

# SceneSplat: Gaussian Splatting-based Scene Understanding with Vision-Language Pretraining

## Supplementary Material

### Contents

<b>A Implementation Details</b>	<b>1</b>
A.1 Language Label Collection . . . . .	1
A.2 Vision-Language Pretraining . . . . .	1
A.3 Masked Gaussian Modeling . . . . .	2
A.4 Self-Distillation Representation Learning . . . . .	2
A.5 Autoencoder for Feature Compression . . . . .	2
A.6 Gaussian Language Alignment . . . . .	3
A.7 Model Architecture . . . . .	3
<b>B Further Experimental Results</b>	<b>3</b>
B.1 SceneSplat Zero-shot Segmentation . . . . .	3
B.2 3DGS Self-Supervised Pretraining . . . . .	4
<b>C Datasets Curation and Statistics</b>	<b>7</b>
C.1 Analysis . . . . .	7
C.2 Visualization . . . . .	9
<b>D Dataset License</b>	<b>9</b>
<b>E Limitations</b>	<b>9</b>
<b>F. Impact Statement</b>	<b>9</b>

### A. Implementation Details

#### A.1. Language Label Collection

Many existing methods align 3D primitives with text embeddings from vision-language models (VLM) [4, 6]. While effective for basic understanding, this approach inherently limits the information captured, as textual descriptions typically only convey categorical and spatial properties, lacking fine-grained visual details and relationships. Methods employing visual captioning models [7, 20] face similar challenges as they struggle to describe all aspects of the target scene content. Even detailed captions inevitably result in information loss during the text embedding alignment process. In contrast, we directly align Gaussians with the image embedding space of VLM, thereby preserving richer latent semantic information.

**Dynamic Weighting Mechanism.** Unlike previous approaches that use a single tight crop around each segment, we adapt the three-crop strategy [16] with dynamic weighting to capture the context. For each segment identified by SAMv2, we extract three distinct features: (1) global feature  $f_g$  from the entire RGB frame, capturing the full scene context; (2) local feature  $f_l$  from the image crop with

background; (3) masked feature  $f_m$  from the image crop without background, focusing solely on the object. During the process, SAMv2/sam2-hiera-large and SigLIP2/siglip2-base-patch16-512 models are used.

Our dynamic weighting mechanism combines the three extracted features through the following equations. First, we calculate the cosine similarity between features with and without background and use it to create a fused local feature:

$$r_{lm} = \text{sim}(f_l, f_m) \quad (1)$$

$$F_l = r_{lm} \cdot f_m + (1 - r_{lm}) \cdot f_l, \quad (2)$$

$$F_l = \text{normalize}(F_l) \quad (3)$$

We then compute the similarity between this fused local feature and the global feature to determine the final weights:

$$\phi_{lG} = \text{sim}(F_l, f_g), \quad w_i = \text{softmax}(\phi_{lG}) \quad (4)$$

$$w_g = w_i, \quad w_m = (1 - w_i) \cdot r_{lm}, \quad (5)$$

$$w_l = (1 - w_i) \cdot (1 - r_{lm}) \quad (6)$$

The final fused feature is computed as a weighted combination and normalized as:

$$f_s = w_g \cdot f_g + w_l \cdot f_l + w_m \cdot f_m, \quad f_s = \text{normalize}(f_s) \quad (7)$$

This mechanism adaptively balances the global context influence via  $w_g$ , local context with background via  $w_l$ , and object-specific features via  $w_m$ . These three features are dynamically combined to create a representation that adapts to the segment’s relationship with its context. For objects that are highly integrated with their surroundings (e.g., a keyboard in front of a monitor), background-inclusive features receive a higher weight. For isolated objects (e.g., a coffee mug), background-excluded features dominate.

#### A.2. Vision-Language Pretraining

Our vision-language 3DGS pretraining model is built on a transformer encoder-decoder architecture adapted from [17]. As detailed in Tab. A, the encoder consists of 4 stages with depths [2,2,2,6], channels [32,64,128,256], and heads [2,4,8,16], while the decoder employs 3 stages with depths [2,2,2], channels [768,512,256], and 16 attention heads each. We process 3D Gaussian primitives with all parameters (center, color, opacity, quaternion, and scale). The model is trained with the AdamW optimizer (initial LR

= 0.006, weight decay = 0.05) using a OneCycle scheduler with cosine annealing. Our loss weights are set to  $\lambda_{\text{cos}} = 1.0$ ,  $\lambda_{\text{L2}} = 1.0$ , and  $\lambda_{\text{con}} = 0.02$ , with contrastive loss (temperature  $\tau = 0.2$ ) activated only in the later 75% of training. During training, we employ extensive data augmentation, including random rotations, scaling, flipping, jittering, and elastic distortions (see Tab. C). We train for 800 data epochs using 4 NVIDIA H100 GPUs.

### A.3. Masked Gaussian Modeling

For all self-supervised pretraining experiments on the full dataset, we employ 4 NVIDIA H100 GPUs (94GB each) and the AdamW optimizer, with learning rates of  $1 \times 10^{-3}$  for the embedding layer and  $1 \times 10^{-4}$  for the attention blocks, coupled with a weight decay of  $1 \times 10^{-3}$ . We use mixed-precision training (FP16) to accelerate convergence and reduce memory overhead. In total, we train for 300k steps. We use an  $\mathcal{L}_2$  loss to enforce consistency between predicted and ground truth Gaussian parameters, focusing only on masked regions. For parameter reconstruction, we design a three-layer MLP projector for each Gaussian attribute, incorporating output activations that adhere to physical constraints: color uses a tanh activation (bounded in  $[-1, 1]$ ), opacity and scale use a sigmoid activation (bounded in  $[0, 1]$ ) because in most indoor scenes, most Gaussians have small scales, rotation, and normals apply tanh followed by  $\ell_2$ -normalization to ensure valid quaternions and normals.

### A.4. Self-Distillation Representation Learning

In self-distilled representation learning, we adopt the framework of [18], combining DINO loss and coding rate regularization on pooled encoder features. We configure a batch size of 24 and set the grid sampling resolution to 0.02 for partitioning Gaussian scenes. After grid sampling, the scenes are randomly partitioned into base views  $\mathcal{G}_b$ , which are then used to generate global and local crops. For global crops, we randomly select a Gaussian splat from  $\mathcal{G}_b$  and sample its  $K$  neighbors, where  $K \sim [0.4, 1.0] \times 256,000$ . For local crops, we sample  $K \sim [0.1, 0.4] \times 102,400$ . Each batch includes  $N_g = 2$  global views and  $N_l = 3$  local views to balance the scene coverage and granularity. For the DINO loss  $\mathcal{L}_{\text{DINO}}$ , we extract tokenized features from the student encoder  $f_\theta(\cdot)$  and teacher encoder  $f_\phi(\cdot)$ , where  $\phi$  is updated via exponential moving average (EMA). A global representation  $\bar{z}$  is obtained by mean pooling over spatial tokens. This representation is projected into a latent space using a 3-layer MLP  $P_{\text{DINO}}$  (dimensions:  $2048 \rightarrow 2048 \rightarrow 256$ ) followed by  $\ell_2$ -normalization. Then, we calculate the

similarity loss:

$$\mathcal{L}_{\text{sim}} = \frac{1}{N_g \times N_l} \sum_{i=1}^{N_l} \sum_{j=1}^{N_g} \frac{P_{\text{DINO}}(\bar{z}_l^{(i)}) \cdot P_{\text{DINO}}(\bar{z}_g^{(j)})}{\|P_{\text{DINO}}(\bar{z}_l^{(i)})\| \|P_{\text{DINO}}(\bar{z}_g^{(j)})\|} \quad (8)$$

For regularization, we use the negative of the coding rate:

$$R_\epsilon(\Gamma) := \frac{1}{2} \log \det \left( \mathbf{I} + \frac{d}{\epsilon^2} \Gamma \right), \quad (9)$$

$R_\epsilon$  approximates the rate-distortion function of a Gaussian random variable with covariance  $\Gamma$ , becoming exact as  $\epsilon \rightarrow 0$ . More specifically, it quantifies the spread of covariance, even when the underlying variables are not strictly Gaussian.

For the iBOT loss, we mask 50% of the global views with masking ratios sampled uniformly from  $r \sim [0.2, 0.7]$ . The masked tokens are replaced by a learnable token ( $T_{\text{mask}}$ ). Note here the masked global view embedding feature goes through the student network and outputs  $\hat{T}'(\theta_s) \in \mathbb{R}^{N' \times d_r}$ , where  $d_r$  denotes the representation dimension, while the unmasked global view embedding is forwarded to the teacher network and outputs  $\hat{T}'(\theta_t) \in \mathbb{R}^{N' \times d_r}$ . We detach the teacher’s output and calculate the simple iBOT loss using 3-layer MLP  $P_{\text{iBOT}}(\cdot)$  (dimensions:  $256 \rightarrow 256 \rightarrow 32$ ) as the projector:

$$\mathcal{L}_{\text{iBOT}} = \frac{1}{|M|} \sum_{i \in M} \underbrace{\frac{P_{\text{iBOT}}(\hat{T}'(\theta_s)^{(i)}) \cdot P_{\text{iBOT}}(\hat{T}'(\theta_t)^{(i)})}{\|P_{\text{iBOT}}(\hat{T}'(\theta_s)^{(i)})\| \|P_{\text{iBOT}}(\hat{T}'(\theta_t)^{(i)})\|}}_{\text{Pairwise Cosine Similarity}} \quad (10)$$

Only the masked regions  $M$  contribute to the iBOT loss calculation. Following [18], we simplify the original iBOT implementation by removing the centering operation and online tokenizer and replacing them with a pairwise cosine-similarity loss for feature alignment. When MGM loss is enabled, for one global view, we will employ less aggressive augmentation, and this view will contribute to both iBOT loss and MGM loss.

### A.5. Autoencoder for Feature Compression

Building on the idea from [12], we train a scene-specific language autoencoder on precomputed vision-language embeddings. This model compresses high-dimensional SigLIP features into a low-dimensional latent space, enabling the efficient storage of language features within Gaussian representations. The encoder (by default 5-layer architecture: [384, 192, 96, 48, 16]) generates compact latent codes, while the symmetric decoder (by default 5-layer architecture: [48, 96, 192, 384, 768]) reconstructs the original high-dimensional embeddings. This design reduces memory overhead while preserving semantic fidelity. Unlike [12], our model is trained on 3D Gaussian features rather

Config	Value
embedding depth	2
embedding channels	32
encoder depth	[2, 2, 2, 6]
encoder channels	[32, 64, 128, 256]
encoder num heads	[2, 4, 8, 16]
encoder patch size	[1024, 1024, 1024, 1024]
decoder depth	[2, 2, 2]
decoder channels	[768, 512, 256]
decoder num heads	[16, 16, 16]
decoder patch size	[1024, 1024, 1024]
down stride	$[\times 2, \times 2, \times 2, \times 2]$
mlp ratio	4
qkv bias	True
drop path	0.3

Table A. **Model Configs for Vision-Language Pretraining.**

Config	Value
embedding depth	2
embedding channels	32
encoder depth	[2, 2, 2, 6, 2]
encoder channels	[32, 64, 128, 256, 512]
encoder num heads	[2, 4, 8, 16, 32]
encoder patch size	[1024, 1024, 1024, 1024]
decoder depth	[2, 2, 2, 2]
decoder channels	[64, 64, 128, 256]
decoder num heads	[4, 4, 8, 16]
decoder patch size	[1024, 1024, 1024]
down stride	$[\times 2, \times 2, \times 2, \times 2]$
mlp ratio	4
qkv bias	True
drop path	0.3

Table B. **Model Configs for GaussianSSL Pretraining and Downstream Semantic Segmentation.**

than 2D image-level data. For detailed ablations on the autoencoder architecture and training paradigm, see Tab. E.

### A.6. Gaussian Language Alignment.

We follow the same cosine similarity loss and  $\mathcal{L}_2$  loss for feature alignment. When combining MGM with language alignment, we adhere to the MGM process by masking the input tokens and applying compressed language alignment loss only to the masked regions. Given the comparable magnitudes of all loss terms, we assign equal weights ( $\omega_{\text{MGM}} = \omega_{\text{DINO}} = \omega_{\text{BOT}} = \omega_{\text{LA}} = 1.0$ ) to maintain the balance of different objectives during training.

### A.7. Model Architecture

The network design choices are described in detail in Tabs. A and B. For the 3D backbone, we adopt the state-of-the-art Point Transformer [17], enhanced with Flash Attention, to substantially improve computational efficiency. To optimize feature embedding for scene-level Gaussians,

we use sparse convolutions, which preserve geometric details while minimizing the memory overhead.

## B. Further Experimental Results

### B.1. SceneSplat Zero-shot Segmentation

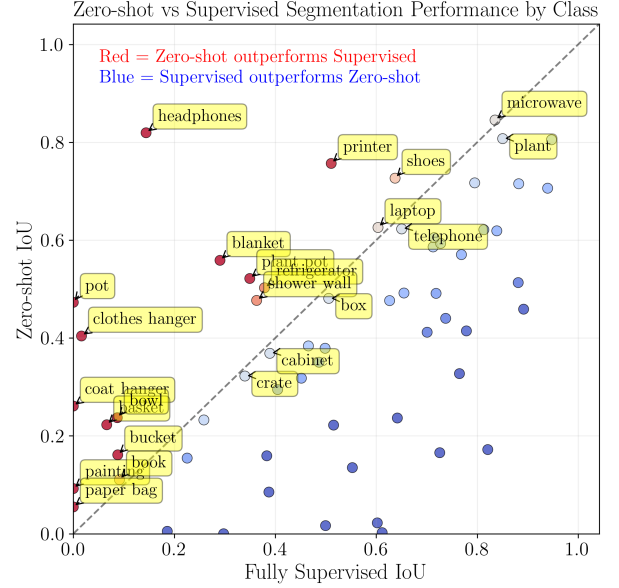


Figure A. **Comparison of SceneSplat Zero-Shot Versus Fully Supervised Segmentation Across Object Classes.** Notably, our zero-shot segmentation after vision language pretraining demonstrates better performance for 18 object classes, predominantly small objects such as headphone, printer, pot, and clothes hanger.

**Zero-shot vs. Fully Supervised.** Fig. A compares the class-wise zero-shot 3D segmentation results with those of the current state-of-the-art supervised method [17]. Points above the diagonal line (colored red) represent classes where zero-shot segmentation outperforms fully supervised approaches, whereas points below (colored blue) indicate the opposite. Our zero-shot segmentation achieves superior performance for 18 object classes in the ScanNet++ benchmark, predominantly small objects such as headphones, printers, pots, and clothes hangers. These results demonstrate the robust prior knowledge acquired by our model through vision-language pretraining.

#### More Qualitative Zero-shot Segmentation Results.

Fig. C presents more qualitative zero-shot semantic segmentation results on ScanNet++ validation scenes. SceneSplat effectively segments the scenes and helps annotate regions with missing labels in the ground truth.

**Consistency Issue in Label Collection.** Tab. D presents our zero-shot segmentation results on the ScanNet20 benchmark. We observe that the performance on this particular benchmark is significantly lower compared to the state-of-the-art results we achieve on the other three benchmarks.

Augmentations	Parameters	Global View	Local View
Base Transform		✓	✓
random rotate	axis: z, angle: [-1, 1], p: 0.5 axis: x, angle: [-1 / 64, 1 / 64], p: 0.5 axis: y, angle: [-1 / 64, 1 / 64], p: 0.5		
random scale	scale: [0.9, 1.1]		
random flip	p: 0.5		
random jitter	sigma: 0.005, clip: 0.02		
elastic distort	params: [[0.9, 0.1]]		
grid sampling	grid size 0.02		
Global Base Transform		✓	-
random flip	p: 0.5		
random crop	ratio: (0.4, 1.0) max: 256000		
Global Transform 0		✓	-
random color jitter	b:0.4, c:0.4, s:0.2 hue:0.1 p:0.8		
random grayscale	p: 0.2		
random Gaussian blur	p: 1.0		
Global Transform 1		✓	-
random dropout	dropout ratio: 0.2, p: 0.2		
random color jitter	b:0.4, c:0.4, s:0.2 hue:0.1 p:0.8		
random grayscale	p: 0.2		
random Gaussian blur	p: 0.2		
Local Base Transform		-	✓
elastic distort	params: [[0.2, 0.4], [0.8, 1.6]]		
random flip	p: 0.5		
random crop	ratio: (0.1, 0.4) max: 102400		
Local Transform		-	✓
random dropout	dropout ratio: 0.2, p: 0.2		
random color jitter	b:0.4, c:0.4, s:0.2 hue:0.1 p:0.8		
random grayscale	p: 0.2		
random Gaussian blur	p: 0.5		

Table C. **Data Augmentations.** Following [11], we design the data augmentations for 3D scenes for global and local views.

Through a detailed analysis, we identify that the issue stems from inconsistencies in the 3DGS language label collection process. Specifically, the SAMv2+SigLIP2 pipeline that we employ does not guarantee temporal consistency, especially for large background objects such as walls and floors, as illustrated in Fig. B. This inconsistency results in corrupted feature fields for Gaussians associated with these regions, subsequently leading to reduced zero-shot segmentation performance. Addressing this temporal consistency issue remains an important direction for future research.

## B.2. 3DGS Self-Supervised Pretraining

In Tab. E, we present an ablation study that compares various autoencoder architectures trained on the ScanNet++

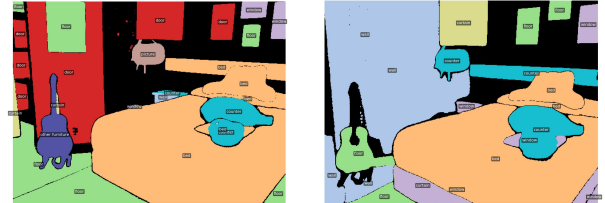


Figure B. **Consistency Issue During 2D Vision-Language Feature Map Collection on ScanNet.** Erroneous 2D feature maps are collected, as shown by the mislabeled regions in the neighboring figures. The root cause is that the SAMv2+SigLip2 process we use does not guarantee temporal consistency.

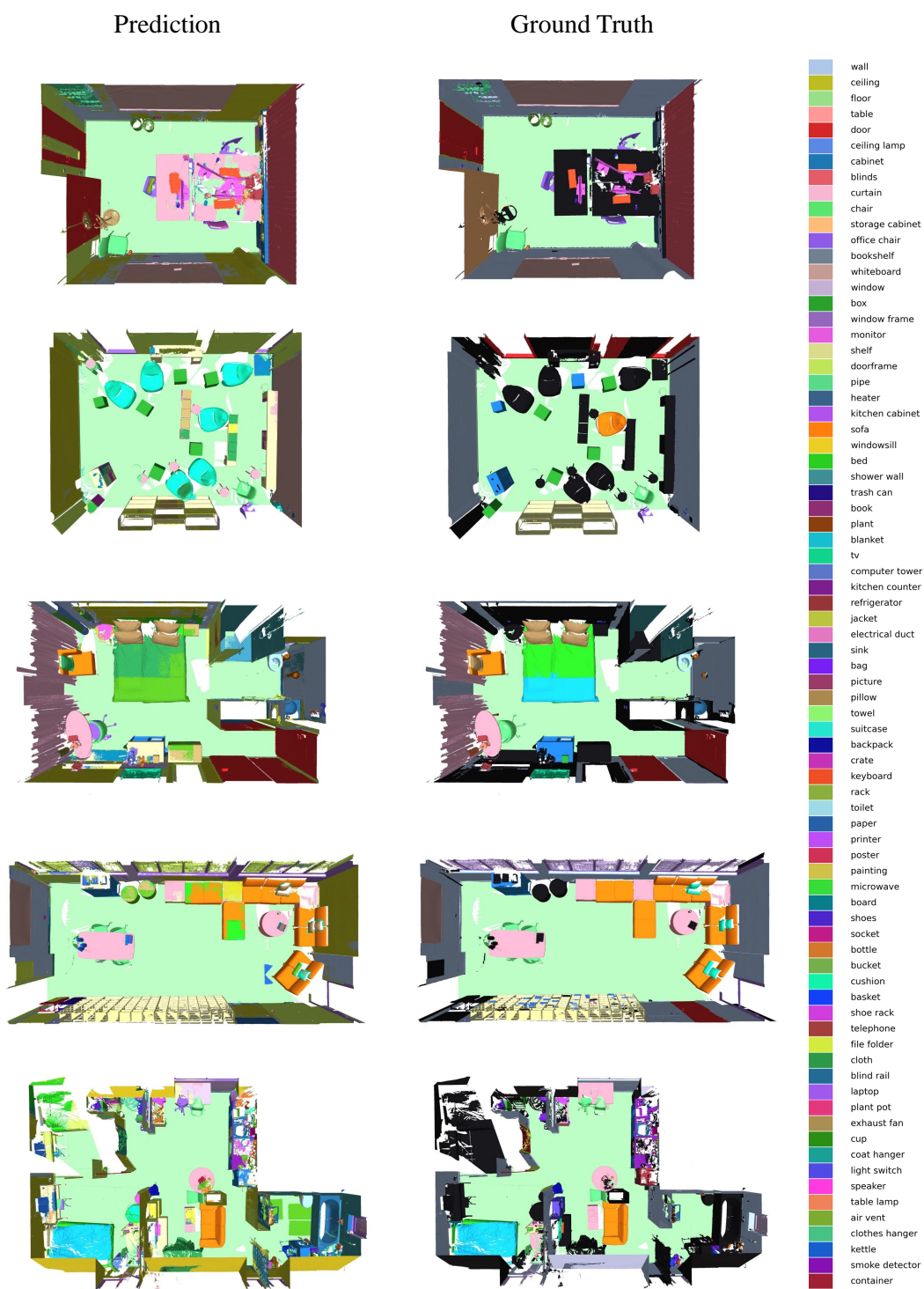


Figure C. **Qualitative Zero-shot Semantic Segmentation Results on ScanNet++ Validation Scenes.** SceneSplat effectively segments the scenes and helps annotate regions with missing labels in the ground truth.



Method	Training Source	ScanNet20 (20)	
		f-mIoU	f-mAcc
OpenScene	ScanNet	57.5	72.4
Mosaic3D	ScanNet	65.0	82.5
PLA	ScanNet	19.1	41.5
RegionPLC	ScanNet	55.6	76.3
OV3D	ScanNet	64.0	76.3
SceneSplat	ScanNet	35.4	57.9

Table D. **Zero-Shot 3D Semantic Segmentation on ScanNet20 Benchmark.** The results for the baselines are taken from [6]. Ours observes many faulty predictions and have poor performance, we identify the issue in the inconsistency during 2D feature map collection when labeling Gaussians, which leads to corrupted 3DGS-feature pairs.

Method	ScanNet++ 2D		ScanNet++ 3D	
	L2	cosine	L2	cosine
VAE16 layer5	0.008	0.182	0.005	0.028
VAE16 layer6	0.008	0.172	0.005	0.023
VAE64 layer5	0.007	0.081	0.004	0.014
VAE64 layer6	0.007	0.079	0.004	0.013

Table E. **AutoEncoder Ablation Experiments.** We report the feature compression performance with  $\mathcal{L}_2$  loss and cosine similarity on unseen 3D language label. We ablate on different autoencoder architectures and training sources.

Method	ScanNet20 (20)	
	mIoU	mAcc
LA16	<b>76.3</b>	<b>83.9</b>
LA64	75.8	82.2
LA16 (MGM)	76.2	83.7

Table F. **Language Alignment Loss Ablation.** We conduct an ablation study on low-dimensional language feature compression and Masked Gaussian Modeling (MGM)-based pretraining to evaluate their impact on semantic segmentation performance.

training set and evaluated on the validation set. In the “ScanNet++ 2D” columns, the autoencoder is trained using SigLIP2 features preprocessed from images, while in the “ScanNet++ 3D” columns, it is trained directly on Gaussian SigLIP2 features. At inference, we measure both the L2 distance and cosine similarity on 3D Gaussian SigLIP2 features. The results indicate that training the autoencoder directly on 3D data yields notably better performance, whereas increasing the network depth from layer5 to layer6 leads to only marginal improvements. Moreover, the 64-dimensional latent space consistently outperforms the 16-dimensional counterpart. Consequently, as detailed in Appendix A.5, we adopt the 3D-trained autoencoder with layer5 and a 16-dimensional latent space as our default configuration, striking a favorable balance between efficiency and accuracy.

We evaluate the impact of language feature alignment loss dimensionality (64 vs.16) when pretraining exclusively on ScanNet++ and testing on the ScanNet20 benchmark. As

Mask Ratio	ScanNet20		Mask Size	ScanNet20	
	mIoU	mAcc		mIoU	mAcc
0.4	76.1	83.6	0.02	76.9	<b>84.5</b>
0.5	76.4	84.1	0.05	<b>77.0</b>	<b>84.5</b>
0.6	<b>77.0</b>	<b>84.4</b>	0.10	76.6	83.8
0.7	76.7	83.8	0.15	76.5	84.1

Table G. **Masked Gaussian Modeling Ablation Experiment.** We analyze the impact of masking ratios and grid sizes in Masked Gaussian Modeling (MGM) on semantic segmentation performance using the ScanNet20 benchmark.

Method	ModelNet10 (10)		Omniobject3D (83)	
	Linear	MLP-3	Linear	MLP-3
MGM	<u>77.3</u>	<u>83.1</u>	<b>50.3</b>	<b>57.5</b>
+DINO	74.5	80.2	40.5	42.4
+iBOT	65.2	73.8	38.4	40.8
+LA	68.1	74.8	26.5	31.2
MGM+LA	<b>83.1</b>	<b>84.6</b>	<u>47.3</u>	<u>53.2</u>

Table H. **Cross Domain Linear Probe Experiments.** We report the mAcc results in the ablation of GaussianSSL on object-level classification tasks using the linear probe.

shown in Tab. F, we implement two 3-layer MLP architectures: one with uniform dimensions ( $64 \rightarrow 64 \rightarrow 64$ ) for 64 dimension language feature and another with progressively reduced dimensions ( $64 \rightarrow 32 \rightarrow 16$ ). We observe that reducing the latent dimension from 64D to 16D improves mIoU by +0.5% in mIoU and +0.7% in mAcc. This suggests that lower-dimensional language features with hierarchical dimensionality reduction preserve critical indoor scene semantic information.

We visualize the pretraining results using Masked Gaussian Modeling (MGM) and language feature alignment loss in Fig. D. The model effectively reconstructs masked Gaussian parameters (*e.g.*, position, scale, and rotation) and predicts semantically meaningful language-aligned features for indoor scene understanding.

Tab. G evaluates how masking ratio and grid size in Masked Gaussian Modeling (MGM) affect downstream semantic segmentation performance. Key observations include: (1) small masking ratios (0.4) degrade mIoU by 0.9% due to insufficient context learning compared to the ratio 0.6; (2) using a larger grid size  $0.15m$  for masking reduces fine-grained detail recovery, leading to a 0.5% drop in mIoU compared to a  $0.05m$  grid size.

To evaluate global scene understanding, we design a cross-domain classification task (Tab. H) using object-level Gaussian splats from [9, 10]. We freeze the encoder and embedding layers and train only (1) a linear probe (final layer) to map the features to logits. (2) a 3-layer MLP ( $512 \rightarrow 256 \rightarrow$  classes) for nonlinear evaluation. We find that progressively adding DINO, iBOT, and language alignment (LA) losses degrades the classification accuracy. This misaligned trend with scene-level benchmarks stems from

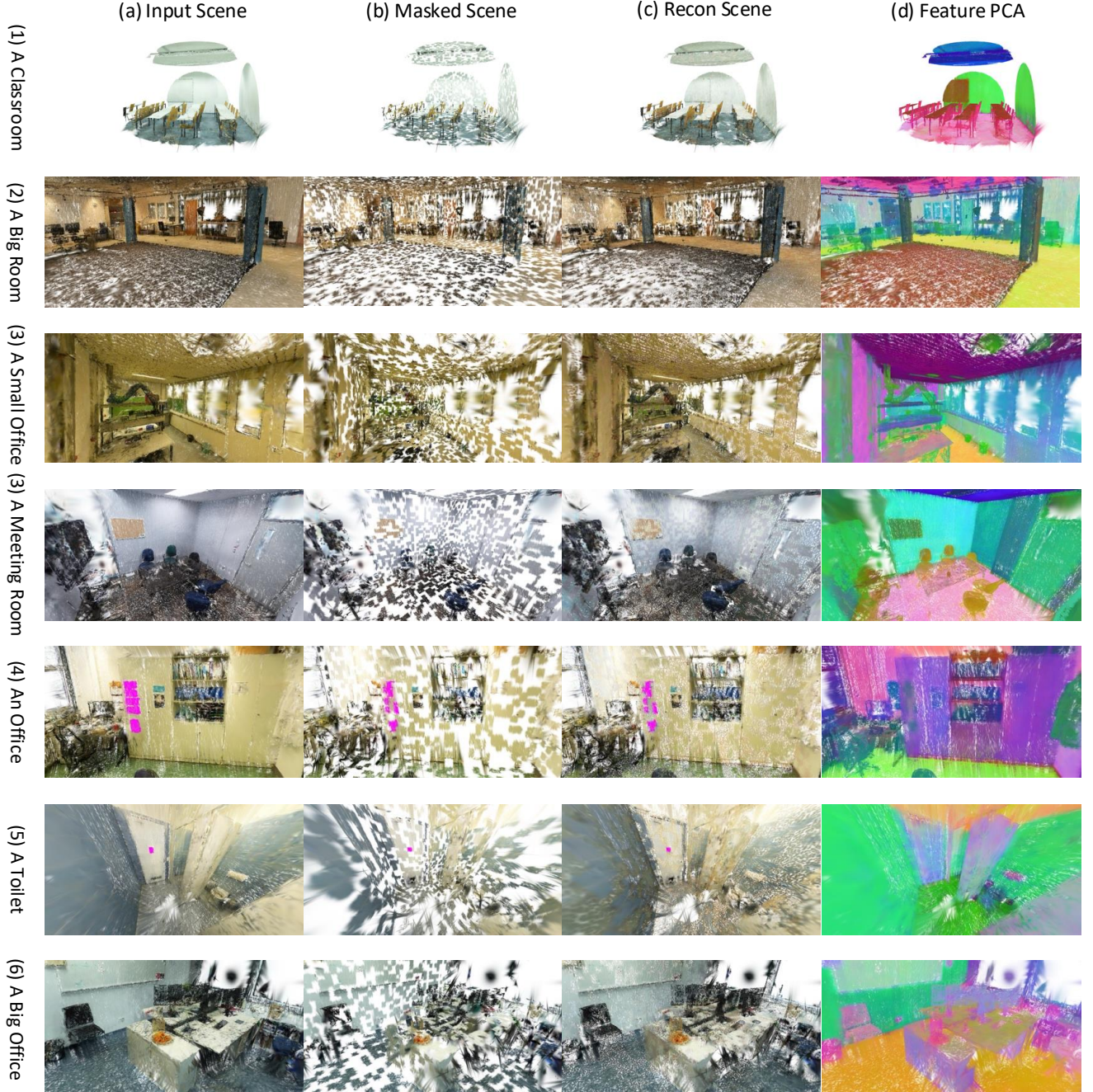


Figure D. **Self-supervised Reconstructions across Multiple Scenes.** Each row shows (left to right) the unmasked input, masked scene, reconstruction, and a PCA projection of features

domain gaps, where the model must map distinct objects (*e.g.*, fridge, oven) to similar scene-level semantics (*e.g.*, "kitchen"). Using masked pretraining and language alignment loss achieves the best performance on ModelNet10 (furniture), whereas MGM alone excels on OmniObject3D (common objects). For ModelNet10, MGM achieves the best 80% accuracy, whereas for the challenging object dataset (*e.g.*, OmniObject3D), it peaks at 50%.

## C. Datasets Curation and Statistics

### C.1. Analysis

Gaussian Splatting (GS) has demonstrated strong reconstruction capabilities, but its performance is heavily dependent on raw dataset quality and input conditions. We analyzed both quantitative metrics and visualization results of reconstructed scenes and identified several common failure



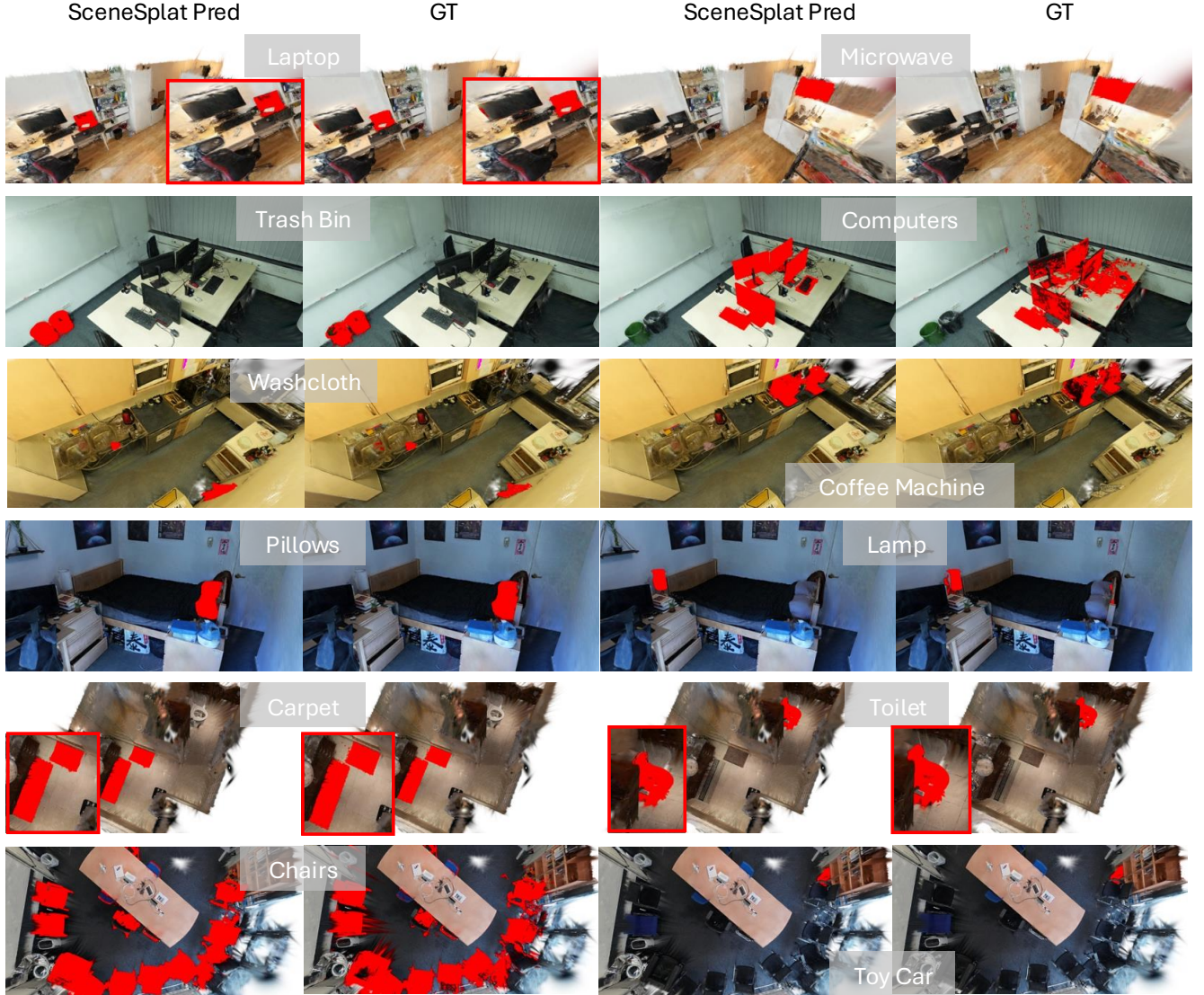


Figure E. Comparison of Scene Query Results Using Our Predictions and GT Language Labels on ScanNet++.

cases. The primary issues include holes, floating artifacts, and non-smooth surfaces. Through our analysis, we categorize the root causes into four main factors.

**Lack of Frames and Filming Angles.** Scenes with fewer than 400 RGB frames often fail to capture a complete representation of the environment. Limited filming angles result in missing perspectives, leading to incomplete indoor structures. Fig. F from the ARKitScenes dataset show partial room reconstructions where corners or ceiling details are lost.

**Motion Blur and Camera Instability.** As seen in Fig. G, blurry inputs are in the 3RScan dataset. The rapid, unsteady filming leads to edge deformation, ghosting, and smeared textures. In the resulted reconstructions, the sharp edges are lost, significantly reducing the fine details in the renderings from the 3DGS scene.



Figure F. Example Incomplete Scenes in ARKitScenes.

**Indoor-outdoor Lighting Changes.** Scenes with dynamic indoor-outdoor lighting changes, such as the case from the Hypersim dataset, exhibit severe blurring and loss of surface textures. Large glass ceilings, skylights, and reflective floors further disrupt GS training, as the algorithm struggles with light inconsistency and high contrast between illumi-





Figure G. Blurry Scenes from 3RScan.

nated and shadowed areas.

**Challenges with Transparent and Glass Objects.** Transparent and glass objects pose a unique challenge, as GS often fails to accurately capture windows or furniture glass, resulting in missing elements or floating artifacts. This issue arises from the inherent difficulty in rendering transparency, where reflection and refraction introduce complexity beyond the current GS capabilities.

## C.2. Visualization

The SceneSplat-7K dataset achieves an impressive average PSNR of 29.64 dB through all the different sources. We provide visualizations that showcase the photorealistic appearance rendering. See Figs. H to K.

## D. Dataset License

SceneSplat-7K is constructed from several established indoor datasets, each attached with a specific license: ARK-itScenes [1] (Apple Open Source License), Replica [14] (Replica Research License), ScanNet [3] (ScanNet Terms of Use), ScanNet++ [19] (ScanNet++ Terms of Use), HyperSim [13] (CC BY-SA 3.0), 3RScan [15] (3RScan Terms of Use), and Matterport3D [2] (Matterport Academic License Agreement). We have carefully structured our distribution approach to respect all original licenses while making our dataset accessible to the research community. For sources that permit redistribution, we plan to release our data on the Hugging Face under their respective terms of use. For datasets that require special permission, we co-host the data only after receiving approval from the original teams or let the original dataset team host the 3DGS data.

## E. Limitations

The quality of our dataset is largely influenced by the source indoor datasets. Low-resolution and blurry images can cause floating artifacts. Downstream tasks are also limited by the annotations of the original dataset. Temporal inconsistency in the current language label obtaining process needs to be addressed, as it can pollute vision-language

pretraining. In addition, we plan to add bounding boxes [5] and language descriptions [8] in the next step.

## F. Impact Statement

This work introduces SceneSplat-7K, the first large-scale indoor dataset of 3D Gaussian Splatting (3DGS). By providing over 7K 3DGS scenes, it enables standardized benchmarking for 3DGS-based understanding. The proposed framework for vision-language pretraining is tailored to 3DGS and enhances semantic alignment and establishes a clear connection between latent representations and 3DGS scenes. By leveraging knowledge distillation from 2D foundation models, combining contrastive learning and masked gaussian modeling (MGM), our approach significantly outperforms existing methods in zero-shot 3D semantic segmentation. This study lays the foundation for advancing scalable 3D scene understanding, with broad implications for autonomous systems and augmented reality applications. By publicly releasing the dataset, model, and code, we foster further innovation and facilitate the development of generalizable solutions for 3D scene understanding.



Figure H. ScanNet++ 3DGS. Ground truth (left) and 3DGS rendering (right).



Figure I. Hypersim 3DGS. Ground truth (left) and 3DGS rendering (right).



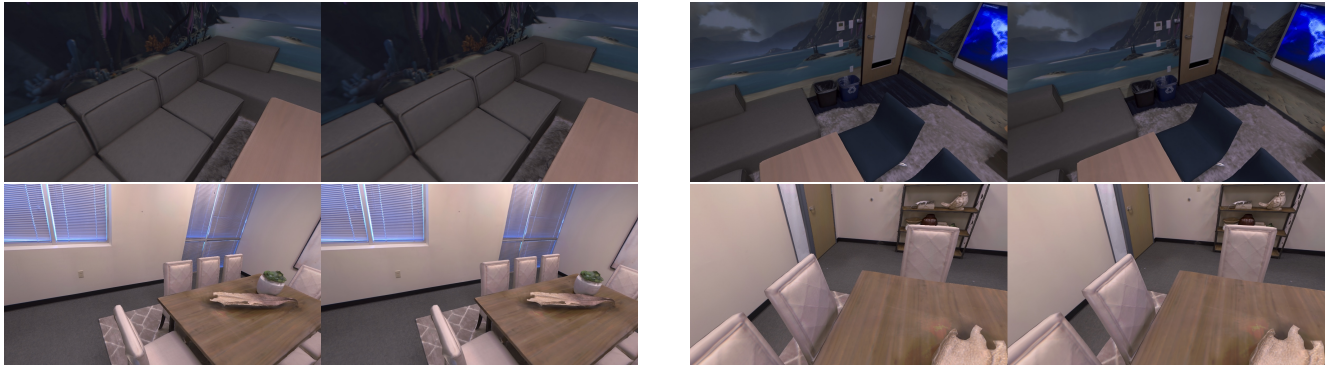


Figure J. Replica 3DGS. Ground truth (left) and 3DGS rendering (right). PSNR: 43.56 dB.



Figure K. 3RScan 3DGS. Ground truth (left) and 3DGS rendering (right). PSNR: 34.43 dB.



## References

- [1] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021. [9](#)
- [2] 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. [9](#)
- [3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. [9](#)
- [4] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7010–7019, 2023. [1](#)
- [5] Muhammad Zubair Irshad, Sergey Zakharov, Vitor Guizilini, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Nerf-mae: Masked autoencoders for self-supervised 3d representation learning for neural radiance fields. In *European Conference on Computer Vision (ECCV)*, 2024. [9](#)
- [6] 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. *arXiv preprint arXiv:2502.02548*, 2025. [1](#), [6](#)
- [7] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306, 2024. [1](#)
- [8] Ruiyuan Lyu, Tai Wang, Jingli Lin, Shuai Yang, Xiaohan Mao, Yilun Chen, Runsen Xu, Haifeng Huang, Chenming Zhu, Dahua Lin, and Jiangmiao Pang. Mmscan: A multi-modal 3d scene dataset with hierarchical grounded language annotations. In *arXiv*, 2024. [9](#)
- [9] Qi Ma, Yue Li, Bin Ren, Nicu Sebe, Ender Konukoglu, Theo Gevers, Luc Van Gool, and Danda Pani Paudel. Shapessplat: A large-scale dataset of gaussian splats and their self-supervised pretraining. In *3D Vision*, 2024. [6](#)
- [10] Qi Ma, Danda Pani Paudel, Ender Konukoglu, and Luc Van Gool. Implicit-zoo: A large-scale dataset of neural implicit functions for 2d images and 3d scenes, 2024. [6](#)
- [11] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. [4](#)
- [12] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. [2](#)
- [13] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021. [9](#)
- [14] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. [9](#)
- [15] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7658–7667, 2019. [9](#)
- [16] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. [1](#)
- [17] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. [1](#), [3](#)
- [18] Ziyang Wu, Jingyuan Zhang, Druv Pai, XuDong Wang, Chandan Singh, Jianwei Yang, Jianfeng Gao, and Yi Ma. Simplifying dino via coding rate regularization. *arXiv preprint arXiv:2502.10385*, 2025. [2](#)
- [19] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023. [9](#)
- [20] Yuqian Yuan, Wentong Li, Jian Liu, Dongqi Tang, Xinjie Luo, Chi Qin, Lei Zhang, and Jianke Zhu. Osprey: Pixel understanding with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28202–28211, 2024. [1](#)