Efficient Encoder-Free Fourier-based 3D Large Multimodal Model

Supplementary Material

A. Introduction

In this supplementary material, we first present additional ablation studies and qualitative results that further demonstrate the effectiveness and efficiency of Fase3D (Sec. B). We then provide additional details of Fase3D to complement the method description in the main paper (Sec. C). Next, we elaborate on the computational complexity analysis of our proposed method (Sec. D). Finally, we present the datasets used for training and evaluation, together with additional implementation details (Sec. E). We use the abbreviations C, B-4, M, and R to denote CIDEr [48], BLEU-4 [37], METEOR [3], and ROUGE-L [29], respectively.

B. Additional results

B.1. Number of LoRA layers

We vary the number of LoRA-inserted layers (0, 4, 8, 10, 12), where 0 indicates that the LLM is not fine-tuned. The largest improvement is observed from 0 to 8 layers (C: +12.9, B-4: +3.3, M: +2.9, R: +5.7). With 10 layers, the QA accuracy reaches its peak (C = 87.05), while 8 layers achieves the best language metrics (B-4/M/R). Increasing to 12 layers leads to degraded performance, suggesting mild overfitting. Overall, 8–10 layers provide the best trade-off between QA accuracy and text quality. We use #Layers = 8 in the final model to balance performance and computational cost.

Table A. Ablation study on the number of LLM's LoRA layers.

#Layers	ScanQA (Validation)			
	R↑	M↑	B-4↑	C↑
0	35.93	14.94	13.39	74.03
4	38.27	16.30	13.76	80.04
8	41.64	17.80	16.70	86.91
10	41.51	17.14	16.67	87.05
12	40.47	16.56	15.78	86.40

B.2. Space-filling curve count

We analyze the effect of the number n_C of space-filling curves (SFCs). As shown in Tab. B, ScanQA performance improves as the number of SFCs increases, but largely saturates beyond $n_C = 4$, with only marginal gains from 4 to 6. We therefore use $n_C = 4$ as a good trade-off between accuracy and computational cost.

B.3. Ablation studies on token merging

We vary the number of LLM input vision tokens after merging (64, 128, 256, 320). Accuracy (C) improves with more tokens, with the largest gain up to 256 (+7.38 C over 64; +2.61 B-4; +1.57 M; +3.68 R). Beyond 256, gains saturate: 320 yields a small accuracy uptick (87.20 *vs*. 86.91) and a negligible ROUGE change (41.73 *vs*. 41.64), while BLEU-4 and METEOR slightly drop. Overall, 256 tokens provides the best language metrics (B-4/M) with near-peak accuracy, offering the best compute–quality trade-off. We set token number as 256 across our experiments.

Table B. Effect of the number of SFC curves n_C in multi-curve serialization on ScanQA (val).

n_C	R↑	M↑	B-4↑	C↑
1	37.42	15.96	14.23	79.06
2	40.49	16.78	15.96	85.64
4	41.64	17.80	16.70	86.91
6	41.72	17.93	17.01	87.14

Table C. Ablation study on the number of LLM's input tokens.

#Token	ScanQA (Validation)			
	R↑	M↑	B-4↑	C↑
64	37.96	16.23	14.09	79.53
128	38.21	15.85	14.88	83.92
256	41.64	17.80	16.70	86.91
320	41.73	17.67	16.61	87.20

Table D. Effect of two-stage training on ScanQA validation.

Method	R↑	M↑	B-4↑	C↑
Scratch	41.64	17.80	16.70	86.91
w/pretrain	43.37	18.61	16.87	91.74

B.4. Ablation studies on two-stage training

Tab. D reports the performance of Fase3D under a two-stage setup compared with training from scratch. Two-stage training (generalist pre-training followed by ScanQA [2] instruction tuning) consistently improves all metrics: R increases from 41.64 to 43.37 (+1.73), M from 17.80 to 18.61 (+0.81), B from 16.70 to 16.87 (+0.17), and C from 86.91 to 91.74 (+4.83). These gains indicate that a generic pre-training stage provides a stronger initialization for downstream ScanQA.

B.5. Comparison with token selection strategies

To evaluate how different token selection and merging mechanisms affect both accuracy and token efficiency, we compare three strategies for reducing the number of 3D tokens: (i) farthest point sampling (FPS), (ii) a Q-Former-style cross-attention pooling module, and (iii) our proposed graph based token merging. All variants share the same backbone and are trained *from scratch* on the ScanQA training set, *without* any pre-training. As shown in Tab. E, FPS already improves over

Table E. Effect of different token selection / merging strategies on ScanQA validation performance. All models are trained from scratch on the ScanQA training split, without any pre-training.

Selector	R↑	M↑	B-4↑	C↑
Q-Former	37.52	15.87	14.06	77.41
FPS	39.67	17.10	15.18	82.54
Fase3D	**41.64**	**17.80**	**16.70**	**86.91**

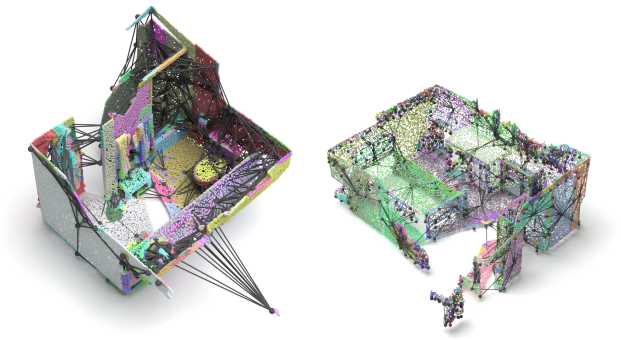

Figure A. Visualization of our SFC-based kNN graph construction via window voting. We show two representative examples of curve-guided neighbor selection.

the Q-Former selector by about +2.1 R, +1.2 M, +1.1 B-4, and +5.1 C, indicating that purely geometric sampling yields stronger 3D language grounding than cross-attention pooling. Our graph–based token merging further boosts performance across all metrics, achieving gains of roughly +4.1 R, +1.9 M, +2.6 B-4, and +9.5 C over the Q-Former baseline.

B.6. More qualitative visualizations.

Fig. A shows how we utilize space-filling curves (SFCs) to efficiently approximate k-NN search and construct our curve-based k-NN graph. The provided examples demonstrate that our method reliably identifies edges between superpoints, effectively preserving their genuine spatial connectivity.

Fig. B compares the space-filling curve (SFC) paths of four variants: Hilbert, transposed Hilbert, Z-order, and transposed Z-order. To make their differences easier to inspect, we render four viewpoints for each curve. While each variant traces the scene with a distinct trajectory, they all provide strong locality-preserving orderings. Specifically, Hilbert-based curves are smoother and exhibit fewer long-range jumps, whereas Z-order variants favor more axis-aligned runs. Including both the standard and transposed variants ensures complementary directional coverage, preventing orientation-specific artifacts and allowing the model to capture structural information consistently across different spatial alignments. This motivates our design choice in Fase3D to use all four curves jointly within the FFT-based token enhancer, ensuring that these complementary locality patterns are seamlessly fused into a single spectral representation.

C. Method details

C.1. Fast Fourier transform

We perform spectral mixing over the serialized token sequence using a real-valued frequency transform. In practice, the transform can be implemented with either FFT/iFFT-based operators or a DCT-style transform for improved efficiency on real-valued inputs. Given a real-valued sequence $x = (x_n)_{n=0}^{N-1}$, its discrete Fourier transform (DFT) and inverse transform are written as [38]

$$\begin{aligned} \mathcal{F}(x)_k &= \sum_{n=0}^{N-1} x_n \exp\left(-i\frac{2\pi}{N}kn\right), \\ \mathcal{F}^{-1}(y)_n &= \frac{1}{N}\sum_{k=0}^{N-1} y_k \exp\left(i\frac{2\pi}{N}kn\right). \end{aligned} \tag{5}$$

For real-valued inputs, the transformed coefficients exhibit conjugate symmetry, *i.e.*, $y_{N-k} = \overline{y_k}$. Therefore, the spectral transform can be implemented efficiently with reduced real-valued operators without changing the overall spectral mixing formulation. To improve efficiency and reduce boundary artifacts caused by discontinuities in the serialized sequence, we process the sequence block-wise using an overlap–add scheme [38]. Specifically, we partition the sequence into overlapping windows of length L_w with stride L_s, reducing the complexity from $O(N \log N)$ to $O(N \log L_w)$. In our implementation, we use a Hann window with $L_s = L_w/2$ (*i.e.*, 50% overlap), which satisfies the constant overlap–add (COLA) condition and ensures stable reconstruction. The smooth tapering at block boundaries also helps mitigate spectral leakage caused by abrupt jumps introduced by curve-based serialization.

C.2. Token serialization

Let $\mathcal{C} = \{c_i\}_{i=0}^{M-1}$ denote the 3D centers of the superpoints. We first normalize these centers relative to the scene bounding box and quantize them onto a b-bit integer grid:

$$\hat{c}_i = \left\lfloor (2^b - 1)\, \frac{c_i - c_{\min}}{c_{\max} - c_{\min}} \right\rfloor, \tag{6}$$

where $c_{\min}, c_{\max} \in \mathbb{R}^3$ are the minimum and maximum coordinates over all superpoint centers. Consequently, the quantized coordinates $\hat{c}_i = (\hat{c}_i^x, \hat{c}_i^y, \hat{c}_i^z)$ are bounded within the discrete space $\{0, \cdots, 2^b - 1\}^3$. From these quantized coordinates, we compute a 1D key $k_i = \mathrm{SFC}(\hat{c}_i)$ using a space-filling curve (*e.g.*, Z-order or Hilbert). This operation effectively maps the 3D spatial layout onto a 1D sequence, assigning each grid cell a unique integer index.

C.3. More details about neighbor searching

Real-world captured point clouds often present holes, disconnections, or sparse regions. Consequently, the initial

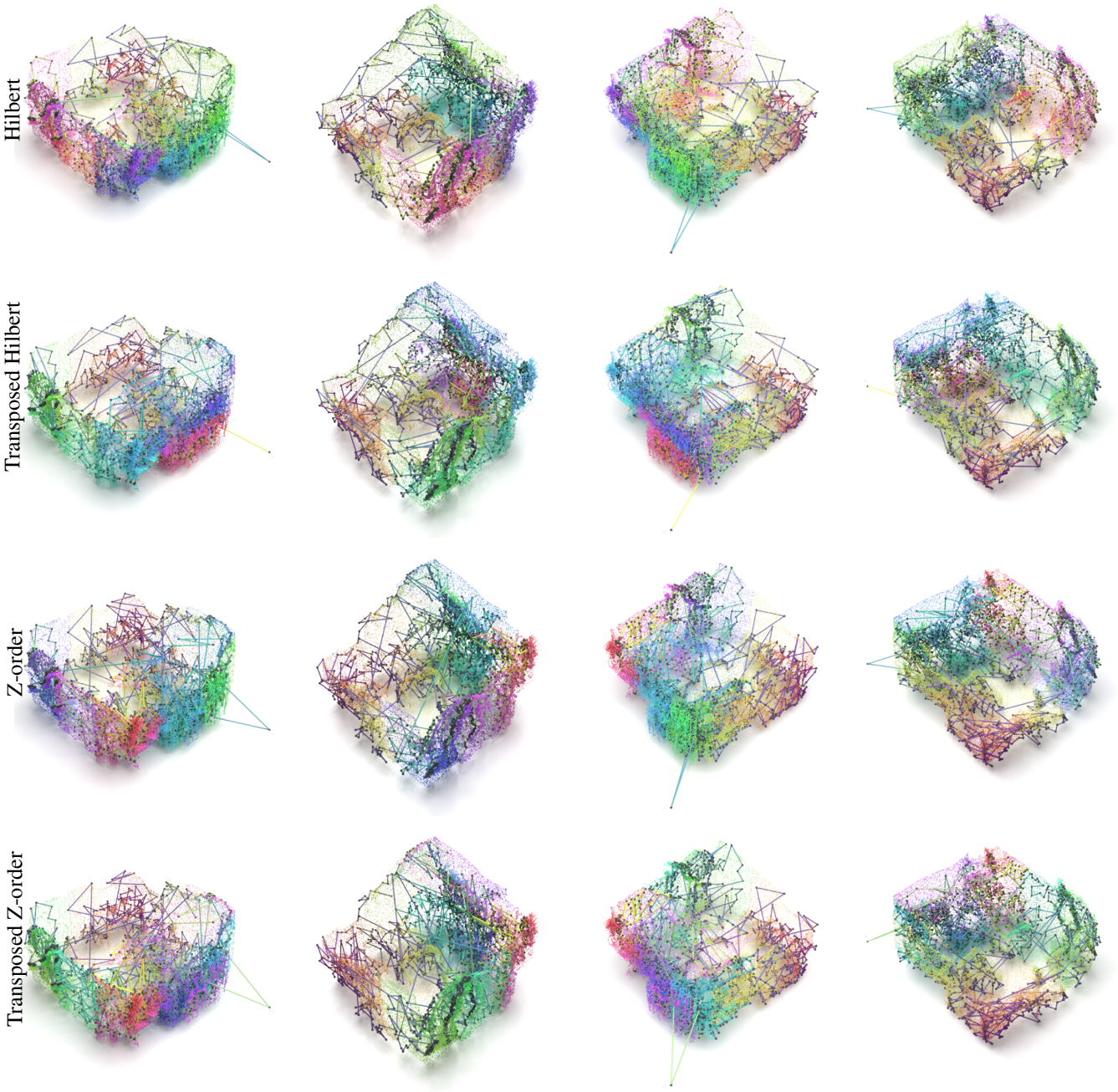

Figure B. Comparison of space-filling curves (Hilbert, transposed Hilbert, Z-order, and transposed Z-order) under four views (columns). All four variants produce broadly similar locality-preserving paths, with Hilbert-type curves appearing smoother and Z-order variants exhibiting more axis-aligned segments. Using both the standard and transposed versions provides complementary directional coverage and reduces orientation bias, so in Fase3D we apply the FFT jointly over all four curves.

connectivity operation can yield spurious edges when attempting to merge object-level superpoints. To reduce incorrect neighbors, we employ a two-factor re-ranking strategy. We prioritize edges based on the *shortest 3D Euclidean distance* as the primary factor. We then use the initial *votes* (*i.e.*, connectivity scores) as a secondary tie-breaker, retaining only the top k nearest neighbors to construct the final, clean graph. Specifically, we sum the neighborhood connectivity votes to obtain a coalesced candidate edge set $\mathcal{E}^{\text{cand}} = \{(s, t, v_{s,t}) \,|\, v_{s_t} > 0\}$. To suppress spurious connections resulting from SFC topological folding, we employ a re-ranking strategy for each source node s. Candidates are ranked using the squared Euclidean center distance $d_{s,t} = \|c_s - c_t\|_2^2$ as the primary sorting key, and the connec-

tivity votes $v_{s,t}$ as the secondary tie-breaker. We then retain the k nearest neighbors that minimize this composite tuple $(d_{s,t}, -v_{s,t})$. The resulting sparse graph $\mathcal{G}$, represented by its adjacency matrix A and degree matrix D.

D. Computational complexity analysis

D.1. Graph construction

Superpoint neighbors in our setting are defined by the minimum distance between their *boundary points*, *i.e.*, points lying near inter-superpoint interfaces. Let N be the number of points in a scene, M the number of superpoints, and B the batch size. Constructing a superpoint adjacency graph is a critical step in 3D tokenization, as it largely determines both the computational cost and the memory footprint of the overall vision–language pipeline.

In this section, we analyze and compare the computational complexity of our proposed *multi-curve voting graph* against a conventional *raw superpoint k-nearest-neighbor (k-NN)* graph constructed from boundary-based distances.

Raw KNN-based graph on superpoints. A brute-force construction compares points across all *distinct* superpoints to identify potential boundary interactions. Boundary-aware distances are computed *a posteriori* by aggregating inter-superpoint point-to-point distances (*e.g.*, taking the minimum), followed by a top-k selection step. Specifically, let p_i denote the number of points in superpoint i, such that the total number of points is $N = \sum_i p_i$. A naive implementation scanning all point pairs across distinct superpoints incurs a computational cost of

$$T_{\text{raw}} = O\Big(B \sum_{i \neq j} p_i p_j\Big). \tag{7}$$

Applying the algebraic identity $\sum_{i \neq j} a_i a_j = (\sum_i a_i)^2 - \sum_i a_i^2$, Eq. (7) then simplifies to:

$$T_{\text{raw}} = O\Big(B\big(N^2 - \textstyle\sum_i p_i^2\big)\Big). \tag{8}$$

In any typical over-segmentation ($M \geq 2, N_M \leq p_i \leq N/2$), the points are distributed across multiple regions, ensuring that the intra-superpoint term is negligible compared to the total interactions ($\sum_i p_i^2 \ll N^2$). $N_M \geq 2$ is a predefined threshold. Consequently, $T_{\text{raw}} = O(BN^2) \gg O(N \log N)$, this brute-force approach asymptotically dominates any efficient near-linear spatial indexing scheme. The memory footprint is likewise prohibitive, requiring the materialization of a dense $M \times M$ distance structure as $M_{\text{raw}} = O\big(BM^2\big)$. Even for moderate over-segmentation (*e.g.*, M=1024), this yields over 10^6 entries per scene. Furthermore, because boundary points are not known *a priori*, the boundary-aware variant amplifies load imbalance: superpoints with complex interfaces or higher densities disproportionately increase the constant factors in the runtime.

Multi-curve sparse voting graph. Our method sidesteps dense global pairwise comparisons by projecting the point cloud onto multiple space-filling curves and restricting neighbor search to short 1D windows along each curve. Let C=$|\pi|$ be the number of curves, W the local window radius on each curve (so each window has at most $2W$+1 points), and s_r the point-sampling stride along the curves. Each source point participates in at most $(2W$+1) 1D neighbors per curve, and we only evaluate these local pairs. Across all curves and all batches, the total number of candidate edges is

$$E \;\approx\; C \cdot B \cdot \frac{N}{s} \cdot (2W{+}1) \;=\; O\Big(CB\frac{NW}{s}\Big). \tag{9}$$

Consequently, the number of candidate edges grows *linearly* with N, rather than quadratically. We then aggregate all per-curve votes into a sparse coordinate (COO) tensor and resolve duplicates via a multi-key stable sort. This yields an overall time complexity of:

$$T_{\text{curve}} = O\Big(CB\frac{NW}{s}\log\big(CB\tfrac{NW}{s}\big)\Big), \tag{10}$$

which is near-linear with respect to the total number of points, bounded only by a small logarithmic sorting factor. The subsequent per-superpoint top-k selection operates strictly on the outgoing edges of each node. This introduces a lower-order term of $O(BMk \log k)$, which is asymptotically negligible compared to the global sparse sorting over E. The memory usage scales linearly with the scene size:

$$M_{\text{curve}} = O(CBN) + O(E). \tag{11}$$

Since all intermediate structures remain sparse, we avoid instantiating a dense $M \times M$ adjacency matrix.

Overall, our multi-curve voting graph replaces the quadratic time and memory complexity of a naive k-NN graph construction with a highly efficient, near-linear alternative. This design ensures robust scalability to high-resolution scenes with a large number of superpoints.

D.2. FFT-based context enhancer

Let D be the feature dimension, and N_h the number of frequency heads (with per-head width $d_h \approx D/N_h$). We analyze the computational complexity of the FFT-based context enhancer along the superpoint axis, specifically focusing on the windowed variant utilized in our implementation.

Windowed FFT mixing. Given a sequence of M tokens, the mixer partitions the input into overlapping windows of length L_w and stride L_s. The number of windows is

$$N_w = \Big\lfloor \frac{M - L_w}{L_s} \Big\rfloor + 1 \;=\; O\Big(\frac{M}{L_s}\Big). \tag{12}$$

For each window, we apply a 1D FFT of length L_w on every head, perform frequency-domain gating, and then apply the

inverse FFT. Since each head has dimension d_h and there are N_h heads with $N_h d_h{=}D$, the total cost of the operations is

$$T_{\text{FFT}} = O\left(B\,\frac{M}{L_s}\,D\,L_w \log L_w\right). \tag{13}$$

The remaining steps, including window weighting, linear input/output projections, and overlap-add reconstruction, contribute only linear overhead. When L_w and L_s are fixed constants, as in our experiments, the dominant cost simplifies to $T_{\text{FFT}} = O(BMD)$, which is effectively linear in the number of tokens M. The memory complexity is also linear as $\mathcal{M}_{\text{FFT}} = O(BMD)$, which accounts for the unfolded windows, FFT buffers, and overlap-add reconstruction buffers.

Multi-curve fusion. When operating on multiple space-filling curves (*e.g.*, Hilbert, transposed Hilbert, Z-order, transposed Z-order) with $|\pi|$ curve traversals, we apply the same windowed FFT mixer independently to each curve and fuse their outputs. This introduces a multiplicative factor in both time and memory:

$$T_{\text{FFT}} = O\big(|\pi|\,BDM\big), M_{\text{FFT}} = O\big(|\pi|\,BMD\big). \tag{14}$$

The subsequent curve-attention scorer introduces only a lightweight $O(B|\pi|D)$ overhead, which remains asymptotically negligible compared to the core FFT operations.

E. Dataset and Implementation details

E.1. Dataset

During training, we use the ScanNet portion of the 3D-LLM dataset [18] as our primary source. We further augment it with complementary datasets, including ScanQA [2], SQA3D [33], ScanRefer [5], and Nr3D [1]. After joint training on this mixture, we fine-tune Fase3D separately on each target dataset. We briefly summarize the datasets below.

3D-LLM dataset [18] comprises: (i) 1,033 textual descriptions across 517 scenes, (ii) 1,864 lines of embodied task planning spanning 510 scenes, and (iii) 2,955 lines of multi-turn embodied dialogues across 517 scenes.

ScanQA dataset [2] is a 3D question answering benchmark built on top of ScanNet [10], designed to evaluate scene understanding via natural language queries. It contains 6,857 unique questions paired with 30,769 answers over 806 reconstructed indoor environments. Questions focus on objects in the scene and cover object attributes, spatial relationships, and scene semantics. On average, each scene has 8.5 questions, encouraging reasoning over object-level details and contextual relations in complex 3D environments.

ScanRefer dataset [5] is a multimodal benchmark for 3D vision-language reasoning built on ScanNet [10], consisting of 1,613 RGB-D scans across 806 unique indoor scenes. It provides 51,583 natural language descriptions for objects in reconstructed 3D scenes, covering 800 ScanNet scenes. Each object is annotated with an average of 4.67 descriptions, resulting in rich linguistic diversity. On average, each scene contains 13.81 objects and 64.48 descriptions, spanning over 250 categories of common indoor objects. Among these, 41,034 descriptions explicitly mention attributes such as color, shape, size, and spatial relations, making ScanRefer a strong benchmark for fine-grained language reasoning.

Nr3D dataset [1] is a natural language 3D object localization benchmark built on ScanNet [10]. It contains 41,503 unique referring expressions for 5,578 objects across 707 scenes. Each description is crafted to unambiguously identify a target object in context, using spatial relations and attributes such as color, shape, and size. On average, each object is associated with 7.4 descriptions, making Nr3D well suited for evaluating fine-grained attribute understanding and spatial reasoning in 3D scenes.

SQA3D dataset [33] is a spatial question answering benchmark in 3D environments, built upon ScanNet [10]. It comprises natural language questions that require explicit 3D spatial reasoning, including object existence, relative positions, and relationships among multiple objects. Each question is grounded in a reconstructed 3D scene and paired with a free-form textual answer, enabling the evaluation of both geometric understanding and language generation. Compared to traditional 2D VQA benchmarks, SQA3D places greater emphasis on 3D-aware reasoning over scene layouts, object configurations, and embodied scene understanding, making it a challenging testbed for holistic 3D VLMs.

E.2. Implementation details

Superpoint generation. We directly use the segmentor provided by ScanNet [11], which is obtained via a graph-cut–based over-segmentation method [15, 21].

Training strategy. Following prior work [6, 35], Fase3D adopts a two-stage training strategy. In the first stage, we pre-train the model on an ensemble of datasets spanning diverse 3D tasks (dense captioning and question answering), so that it acquires a broad understanding of 3D scenes and functions as a 3D *generalist* model. In the second stage, we perform instruction-following fine-tuning on task-specific datasets for 3D dense captioning and 3D question answering, thereby specializing the model for the downstream benchmarks. Note that during both stages we do not use proposals from Mask3D [45]; all supervision is applied directly on our own tokenization pipeline.

Hyperparameters. Unless otherwise specified, for graph construction, window-based voting uses a window size of 64 and a stride of $s_r = 16$ along each space-filling curve. The number of initial superpoints, M, depends on the input point cloud and thus varies across samples. For the prompt embedding, we set 16 neighbors for the k-NN searching.