Chorus: Multi-Teacher Pretraining for Holistic 3D Gaussian Scene Encoding

Supplementary Material

Contents

A. Method Details

A.1. Teacher-Specific Contrastive Loss

When the source dataset for a teacher t provides semantic or instance labels, we optionally add a compact contrastive regularizer $\mathcal{L}_{\text{con}}^{(t)}$ on top of the per-Gaussian matching losses. This follows the aggregated InfoNCE formulation of SceneSplat [9], but is applied independently per teacher and per supervision type (semantic & instance).

Let $\hat{F}^{(t)} \in \mathbb{R}^{N\times d_t}$ denote the predicted features for teacher t, where N is the number of Gaussians in the scene. We assume that the Gaussians are partitioned into a collection of labeled groups $\{\mathcal{G}_g\}_{g\in\mathcal{C}^{(t)}}$, where g indexes either a semantic class (for SigLIP2) or an instance (for PE-Spatial), and $\mathcal{C}^{(t)}$ is the set of valid groups for teacher t.

For each group $g \in \mathcal{C}^{(t)}$ with sufficiently many Gaussians, we randomly split its elements into two disjoint subsets $\mathcal{G}_g^A, \mathcal{G}_g^B \subset \mathcal{G}_g$ and compute pooled features

$$\bar{F}_g^A = \text{mean}\{\hat{F}_i^{(t)} : i \in \mathcal{G}_g^A\},\ \bar{F}_g^B = \text{mean}\{\hat{F}_i^{(t)} : i \in \mathcal{G}_g^B\}. \tag{1}$$

Stacking these vectors over all groups yields matrices $\bar{F}^A, \bar{F}^B \in \mathbb{R}^{|\mathcal{C}^{(t)}|\times d_t}$, which we ℓ_2-normalize row-wise and use to form bidirectional InfoNCE logits:

$$Z^A = \bar{F}^A(\bar{F}^B)^\top/\tau^{(t)},\ Z^B = \bar{F}^B(\bar{F}^A)^\top/\tau^{(t)}, \tag{2}$$

where $\tau^{(t)}$ is a learnable temperature. Each row of Z^A and Z^B corresponds to a query group, and the diagonal entries are the positive pairs (same group across the A/B split). The teacher-specific contrastive loss is then

$$\mathcal{L}_{\text{con}}^{(t)} = \frac{1}{2|\mathcal{C}^{(t)}|}\sum_{X\in\{A,B\}}\sum_{g\in\mathcal{C}^{(t)}} -\log\frac{\exp\left(Z_{g,g}^X\right)}{\sum_{g'\in\mathcal{C}^{(t)}}\exp\left(Z_{g,g'}^X\right)}. \tag{3}$$

We use the same formulation in Eq. (3) for different supervision signals:

- **SigLIP2 (semantic).** Here, each group g is a semantic class c, and $\mathcal{G}_c$ is the set of Gaussians assigned to it. We thus pool class-wise features $\bar{F}_c^A, \bar{F}_c^B$ from SigLIP2-predicted features $\hat{F}^{(t)}$ and apply Eq. (3) over $\mathcal{C}^{(t)} = \{c\}$.
- **PE-Spatial (instance).** Here, each group g is an instance k, and $\mathcal{G}_k$ collects Gaussians belonging to it. We pool instance-wise means $\bar{F}_k^A, \bar{F}_k^B$ and apply the same loss over $\mathcal{C}^{(t)} = \{k\}$, encouraging distinct instances to occupy well-separated regions in the embedding space.

In both cases, $\mathcal{L}_{\text{con}}^{(t)}$ is enabled only after the warm-up epochs of the teacher projector so that the features first stabilize under the regression objectives before being refined by contrastive separation.

A.2. Rendering-Based Adaptation

We further explain the pose sampling and selection pipeline. The primary requirement for sampling camera poses is finding valid locations within free space. For Gaussian Splats, this can be achieved by computing the signed distance field through rendered depth map fusion. For convenience, we leverage the pre-computed 2D occupancy maps provided by the InteriorGS dataset. To avoid poses that are too close to obstacles (where the occlusion map may contain errors and the resulting view is less informative), we first dilate the occlusion mask as shown in Fig. A. Next, we sample the 2D camera locations within the dilated occupancy map. To address issues that occlusion map incorrectly marks areas outside the room as free space, we ignore all poses that fall outside the XY projection boundary of the Gaussian splats.

Determining the optimal camera height (Z-coordinate) is non-trivial, as simply using the mean or median height ignores variations like those found in multi-story environments. We propose a label-guided height decision method:

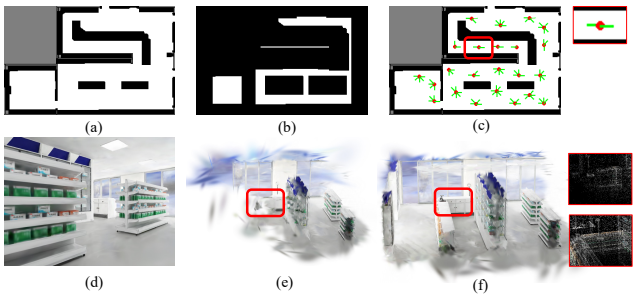


Figure A. **Visualization of Pose Sampling and Visibility Crop.** (a) The 2D occupancy map. (b) The dilated occupancy map. (c) The sampled locations and view directions, with a zoomed-in view showing that viewpoints near obstacles are excluded. (d) The 2D rendered image for the given pose. (e) The naive visibility crop for the given pose. (f) The box-augmented visibility crop, with a zoomed-in image showing that more complete 3D structure is maintained.

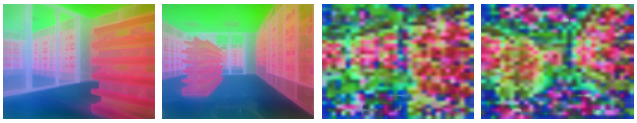


Figure B. **Cross-View 2D Teacher Feature Visualization.** PCA visualizations of DINOv3 features in (a) View 1 and (b) View 2, compared to SigLIP2 features in (c) View 1 and (d) View 2. DINOv3 generally demonstrates good 3D consistency across views. In contrast, SigLIP2 patch features are less smoothed and consistent across views.

For a sampled XY position, we query labeled points within a 2-meter radius, prioritizing furniture labels, then floor labels, and finally ceiling labels. We use the height of these labeled points to determine the camera's Z-location (specifically, ± 0.5m relative to furniture, $+1.5$m relative to the floor, and -1m relative to the ceiling), thereby avoiding viewpoints that are either too low or too high.

We sample eight uniform view directions from the given location. For each direction, we project the center ray onto the XY plane and check if this ray intersects any obstacles within a 2-meter radius. If an intersection occurs, we disregard that direction. As illustrated in Fig. A, only view directions parallel to the narrow corridor are retained.

Next, we determine the visible Gaussians by selecting the splats that have valid accumulated transmittance during rasterization. As illustrated in Fig. A, this visibility is linked primarily to the 2D rendering rather than the underlying 3D scene structure. If we supervise the network using only these visible splats, critical geometry (such as the desk structure) can be missed. To address this, we compute the minimum 2D bounding box that encloses all visible splats and augment it with an additional 0.2m margin. This results in a complete and larger scene region for the subsequent Chorus encoder forward pass.

The enclosed splats within the augmented bounding box are saved. Next, we find three additional poses from the rest

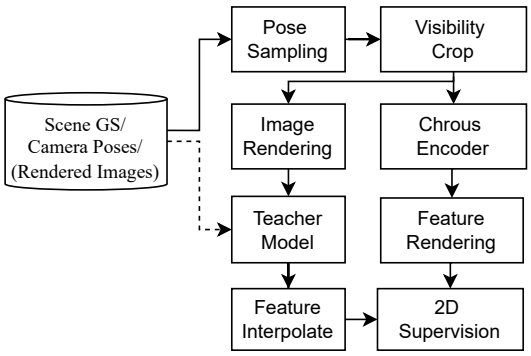


Figure C. **Rendering-Based Adaptation Diagram.**

pose set that exhibit the largest overlap ratio with this specific bounding box. These selected cameras are then used in the subsequent training step. This strategy allows us to perform a forward pass on a single cropped 3D scene and supervise it using up to four images from different viewpoints, which effectively promote the inter-view encoding consistency, as analyzed in Fig. B.

Following the above preprocessing, we present the complete training pipeline. Fig. C shows the high-level workflow and Algorithm 1 provides the details. For each batch, we first select multiple scenes. Within each selected scene, we sample one main camera pose and then sample $n-1$ additional camera poses for rendering supervision. We then merge the visibility masks from these poses to form a co-visibility mask. As demonstrated in Tab. B, our overlap ratio ranking strategy helps prevent the inclusion of excessive Gaussians during merging.

The pipeline then proceeds in two parallel branches. First, we render the co-visible Gaussians to synthesize batch images for each pose (or utilize pre-rendered views). These images are fed into the 2D teacher model to extract features, which are subsequently interpolated to the target resolution. Simultaneously, the co-visible Gaussians are forwarded to the Chorus encoder to obtain per-Gaussian features. These per-Gaussian features are then rendered back onto the 2D plane to obtain the predicted 2D feature maps $\hat{F}_j$ for the sampled poses. Finally, the loss is computed between the predicted features $\hat{F}_j$ and the teacher features $\tilde{F}_j$.

A.3. 3DGS-Aware Augmentations

We now detail the two 3DGS-aware augmentations introduced in main paper, which are designed to perturb splat parameters in ways that are approximately rendering-preserving and consistent with observed 3DGS optimization dynamics.

Rendering-equivalent perturbation. Let $R(\mathbf{q}_i) \in \mathrm{SO}(3)$ be the rotation matrix of quaternion $\mathbf{q}_i$, and the world-space

Algorithm 1 Rendering-Based Adaptation

Input: Gaussian scene $G = \{G_i\}_{i=1}^{N}$, training poses $\{P_j\}_{j=1}^{M}$ and views $\{I_j\}_{j=1}^{M}$ (optional) of this scene, overlap set size n, pose visibility lookup table $\{V_j\}_{j=1}^{M}$.
Models: 2D teacher model t and Chorus encoder g_θ.
Output: Loss for Rendering-Based Adaptation.
Step 1: Pose Sampling and Scene Crop
Uniformly sample $P_s \sim \mathcal{U}(\{P_j\}_{j=1}^{M})$
Given lookup table, calculate overlap counts between P_s and all other poses P_g:

$$E(P_s, P_g) = |V_s \cap V_g| \quad \forall P_g \in \mathcal{P}, g \neq s$$

Define selection probability $\mathbf{Prob}(P_g)$ proportional to overlap:

$$\mathbf{Prob}(P_g) = \frac{E(P_s, P_g)}{\sum_{k \neq s} E(P_s, P_k)}$$

Sample $n-1$ overlap poses Pair_s to form the batch $\mathcal{B}$:

$$\mathcal{B} \leftarrow \{P_s\} \cup \text{Sample}(\mathcal{P} \setminus \{P_s\}, n-1, \mathbf{Prob})$$

Step 2: Visibility Crop
Create the co-visibility mask of the batch with $\mathbf{V} = \bigcup_{V_i \in \mathcal{B}} V_i$, and then crop the scene $G' = \text{Crop}(G, \mathbf{V})$
Step 3: Image Rendering
for $P_j \in \mathcal{B}$ **do**
    Obtain RGB image $\tilde{I}_j$ (Render G' via rasterization or retrieve pre-rendered I_j).
end for
Step 4: Chorus Encoding
Encode the cropped scene G' to get $g_\theta(G')$.
Step 5: Teacher Supervision
for $P_j \in \mathcal{B}$ **do**
    Teacher: Run inference and extract teacher 2D features $\tilde{F}_j^{(t)}$ from image $\tilde{I}_j$.
    Chorus: Render predicted feature map using $g_\theta(G')$:

$$\hat{F}_j = \text{Render}(G', g_\theta(G'), P_j)$$

    , with resolution $H_{\hat{F}}, W_{\hat{F}}$
end for
Step 6: Loss calculation
To align the feature we interpolate the teacher feature resolution to $H_{\hat{F}}$ and calculate loss $\mathbf{L} = \mathcal{L}(\hat{F}_j^{(t)}, \text{Interp}(\tilde{F}_j^{(t)}))$.
Return: Rendering-Based Adaptation loss $\mathbf{L}$

covariance of the i-th splat be

$$\Sigma_i \;=\; R(\mathbf{q}_i)\,\mathrm{diag}(\mathbf{s}_i^2)\,R(\mathbf{q}_i)^\top. \tag{4}$$

Motivated by *exploration steps* in 3DGS-MCMC [7], where low-contribution splats are perturbed or relocated without harming the converged quality, we add small, zero-mean displacements to the ellipsoid centers and modulate their magnitude by the splat's opacity. Concretely, for each splat we sample

$$\mathbf{x}_i' \;=\; \mathbf{x}_i \;+\; \eta\, w(\alpha_i)\,\Sigma_i \boldsymbol{\xi}_i, \quad \boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, I). \tag{5}$$

where $\eta > 0$ is a small scale and $w(\alpha)$ is a monotonic weight emphasizing *low-opacity* (i.e., less influential or certain) splats. In practice, we use the sharp logistic as in [7]

$$w(\alpha) = \sigma\big(k(\tau - \alpha)\big), \quad k = 100,\ \tau = 0.005, \tag{6}$$

so that low-opacity splats (small α) are perturbed more strongly than high-opacity ones.

Why this preserves appearance in expectation. Let $\mathbf{C}(\mathbf{u}; \mathbf{X})$ denote the rendered color at pixel $\mathbf{u}$ given centers $\mathbf{X} = \{\mathbf{x}_i\}$, and let $\mathbf{X}' = \{\mathbf{x}_i'\}$ be the perturbed centers. A second-order Taylor expansion of $\mathbf{C}$ around $\mathbf{X}$ yields

$$\mathbb{E}_{\boldsymbol{\xi}}\big[\mathbf{C}(\mathbf{u}; \mathbf{X}')\big] \approx \mathbf{C}(\mathbf{u}; \mathbf{X}) \;+\; \sum_i \nabla_{\mathbf{x}_i} \mathbf{C}(\mathbf{u}; \mathbf{X})^\top \mathbb{E}[\Delta \mathbf{x}_i] \;+\; \frac{1}{2} \sum_i \mathrm{tr}\Big(H_i(\mathbf{u})\,\mathrm{Cov}[\Delta \mathbf{x}_i]\Big), \tag{7}$$

where $H_i(\mathbf{u})$ is the Hessian of $\mathbf{C}(\mathbf{u}; \mathbf{X})$ *w.r.t.* $\mathbf{x}_i$.

From Eq. (5), we have $\mathbb{E}[\Delta \mathbf{x}_i] = \mathbf{0}$ and, using that Σ_i is symmetric,

$$\mathrm{Cov}[\Delta \mathbf{x}_i] = \eta^2 w(\alpha_i)^2 \Sigma_i \mathrm{Cov}[\boldsymbol{\xi}_i] \Sigma_i^\top = \eta^2 w(\alpha_i)^2 \Sigma_i^2. \tag{8}$$

Thus, the *first-order* term in Eq. (7) vanishes, and the remaining *second-order* bias is controlled by $\eta^2 w(\alpha_i)^2 \Sigma_i^2$, *i.e.*, it is (i) small for small η and (ii) concentrated on low-opacity splats due to $w(\alpha_i)$.

Immature-manifold perturbation. RAIN-GS [6] shows that 3DGS optimization follows a coarse-to-fine trajectory: early stages emphasize low-frequency structure, while finer details appear as covariances shrink. We mimic an earlier parameter state by inflating per-splat scales:

$$\mathbf{s}_i' = \mathbf{s}_i \odot \Big(1 + (\beta - 1)\,\omega(\mathbf{s}_i)\Big),\ \beta \sim \mathcal{U}[\beta_{\min}, \beta_{\max}],\ \beta \geq 1, \tag{9}$$

where $\odot$ denotes element-wise multiplication, and $\omega(\mathbf{s}_i) \in [0, 1]$ is a scale-dependent weight that prioritizes small (high-frequency) splats. We define

$$\bar{s}_i \;=\; \tfrac{1}{3} \sum_{k=1}^{3} s_{ik}, \qquad \omega(\mathbf{s}_i) = \exp\Big(-\frac{\bar{s}_i}{\mathrm{median}_j\, \bar{s}_j}\Big), \tag{10}$$

so that splats with below-median size (small $\bar{s}_i$) receive larger inflation factors.

The induced covariance becomes

$$\Sigma_i' = R(\mathbf{q}_i)\,\mathrm{diag}((\mathbf{s}_i')^2)\,R(\mathbf{q}_i)^\top, \tag{11}$$

and under local projection, the 2D covariance satisfies

$$\Sigma_i^{2D} \approx J_i \Sigma_i J_i^\top, \tag{12}$$

where J_i is the Jacobian of the 3D→2D projection at $\mathbf{x}_i$. A 2D Gaussian with covariance Σ^{2D} acts locally as a low-pass filter with Fourier-domain transfer function

$$\exp\Big(-\tfrac{1}{2}\boldsymbol{\kappa}^\top \Sigma^{2D} \boldsymbol{\kappa}\Big), \tag{13}$$

for spatial frequency $\boldsymbol{\kappa}$. Inflating the scales from $\mathbf{s}_i$ to $\mathbf{s}'_i$ therefore increases the local 2D covariance from Σ_i^{2D} to $\Sigma_i'^{2D}$, which is equivalent to applying an additional Gaussian blur with covariance

$$\Delta\Sigma_i^{2D} = \Sigma_i'^{2D} - \Sigma_i^{2D}. \tag{14}$$

This extra blur yields stronger high-frequency suppression while preserving coarse structure, which targets precisely the "immature" yet plausible states encountered early in 3DGS optimization.

B. Pretraining Data

B.1. Processing of Teacher Pseudo-Labels

We employ three 2D teachers during Chorus pretraining: `SigLIP2-so400m-p16-512`, `DINOv3-ViT-L/16`, and `PE-Spatial-L14-448`. Their dense feature maps have dimensionality 1152 (SigLIP2 [18]) and 1024 (DINOv3 [16], PE-Spatial [2]), respectively. For the selected RGB frames used in 3DGS optimization, we run all teachers offline and obtain the corresponding 2D feature maps. The input image resolutions and output feature map sizes for each teacher on each data source are summarized in Tab. A.

Input/Output Resolution	DINOv3		SigLIP2		PE-Spatial	
	input	output	input	output	input	output
ScanNet	1296*968	81*60	640*480	616*456	640*480	45*34
ScanNet++	1752*1168	109*73	876*584	876*584	876*584	62*41
Matterport3D	2560*2048	160*128	640*512	640*512	640*512	45*36

Table A. **Input Image and Output Size During 2D Feature Map Collection for Teachers.**

For SigLIP2 and PE-Spatial, we use the same input resolutions as those used during 3DGS optimization. Differently, DINOv3, due to its high-resolution-adapted training, is applied with high-resolution inputs: on ScanNet and ScanNet++ we use the original high-resolution RGB images, and on Matterport3D we use upsampled images. SigLIP2 feature maps are obtained jointly with SAM2 [14] inference, following Algorithm 1 in [9]. The algorithm outputs feature maps with the same spatial size as its input images; the only exception is ScanNet, where we crop image borders, resulting in slightly smaller feature maps.

To obtain pseudo-labels on the Gaussians for each teacher, we lift the 2D teacher feature maps to the 3DGS, with the feature map spatial resolution being bilinearly interpolated to the image size during 3DGS optimization. For SigLIP2, we adapt the Occam's LGS pipeline [4], projecting each Gaussian into all views with valid features and aggregating the corresponding per-pixel embeddings into a single embedding per Gaussian based on rasterization weights. For DINOv3 and PE-Spatial, we use the adapted LUDVIG [12] codebase to perform uplifting, yielding 1024-D pseudo-labels per Gaussian for these two teachers. These per-Gaussian pseudo-labels form the supervision targets used in our multi-teacher pretraining framework.

B.2. Update in SceneSplat-7K Data

We largely use the 3DGS scenes collected in SceneSplat-7K [9] dataset. The only modification is on the Matterport3D [3] subset, where we observed limited 3DGS quality in its release. SceneSplat-7K is built from the `region_segmentations` version provided by Matterport3D, and the original 3DGS optimization was performed independently for each region. Because RGB images were also splitted per region, each 3DGS was trained with only a small set of images, resulted in reduced multi-view coverage and degraded reconstruction quality.

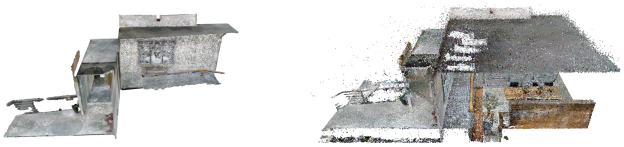
(a) 3DGS Before (b) 3DGS After (c) Scene Geometry

Figure D. **Rendering Example of Updated Matterport3D 3DGS in SceneSplat-7K.**

In this work, we re-process the Matterport3D subset by first optimizing a single 3DGS scene per house using all available views, and then obtaining region-level 3DGS via cropping. Concretely, for each region we compute the axis-aligned bounding box of the corresponding region point clouds and enlarge it by $0.25\,\mathrm{m}$ in all directions before cropping the house-level Gaussians. This simple change yields noticeably better rendering quality for Matterport3D regions, as shown in Fig. D. The example of the original region point clouds and the resulting cropped Gaussians is shown in Fig. E.

B.3. Processing of InteriorGS Data

InteriorGS [17] contains 1,000 indoor scenes in Gaussian splats. Each scene provides instance-level semantic labels, instance IDs, 3D bounding boxes, and an occupancy map. We process InteriorGS into a dataset suitable for our downstream benchmarks, including four main steps: semantic



(a) Region Point Clouds (b) Cropped Gaussians (centers)

Figure E. **Example of Matterport3D Region Point Clouds and the Resulting Cropped Gaussians.**

class mapping, assigning semantic and instance labels to Gaussians, outlier removal, and defining splits.

Semantic class mapping. The original annotation uses 755 fine-grained semantic categories, many of which are overly specific. We group and map these into 72 classes aimed at semantic clarity and hierarchy, while maintain privacy-preserving. Concretely, we remove any labels related to people, discard ambiguous tags such as *"misc","set","object"* by mapping them to the ignore label (-1), and merge long-tail categories into broader concepts while keeping frequently occurring object types separate. The resulting 72 mapped classes each appear in at least 10 instances and together cover over 83.5% of all labeled instances in InteriorGS.

The mapped classes are summarized in the following:

- Structural & fixtures: bathtub, ceiling, ceiling light, column, countertop, door, faucet, floor, lamp, mirror, shower, sink, staircase, toilet, ventilation and heating, wall, window, window accessory.
- Furniture & appliances: bed, bedding, bench, bookshelf, cabinet, chair, clock, computer, computer peripheral, cooktop, decor, desk, dishwasher, dresser, drawer, fan, floor covering, mattress, microwave, monitor, oven, plant, range hood, refrigerator, shelving, sofa, stool, stove, table, television, trash can, vase, wall art, wardrobe, washing machine.
- Goods & supplies (coarse): bakery, beverage, books, container, cookware, electronics accessory, medicine, packaged food, paper goods, produce, snacks and candy, stationery, tableware, tobacco, toiletries, towel, toy, utensils, wine.

Assigning semantic and instance labels to Gaussians. In the original InteriorGS annotations, semantics and instance IDs are attached to 3D bounding boxes, not directly to Gaussians. Simply labeling every Gaussian inside a box is problematic when boxes overlap, as this causes label "bleeding" between neighboring objects (*e.g.*, a table partially swallowed by a chair's bounding box). To make the assignment more accurate, we label each box using a connected-components heuristic.

For each annotated bounding box, we first obtain Gaussians whose centers lie inside the box and voxelize these points on a sparse 3D grid. We then compute connected components over the occupied voxels and retain only the largest connected component (LCC). Semantic and instance labels from the box are assigned only to Gaussians whose voxels belong to this LCC, which suppresses overlaps with adjacent bounding boxes while preserving the target instance. If the box contains fewer than two occupied voxels, we skip this bounding box label. After this step, each Gaussian is associated with at most one instance ID and a semantic label from the 72-class taxonomy. Examples of assigned semantic labels are visualized in Fig. F.

Outlier removal. Some InteriorGS scenes contain a small number of Gaussians located far from the scene contents. We remove such outliers using a simple quantile-based filter on the Gaussian centers. For each scene and each axis (x, y, z), we compute the q and $(1-q)$ percentiles with $q = 10^{-3}$, giving an interval $[\text{lower}, \text{upper}]$. We then enlarge this interval by 5% of its span on each side and keep only Gaussians whose centers lie within these expanded bounds on all three axes. In other words, we calculate

$$\text{span} = \text{upper} - \text{lower}, \quad \text{buffer} = 0.05 \cdot \text{span}. \tag{15}$$

The resulting valid interval is $[\text{lower} - \text{buffer}, \text{upper} + \text{buffer}]$. This procedure effectively removes outlier Gaussians that are far away, accounting for 0.20% of the total splats.

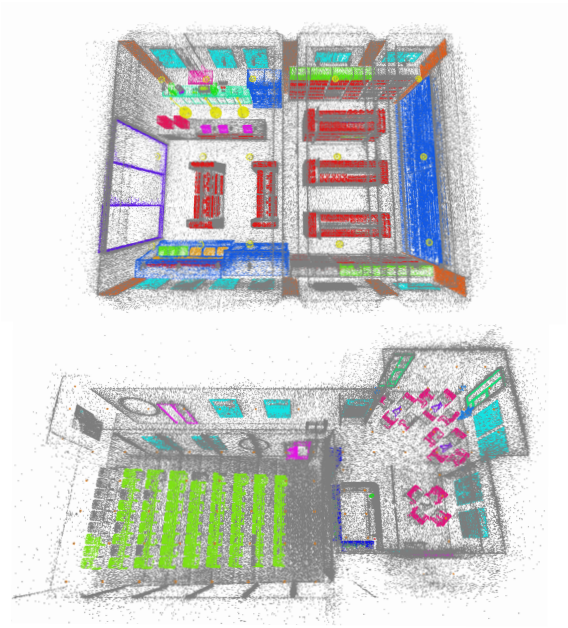


Figure F. **Semantic Labels Visualization (Gaussian centers) of InteriorGS Scenes.** Different classes are in distinct colors with the ignore label in dark grey.

Dataset splits. Finally, we define a simple scene-level split of InteriorGS for benchmarking: scenes `001-100` are used as the test set, `101-200` as the validation set, and `201-1000` as the training set. Note that InteriorGS is *not* included in the pretraining data of Chorus, it is reserved

entirely for zero-shot evaluation, downstream probing, and rendering-based adaptation.

C. Implementation Details

C.1. Open-Vocabulary Segmentation

In main paper Tab. 1 and 2, when running Chorus and [9] for the open-vocabulary semantic and instance segmentation on ScanNet, ScanNet++, and Matterport3D benchmarks, the predictions are first obtained at the Gaussians' centers and then propagated to the ground truth labeling locations through nearest neighbor voting. When running point clouds baseline [8] on InteriorGS, we compared two approaches when evaluating: *(i)* input Gaussians' centers, colors and estimated normals, *(ii)* sample viewpoints and fuse the rendered depth map from the test scenes into point clouds, and interpolate the colors from Gaussians' colors, then input this derived point clouds instead. Approach *(i)* is chosen for it leads to better results for the baseline.

C.2. Language Model-Based Question Answering

We follow GaussianVLM [5] to conduct VLM experiments with our Chorus encoder. We represent each 3D scene using 40k randomly sampled Gaussians from SceneSplat [9, 11]. For the language model, we adopt OPT-1.3B [22], following the LL3DA training protocol. The LLM is loaded in `float16` for memory efficiency and finetuned using LoRA. Our training procedure adheres to the standard protocol: 5 epochs of alignment followed by 32 epochs of instruction tuning. Training completes in under one day on two H200 GPUs. Additionally, we pretrain our task-guided sparsifier on the object-captioning task for 5 epochs. We employ the AdamW optimizer with a weight decay of 0.1 and a cosine-annealing learning-rate schedule, decaying from 10^{-4} to 10^{-6}. Evaluation is performed every 8 epochs.

For the connector, we preserve the overall sparsifier architecture and only modify the task-guided sparsifier: we reduce it to a single attention block and feed it only the final-layer Chorus features as input, leaving all other components unchanged.

Evaluation Metrics. For a predicted sentence $\hat{y}$ and reference sentence y, we use the following metrics; all scores are averaged over the evaluation set.

- **EM1z (Top-1 Exact Match).** This metric measures the fraction of predictions that exactly match the reference answer (after normalization such as lowercasing and stripping punctuation). For N samples with predictions $\{\hat{y}_i\}_{i=1}^{N}$ and references $\{y_i\}_{i=1}^{N}$, we compute

$$\mathrm{EM}_1 = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\big[\hat{y}_i = y_i\big], \tag{16}$$

where $\mathbb{I}[\cdot]$ is the indicator function.

- **M (METEOR)** [1]. METEOR computes a unigram alignment between $\hat{y}$ and y (accounting for stem, synonym, and paraphrase matches), and then derives precision and recall over matched unigrams:

$$P = \frac{\#\text{ matched unigrams}}{\#\text{ unigrams in }\hat{y}}, \qquad R = \frac{\#\text{ matched unigrams}}{\#\text{ unigrams in }y}. \tag{17}$$

A recall-weighted F-score is then defined as

$$F_\beta = \frac{(1+\beta^2)\,PR}{\beta^2 P + R}, \tag{18}$$

where $\beta > 1$ emphasizes recall (we use the default METEOR setting). Finally, a fragmentation penalty is applied based on the number of contiguous matched chunks ch and the total number of matched unigrams m:

$$\text{Penalty} = \gamma\left(\frac{ch}{m}\right)^{\theta}, \qquad \text{METEOR} = (1-\text{Penalty})\cdot F_\beta, \tag{19}$$

with γ and θ as method-specific hyperparameters.

- **R (ROUGE)** [10]. We report ROUGE-L, which is based on the length of the longest common subsequence (LCS) between $\hat{y}$ and y. Let $\mathrm{LCS}(\hat{y}, y)$ denote the LCS length, and let $|\hat{y}|$ and $|y|$ be the sequence lengths. We define

$$P_L = \frac{\mathrm{LCS}(\hat{y}, y)}{|\hat{y}|}, \qquad R_L = \frac{\mathrm{LCS}(\hat{y}, y)}{|y|}, \tag{20}$$

and compute an F-score:

$$F_\beta^{\text{ROUGE-L}} = \frac{(1+\beta^2)\,P_L R_L}{\beta^2 P_L + R_L}, \tag{21}$$

where β controls the relative weighting of recall (we use the standard ROUGE-L configuration).

- **Sim (Sentence Similarity)** [15]. To capture semantic similarity beyond surface-level overlap, we encode $\hat{y}$ and y with Sentence-BERT to obtain embeddings $\mathbf{e}_{\hat{y}}, \mathbf{e}_y \in \mathbb{R}^d$. We then compute cosine similarity:

$$\mathrm{Sim}(\hat{y}, y) = \cos\left(\mathbf{e}_{\hat{y}}, \mathbf{e}_y\right) = \frac{\mathbf{e}_{\hat{y}}^{\top}\mathbf{e}_y}{\|\mathbf{e}_{\hat{y}}\|_2\,\|\mathbf{e}_y\|_2}. \tag{22}$$

C.3. Chorus on Point Clouds Tasks

For all results reported in main paper Tab. 4, 5, and 6 on Semantic Segmentation Probing & Finetuning, ScanNet Data-Efficient Benchmark, and Instance Segmentation Probing and Finetuning, we conduct the experiments using data epochs of 400. We reproduced [21] using the official checkpoint and report results for [19, 20] from [23]. During Semantic Segmentation Probing & Finetuning on InteriorGS, we use the standard Chorus encoder❄ that takes

Gaussian parameters as input, and for the point clouds baseline [21], similarly to Sec. C.1, we have chosen input with Gaussians' centers, colors and estimated normals, as it leads to better results for the baseline. For all other benchmarks, we use the Chorus variant • that takes the point clouds parameters as input.

C.4. Rendering-Based Adaptation

We present the 2D statistical results for the final selected scenes in Tab. B, which were filtered based on two criteria: the minimum number of available poses must be greater than 10, and scenes with too many splats (where the minimum visible Gaussian count exceeds 819k) are ignored. Crucially, the box augmentation strategy significantly increases the number of splats. This effectively increases the number of output features supervised by the 2D teacher models.

We use a base resolution of 480×640 for image rendering, as outlined in step 3 of Algorithm 1. To generate a larger and higher-resolution feature map, we interpolate the images before feeding them into the 2D teacher model (rendering directly to high-resolution images would be slightly slower). In the resolution experiments, we interpolate images up to 960×1280 to generate the 60×80 DINOv3 feature map. By default, the predicted feature map is rendered to a resolution of 120×160 (step 5 in Algorithm 1), which is also the dimension used for the interpolation of the teacher feature.

Avg. Frame Number	Avg. per-frame visible Gaussians	+ box cropping	+ overlapping pair merge
37	126k (16%)	348k (43%)	419k (52%)

Table B. **Pose Sampling Statistics on InteriorGS.** In 582 selected scenes we report average visible gaussian in each frame and its ratio compared to total number of Gaussians in the scene.

D. Additional Experiment Results

D.1. Language Teacher Ablation

Training source	Teachers			Linear Probing		Decoder Probing	
	DINO	Lang	PE	mIoU	mAcc	mIoU	mAcc
ScanNet	✓	-	-	23.7	34.2	26.7	37.6
	✓	✓	-	**24.7**	**35.6**	**27.5**	**38.2**
ScanNet++ v2	✓	-	-	24.6	35.2	26.2	37.4
	-	✓	-	24.4	35.1	27.4	40.0
	✓	✓	-	25.1	36.7	**28.9**	40.4
	✓	✓	✓	**25.6**	**37.3**	28.3	**40.6**

Table C. **Language Teacher Ablation on InteriorGS.** We report linear probing and decoder probing results of the corresponding pretrained Chorus encoder.

In the main paper Tab. 9, we ablate the teachers by fixing the language teacher (SigLIP [18]) and then add DINO [16], followed by PE-Spatial [2], by the zero-shot segmentation performance on InteriorGS. Tab. C presents another view where we start with the DINO teacher, then add SigLIP and PE-Spatial. In this setting, we compare the probing performance on the InteriorGS dataset, as an encoder trained with the DINO teacher alone does not support zero-shot segmentation. The results are mostly consistent with Tab. 9: incorporating both the SigLIP and PE-Spatial teachers yields an improvement in probing accuracy, verifying that language-aligned and object-aware cues are complementary to the generalist features from DINO.

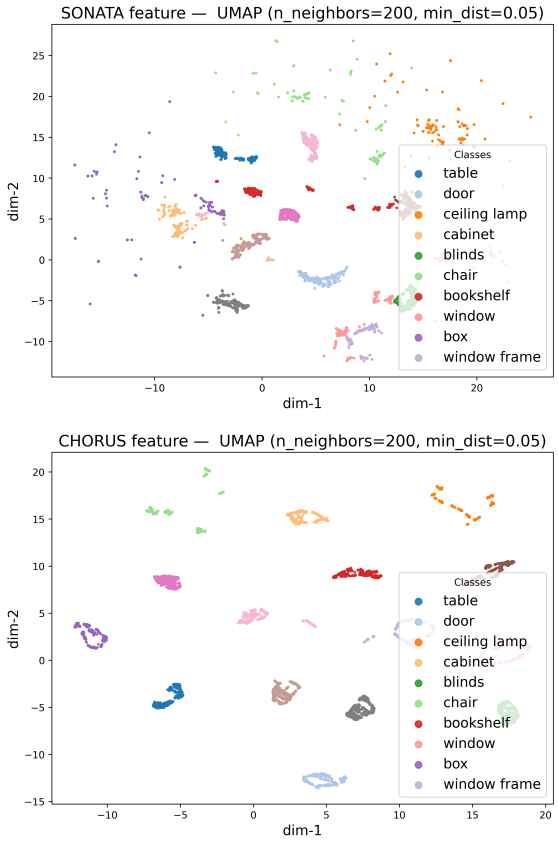


Figure G. **UMAP Feature Analysis.** We select 15 semantic classes, sample one instance per class from 10 scenes in the ScanNet++ Val split, and then project per-point encoded features from each method using UMAP. Chorus leads to more compact and well-separated clusters.

D.2. UMAP Feature Analysis

To further support our hypothesis on why Chorus works well on point clouds, we compare the Chorus variant that operates directly on point clouds and the Sonata [21] encoder trained with the self-supervised scheme, by visualizing the encoded features with UMAP [13]. We select 15 semantic classes, sample one instance per class, and subsample at most 1000 points per instance from 10 scenes in the ScanNet++ Val split. We then project per-point encoded

Figure H. **Inference Feature PCA Comparison** (best viewed zoom-in).

features from each method using UMAP with $n_{\text{neighbors}} = 200$ and $\text{min\_dist} = 0.05$. We choose UMAP instead of t-SNE because it better preserves global neighborhood structure, making inter-cluster distances more interpretable.

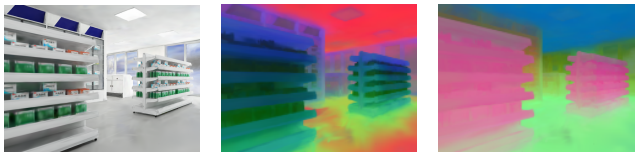
As shown in Fig. G, Chorus produces noticeably more compact and well-separated clusters than Sonata, especially for classes such as *ceiling lamp* and *box*, where Sonata exhibits more scatter and overlap with other categories. The tight within-instance clusters and clear inter-class margins indicate that Chorus learns semantically consistent and geometry-aware embeddings even when only Gaussians' centers, colors, and normals are used as inputs during pretraining. These observations align with our retrieval results in the main paper ablation study: 3DGS-based pretraining behaves like a strong augmentation that stabilizes point-cloud features, while multi-teacher supervision encourages a more structured feature space that transfers effectively to point clouds despite the inference distribution gap.

D.3. Inference Feature PCA Visualization

We provide additional qualitative examples in Fig. H comparing the scene encodings of Chorus, SceneSplat [9], and Sonata [21] on diverse scenes from the ScanNet++ Val split and the InteriorGS Test split. Compared to SceneSplat, Chorus exhibits stronger semantic consistency, for example, instances of the semantic classes *chair*, *plant*, *food package*, and *book* are more coherently clustered. Chorus also presents superior geometry awareness: even a stylized chair with many thin structures is correctly encoded, *i.e.*, its PCA colors resemble those of other chairs, whereas SceneSplat fails to recognize it. Meanwhile, the encodings from Sonata exhibit noticeable grid artifacts and reduced semantic consistency. These qualitative trends are consistent with the performance gains of our Chorus encoder on 3DGS-native and point-cloud benchmarks reported in the main paper.

D.4. 2D Adaptation Feature PCA Visualization

Figure I visualizes the features of an unseen InteriorGS [17] scene processed by Chorus. The adaptation notably reduces inconsistency on the ground plane. However, the lamp and ceiling are not clearly distinguishable, and the edges are less distinct. This degradation is likely due to the use of a low-resolution feature map for supervision. As ablated in our main paper, employing higher-resolution feature maps is preferred to improve adaptation performance.



(a) (b) (c)

Figure I. **Inference Features of the Chorus DINO Projector Head Before and After Rendering-Based Adaptation.** (a) Rendered RGB image. (b) PCA of rendered features before adaptation. (c) PCA of rendered features after adaptation.

E. Impact Statement

Chorus advances 3D scene understanding by offering the first holistic, feed-forward 3DGS encoder pretrained with multi-teacher distillation. By unifying complementary signals from language-aligned, general-purpose, and object-aware 2D foundation models, Chorus produces highly structured scene embeddings that generalize across tasks, datasets, and even modalities (3DGS and point clouds) with remarkable data efficiency. This capability enables more reliable open-vocabulary perception and more scalable scene understanding in applications like XR and embodied AI systems. Beyond in-distribution performance, our lightweight render-and-distill adaptation pipeline lowers the barrier for deploying the model to new environments without expensive 3D annotations. Overall, Chorus contributes a practical and scalable route for the next-generation 3D scene encoder.

References

[1] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005. 6

[2] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025. 4, 7

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 4

[4] Jiahuan Cheng, Jan-Nico Zaech, Luc Van Gool, and Danda Pani Paudel. Occam's lgs: A simple approach for language gaussian splatting. *arXiv e-prints*, pages arXiv–2412, 2024. 4

[5] Anna-Maria Halacheva, Jan-Nico Zaech, Xi Wang, Danda Pani Paudel, and Luc Van Gool. Gaussianvlm: Scene-centric 3d vision-language models using language-aligned gaussian splats for embodied reasoning and beyond. *arXiv preprint arXiv:2507.00886*, 2025. 6

[6] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. 2024. 3

[7] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024. 3

[8] Junha Lee, Chunghyun Park, Jaesung Choe, Yu-Chiang Frank Wang, Jan Kautz, Minsu Cho, and Chris Choy. Mosaic3d: Foundation dataset and model for open-vocabulary 3d segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14089–14101, 2025. 6

[9] Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, et al. Scenesplat: Gaussian splatting-based scene understanding with vision-language pretraining. *arXiv preprint arXiv:2503.18052*, 2025. 1, 4, 6, 9

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. 6

[11] Mengjiao Ma, Qi Ma, Yue Li, Jiahuan Cheng, Runyi Yang, Bin Ren, Nikola Popovic, Mingqiang Wei, Nicu Sebe, Luc Van Gool, et al. Scenesplat++: A large dataset and comprehensive benchmark for language gaussian splatting. In *NeurIPS*, 2025. 6

[12] Juliette Marrie, Romain Ménégaux, Michael Arbel, Diane Larlus, and Julien Mairal. Ludvig: Learning-free uplifting of 2d visual features to gaussian splatting scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7440–7450, 2025. 4

[13] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 7

[14] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 4

[15] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. 6

[16] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 4, 7

[17] Manycore Tech Inc. SpatialVerse Research Team. Interiorgs: A 3d gaussian splatting dataset of semantically labeled indoor scenes. https://huggingface.co/datasets/spatialverse/InteriorGS, 2025. 4, 9

[18] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. 4, 7

[19] Xiaoyang Wu, Xin Wen, Xihui Liu, and Hengshuang Zhao. Masked scene contrast: A scalable framework for unsupervised 3d representation learning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9415–9424, 2023. 6

[20] Xiaoyang Wu, Zhuotao Tian, Xin Wen, Bohao Peng, Xihui Liu, Kaicheng Yu, and Hengshuang Zhao. Towards large-scale 3d representation learning with multi-dataset point prompt training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19551–19562, 2024. 6

[21] Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22193–22204, 2025. 6, 7, 9

[22] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 6

[23] Yujia Zhang, Xiaoyang Wu, Yixing Lao, Chengyao Wang, Zhuotao Tian, Naiyan Wang, and Hengshuang Zhao. Concerto: Joint 2d-3d self-supervised learning emerges spatial representations. *arXiv preprint arXiv:2510.23607*, 2025. 6