot 805459861) \mod T \tag{4.10}$$

The concatenated multi-resolution feature:

$$\mathbf{h}_i = [\mathbf{h}_i^{(0)} \| \mathbf{h}_i^{(1)} \| \cdots \| \mathbf{h}_i^{(L-1)}] \in \mathbb{R}^{32} \tag{4.11}$$

Total hash parameters: $L \times T \times F = 16 \times 524288 \times 2 \approx 16.8\text{M}$ parameters. These concatenate with raw Gaussian parameters (14D) to form 46-dimensional inputs:

$$\mathbf{F}_0 = [\tilde{\boldsymbol{\mu}}_i \| \log \tilde{\mathbf{s}}_i \| \mathbf{q}_i \| \mathbf{c}_i \| \alpha_i \| \mathbf{h}_i] \in \mathbb{R}^{N \times 46} \tag{4.12}$$

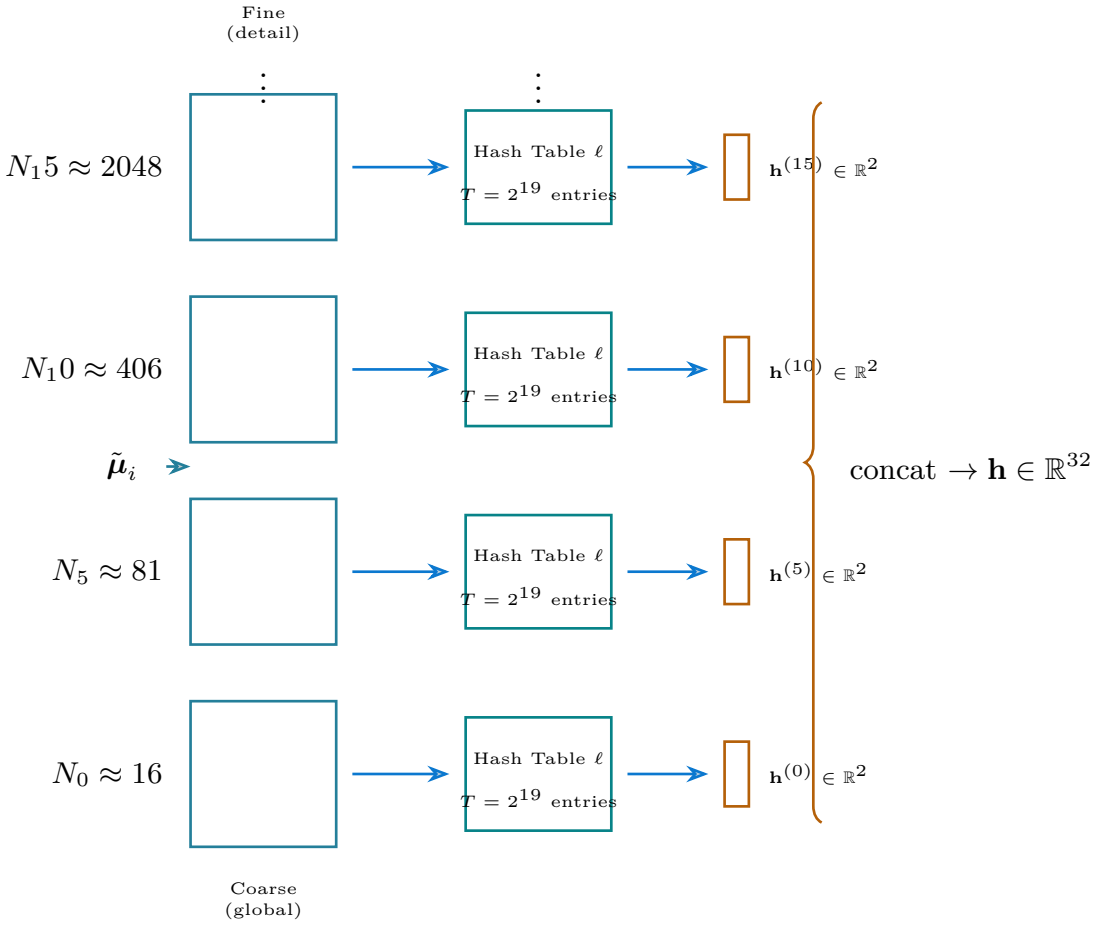

Figure 4.3: Multi-resolution hash encoding. Each position queries 16 hash tables at exponentially increasing resolutions ($16 \rightarrow 2048$); representative levels are shown. Coarse levels capture global structure; fine levels capture local detail. Features concatenate to form 32-dimensional spatial embeddings.

The hash encoding mechanism deserves careful explanation, for it solves a problem that has plagued neural network approaches to spatial data since their inception. Consider the task of learning a function from 3D coordinates to some output – perhaps a color, a density, or a feature vector. The naive approach is to pass the coordinates (x, y, z) through a

multilayer perceptron. But MLPs are smooth functions of their inputs; they cannot represent sharp transitions or fine-scale spatial variation without becoming impractically deep. A point at $(0.5, 0.5, 0.5)$ and a point at $(0.5001, 0.5001, 0.5001)$ will receive nearly identical feature representations, because the smooth MLP cannot change rapidly over such small distances. This limitation is sometimes called the "spectral bias" of neural networks: they learn low-frequency functions easily and high-frequency functions slowly, if at all. For spatial problems where fine detail matters – and in 3D graphics, fine detail always matters – this bias is crippling. Hash encoding circumvents this limitation by providing the network with explicit spatial features at multiple scales. Rather than asking the network to learn spatial relationships from raw coordinates, we precompute a set of learnable features arranged on grids of varying resolution, and look up these features based on the input coordinates. The key insight, due to Müller et al. [Müller et al., 2022], is that dense grids at all resolutions – which would require prohibitive memory – can be replaced by compact hash tables with no explicit collision handling. At coarse resolutions, the mapping from grid vertices to table entries is injective. At fine resolutions, multiple grid points alias to the same entry, and collisions are resolved implicitly through gradient-based optimization: the gradients of high-importance samples (e.g., visible surfaces) dominate the collision average, while low-importance samples (e.g., empty space) contribute negligibly. The MLP then disambiguates residual aliasing through the combination of multiple resolution levels, where simultaneous collision at all levels for a given point pair is statistically unlikely. Müller et al.'s framing is strictly operational: collisions are a tolerable cost that the system works *despite*, not *because of*. Recent work by Luo [Luo, 2025] advances understanding of the grid component through a "domain manipulation" perspective – showing how feature grids increase MLP expressivity by warping and flipping subdomains of the input space, multiplying the number of locally linear segments the downstream network can represent. However, Luo explicitly scopes out the collision question, noting that "the relation between the hashing solution and the neural field's expressivity is yet to be explored" and that "it is unclear how the MLP could serve

both as the underlying signal provider and a hash collision resolver." The grid's contribution to expressivity is now well-characterized; the collision mechanism's contribution to the *learned representation* remains an open problem.

ARCHITECTURAL INSIGHT

A possible interpretation is that fine-resolution hash collisions act as a form of structured aliasing and may regularize which local features persist. The present experiments do not isolate collision rate or compare matched collision-free encodings, so this mechanism should be treated as a hypothesis rather than an established property of GLEA. A direct test would vary table size at fixed model capacity and measure reconstruction, feature allocation across levels, and sensitivity to collision patterns.

The multi-resolution structure is essential. Coarse levels, with low-resolution grids, capture global spatial structure: which region of the scene does this point belong to? Fine levels, with high-resolution grids, capture local detail: what are the precise geometric relationships in this neighborhood? By concatenating features from all levels, we provide the network with a complete spatial context spanning from global to local. One subtle but important detail: we encode $\log \mathbf{s}$ rather than raw scales. The scale values in Gaussian splats span a wide range – from 0.001 for fine detail to 10.0 for large diffuse regions – and this dynamic range creates numerical difficulties for gradient-based optimization. The logarithmic encoding compresses this range, improving numerical stability and ensuring that gradients flow effectively regardless of the current scale magnitude. Hash encoding therefore supplies explicit multi-scale locality features that complement the MLP's learned representation. Its specific regularization effect in this architecture remains to be tested.

4.2.4 Feature Grouping: Computational Tractability

We now confront a computational obstacle that is fundamental to attention-based architectures: the quadratic scaling of self-attention with sequence length. Standard attention computes pairwise interactions between all elements of a sequence; for N elements, this requires $O(N^2)$ operations. When N is small – a few hundred tokens in a language model – this is manageable. But a Gaussian splat scene may contain $N \sim 2000$ primitives, yielding $N^2 = 4{,}000{,}000$ pairwise interactions per attention layer. This quickly becomes prohibitive, especially when training on consumer hardware. The solution, following ShapeSplat [Ma et al., 2025], is to reduce the effective sequence length through learned grouping. Rather than computing attention over all N points, we first aggregate points into $K = 256$ groups, then compute attention over these groups. The quadratic cost drops from $O(N^2) \approx O(4 \times 10^6)$ to $O(K^2) = O(65{,}536)$ – a 60× reduction. But how do we aggregate points into groups without losing information? The key is to use soft, learned assignments rather than hard clustering. We learn a grouping matrix that assigns each point to each group with some probability, then compute group features as weighted averages. Hash-augmented features first pass through 1D convolutions that progressively expand the channel dimension, building up representational capacity:

$$\mathbf{F} = \text{Conv}_{512}(\text{Conv}_{256}(\text{Conv}_{128}(\mathbf{F}_0))) \in \mathbb{R}^{N \times 512} \tag{4.13}$$

Each convolutional block includes LayerNorm (for training stability), ReLU activation (for nonlinearity), and dropout with probability $p = 0.1$ (for regularization). The resulting 512-dimensional features encode rich per-point information combining the original Gaussian parameters with multi-resolution spatial context. Soft grouping then projects these features into assignment logits and computes group features as weighted averages:

$$\mathbf{A} = \text{softmax}(\mathbf{W}_g \mathbf{F}), \quad \mathbf{F}_g^{(k)} = \frac{\sum_i A_{ik} \mathbf{F}_i}{\sum_i A_{ik}} \tag{4.14}$$

With $K = 256$ groups, subsequent transformer attention operates in $O(K^2) = O(65536)$ rather than $O(N^2) \approx O(4 \times 10^6)$ – a 60× reduction. The soft assignments $\mathbf{A}$ are learned end-to-end. Semantic part correspondence is not directly supervised or evaluated in the current experiments.

4.2.5 Bhattacharyya Attention: Information-Geometric Kernel

We now arrive at the core geometric innovation – the component where the statistical manifold structure of Gaussian distributions is encoded directly into the architecture's computational mechanism. Standard transformer attention computes weights via softmax-normalized dot products of learned query and key projections:

$$\text{Attention}_{\text{standard}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{4.15}$$

This mechanism is geometrically agnostic. The similarity $\mathbf{q}_i^\top \mathbf{k}_j$ reflects learned feature correlations with no intrinsic connection to the statistical structure of Gaussian distributions. The question is whether adding an analytically defined distribution-overlap score improves the learned content-based comparison.

4.2.5.1 The Statistical Manifold of Gaussians

Each 3D Gaussian splat $G = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is not merely a point with attributes – it is a probability distribution on $\mathbb{R}^3$. The space of all multivariate Gaussians forms a Riemannian manifold $\mathcal{M}$ with dimension 9 (3 mean + 6 covariance parameters). The Fisher information matrix defines the natural Riemannian metric on $\mathcal{M}$:

$$ds^2_{FR} = d\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1} d\boldsymbol{\mu} + \frac{1}{2}\text{tr}(\boldsymbol{\Sigma}^{-1} d\boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} d\boldsymbol{\Sigma}) \tag{4.16}$$

The first term is Mahalanobis distance in the mean: changes along the "thin" axis of an ellipsoid represent larger statistical changes than movements along the "fat" axis. The second term captures shape mismatch.

4.2.5.2 Bhattacharyya Coefficient and Local Fisher-Rao Geometry

GEOMETRIC FOUNDATION

Bhattacharyya Coefficient for Gaussians For two multivariate Gaussians $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, the Bhattacharyya coefficient has closed form:

$$\mathrm{BC}(G_1, G_2) = \frac{|\boldsymbol{\Sigma}_1|^{1/4}|\boldsymbol{\Sigma}_2|^{1/4}}{|\bar{\boldsymbol{\Sigma}}|^{1/2}} \exp\left(-\frac{1}{8}D_\mu\right) \tag{4.17}$$

where $\bar{\boldsymbol{\Sigma}} = \frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2}$ and $D_\mu = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \bar{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$.

The formula rewards careful examination. The Bhattacharyya distance $D_B = -\log \mathrm{BC}$ decomposes into two interpretable terms:

$$D_B = \underbrace{\frac{1}{8}D_\mu}_{\text{mean separation}} + \underbrace{\frac{1}{2}\log \frac{|\bar{\boldsymbol{\Sigma}}|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}}_{\text{shape mismatch}} \tag{4.18}$$

The first term measures how far apart the distributions are centered, using a Mahalanobis-like metric weighted by the average covariance. The second term measures how different the distributions are in shape – their relative "roundness" and orientation. Two distributions can be far apart in Bhattacharyya distance either by being centered in different locations, or by having different shapes, or both. For infinitesimally separated Gaussians, $D_B \approx \frac{1}{8}ds^2_{FR}$. Thus, the Bhattacharyya distance agrees locally with the Fisher-Rao metric to second order. This is a local statement; D_B is not the exact global Fisher-Rao geodesic distance.

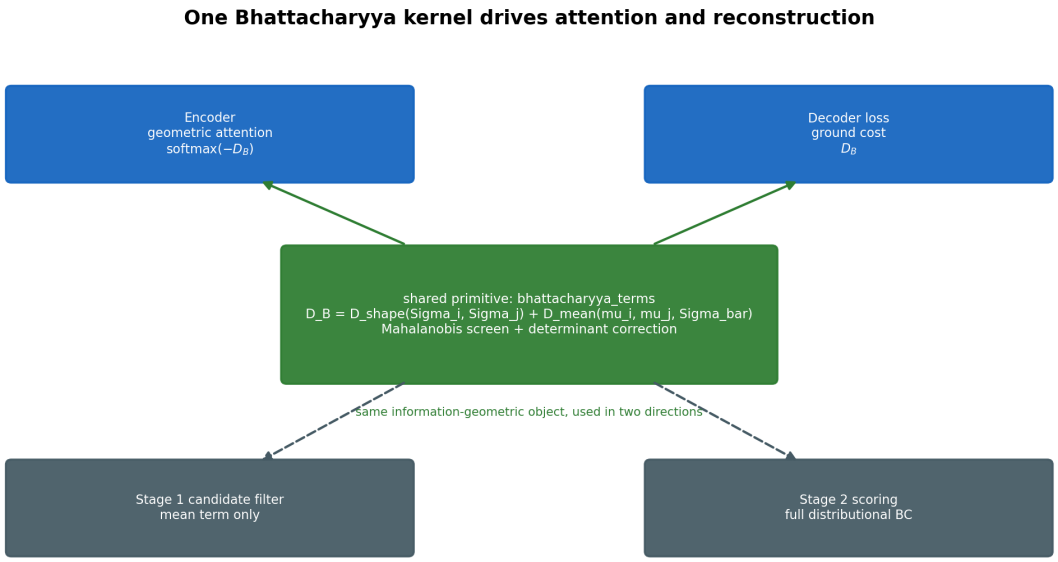

Figure 4.4: Reuse of a Bhattacharyya-derived primitive comparison. The geometric attention branch uses a Bhattacharyya score, while the sparse reconstruction objective uses the corresponding distance after candidate screening. The local relationship to Fisher-Rao geometry motivates both uses without implying an exact global geodesic or an active optimal-transport objective.

4.2.5.3 Kernel Validity

For the Bhattacharyya coefficient to serve as an attention kernel, it must satisfy a mathematical property called positive semi-definiteness (PSD). A kernel $k(x, y)$ is PSD if, for any finite set of inputs $\{x_1, \ldots, x_n\}$, the Gram matrix $K_{ij} = k(x_i, x_j)$ has only non-negative eigenvalues. Positive semidefiniteness is not required by ordinary softmax attention. Its value here is that it gives the Bhattacharyya coefficient a valid kernel interpretation and supports feature-map reasoning. The Bhattacharyya coefficient is indeed PSD, and the proof is elegant. Consider the L^2 space of functions, where the inner product is $\langle f, g \rangle = \int f(x)g(x)dx$. Define a feature map that sends each probability density p to its square root: $\phi(p) = \sqrt{p}$. Then:

$$\mathrm{BC}(p, q) = \int \sqrt{p(x)q(x)}\, dx = \int \sqrt{p(x)} \cdot \sqrt{q(x)}\, dx = \langle \phi(p), \phi(q) \rangle_{L^2} \tag{4.19}$$

The Bhattacharyya coefficient is an inner product in L^2 space. By reproducing kernel Hilbert

space (RKHS) theory, any inner product defines a PSD kernel. The BC is not merely inspired by inner products or analogous to them – it *is* one, in the space of square-root probability densities. The square-root density embedding gives the coefficient a precise kernel interpretation. In GLEA, this kernel supplies an analytically defined overlap score rather than asking the content branch to infer that score entirely from data. The complete layer is still learned: a gate mixes the geometric score with asymmetric dot-product content attention before softmax normalization.

4.2.5.4 What Geometric Attention "Sees"

The following figure illustrates the distinctive behavior of Bhattacharyya-based attention. Unlike Euclidean distance, which considers only centers, or feature-based attention, which compares learned embeddings, Bhattacharyya attention directly measures volumetric overlap between the actual distributions.

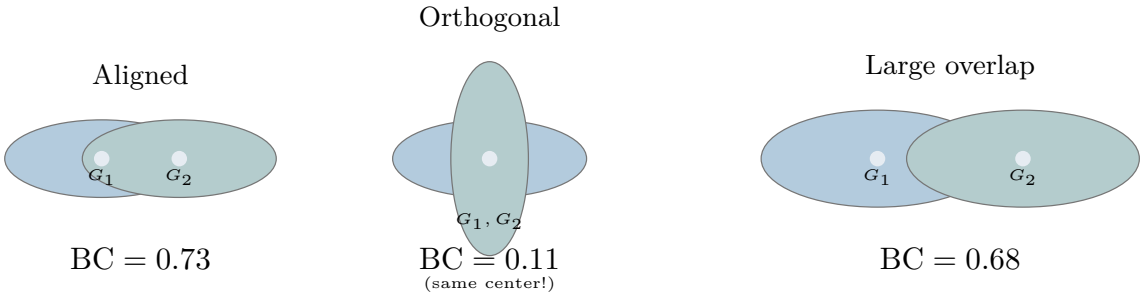

Figure 4.5: Bhattacharyya coefficient responds to covariance as well as center displacement. Left: moderate overlap with a small center separation. Middle: lower overlap despite identical centers because the covariance orientations differ. Right: substantial overlap despite a larger center separation because the covariances are broad. The numerical values are illustrative.

The middle case isolates the limitation of center-only comparison: the means coincide, but the covariance orientations reduce the overlap score. A learned point-set encoder may still recover this distinction from scale and rotation features. The geometric branch makes the distinction explicit in one component of the attention score. Whether this produces semantically meaningful grouping is not evaluated here; the current ablation tests only its

effect on reconstruction.

4.2.5.5 Optional Linear Attention Approximation

The closed-form expression for BC between Gaussians, while elegant, requires computing covariance inverses and determinants. For K groups, computing all pairwise BCs requires $O(K^2)$ such operations – manageable for $K = 256$, but a potential bottleneck for larger scenes or deeper networks. The codebase also contains a linear attention approximation intended to reduce complexity. This path is not used by the reported baseline, which uses full gated Bhattacharyya attention after grouping. The key insight is to learn a feature map $\phi_{\text{geo}} : \mathbb{R}^{10} \to \mathbb{R}^{d_k}$ that approximately factorizes the BC kernel:

$$\phi_{\text{geo}}(G_i) = \text{ELU}(\mathbf{W}_{\text{geo}}[\boldsymbol{\mu}_i \| \log \mathbf{s}_i \| \mathbf{q}_i]) + 1 \tag{4.20}$$

The ELU+1 activation ensures that all feature components are strictly positive – a requirement for the kernel trick to yield valid attention weights. The feature map takes as input the raw Gaussian parameters (position, log-scale, quaternion) and projects them to a space where dot products approximate Bhattacharyya coefficients. With this feature map, linear attention computes:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \frac{\phi(\mathbf{Q})(\phi(\mathbf{K})^\top \mathbf{V})}{\phi(\mathbf{Q})(\phi(\mathbf{K})^\top \mathbf{1})} \tag{4.21}$$

Computing $\phi(\mathbf{K})^\top \mathbf{V}$ first gives linear sequence-length dependence for fixed feature and value dimensions; the full cost also depends on those dimensions. The full-attention implementation mixes geometry and content before normalization:

$$s_{ij} = \lambda \log(\text{BC}(G_i, G_j) + \epsilon) + (1 - \lambda)\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}, \tag{4.22}$$

$$\alpha_{ij} = \text{softmax}_j(s_{ij}). \tag{4.23}$$

Here $\lambda = \sigma(\texttt{geometric\_mix\_logit})$ is the geometric-mix gate: $\lambda \to 1$ selects the Bhattacharyya kernel, while $\lambda \to 0$ falls back to ordinary dot-product attention. Chapter 5 reports the measured gate behavior.

4.2.6 Hierarchical Latent Decomposition

The three-level hierarchy encodes the hypothesis that whole-object, regional, and local information benefit from separate intermediate summaries before compression. The labels "global," "regional," and "local" describe the aggregation structure; they do not by themselves establish semantic disentanglement, part discovery, or topological encoding. Those interpretations require dedicated probes. After transformer processing, we decompose the representation into three hierarchical levels reflecting this multi-scale structure: **Global latent** $\mathbf{z}_g \in \mathbb{R}^{128}$: Mean-pools all group features to form a whole-object summary. **Regional latents** $\mathbf{z}_r \in \mathbb{R}^{8\times64}$: Softly aggregates features into 8 learned regions. Assignment logits $\mathbf{A}_r = \text{softmax}(\mathbf{W}_r\mathbf{Z}/\tau)$ with temperature $\tau = 0.1$ encourage peaked (near-hard) assignments while remaining differentiable. **Local latents** $\mathbf{z}_l \in \mathbb{R}^{8\times8\times32}$: Forms 8 learned local summaries within each region, for 64 local blocks in total. The hierarchical structure satisfies dimension constraint: $128 + 8 \times 64 + 8 \times 8 \times 32 = 128 + 512 + 2048 = 2688$ total dimensions, projected to 512-dimensional bottleneck via learned compression.

4.2.7 The Soft Assignment Blur Problem

Standard softmax produces soft assignments:

$$p(\text{region} = r|G_i) = \frac{\exp(\text{logit}_r)}{\sum_{r'} \exp(\text{logit}_{r'})} \tag{4.24}$$

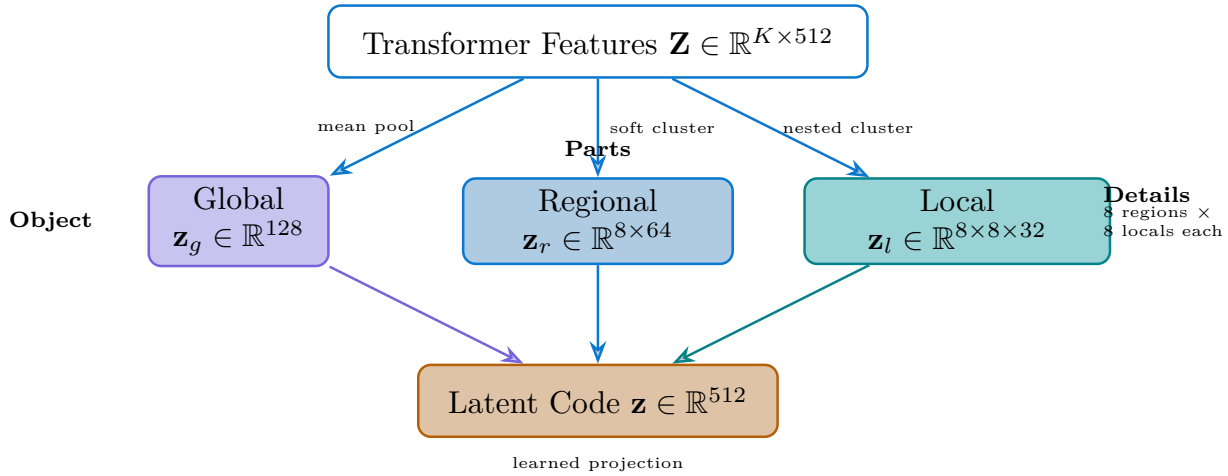

Figure 4.6: Hierarchical latent decomposition. Global captures object identity, regional captures part structure, local captures fine detail. All levels compress to 512-dimensional bottleneck while preserving multi-scale information.

For a Gaussian near the spatial center, logits across 8 regions might be similar, yielding:

$$\mathbf{p} \approx [0.13, 0.12, 0.14, 0.11, 0.13, 0.12, 0.13, 0.12] \tag{4.25}$$

When computing regional features via weighted average:

$$\mathbf{f}_{\text{region}_r} = \sum_i p(\text{region} = r | G_i) \cdot \mathbf{f}_i \tag{4.26}$$

all regions receive similar blurred features – defeating spatial decomposition.

4.2.8 Temperature-Controlled Sharpening

We introduce temperature τ to control assignment sharpness:

$$p(\text{region} = r | G_i) = \frac{\exp(\text{logit}_r / \tau)}{\sum_{r'} \exp(\text{logit}_{r'} / \tau)} \tag{4.27}$$

The temperature has dramatic effects:

Table 4.1: Effect of assignment temperature on regional feature grouping.

τ	Assignment Character	Gradient Behavior
$\tau = 1.0$	Soft, uniform-ish	Full gradients, but blurred features
$\tau = 0.5$	Moderately peaked	Reduced but stable gradients
$\tau = 0.1$	Near-hard, sparse	Very small gradients for non-winners
$\tau \to 0$	One-hot (argmax)	Zero gradients (non-differentiable)

We use $\tau = 0.1$, empirically balancing peaked assignments (distinct regional features) with sufficient gradient flow.

ARCHITECTURAL INSIGHT

The temperature $\tau = 0.1$ is critical. With $\tau = 1.0$ (standard softmax), assignments are too soft – a Gaussian near the center gets nearly uniform weights across regions, destroying spatial decomposition. With $\tau \to 0$ (hard assignment), gradients vanish. The low temperature provides "soft hard" assignments: peaked enough for meaningful regional features, smooth enough for gradient flow.

4.3 Decoder Design

A critical design decision distinguishes our architecture from typical autoencoders: the decoder operates **entirely independently** of the encoder. No skip connections, no cross-attention to encoder features, no information flow from encoder to decoder except through the latent bottleneck. This constraint, while making reconstruction more challenging, is essential for the intended application.

4.3.1 Why Independence Matters

A latent-only decoder is useful whenever codes must be stored, transmitted, or processed without retaining encoder activations. It also creates the architectural precondition for future experiments that treat scene codes as inputs to multimodal models. The present thesis does

not test language-model integration, semantic reasoning, or latent editing; "3D token" is therefore a proposed use case rather than a demonstrated property.

ARCHITECTURAL INSIGHT

Because no encoder features bypass the bottleneck, every reconstruction-relevant signal available to the decoder must pass through the 512-dimensional code. This makes the code self-contained for the evaluated reconstruction task. It does not imply that the code preserves every property of the original scene or is sufficient for untested downstream tasks.

4.3.2 Decoder Architecture

The logged v48 decoder does not expand the code directly into an arbitrary $N \times 512$ tensor in one step. It uses a fixed seed-and-upsampling structure. The 512-dimensional decoder code conditions 64 learned seed features, and each seed is expanded into 50 child slots, producing

$$64 \times 50 = 3{,}200 \tag{4.28}$$

output slots. This fixed maximum agrees with the padded training representation and the recorded average predicted point count of 3,200. Learned count prediction is disabled in the reported configuration.

A schematic formulation is

$$\mathbf{H}_{\text{seed}} = f_{\text{seed}}(\mathbf{z}) \in \mathbb{R}^{64 \times d}, \tag{4.29}$$

$$\mathbf{H}_{\text{child}} = f_{\text{up}}(\mathbf{H}_{\text{seed}}, \mathbf{z}) \in \mathbb{R}^{3200 \times d}, \tag{4.30}$$

$$\mathbf{H}_{\text{refined}} = f_{\text{dec}}(\mathbf{H}_{\text{child}}), \tag{4.31}$$

where f_{dec} contains decoder-side refinement and no cross-attention to encoder activations.

Final attribute heads map each refined slot to the Gaussian parameters

$$\hat{\boldsymbol{\mu}} = \mathrm{MLP}_{\mu}(\mathbf{H}_{\text{refined}}), \tag{4.32}$$

$$\hat{\mathbf{s}} = \exp(\mathrm{MLP}_{s}(\mathbf{H}_{\text{refined}}))\,, \tag{4.33}$$

$$\hat{\mathbf{q}} = \mathrm{normalize}(\mathrm{MLP}_{q}(\mathbf{H}_{\text{refined}}))\,, \tag{4.34}$$

$$\hat{\mathbf{c}} = \sigma(\mathrm{MLP}_{c}(\mathbf{H}_{\text{refined}}))\,, \tag{4.35}$$

$$\hat{\alpha} = \sigma(\mathrm{MLP}_{\alpha}(\mathbf{H}_{\text{refined}}))\,. \tag{4.36}$$

The positivity, unit-quaternion, and bounded-range transforms encode basic validity constraints in the output parameterization.

4.3.3 Inactive Variational Output Path

The code retains optional variational output heads, but the reported v48 runs disable that path. The active decoder is therefore deterministic. If this optional path is enabled, each attribute head predicts a mean and log-variance:

$$\boldsymbol{\mu}_{\text{out}}, \log \boldsymbol{\sigma}_{\mu}^{2} = \mathrm{PositionHead}(\mathbf{F}_{\text{dec}}) \tag{4.37}$$

$$\mathbf{s}_{\text{out}}, \log \boldsymbol{\sigma}_{s}^{2} = \mathrm{ScaleHead}(\mathbf{F}_{\text{dec}}) \tag{4.38}$$

$$\vdots \tag{4.39}$$

The optional path samples via reparameterization $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Its KL divergence $D_{KL}(q(\mathbf{x}|\mathbf{z})\|p(\mathbf{x}))$ is weighted by $\lambda_{KL} = 0.001$ when enabled. This KL term is not part of the active v48 objective.

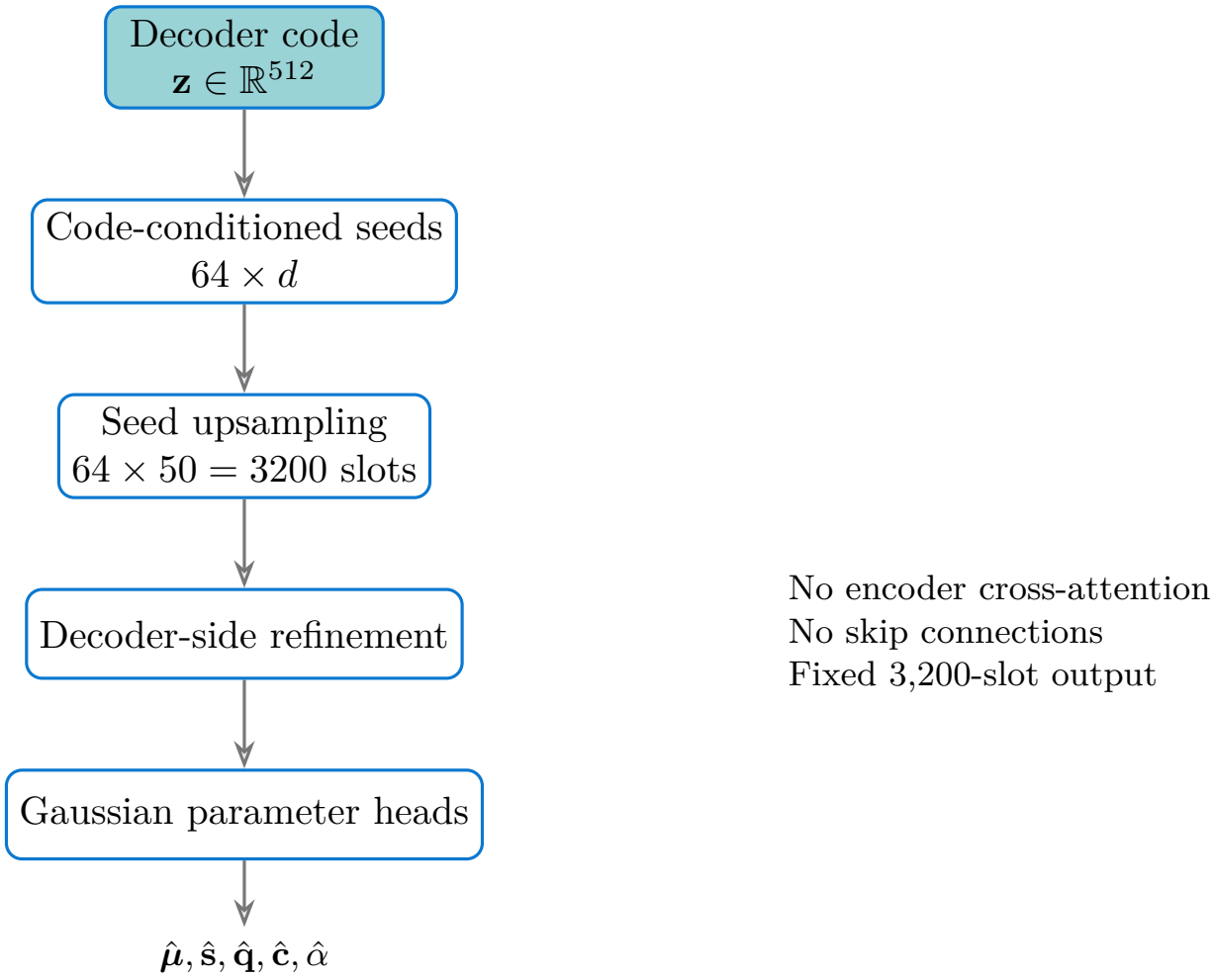

Figure 4.7: Reported latent-only decoder path. The training log records 64 seeds, a factor-50 expansion, and 3,200 output Gaussian slots. Optional count prediction, variational output heads, and noise injection are inactive.

4.3.4 Inactive StyleGAN-Inspired Noise Injection Path

The codebase includes an optional StyleGAN-inspired [Karras et al., 2019] per-point noise path, but it is disabled in the reported experiments:

$$\mathbf{F}'_i = \mathbf{F}_i + \gamma_i \cdot \boldsymbol{\epsilon}_i, \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \mathbf{I}) \tag{4.40}$$

The noise scale γ_i is learned per spatial position, initialized to 0.0. This allows the network to discover which aspects of reconstruction benefit from stochasticity (typically fine surface detail) versus deterministic prediction (global structure).

4.4 Loss Functions

4.4.1 Geometric Reconstruction: Sparse Bhattacharyya Chamfer

Standard point cloud losses (e.g., Chamfer distance) treat Gaussians as point masses. The reported v48 training runs instead use a covariance-aware sparse Chamfer objective with the Bhattacharyya distance as ground cost, extending the encoder's geometric kernel into the reconstruction loss. Given input $\mathcal{G} = \{G_i\}_{i=1}^N$ and reconstruction $\hat{\mathcal{G}} = \{\hat{G}_j\}_{j=1}^M$, define

$$C_{ij} = D_B(G_i, \hat{G}_j). \tag{4.41}$$

Candidate correspondences are screened to the top $k = 32$ neighbors per primitive, yielding sparse neighborhoods $\mathcal{N}_{32}(i)$ and $\hat{\mathcal{N}}_{32}(j)$. The active geometric loss is

$$\mathcal{L}_{\text{geo}} = \frac{1}{N}\sum_{i=1}^{N} \min_{j \in \mathcal{N}_{32}(i)} C_{ij} + \frac{1}{M}\sum_{j=1}^{M} \min_{i \in \hat{\mathcal{N}}_{32}(j)} C_{ij}. \tag{4.42}$$

Appearance terms for color, opacity, scale, and rotation are added in the surrounding

`geometric_appearance_loss`. The sparse entropic-OT path with $\epsilon = 0.05$ and 15 Sinkhorn iterations remains available for `bc_ot` and evaluation, but the reported v48 training objective uses the faster `bc_chamfer` branch.

ARCHITECTURAL INSIGHT

The Bhattacharyya cost ensures that displacing a thin Gaussian along its minor axis, where it is already well-localized, incurs higher cost than moving it along the major axis, where the distribution has large spatial extent. This is precisely correct for rendering: shifting a disk-shaped Gaussian perpendicular to its plane produces a far more visible artifact than shifting it within the plane. Point-cloud losses would treat both displacements equally, systematically underpenalizing the errors that matter most for visual fidelity.

4.4.2 Rendering Loss

We render both input and reconstructed Gaussians from 5 viewpoints at 256×256 resolution and compute the content-aware multi-scale rendering loss

$$\mathcal{L}_{\text{render}} = \frac{\mathcal{L}_{\text{pyr}} + 0.5\,\mathcal{L}_{\text{contrast}} + 0.1\,\mathcal{L}_{\alpha} + 0.3\,\mathcal{L}_{\text{perc}}}{\rho}. \tag{4.43}$$

Here $\mathcal{L}_{\text{pyr}}$ is a 5-level image pyramid MSE, $\mathcal{L}_{\text{contrast}}$ is the local-contrast pseudo-SSIM term, $\mathcal{L}_{\alpha}$ penalizes alpha-mask disagreement, and ρ is the foreground coverage ratio. In v48, `render_use_perceptual=false`, so $\mathcal{L}_{\text{perc}} = 0$. Camera positions are adaptively computed based on object bounding boxes with margin factor 0.05 and distance range $[1.5, 8.0]$.

4.4.3 Flow Matching Latent Regularization

We regularize the latent space via Conditional Flow Matching (CFM) on linear conditional paths [Lipman et al., 2023], trained jointly with the encoder. The current experiments do

not directly compare posterior collapse or generative quality against a VAE baseline. The regularizer maintains a velocity network $v_\psi : \mathbb{R}^{512} \times [0,1] \to \mathbb{R}^{512}$. For each training step, given encoder outputs $\mathbf{z}_1 = E_\phi(\mathcal{G})$ and prior samples $\mathbf{z}_0 \sim \mathcal{N}(0, I)$, we construct a linear conditional path:

$$\mathbf{z}_t = (1-t)\mathbf{z}_0 + t\mathbf{z}_1, \quad t \sim \mathcal{U}(0,1) \tag{4.44}$$

For each paired endpoint sample, the target velocity along this linear path is constant and time-independent:

$$v^* = \mathbf{z}_1 - \mathbf{z}_0 \tag{4.45}$$

The CFM loss:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\,\mathbf{z}_0,\,\mathbf{z}_1}\left\| v_\psi(\mathbf{z}_t + \sigma_{\min}\boldsymbol{\epsilon},\, t) - (\mathbf{z}_1 - \mathbf{z}_0) \right\|^2 \tag{4.46}$$

with $\sigma_{\min} = 10^{-4}$ for numerical stability. The velocity network is a 6-layer MLP (1024 hidden dim) with sinusoidal time embeddings (256-dim), LayerNorm, GELU activations, and zero-initialized output layer for training stability. **The regularization mechanism is joint.** Gradients of $\mathcal{L}_{\text{CFM}}$ with respect to $\mathbf{z}_1$ flow back into the encoder, training it to produce latent codes whose displacement from Gaussian noise is predictable by a smooth velocity field. The objective encourages encoder outputs for which displacement from prior samples can be predicted by the velocity network. Whether this improves interpolation semantics or generative sampling must be evaluated separately. This differs structurally from concurrent work using flow models as latent regularizers [Li et al., 2025], which aligns latent spaces against a *fixed, pretrained* flow model. Here, the flow and the encoder co-evolve: the velocity network learns the current encoder's output geometry; the encoder learns to produce outputs the flow can predict. The implementation can integrate the learned velocity field from $t = 0$ to $t = 1$. Unconditional generation is established only when the transported endpoint is exactly the decoder input and direct samples are evaluated.

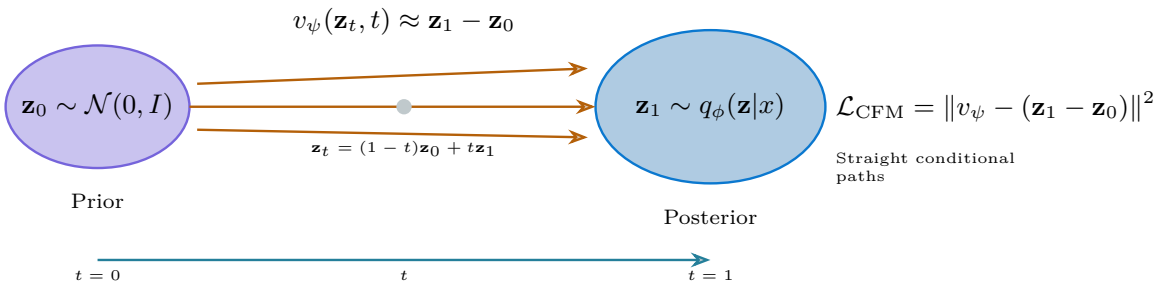

Figure 4.8: Conditional Flow Matching regresses a velocity field along straight conditional paths between prior samples and encoder outputs. The target velocity $v^* = \mathbf{z}_1 - \mathbf{z}_0$ is constant for each paired path. Calling these paths optimal transport requires a separately documented OT endpoint coupling.

4.4.4 Total Loss

$$\mathcal{L} = 1.0\,\mathcal{L}_{\text{geo}} + 1.0\,\mathcal{L}_{\text{render}} + 0.3\,\mathcal{L}_{\text{latent}} + 0.05\,\mathcal{L}_{\text{org}}. \tag{4.47}$$

The latent term is the Conditional Flow Matching objective with `flow_loss_weight=1.0` inside the latent branch and an outer static weight of 0.3. The lightweight organization term regularizes assignment structure with weight 0.05. Hierarchical WAE, volumetric IoU, T-Net orthogonality, diversity, feature decorrelation, count prediction, adaptive uncertainty weighting, and variational KL are inactive in the reported v48 runs.

4.5 Implementation, Complexity, and Numerical Stability

4.5.1 Active and Inactive Paths

The codebase contains more options than the reported model. Table 4.2 separates the active v48 path from retained alternatives. This distinction is necessary because a configurable implementation can otherwise make the paper describe a model that was not actually evaluated.

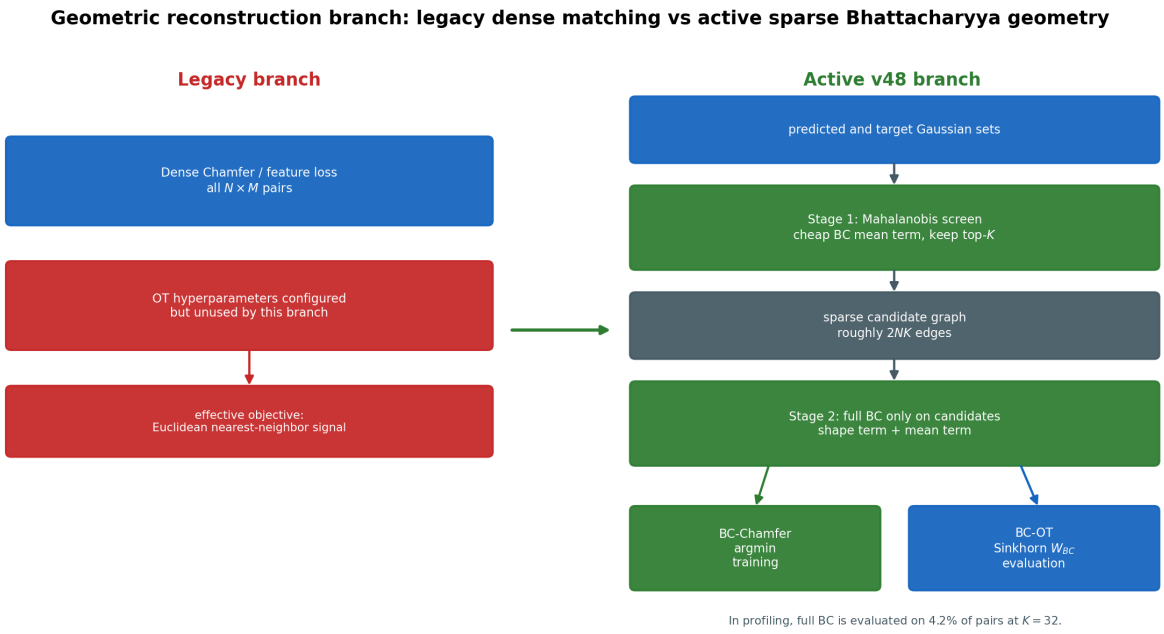

Figure 4.9: Geometric reconstruction branch. The left column summarizes the legacy dense Euclidean path retained in the codebase; the right column is the active v48 sparse Bhattacharyya-Chamfer path. This diagram concerns $\mathcal{L}_{\text{geo}}$ specifically. The complete objective additionally includes multi-view rendering, Conditional Flow Matching, and organization terms. The active geometric branch uses the same Bhattacharyya primitive comparison as the encoder attention in Figure 4.4.

Table 4.2: Implementation paths in the reported v48 baseline.

Component	**State**	**Reported configuration**
PCA pose alignment	Active	Applied before cuboid normalization
Feature grouping	Active	Variable input reduced to 256 groups
Full Bhattacharyya attention	Active	Trainer replaces the linear module with full $O(K^2)$ gated attention after grouping
Linear attention approximation	Inactive in reported path	Retained in code but not used for the evaluated transformer layers
Latent-only decode	Active	Enabled by the logged latent-only decoder switch; no encoder cross-attention
Point-count prediction	Inactive	Decoder emits 3,200 slots
Variational decoder heads	Inactive	Deterministic attribute outputs
Noise injection	Inactive	No stochastic per-slot noise
Conditional flow matching	Active	Six-layer velocity MLP, 1024 hidden units, 256-dimensional time embedding
Perceptual render term	Inactive	Render loss uses pyramid, contrast, and alpha terms; LPIPS is evaluation-only

4.5.2 Sparse Bhattacharyya Candidate Screening

A dense full Bhattacharyya matrix over N target and M predicted primitives requires $O(NM)$ covariance solves and log-determinant evaluations. The active reconstruction path first computes a cheaper mean-separation screen, retains $k = 32$ candidate neighbors, and evaluates the full covariance-aware cost only on the sparse candidate graph. The screening pass remains quadratic in pair count, but expensive covariance scoring is reduced from $O(NM)$ to approximately $O(k(N + M))$ after bidirectional candidate construction.

The active training objective uses bidirectional sparse nearest-neighbor aggregation (BC–Chamfer). A sparse entropic Sinkhorn path with 15 iterations and regularization $\epsilon = 0.05$ remains available for BC–OT evaluation. Calling the training path optimal transport would be inaccurate: its aggregation is Chamfer-like even though its ground cost is distribution-aware.

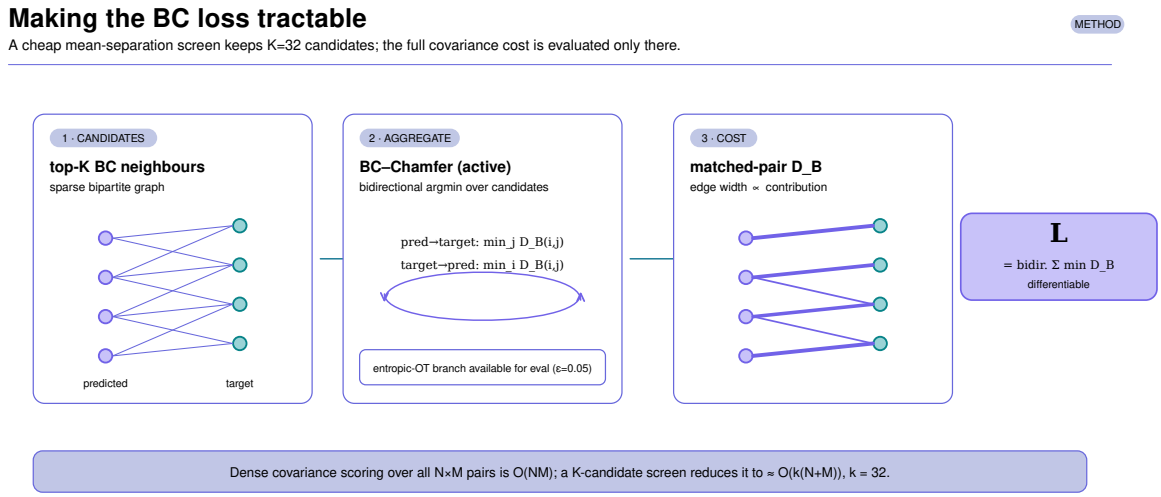

Figure 4.10: Mechanism of the active reconstruction cost. A cheap mean-separation screen retains $k = 32$ candidate neighbors per primitive; the full covariance-aware Bhattacharyya cost is evaluated only on that sparse graph, reducing covariance scoring from $O(NM)$ to approximately $O(k(N + M))$. Aggregation over the retained candidates is the bidirectional sparse nearest-neighbor rule that gives the objective its BC–Chamfer character; the entropic Sinkhorn path ($\epsilon = 0.05$) shown as the available branch is used only for BC–OT evaluation, not for the reported training runs.

4.5.3 Numerical Hardening

Covariance-aware computation is sensitive to nearly degenerate predicted scales. In finite precision, a nominally positive covariance can become ill-conditioned, and direct inversion can amplify errors. The implementation regularizes covariance matrices before solves, uses stable log-determinant operations, clamps unsafe ranges, and replaces non-finite intermediate values where necessary. Conceptually, a regularized covariance takes the form

$$\boldsymbol{\Sigma}_{\text{safe}} = \boldsymbol{\Sigma} + \eta(\boldsymbol{\Sigma})\mathbf{I}, \tag{4.48}$$

where η combines an absolute floor with a scale-relative term. This changes the cost slightly near degeneracy, but it preserves finite gradients and makes the objective usable during early training when output scales are poorly calibrated.

4.5.4 Algorithmic Summary

High-level sparse Bhattacharyya reconstruction procedure.

```
# pred and target are sets of Gaussian primitives.
mean_screen = pairwise_mean_component(pred, target)
forward = topk_smallest(mean_screen, k=32, axis="target")
backward = topk_smallest(mean_screen, k=32, axis="prediction")
edges = union(forward, backward)

full_bc_cost = bhattacharyya_distance(pred, target, edges)
loss_geo = bidirectional_sparse_min(full_bc_cost)
```

The same closed-form primitive comparison is used in two roles: a log-coefficient contributes to encoder attention, while distance contributes to reconstruction. The parameters and aggregations differ, so the two uses should not be called identical objectives. They are, however, grounded in the same distributional notion of primitive overlap.

CHAPTER 5: EMPIRICAL RESULTS

5.1 Experimental Setup

We evaluate GLEA on the ShapeSplat dataset [Ma et al., 2025], which contains 36,248 3D Gaussian splat scenes across 32 ShapeNet object categories. We use a disjoint split of 34,431 training and 1,817 held-out scenes (seed 20260627). Each scene is a variable-length set of between roughly 500 and 3,200 Gaussian primitives, and every primitive is a 14-dimensional vector $(\boldsymbol{\mu}, \mathbf{s}, \mathbf{q}, \mathbf{c}, \alpha)$ comprising position, scale, rotation quaternion, colour, and opacity. Scenes are padded to a fixed maximum of 3,200 primitives with validity masks so that the batched computation never conflates padding with content.

5.1.1 Optimization

The reported models are trained with AdamW using weight decay 10^{-8} and a peak learning rate of 10^{-3}. The learning rate is warmed up over the first 5% of training and then cosine-annealed to 10^{-4}. Gradients are clipped to a maximum norm of 1.0. The matched ablation runs use random seed 20260627, batch size 2, a four-epoch training budget, and the same checkpoint-selection procedure. The flow-matching velocity network uses twice the main learning rate. Because the experiments use a single training seed and a short budget, the values are controlled descriptive comparisons rather than estimates of across-run variability.

5.1.2 Loss Configuration

The training objective is a static weighted sum of four active terms,

$$\mathcal{L} = 1.0\,\mathcal{L}_{\text{geometric}} + 1.0\,\mathcal{L}_{\text{render}} + 0.3\,\mathcal{L}_{\text{latent}} + 0.05\,\mathcal{L}_{\text{organization}}, \tag{5.1}$$

with the remaining auxiliary terms (hierarchical WAE, volumetric IoU, T-Net orthogonality, spatial diversity, feature decorrelation, count prediction, variational KL, and adaptive uncertainty weighting) disabled for the reported v48 experiments. The geometric term is a covariance-aware Chamfer reconstruction whose ground cost is the Bhattacharyya distance between primitives (`mahalanobis_distance_metric = bhattacharyya`); candidate correspondences are established with a sparse screen rather than dense all-pairs matching. The render term is a content-aware multi-view photometric loss combining pyramid MSE with a local-contrast pseudo-SSIM term over 256×256 views. The latent term is the Conditional Flow Matching objective of Section 4.4.3, trained jointly with the encoder; its branch-internal `flow_loss_weight` is 1.0 before the outer static weight of 0.3. The organization term is the lightweight assignment-structure regularizer used in the v48 run logs.

5.1.3 Architecture Configuration

Unless an ablation states otherwise, the encoder applies PCA pose alignment and cuboid normalization, followed by a 16-level multi-resolution hash encoding, learned grouping to 256 groups, and four layers of full gated attention. Each layer mixes a Bhattacharyya-based geometric score with a content dot-product score before softmax normalization. Although the codebase contains a linear-attention approximation, the reported baseline uses full $O(K^2)$ Bhattacharyya attention for $K = 256$ groups. The deterministic decoder receives only the 512-dimensional code and emits a fixed maximum of 3,200 primitive slots. Count prediction, variational heads, and noise injection are disabled.

5.1.4 Evaluation Protocol

Reconstruction quality is measured by PSNR, SSIM, and LPIPS computed from renders, plus Chamfer distance evaluated directly on Gaussian centres. Baselines and ablations share the same decoder, optimizer, split, seed, and four-epoch training budget; only the tested component varies.

5.2 Metrics and Reporting Conventions

The evaluation combines primitive-space and image-space measurements. Center Chamfer distance is the bidirectional nearest-neighbor distance between reconstructed and target primitive centers; it does not compare covariance, color, or opacity. PSNR measures image reconstruction on a logarithmic signal-to-error scale. SSIM measures local structural agreement, and LPIPS measures distance in a learned perceptual feature space. Lower is better for Chamfer and LPIPS; higher is better for PSNR and SSIM.

All principal v48 tables report the arithmetic mean over 128 objects. Where space permits, the accompanying standard deviation describes object-to-object dispersion. These standard deviations are not uncertainty over training runs because the model was trained with one seed. The train-minus-held-out difference is a descriptive sample comparison, not a confidence interval and not an equivalence test.

The v48 post-hoc renderer uses four views at 256×256 resolution. Training render supervision uses five views at the same nominal resolution. The distinction matters: the number of training views is a loss configuration, whereas the four-view protocol defines the reported image metrics.

5.2.1 Checkpoint Selection and the Meaning of "Held-Out"

The object split is disjoint: the 34,431 training paths and 1,817 held-out paths do not overlap under seed 20260627. The selected baseline checkpoint is the epoch-2 checkpoint with recorded validation total loss 13.18. However, the training process used batches from the held-out partition for validation and checkpoint selection. The partition is therefore held out from gradient updates, but it is not a sealed final test set untouched by model selection. The thesis uses "held-out" in this precise sense and does not call the partition an independent final test set.

A publication-grade follow-up should divide the current held-out pool into validation and test subsets, select checkpoints only on validation, and report the final test once. The current comparison is still useful for controlled reconstruction and ablation because every v48 variant uses the same split and selection procedure, but checkpoint exposure limits the strength of the generalization claim.

5.3 Train and Held-Out Reconstruction Comparison

The disjoint v48 split is the primary held-out reconstruction result used in this thesis. The older v46 run is useful for latent diagnostics, but it is not used to support train-versus-held-out claims because its nominal validation split was later found to contain training objects.

Table 5.1: v48 reconstruction metrics over 128 objects per group. Entries are mean ± object-level standard deviation. Lower is better for center Chamfer and LPIPS; higher is better for PSNR and SSIM.

Group	**Center Chamfer ↓**	**PSNR (dB) ↑**	**SSIM ↑**	**LPIPS ↓**
Training sample	0.001765 ± 0.002083	25.42 ± 2.35	0.955 ± 0.015	0.129 ± 0.029
Held-out sample	0.001536 ± 0.000878	25.57 ± 2.64	0.956 ± 0.016	0.124 ± 0.032

Mean held-out PSNR is 25.57 dB, compared with 25.42 dB for the sampled training objects,

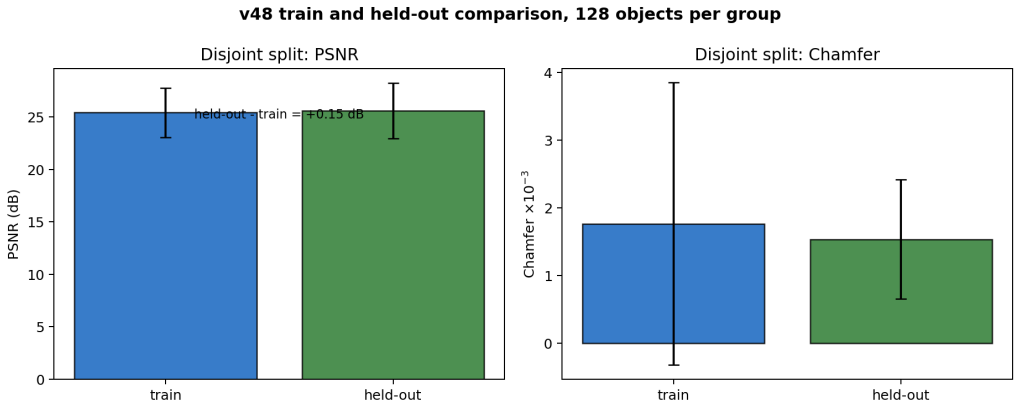

Figure 5.1: Training versus held-out reconstruction under one renderer and metric suite. The sampled held-out mean is close to the sampled training mean under the same evaluation protocol.

so the train-minus-held-out difference is -0.15 dB. This sample does not show a worse held-out mean. The result is consistent with limited instance-level memorization under the disjoint split, but it is not a statistical equivalence test and does not establish category-level or cross-dataset generalization.

5.4 Qualitative Reconstruction Results

Figure 5.2 shows the metric-matched held-out v48 example across multiple viewpoints. The logged renderer panels have been cropped to remove empty white margins and enlarged uniformly so that silhouette, orientation, and view consistency can be inspected at print scale; the underlying input and reconstruction pixels are otherwise unchanged.

5.5 Quantitative Ablation Study

Table 5.2 reports the matched v48 ablation ladder. Each row removes one component while holding the split, seed, architecture, optimizer, and four-epoch training budget fixed. The hierarchy row is approximate because it collapses the hierarchical latent to a one-region form and was trained under the final hardened Bhattacharyya kernel, so it should be replicated

Figure 5.2: Qualitative v48 reconstruction across viewpoints, reformatted for legibility from the original logged panel. The top row shows input renders and the bottom row shows the GLEA reconstruction decoded from the latent code alone. Object identity, silhouette, approximate orientation, and broad appearance are recovered consistently across views, while fine detail and thin structures remain visibly smoothed.

before being treated as definitive.

Table 5.2: v48 ablation necessity map on 128 held-out objects. Entries are mean ± object-level standard deviation; ΔPSNR is relative to the full model. The hierarchy row is approximate.

Variant	PSNR ↑	ΔPSNR	Center Chamfer ↓	SSIM ↑	LPIPS ↓
GLEA full model	$\mathbf{25.57 \pm 2.64}$	—	$\mathbf{(1.54 \pm 0.88) \times 10^{-3}}$	$\mathbf{0.956 \pm 0.016}$	$\mathbf{0.124 \pm 0.032}$
w/o PCA pose alignment	24.90 ± 2.49	-0.67	$(1.73 \pm 1.08) \times 10^{-3}$	0.954 ± 0.016	0.133 ± 0.032
w/o Bhattacharyya attention	24.37 ± 2.83	-1.20	$(1.95 \pm 1.40) \times 10^{-3}$	0.949 ± 0.019	0.140 ± 0.036
w/o hierarchy*	25.39 ± 2.91	-0.18	$(1.71 \pm 1.30) \times 10^{-3}$	0.953 ± 0.018	0.126 ± 0.034

The necessity ordering is sharpest at its top: removing Bhattacharyya attention costs 1.20 dB of PSNR, the largest drop in the ladder, and also produces the worst Chamfer, SSIM, and LPIPS. That matters because PSNR, SSIM, LPIPS, and Euclidean Chamfer contain no Bhattacharyya coefficient. The degradation is not a self-referential score improving itself; the geometry improves external image and positional metrics.

PCA pose alignment is the second-most visible component in this short-budget ladder, costing 0.67 dB when removed. The hierarchy ablation is the most provisional row: it costs only 0.18 dB and carries the kernel-hardening confound described above. The hierarchy row should be treated as provisional because the artifact is marked approximate. The geometric-attention difference is larger, but it also requires longer-budget and multi-seed replication before its

v48 BC-Chamfer ablation: 128 held-out objects, matched 4-epoch budget

variant	Chamfer ↓	PSNR ↑	SSIM ↑	LPIPS ↓	ΔPSNR
baseline	1.54e-3	25.57	0.956	0.124	--
- PCA pose-norm	1.73e-3	24.90	0.954	0.133	-0.67
- geom attention	1.95e-3	24.37	0.949	0.140	-1.20
- hierarchy*	1.71e-3	25.39	0.953	0.126	-0.18

red = largest observed degradation; green = smallest observed degradation; * approximate 1x1 ablation

Figure 5.3: v48 ablation summary generated from the local ablation artifacts. The largest degradation occurs when Bhattacharyya attention is removed.

stability can be estimated. Matched-budget ablations for removing flow matching and hash encoding remain future work rather than blank rows in the results table.

5.6 The Full-Latent Regime: Fidelity Versus Interpretability

Every v48 number reported above is produced under `use_latentz_decode=true`: the decoder receives only the compressed, flow-regularized 512-dimensional `latent_z`. This is a deliberately demanding configuration. It forces the single vector that the conditional flow-matching objective transports toward $\mathcal{N}(0, I)$ to be the exact vector the decoder consumes, which is what makes the latent diagnostics of Section 5.2 ff. and the generative-wiring audit of Chapter 6 meaningful in the first place. The same constraint withholds most of the encoder's hierarchical detail from reconstruction.

The architecture exposes the opposite setting as a one-flag change. With `use_latentz_decode=false` the decoder upsamples the full hierarchical intermediate representation directly—the global, regional, and local blocks whose 2,688 concatenated dimensions would otherwise be projected down to the code—so that far more of the encoded signal reaches the output. We call this the *full-latent* regime. It is expected to reconstruct more faithfully, because the decoder is no longer squeezed through a 512-dimensional bottleneck, and it is correspondingly less interpretable, because there is no longer a single compact, regularized

code to analyze or to sample: the participation-ratio spectrum of Section 5.2 and the prior-transport path of Chapter 6 describe the bottleneck vector, not the full hierarchy that now drives reconstruction.

The project's full-hierarchy-decode runs bear this out. Under the earlier `version_28`/v46 configuration, which decodes from the full hierarchy, reconstructions reach roughly 25.8 dB PSNR, 0.963 SSIM, and 0.094 LPIPS on the broach validation objects and roughly 28.2 dB PSNR, 0.967 SSIM, and 0.083 LPIPS on random-splat validation—perceptually much sharper (LPIPS 0.083 versus 0.124) than the bottleneck decoder on the disjoint v48 split. Table 5.3 places the two regimes side by side.

Table 5.3: The full-latent regime trades interpretability for reconstruction fidelity. The bottleneck decoder (reported v48) consumes only the compact, flow-regularized code and supports the latent diagnostics and generative path; the full-latent decoder consumes the entire hierarchy and reconstructs more sharply but exposes no single analyzable code. The two rows are *not* measured on the same split: the full-latent figures come from the earlier `version_28`/v46 regime whose held-out partition is contaminated (Appendix B), so they bound the available fidelity and illustrate the tradeoff rather than furnishing a matched comparison.

Decode path	**Decoder input**	**PSNR (dB) ↑**	**SSIM ↑**	**LPIPS ↓**	**Compact code / prior?**
Bottleneck (reported v48)	`latent_z` (512-d)	25.57	0.956	0.124	yes / yes
Full-latent (`v28`/v46)†	full hierarchy (2,688-d)	∼28.2 / 25.8	0.967 / 0.963	0.083 / 0.094	no / no

†Random-splat / broach validation, prior-regime split; see Appendix B.

The reading is not that one decoder is universally better. It is that the latent-decode flag selects between two genuinely different objects: a maximally faithful reconstructor and an analyzable, samplable representation. Because the higher-fidelity figures were obtained under the earlier split discipline of Appendix B, whose `holdout` partition is flagged contaminated, they should be read as an upper bound on what the full-latent decoder recovers, not as a clean held-out result. The confirming experiment is inexpensive and fully specified: rerun the v48 baseline command on the identical disjoint split, seed, renderer, and metric implementation

with `use_latentz_decode` toggled, and report both decoders once on a sealed test set. The predicted outcome is a uniform improvement in the image and center metrics for the full-latent decoder, purchased by giving up the compact, low-participation, flow-compatible code that the rest of this thesis analyzes. Which regime is preferable is therefore decided downstream, by whether the task needs reconstruction fidelity or a latent that can be measured and generated from.

5.7 Supplementary and Prior-Regime Material

Late-epoch `version_28` reconstructions, the contaminated v46 reconstruction regime, and additional prior-sampling context are retained in Appendix B. They are useful for understanding project development and decoder behavior, but they are not mixed with the primary v48 generalization claim because their configurations or split discipline differ.

5.8 Threats to Validity

Four limitations constrain interpretation of the main tables. First, all variants use one training seed and a four-epoch budget. Object-level standard deviations describe heterogeneous examples, not reproducibility across optimization runs. Second, the evaluation uses 128 sampled objects per group rather than the full split. Third, categories are shared across training and held-out partitions, so the study measures instance-level generalization within ShapeSplat rather than category-held-out transfer. Fourth, the held-out partition participates in checkpoint selection, as described in Section 5.2.

The ablation ladder also varies in evidentiary strength. The no-PCA and no-geometric-attention variants are exact toggles under matched commands. The hierarchy row is marked approximate in the artifact metadata and should not be treated as a definitive estimate of the hierarchy's contribution. No matched v48 ablations are available for hash encoding or

flow matching, and no external autoencoder baseline was retrained under the same renderer, split, and metric implementation.

Image metrics depend on camera selection, background, rasterizer settings, and view count. They are therefore meaningful within the documented protocol, not as implementation-independent constants. Center Chamfer ignores covariance and appearance, while image metrics conflate several primitive errors through rendering. Their agreement in the geometric-attention ablation is informative precisely because each metric observes a different projection of reconstruction quality.

5.9 Latent-Code Diagnostics

Reconstruction fidelity establishes that the model can reconstruct the evaluated objects; the following measurements probe how variation is organized in the sampled codes. The narrower question is whether the code distribution concentrates along a small number of linear directions or behaves like an unstructured collection of per-object encodings.

5.9.1 Effective Linear Dimensionality

We encode 2,048 objects and measure the participation ratio $\mathrm{PR}(\lambda) = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$ of each latent block's covariance spectrum, alongside the number of principal components needed to explain 90, 95, and 99% of variance.

Table 5.4: Participation-ratio and principal-component diagnostics for the latent blocks over 2,048 encoded objects. PR estimates effective linear dimensionality in the sampled codes; the last three columns give the number of principal components needed for 90/95/99% of variance.

Latent block	**Stored dim.**	**PR**	**90%**	**95%**	**99%**
Global	128	6.10	7	9	11
Regional	512	6.33	29	62	169
Local	2048	7.24	33	91	377
`latent_z`	512	11.90	74	131	270

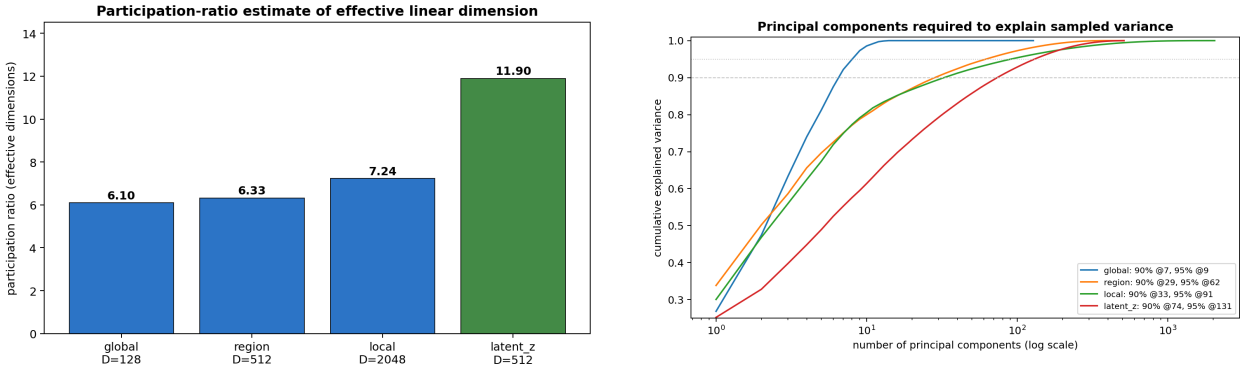

Figure 5.4: Effective linear dimensionality of sampled latent codes. Left: participation ratio per block. Right: cumulative explained variance. These are covariance-spectrum diagnostics, not estimates of global topology or the dimension of all valid scenes.

Each sampled decoder-input block concentrates variance in far fewer linear directions than its storage size. The global latent concentrates most variation in roughly six effective directions out of 128 and reaches 90% cumulative variance in seven components. Region and local latents show the same pattern despite larger storage sizes. The interpretation is not that the codes are rank-deficient; the 99% tails run to hundreds of components. The result is a low effective linear dimension with a longer residual tail. This characterizes variance concentration in the sampled learned codes; it does not establish a manifold chart or semantic coordinate system.

5.9.2 Latent Interpolation

If the encoder had learned isolated per-object codes, a straight line between two encodings would pass through invalid intermediate splats. Instead, linear interpolation between encoded objects produces smooth, coherent morphs (Figure 5.5), providing qualitative evidence that the decoder behaves smoothly along this selected path. Broader connectedness requires many endpoint pairs and quantitative validity measures.

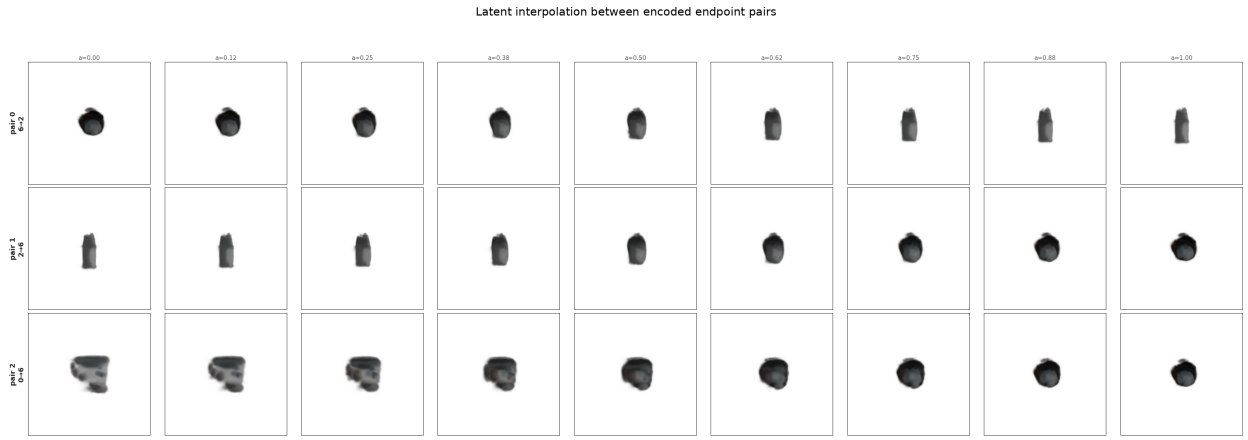

Figure 5.5: Latent interpolation between two encoded objects. Along this selected line segment, decoded outputs change smoothly and remain visually coherent. This is a qualitative path-specific observation, not a test of global connectedness or manifold topology.

5.9.3 Matched-Budget Geometric-Attention Gate

Each Bhattacharyya-attention layer carries a learnable gate $\lambda = \sigma(\texttt{geometric\_mix\_logit})$ that blends the geometric kernel with ordinary dot-product attention. Table 5.5 compares two four-epoch runs with the same split, seed, and architecture. The comparison therefore isolates the reconstruction objective more cleanly than the much longer historical v46 trajectory.

Table 5.5: Geometric-mix gate λ under matched four-epoch conditions. The rows share the same split, seed, architecture, and budget; only the reconstruction objective differs.

Objective	**Budget**	**λ mean**	**Layer-0 gate**
`volumetric_iou`	4 epochs	0.359	0.297
BC–Chamfer	4 epochs	0.233	0.203

The gate and ablation answer different questions. The gate measures how much geometric mixing a trained model chooses to use; the ablation measures how much performance is lost when the branch is removed and the model is retrained. At matched budget there is no contradiction: the BC–Chamfer model retains a nonzero mean geometric gate (0.233), while removing Bhattacharyya attention costs 1.20 dB. A relatively small gate can still carry a non-substitutable signal that is useful only for particular layers, examples, or stages of optimization.

The long-budget v46 gate trajectory and the earlier empirical-versus-prior sampling diagnostic are retained in Appendix B. Their configurations differ from the primary v48 path, so they are used as historical architecture diagnostics rather than as primary reconstruction or generation results.

CHAPTER 6: DISCUSSION: LIMITATIONS AND CONCLUSION

This thesis introduced the Geometric Latent Encoding Autoencoder (GLEA), a class-agnostic autoencoder for 3D Gaussian-splat scenes. The system is motivated by a representational observation: an anisotropic Gaussian primitive is characterized not only by its center but also by a covariance that determines its orientation and spatial extent. GLEA therefore augments learned content attention with a geometric score derived from the Bhattacharyya coefficient, combines that score with canonicalization and learned grouping, compresses a hierarchical intermediate representation to 512 dimensions, and reconstructs a fixed maximum of 3,200 primitives without encoder skip connections.

The value of the project lies less in any single module than in the alignment between question, architecture, and evidence. The mathematical question is whether full Gaussian overlap provides a principled comparison unavailable to center-only geometry. The architectural question is whether that comparison can be inserted as a useful, learnable branch rather than a rigid replacement for content attention. The empirical question is whether the branch improves reconstruction when removed under matched conditions. This thesis deliberately keeps these questions separate because each requires a different kind of support.

6.1 What the Current Evidence Supports

The primary v48 evaluation supports three conclusions within the documented setting. First, a latent-only decoder can reconstruct held-out ShapeSplat objects with mean PSNR of 25.57 dB, SSIM of 0.956, LPIPS of 0.124, and center Chamfer distance of 1.54×10^{-3} over 128 objects. The decoder receives the 512-dimensional code rather than cached encoder activations, so the result establishes that the code is sufficient for the evaluated reconstruction

path.

Second, the sampled training and held-out means are close under the same four-view renderer and metric implementation. The train-minus-held-out PSNR difference is -0.15 dB. This observation is consistent with limited instance-level memorization on the disjoint object split. It should not be interpreted as a statistical equivalence test, a category-transfer result, or a guarantee about the full held-out population.

Third, removing Bhattacharyya attention lowers mean held-out PSNR by 1.20 dB and worsens SSIM, LPIPS, and center Chamfer. This is the clearest evidence for the central architectural hypothesis. The external metrics do not contain the Bhattacharyya coefficient, so the result is not a tautology in which a model merely improves the quantity it explicitly optimizes. Removing PCA pose alignment also lowers PSNR by 0.67 dB, indicating that canonicalization is consequential under the short training budget. The approximate hierarchy row shows a smaller difference but is not sufficiently controlled for a definitive conclusion.

EVIDENCE BOUNDARY

The defensible empirical statement is conditional: covariance-aware attention is load-bearing in the evaluated v48 architecture, split, seed, checkpoint-selection procedure, and four-epoch budget. The experiment does not establish that Bhattacharyya attention is optimal among all covariance-aware alternatives or necessary in every sufficiently large model.

6.2 What the Theory Establishes – and What It Does Not

The Bhattacharyya coefficient is a mathematically coherent affinity for probability densities. The square-root density embedding writes it as an L^2 inner product, which proves positive semidefiniteness. For nearby distributions, the associated divergence induces the Fisher information metric to second order. For Gaussian primitives, the closed form separates mean

displacement from covariance mismatch and can distinguish equal-center primitives with different spatial support.

These facts justify the branch at the level of primitive comparison. They do not imply that softmax attention requires a positive-semidefinite kernel, that the Bhattacharyya distance equals the global Fisher–Rao geodesic distance, or that the final 512-dimensional code inherits Fisher–Rao geometry. Grouping, nonlinear attention, pooling, projection, the decoder, and the training objective can each distort pairwise relationships. Demonstrating metric preservation would require scene-level distance definitions and direct measurements of neighborhood preservation or distortion.

The strongest theoretical interpretation is therefore local and architectural. Information geometry identifies a natural reference for infinitesimal statistical distinguishability. Bhattacharyya overlap provides a tractable, symmetric, covariance-sensitive surrogate. GLEA makes that surrogate available to the network through a learned gate. The ablation then determines whether the resulting inductive bias is useful.

6.3 Interpreting the Latent Diagnostics

The participation-ratio analysis shows strong variance concentration in sampled codes. The global block has participation ratio 6.10 despite 128 stored dimensions, while the regularized `latent_z` has participation ratio 11.90 despite 512 stored dimensions. These values indicate low effective linear dimension in the observed covariance spectra. They do not identify the intrinsic topological dimension of all valid scenes, and they do not show that the active coordinates are semantically disentangled.

Likewise, the interpolation figure demonstrates smooth decoder behavior along one selected line segment. It is evidence against a visibly discontinuous transition for that pair, not proof that the encoded set is globally connected or that straight lines are geodesics. A

systematic interpolation study should sample many endpoint pairs, compare in-distribution and cross-category paths, and quantify validity, reconstruction energy, and local Jacobian behavior along each path.

Empirical-posterior sampling produces object-like outputs because it samples near codes the decoder has observed. Naive unit-Gaussian sampling is not yet a valid generative test unless the learned flow transports the prior to the exact decoder-input code. The active command enables `use_latentz_decode=true`, but the project history contains earlier regimes in which the regularized code and decoder input were not identical. The generative path therefore needs an end-to-end wiring audit and direct sample evaluation rather than inference from reconstruction alone.

6.4 Limitations and Alternative Explanations

6.4.1 Single-Seed, Short-Budget Training

Every principal model was trained with seed 20260627 for four epochs. A component can appear important at short horizon because it accelerates optimization, even if a larger or longer-trained baseline eventually compensates. Conversely, a useful component can be underestimated if it requires more time to integrate. Multi-seed learning curves, not only final checkpoints, are needed to separate optimization acceleration from asymptotic representational advantage.

6.4.2 Checkpoint Selection and Split Semantics

The v48 split is object-disjoint, correcting the leakage found in the earlier v46 regime. However, the held-out partition was used for validation and checkpoint selection. It is held out from gradient updates but not from model selection. This weakens a strict test-set interpretation

and can introduce optimistic selection bias. A future evaluation should maintain separate training, validation, and sealed test manifests.

The split also shares the same 32 ShapeNet categories across partitions. The result therefore concerns new instances from known categories. Category-held-out evaluation, where entire semantic classes are absent during training, would test a substantially stronger form of transfer.

6.4.3 Metric and Renderer Dependence

PSNR, SSIM, and LPIPS depend on the selected views, rasterizer, resolution, background, and camera normalization. Center Chamfer ignores covariance and appearance. No single metric fully characterizes a Gaussian-splat reconstruction. The current agreement across image and center metrics is encouraging, but future work should add covariance-aware assignment metrics, opacity-aware rendering diagnostics, silhouette measures, and per-category breakdowns.

6.4.4 Unablated Components and External Comparisons

The model combines PCA alignment, cuboid normalization, hash features, grouping, geometric attention, hierarchy, rendering supervision, sparse Bhattacharyya reconstruction, and flow matching. Only a subset has a completed matched ablation. Hash encoding and flow matching remain unablated in the primary v48 regime, and the hierarchy result is approximate. The thesis also lacks an external point-set or native-splat autoencoder retrained with the same data split, output cardinality, renderer, and evaluation code. Consequently, state-of-the-art or comparative superiority claims are not warranted.

6.4.5 Fixed Output Cardinality

The decoder emits 3,200 slots because learned count prediction is disabled. This simplifies batching and ensures a uniform output tensor, but it can allocate unnecessary primitives to simple objects and may encourage low-opacity or redundant slots. A principled variable-cardinality decoder would need to predict existence probabilities or a count distribution and would require metrics robust to empty or inactive slots.

6.5 Prioritized Future Experiments

The next experiments should be chosen to discriminate among competing explanations rather than merely add more metrics.

1. **Multi-seed replication with paired statistics.** Train the full model, no-geometric-attention model, and no-PCA model for at least five seeds. Evaluate the same objects for every seed and report paired per-object differences, bootstrap confidence intervals, learning curves, and seed-level variation.
2. **Longer-budget necessity map.** Repeat the ablation ladder until validation curves plateau. Add exact hierarchy, hash-encoding, geometric-loss, and flow-matching toggles. This separates components that improve optimization speed from those that improve the attainable solution.
3. **Sealed test protocol.** Freeze train, validation, and test manifests before model development. Use validation for early stopping and hyperparameter selection, then evaluate test exactly once. Publish hashes of all manifests with the artifacts.
4. **Matched external baselines.** Retrain a content-only set autoencoder and at least one native-splat representation model with the same data, 512-dimensional bottleneck, decoder cardinality, renderer, views, and metrics. This establishes whether the gain comes from covariance-aware comparison rather than unrelated capacity differences.
5. **Alternative geometric affinities.** Compare Bhattacharyya overlap with center distance, symmetrized KL, 2-Wasserstein Gaussian distance, affine-invariant covariance distance, and learned covariance features. Such a study would test whether the benefit is specific to Bhattacharyya structure or to covariance awareness more generally.
6. **Latent metric tests.** Define a scene-level distance using matched Gaussian distributions or rendered views, then measure rank correlation, neighborhood recall, local distortion, and decoder Jacobian conditioning in the 512-dimensional code. These

experiments directly test the hypothesis that primitive-level geometry induces useful scene-level organization.

7. **Category and dataset transfer.** Hold out complete ShapeNet categories and evaluate on an independently generated splat dataset. Transfer should be separated into reconstruction, retrieval, classification, and adaptation settings because each probes a different property.
8. **End-to-end generation audit.** Confirm that the flow endpoint is exactly the decoder-input code, integrate the learned velocity field from prior samples, and report validity, diversity, nearest-neighbor distance to training data, and reconstruction-versus-generation tradeoffs. Compare unit-Gaussian, empirical-posterior, and transported-prior samples.

6.6 Conclusion

The central contribution of this thesis is a concrete, testable way to incorporate covariance-aware similarity into representation learning for Gaussian-splat scenes. The work begins from a first-principles mismatch: a Gaussian primitive has spatial extent and orientation, while center-only geometry cannot express either. It then supplies a tractable distributional affinity, integrates that affinity with learned content attention, and evaluates its necessity through a controlled removal.

The present results do not establish every broader implication of information-geometric latent spaces. They do establish a meaningful empirical signal: in the documented v48 experiment, removing the Bhattacharyya attention branch degrades reconstruction in image space and primitive-center space. That result justifies continued investigation of geometric inductive biases for 3DGS encoders.

The broader research program remains compelling precisely because it is falsifiable. Primitive-level geometry should be retained only when it improves controlled comparisons; scene-level geometry should be claimed only when measured; and generative use should be claimed only when the prior, transport, and decoder are connected and tested end to end. In that form, the thesis is not an assertion that one geometry solves representation learning. It is a

framework for turning geometric commitments into architectural choices and then subjecting those choices to evidence.

REFERENCES

[Amari, 1998] Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

[Calvo-Ordoñez et al., 2025] Calvo-Ordoñez, S., Meunier, M., Cartea, A., Reisinger, C., Gal, Y., and Hernández-Lobato, J. M. (2025). Weighted conditional flow matching. *arXiv preprint arXiv:2507.22270*.

[Chen et al., 2018] Chen, R. T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Groueix et al., 2018] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018). A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224.

[Karras et al., 2019] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410.

[Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14.

[Li et al., 2025] Li, Y., Ge, Y., Ge, Y., Shan, Y., and Luo, P. (2025). Aligning latent spaces with flow priors. *arXiv preprint arXiv:2506.05240*.

[Lipman et al., 2023] Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*.

[Luiten et al., 2024] Luiten, J., Kopanas, G., Leibe, B., and Ramanan, D. (2024). Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision (3DV)*.

[Luo, 2025] Luo, S. T. S. (2025). A new perspective to understanding multi-resolution hash encoding for neural fields. *arXiv preprint arXiv:2505.03042*.

[Ma et al., 2025] Ma, Q., Li, Y., Ren, B., Sebe, N., Konukoglu, E., Gevers, T., Van Gool, L., and Paudel, D. P. (2025). A large-scale dataset of gaussian splats and their self-supervised pretraining. In *2025 International Conference on 3D Vision (3DV)*, pages 145–155. IEEE.

[Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15.

[Musgrave et al., 2020] Musgrave, K., Belongie, S., and Lim, S.-N. (2020). A metric learning reality check. In *European Conference on Computer Vision (ECCV)*, pages 681–699.

[Niedermayr et al., 2024] Niedermayr, S., Stumpfegger, J., and Westermann, R. (2024). Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Tang et al., 2024] Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G. (2024). DreamGaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations (ICLR)*.

[Tolstikhin et al., 2018] Tolstikhin, I. O., Bousquet, O., Gelly, S., and Schölkopf, B. (2018). Wasserstein auto-encoders. In *International Conference on Learning Representations (ICLR)*.

[Xin et al., 2025] Xin, Y., Liu, Y., Xie, X., and Li, X. (2025). Learning unified representation of 3d gaussian splatting. *arXiv preprint arXiv:2509.22917*.

[Yang et al., 2018] Yang, Y., Feng, C., Shen, Y., and Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215.

[Yao et al., 2025] Yao, L., Wang, Y., Zhang, Y., Liu, M., and Chau, L.-P. (2025). Gaussian-Cross: Cross-modal self-supervised 3d representation learning via gaussian splatting. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 6500–6509.

[Youn et al., 2025] Youn, S., Lee, S., Kim, G., Kwon, W., Bae, S.-H., and Oh, J. (2025). SUCCESS-GS: Survey of compactness and compression for efficient static and dynamic gaussian splatting. *arXiv preprint arXiv:2512.07197*.

[Zhang et al., 2024] Zhang, B., Cheng, Y., Yang, J., Wang, C., Zhao, F., Tang, Y., Chen, D., and Guo, B. (2024). GaussianCube: A structured and explicit radiance representation for 3d generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Zhang et al., 2016] Zhang, H., Reddi, S. J., and Sra, S. (2016). Riemannian svrg: Fast stochastic optimization on riemannian manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4592–4600.

APPENDIX A: REPRODUCIBILITY AND CONFIGURATION RECORD

This appendix records the computational path that supports the primary v48 claims. It is intentionally more explicit than the Methods chapter because the implementation contains historical and optional branches. The accompanying source package includes the metric JSON files, ablation summaries, configuration-bearing logs, figures, and plotting script used by this manuscript.

A.1 Primary Training Command

The baseline training log begins with the following command:

Primary v48 baseline command recorded in the packaged log.

```
python Trainer.py \
  -train_path_base z_drive_split_train \
  -test_path_base z_drive_split_holdout \
  -num_epochs 4 \
  -seed 20260627 \
  -output_dir ablation_v48/runs/baseline \
  -geometric_loss_strategy bc_chamfer \
  -use_latentz_decode true \
  -use_flow_matching true \
  -ot_k_candidates 32 \
  -ot_sinkhorn_iters 15
```

The run used one NVIDIA RTX 4090, batch size 2, and standard 32-bit computation; mixed precision was disabled. The training directory contained 34,431 objects and the held-out directory contained 1,817 objects. With batch size 2, the log records 17,216 training batches per epoch.

A.2 Data Partition and Evaluation Sample

The v48 split was generated with seed 20260627 and contains disjoint object paths. The post-hoc generalization summary evaluates 128 training objects and 128 held-out objects with the same evaluation seed, renderer, four views, and 256×256 resolution. The baseline and ablation evaluation summaries reference the same held-out protocol.

The held-out directory was also used as the Lightning validation source during training, with `limit_val_batches=5`. Checkpoint selection therefore observes a small subset of that partition. This is why the thesis calls it a held-out-from-training partition rather than a sealed test set.

A.3 Core Model Configuration

A.4 Active Loss Configuration

The total active objective is

$$\mathcal{L} = 1.0\,\mathcal{L}_{\text{geo}} + 1.0\,\mathcal{L}_{\text{render}} + 0.3\,\mathcal{L}_{\text{latent}} + 0.05\,\mathcal{L}_{\text{org}}. \tag{A.1}$$

The geometric branch uses `bc_chamfer` with 32 candidates. The flow loss has internal weight 1.0 inside the latent branch. The render loss uses five training views and a five-level pyramid. The perceptual training term is disabled, even though LPIPS is reported as an evaluation metric.

The following terms have zero static weight or are disabled in the primary run: diversity, feature decorrelation, hierarchical WAE, volumetric IoU, T-Net regularization, count prediction, adaptive uncertainty weighting, variational KL, and stochastic decoder noise. Their presence in the codebase should not be read as evidence that they influenced the reported checkpoint.

Table A.1: Core v48 architecture settings extracted from the baseline log (part 1 of 2).

Setting	**Recorded value**
Input cardinality	Up to 3,200 primitives with validity masks
Primitive attributes	Position 3, scale 3, quaternion 4, color 3, opacity 1
PCA alignment	Enabled
Cuboid normalization	Enabled
Hash encoding	16 levels, 2 features per level, 32 features total; nominal resolutions 16–2,048
Feature grouping	Enabled, 256 groups
Encoder transformer	4 layers, 4 heads
Reported attention module	Full gated Bhattacharyya/content attention after trainer replacement
Global latent	128 dimensions
Regional latent	8 regions $\times$ 64 dimensions
Local latent	8 regions $\times$ 8 locals $\times$ 32 dimensions
Concatenated hierarchy	2,688 dimensions before projection
Decoder code	512 dimensions
Decoder expansion	64 seeds $\times$ 50 children = 3,200 slots
Cross-attention decoder	Disabled
Learned count prediction	Disabled; count loss weight 0
Variational decoder	Disabled
Noise injection	Disabled

Table A.2: Core v48 optimization and numerical settings extracted from the baseline log (part 2 of 2).

Setting	**Recorded value**
Flow network	6 layers, 1,024 hidden dimensions, 256-dimensional time embedding
Flow noise floor	$\sigma_{\min} = 10^{-4}$
Optimizer	AdamW
Peak learning rate	10^{-3}
Final learning-rate ratio	0.1
Warmup	First 5% of training
Weight decay	10^{-8}
Gradient clipping	Norm 1.0
Training epochs	4
Training seed	20260627

A.5 Checkpoint and Metric Provenance

Table A.3: Packaged artifacts supporting the main numerical claims.

Artifact	Role
`data/gap_v48_baseline.json`	Train/held-out means, medians, standard deviations, evaluation seed, view count, resolution, and checkpoint path
`evidence/ablation_results.json`	Baseline and component-removal metrics, standard deviations, deltas, and approximate hierarchy flag
`evidence/gate_objective_comparison.json`	Matched-budget and long-budget geometric-gate summaries
`evidence/status.json`	Run status and exact/approximate annotations
`evidence/baseline_training.log`	Command, model configuration, optimizer settings, active switches, checkpoint metrics, and hardware
`data/intrinsic_dim_full.json`	Latent covariance spectra and participation-ratio summaries over 2,048 objects
`data/generalization_gap.json`	Historical v46 diagnostic metrics and contamination metadata

The primary checkpoint path recorded in the v48 metric summary is

```
ablation_v48/runs/baseline/
weights_mark28_lightning_epoch=02_val/total_loss=13.18.ckpt
```

Ablation checkpoints are selected by the same validation-loss procedure. The no-geometric-attention checkpoint has a lower recorded total training objective than the baseline while worse external reconstruction metrics. This is not contradictory because the checkpoint objective combines several weighted terms and is not numerically identical to PSNR, SSIM, LPIPS, or center Chamfer.

A.6 Building the Thesis

From the source-package root, the intended build command is

```
latexmk -pdf thesis.tex
```

The plotting script can regenerate the derived JSON-based figures with

```
python3 scripts/make_figures.py
```

The script writes the derived figures from the packaged data summaries. The PDF also contains TikZ diagrams compiled directly by LaTeX.

A.7 Reproducibility Boundary

The package is sufficient to audit the manuscript's numerical summaries and rebuild the thesis. It is not a complete release of the training code, dataset, or checkpoint tensors. Exact computational reproduction therefore requires the original GLEA repository, ShapeSplat-derived paths, renderer dependencies, and saved checkpoints in addition to this document package. The appendix distinguishes documentary reproducibility – tracing a claim to recorded artifacts – from full executable reproducibility – rerunning the model from raw data.

APPENDIX B: SUPPLEMENTARY DIAGNOSTICS AND HISTORICAL REGIMES

This appendix preserves useful evidence from earlier project variants without allowing it to carry the primary v48 generalization claim. The material is included because it illuminates decoder capacity, latent behavior, gate dynamics, and the consequences of split contamination. Configuration differences are stated explicitly.

B.1 Late-Epoch 3,200-Slot Reconstruction Snapshots

The `version_28` run produced late-epoch TensorBoard-style reconstruction snapshots with the same nominal 3,200 input/output maximum, 512-dimensional code, and 128/64/32 global, regional, and local latent dimensions used by the later architecture. The images are qualitative rather than metric-matched to v48.

The validation panel preserves object pose, silhouette, and broad color layout across views. The training panel shows that the 3,200-slot decoder can represent a more articulated form. Fine details remain smoothed, and the snapshots lack the standardized four-view metric protocol used by v48.

B.2 Selected Additional Reconstruction Snapshots

Figure B.2 uses the strongest supplied log snapshots for print-scale qualitative inspection. The first block is marked validation in the source image; the remaining two are marked training. The examples were selected for structural diversity and clear multi-view comparison rather than by a metric-ranked, pre-registered procedure. They are therefore visual diagnostics only and do not replace the primary v48 held-out metrics.

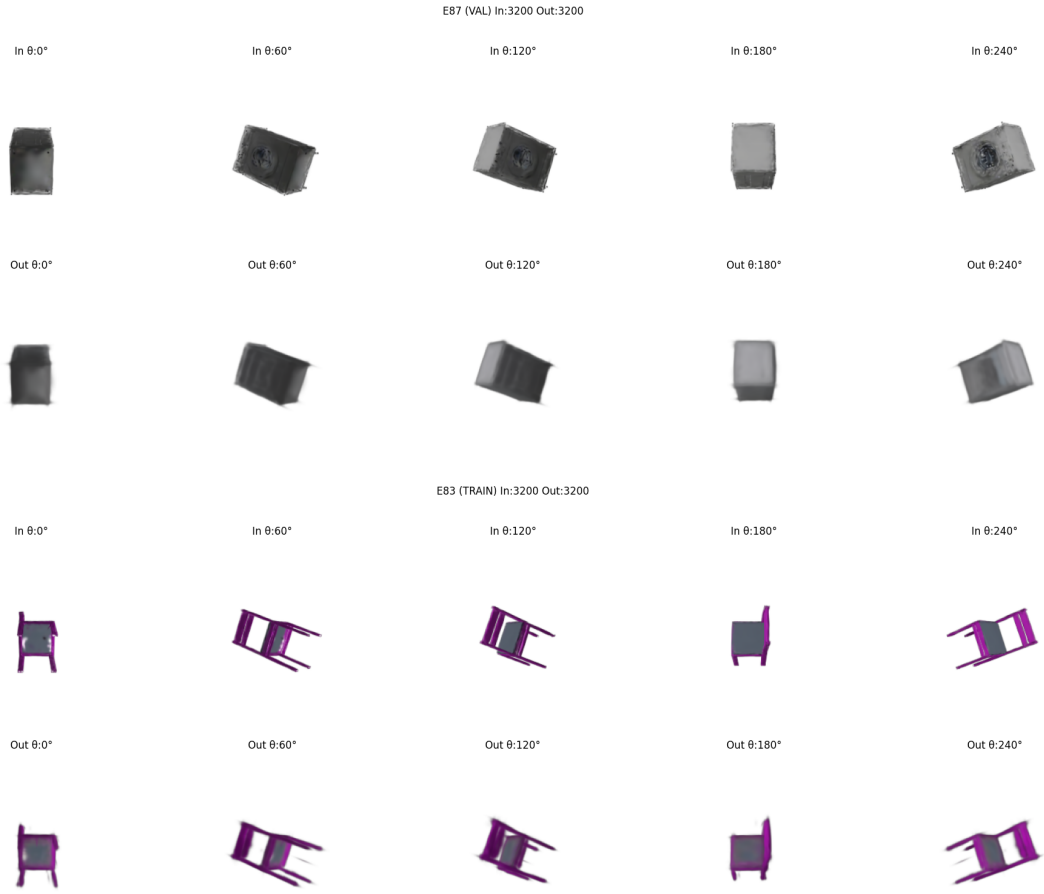

Figure B.1: Historical late-epoch `version_28` snapshots. Top: epoch-87 validation reconstruction from the recorded validation source. Bottom: epoch-83 training reconstruction of a more complex chair-like object. These images illustrate convergence and capacity, not a clean train–test comparison.

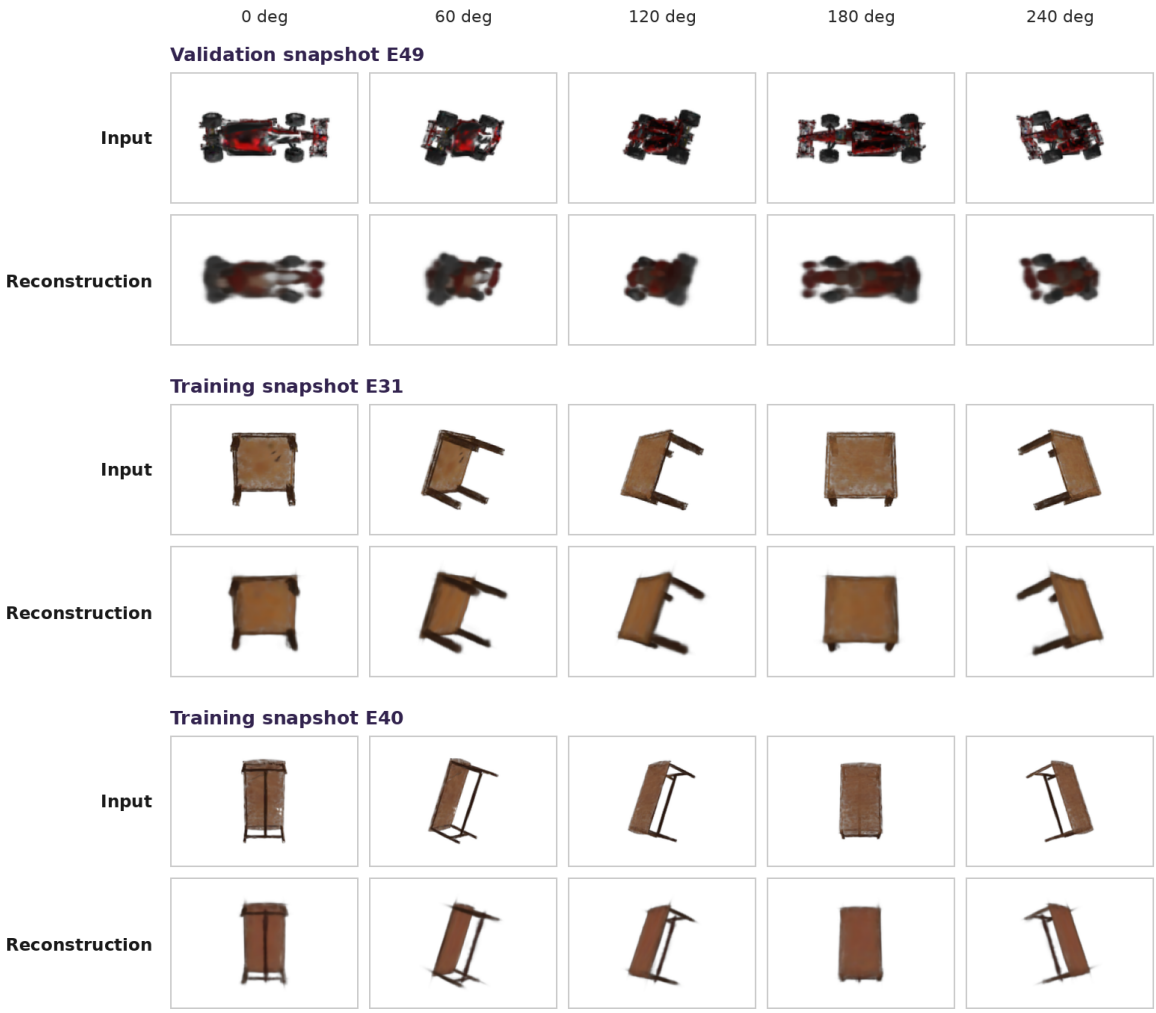

Figure B.2: Selected model-output gallery from the supplied reconstruction logs. Each block shows five input views above the corresponding latent-only reconstructions. The validation racecar tests a more articulated shape, while the two training examples show furniture with planar surfaces and thin supports. Cropping removes empty margins and enlarges the logged renderings without altering their content. These snapshots are not used in the v48 quantitative generalization claim.

B.3 The Contaminated v46 Regime

The v46 regime contains a longer run and rich latent diagnostics, but four of five nominal validation objects were later found in the training directory. Its aggregate "holdout" metrics cannot establish generalization. Table B.1 reports the numbers only as historical diagnostics.

The one extra object is not enough to estimate a population generalization gap. The v46

Table B.1: Historical v46 reconstruction diagnostics. The nominal holdout is contaminated and must not be interpreted as a test set.

Group	Center Chamfer	PSNR	SSIM	LPIPS
Training, $n = 64$	0.000764	27.67	0.966	0.069
Nominal holdout, $n = 64$	0.000627	28.29	0.967	0.066
One extra object, $n = 1$	0.001077	27.61	0.953	0.090

numbers are retained because they help contextualize long-run gate trajectories and latent analyses that were not rerun under every later configuration.

B.4 Geometric Gate Dynamics Across Objectives and Budgets

Each geometric-attention layer has a gate $\lambda = \sigma(\ell)$ that mixes the geometric and content scores. At a matched four-epoch budget, the volumetric-IoU baseline ends with mean $\lambda = 0.359$, while the BC–Chamfer baseline ends with mean $\lambda = 0.233$. The latter therefore does not preserve or increase the gate relative to the older objective. The geometric-attention ablation is nevertheless load-bearing at the same budget.

The long v46 run reaches mean $\lambda = 0.095$ after approximately 1.7 million steps, with the first layer near 0.007. That long-run value is not directly comparable to the four-epoch v48 ablation because both objective and budget differ.

The gate and ablation answer different questions. A gate measures how a trained mixed model allocates weight between available branches. An ablation measures the performance of a model trained without one branch. A small gate can still correspond to a useful signal if it is needed only in specific examples or layers, if it improves early optimization, or if its effect is not substitutable by the content branch.

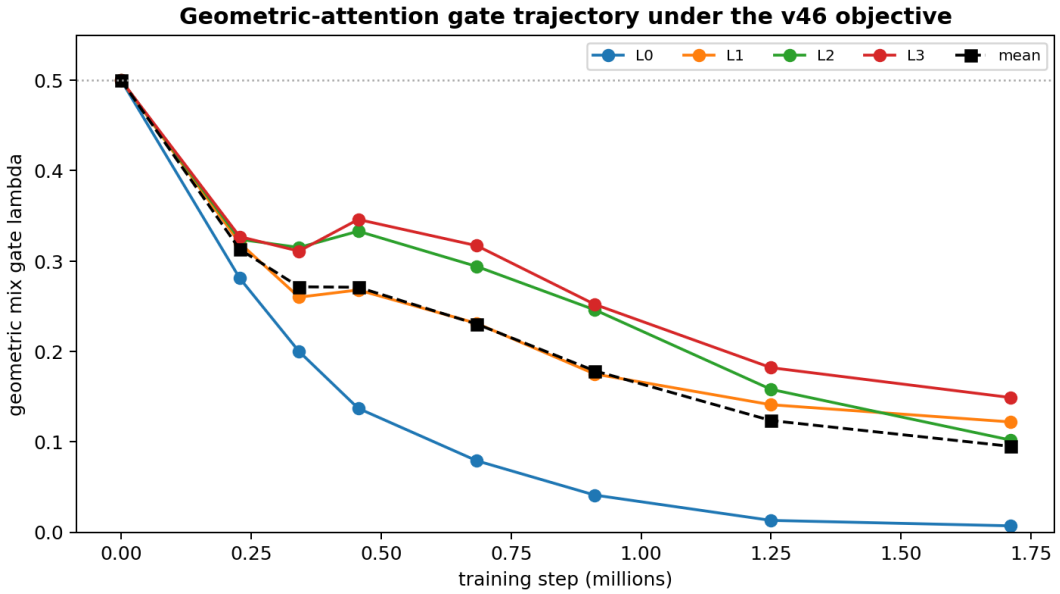

Figure B.3: Historical long-budget v46 geometric-gate trajectory. The first layer nearly removes the geometric branch, while later layers retain a small residual. The figure should be interpreted together with the matched-budget table in Chapter 5.

B.5 Empirical-Posterior Versus Naive Prior Sampling

The project includes a diagnostic in which decoder-input codes are sampled near their empirical distribution and compared with naive samples from $\mathcal{N}(0, I)$. Empirical-posterior samples remain near the code support observed during training and yield more coherent object-like forms.

The correct conclusion is a wiring requirement. A latent regularizer is generative only when the regularized or transported variable is exactly the variable consumed by the decoder. If a separate hierarchy or projection bypasses that variable, successful empirical-posterior sampling says little about unit-prior generation. Future experiments should log the exact tensor at the flow endpoint and at the decoder input, verify equality by construction, and evaluate transported samples directly.

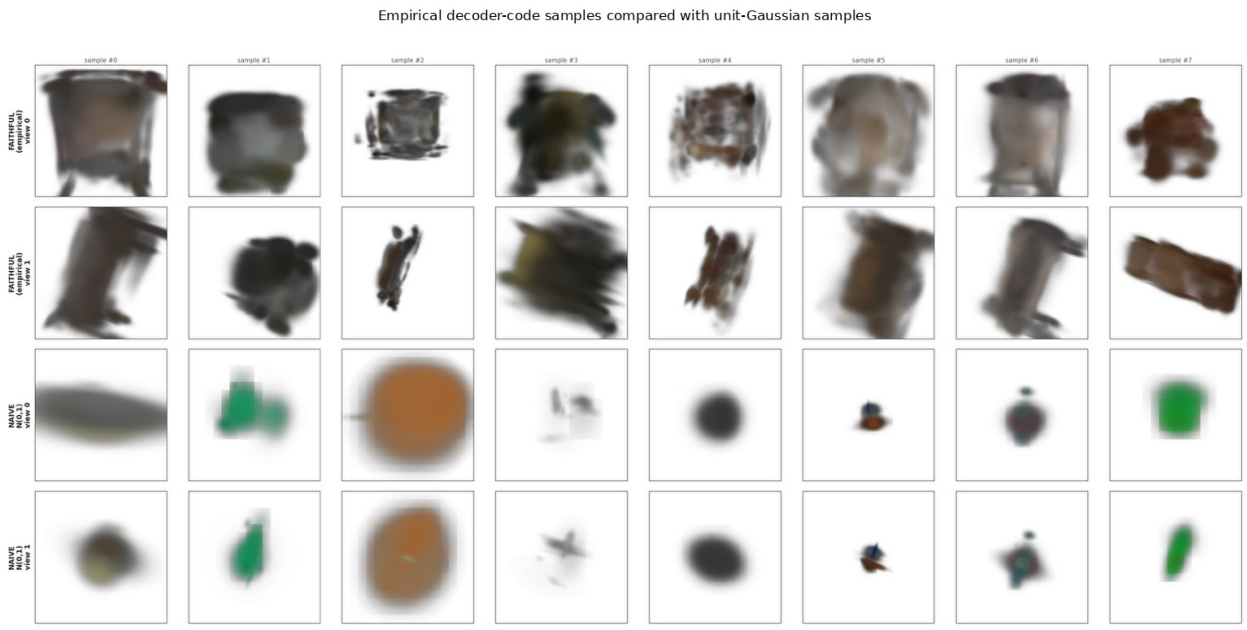

Figure B.4: Historical prior diagnostic. Empirical-posterior samples remain near observed decoder-input codes; naive unit-Gaussian samples can fall outside that support. The image does not establish unconditional generation through the current flow.

B.6 Why the Historical Material Remains Useful

Discarding contaminated or mismatched experiments entirely would erase information about model behavior, while mixing them with the main result would overstate the evidence. The appendix provides a middle path. The v28 images and selected log gallery support qualitative capacity claims, the v46 gates support a hypothesis about branch allocation over long training, and the prior figure exposes an architectural dependency. None is used to estimate the primary v48 generalization gap.

APPENDIX C: MATHEMATICAL AND NUMERICAL NOTES

This appendix collects derivations and implementation considerations that are useful for auditing the geometric kernel but would interrupt the main architectural narrative.

C.1 Bhattacharyya Coefficient for Multivariate Gaussians

Let

$$p_i(x) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\Big[-\frac{1}{2}(x-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(x-\boldsymbol{\mu}_i)\Big]. \tag{C.1}$$

The coefficient is

$$\mathrm{BC}(p_1, p_2) = \int_{\mathbb{R}^d} \sqrt{p_1(x)p_2(x)}\, dx. \tag{C.2}$$

Multiplying the square-root densities yields a quadratic form in x. Completing the square converts the integral to the normalization constant of another Gaussian. With

$$\overline{\boldsymbol{\Sigma}} = \frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}, \qquad \boldsymbol{\delta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2, \tag{C.3}$$

the resulting coefficient is

$$\mathrm{BC}(p_1, p_2) = \frac{|\boldsymbol{\Sigma}_1|^{1/4}|\boldsymbol{\Sigma}_2|^{1/4}}{|\overline{\boldsymbol{\Sigma}}|^{1/2}} \exp\Big[-\frac{1}{8}\boldsymbol{\delta}^\top \overline{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\delta}\Big]. \tag{C.4}$$

Taking the negative logarithm gives the sum of a mean term and a covariance term,

$$D_{\mathrm{B}} = \underbrace{\frac{1}{8}\boldsymbol{\delta}^\top \overline{\boldsymbol{\Sigma}}^{-1}\boldsymbol{\delta}}_{\text{mean separation}} + \underbrace{\frac{1}{2}\log \frac{|\overline{\boldsymbol{\Sigma}}|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}}_{\text{covariance mismatch}}. \tag{C.5}$$

Both terms are nonnegative. The determinant term is zero when the covariances are equal. The mean term is zero when the centers coincide.

C.2 Verification of the Equal-Center Example

For

$$\boldsymbol{\Sigma}_1 = \mathrm{diag}(1, 1, 0.01), \qquad \boldsymbol{\Sigma}_2 = \mathrm{diag}(0.01, 0.01, 1), \tag{C.6}$$

the equal means make the exponential term one. The determinants are

$$|\boldsymbol{\Sigma}_1| = 0.01, \qquad |\boldsymbol{\Sigma}_2| = 0.0001, \tag{C.7}$$

and

$$\overline{\boldsymbol{\Sigma}} = \mathrm{diag}(0.505, 0.505, 0.505), \qquad |\overline{\boldsymbol{\Sigma}}| = 0.505^3. \tag{C.8}$$

Therefore

$$\mathrm{BC} = \frac{0.01^{1/4} 0.0001^{1/4}}{(0.505^3)^{1/2}} \approx 0.0881. \tag{C.9}$$

The qualitative point is unchanged: center distance reports exact coincidence while Gaussian overlap reflects substantial covariance disagreement.

C.3 Positive-Semidefinite Kernel Proof

For normalized densities p and q, define $\Phi(p) = \sqrt{p} \in L^2$. Then

$$\langle \Phi(p), \Phi(q) \rangle_{L^2} = \int \sqrt{p(x)q(x)}\, dx = \mathrm{BC}(p, q). \tag{C.10}$$

For any finite set of densities $p_1, \ldots, p_n$ and real coefficients $a_1, \ldots, a_n$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{BC}(p_i, p_j) = \sum_{i,j} a_i a_j \langle \Phi(p_i), \Phi(p_j)\rangle_{L^2} \tag{C.11}$$

$$= \left\langle \sum_i a_i \Phi(p_i), \sum_j a_j \Phi(p_j) \right\rangle_{L^2} \tag{C.12}$$

$$= \left\| \sum_i a_i \Phi(p_i) \right\|_{L^2}^2 \geq 0. \tag{C.13}$$

Thus the Gram matrix $K_{ij} = \mathrm{BC}(p_i, p_j)$ is positive semidefinite. The proof applies to normalized densities generally, not only Gaussians.

C.4 Local Relation to Fisher–Rao Geometry

Let p_θ be a smooth regular statistical model, and consider a nearby distribution $p_{\theta+d\theta}$. Expanding the squared Hellinger distance around $d\theta = 0$ gives

$$H^2(p_\theta, p_{\theta+d\theta}) = \frac{1}{8} d\theta^\top I(\theta) d\theta + O(\|d\theta\|^3), \tag{C.14}$$

where $I(\theta)$ is the Fisher information matrix. Since $H^2 = 1 - \mathrm{BC}$ and $-\log(1-u) = u + O(u^2)$,

$$D_{\mathrm{B}}(p_\theta, p_{\theta+d\theta}) = \frac{1}{8} d\theta^\top I(\theta) d\theta + O(\|d\theta\|^3). \tag{C.15}$$

The scale factor depends on the divergence convention. The key statement is local second-order agreement. The equation does not identify Bhattacharyya distance with the finite-separation Fisher–Rao geodesic distance.

C.5 Stable Computation

Direct determinant and inverse operations are numerically fragile. A stable implementation should operate in log space and use matrix factorizations. For a symmetric positive-definite matrix $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$ from Cholesky factorization,

$$\log |\mathbf{A}| = 2 \sum_k \log L_{kk}, \tag{C.16}$$

and the quadratic form should be computed by triangular solves rather than forming $\mathbf{A}^{-1}$ explicitly. The mean term becomes

$$\boldsymbol{\delta}^\top \mathbf{A}^{-1} \boldsymbol{\delta} = \|\mathbf{L}^{-1}\boldsymbol{\delta}\|_2^2. \tag{C.17}$$

Predicted scales can approach zero, so an implementation should regularize both individual and averaged covariances. A generic rule is

$$\mathbf{A}_{\text{safe}} = \frac{\mathbf{A} + \mathbf{A}^\top}{2} + \max\left(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} \frac{\operatorname{tr}(\mathbf{A})}{d}\right) \mathbf{I}. \tag{C.18}$$

Symmetrization removes small antisymmetric floating-point errors; the absolute floor protects near-zero matrices; and the relative floor scales with the primitive. Failed Cholesky factorizations should be surfaced in diagnostics even when a fallback keeps training finite.

C.6 Sparse Screening and Approximation Error

Top-k screening is computationally useful only if the candidate score has high recall for the true low-cost full-Bhattacharyya neighbors. Using the mean component as a screen can miss pairs whose centers are farther apart but whose broad covariances yield substantial overlap. The approximation error is therefore data dependent.

A rigorous screening study would compute dense full costs for a manageable subset and report

$$\text{Recall@}\, k = \frac{1}{N} \sum_{i} \mathbf{1}[j_i^\star \in \mathcal{N}_k(i)]\,, \tag{C.19}$$

where $j_i^\star$ is the true dense minimum and $\mathcal{N}_k(i)$ is the screened candidate set. It should also compare sparse and dense loss values, gradients, wall-clock time, and memory. The current thesis documents the approximation and its active $k = 32$ setting but does not claim an error bound.

C.7 Attention Score Calibration

The geometric and content branches have different natural scales. The reported score has the form

$$s_{ij} = \lambda \log(\mathrm{BC}_{ij} + \epsilon) + (1 - \lambda) \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d}}, \tag{C.20}$$

followed by row-wise softmax. The gate $\lambda \in (0, 1)$ controls interpolation between scores, but it does not guarantee equal variance or equal gradient magnitude. A small λ can have a large effect if the geometric logits are more dispersed, and a large λ can have little effect if they are nearly constant within a row.

For that reason, gate values should be interpreted together with logit distributions and ablations. Future instrumentation should record per-layer means, standard deviations, entropy, and gradient norms for both branches. Such measurements would distinguish numerical scale effects from genuine model preference.