Diffusion Fine-Tuning: Iterative Refinement for Advanced Grounding with Diffusion Large Language Models

Zhangyang Qi[1,2] **Jinsong Li**[3] **Jiaqi Wang**[2] **Hengshuang Zhao**[1]*
[1]The University of Hong Kong [2]JD Explore Academy
[3]The Chinese University of Hong Kong

Abstract

Large Vision-Language Models (LVLMs) are strong at bounding-box grounding but struggle with precise spatial outputs such as polygon outlines. In vertex-level autoregressive (AR) decoding, early decisions cannot use the unresolved contour, and their errors can propagate along the sequence. We propose **Diffusion Fine-Tuning (DFT)**, which jointly denoises the coordinate digits of all vertices. Hierarchical Coordinate Decomposition groups hundreds, tens, and units into place-value channels, and Hierarchical Curriculum Learning progressively activates coarse-to-fine supervision. With ground-truth-conditioned training and token-wise decoding, DFT improves 16-point polygon grounding over matched-fine-tuned AR baselines while matching or modestly improving 2D bounding-box performance; the same formulation extends to 9-DoF monocular 3D grounding. Additional controlled experiments separate the contributions of AR output ordering, curriculum, diffusion, and block-wise search, and examine annealed self-conditioning and an auxiliary coordinate loss. Higher-vertex evaluations distinguish train–test count mismatch from matched-count performance: block-wise decoding improves both, but does not eliminate the difficulty of long contours. Inference uses joint denoising steps rather than sequential hundreds/tens/units passes; diffusion can still make early commitment errors and remains sensitive to training and decoding choices.

1 Introduction

Large Vision-Language Models (LVLMs) such as GPT-4o [OpenAI, 2024] and LLaVA [Liu et al., 2024a] excel at coarse bounding-box grounding but fall short on tasks requiring high spatial precision, such as Polygon Grounding, where a limited vertex budget must be allocated by global planning to outline an object accurately.

We attribute this bottleneck to the **vertex-level autoregressive (AR)** decoding employed by most leading LVLMs [Bai et al., 2025, Chen et al., 2024]: the $2N$ coordinate tokens of an N-vertex polygon (one (x, y) pair per vertex) are emitted left-to-right, with each token conditioned only on previously emitted ones. This induces two failure modes (Figure 1): (i) early-vertex errors are uncorrectable and propagate into the rest of the contour, and (ii) vertex allocation is committed before the full contour is observed, so the model overspends tokens on smooth segments and starves detailed regions.

To address these issues, we adopt a **Diffusion Language Model (DLM)** [Nie et al., 2025, Yang et al., 2025] backbone that places all $2N$ coordinate tokens under a single discrete-diffusion denoiser, removing the vertex-level left-to-right dependency. Naive DLM training, however, fails to converge—staying below 5 Acc@0.1 on RefCOCOg-test, far under the matched-FT AR baseline at 11.4 (Figure 2,

*Corresponding author: hszhao@cs.hku.hk.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

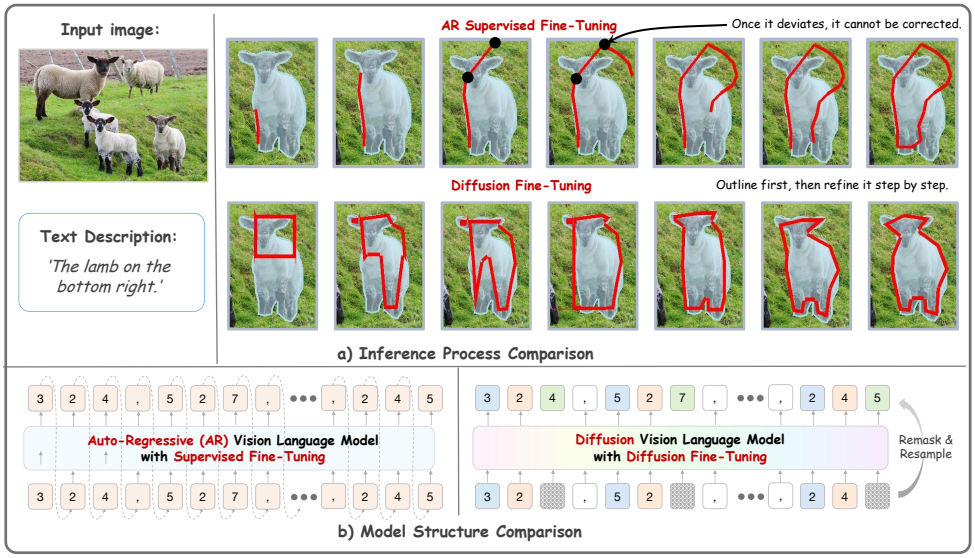

Figure 1: **Autoregressive (AR) models vs. Diffusion Fine-Tuning (DFT).** (a) An AR model generates vertices sequentially, making early errors irreversible, whereas DFT resolves positions using context distributed across the contour. Diffusion can still make early commitment errors. (b) AR uses unidirectional sequence generation, while DFT denoises all vertices in parallel.

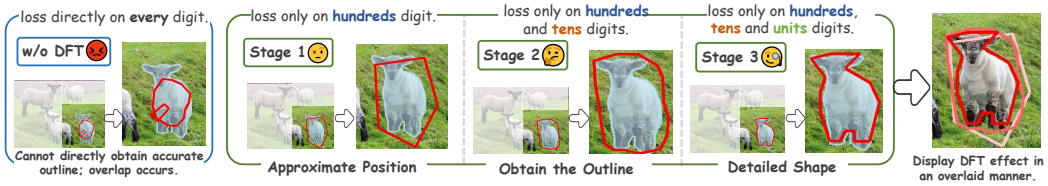

Figure 2: **DLM Training.** Direct DLM training (left) fails to converge, producing a coarse, invalid output with overlapping contours. Right part displays the outputs from the three progressive stages of Hierarchical Curriculum Learning (HCL): from a coarse outline, to a refined contour, and finally to the detailed shape.

left)—indicating that a structured fine-tuning recipe is required to make diffusion competitive with AR on high-precision localization.

We therefore introduce the **Diffusion Fine-tuning (DFT)** framework with two structural ingredients. **Hierarchical Coordinate Decomposition (HCD)** groups coordinate digits (hundreds, tens, units) into place-value channels across the contour. The AR LVLM baselines already tokenize coordinates as digits; HCD adds this channel structure, rather than giving DFT an easier output alphabet. The loss factorises across these channels via a coarse-to-fine chain rule, induced by training with **Hierarchical Curriculum Learning (HCL)**, a three-stage schedule that progressively activates the loss for the hundreds, tens, and units digits (Figure 2, right). The net effect: vertex-level sequential dependence is replaced by $K = 12$ joint denoising passes; the digit hierarchy shapes training supervision and conditioning, not a sequence of per-channel inference passes.

Our main tables retain the measured ground-truth-conditioned HCL recipe with token-wise top-1 decoding. Separately labelled experiments in Section M test annealed self-conditioning, an auxiliary coordinate loss, and alternative decoders; their gains are not combined or extrapolated to unmeasured table cells. We validate the measured recipe in three settings: (i) **RefCOCO-Polygon**, a new 16-point polygon benchmark derived from RefCOCO [Mao et al., 2016], where DFT reaches 26.4 Acc@0.1 on RefCOCOg-test vs. 19.7 for matched-FT Qwen3-VL 8B (+6.7); (ii) standard bounding-box benchmarks RefCOCO/+/g, where DFT matches or modestly exceeds the strongest AR baselines; and (iii) **Ref3D**, a 9-DoF monocular 3D grounding benchmark integrating four indoor sources (Objectron [Ahmadyan et al., 2021], SUN RGB-D [Song et al., 2015], ARKitScenes [Baruch et al., 2021], Hypersim [Roberts et al., 2021]) and two outdoor sources (KITTI [Geiger et al., 2013], nuScenes [Caesar et al., 2020]), where the same network and recipe extend without architectural change.

The main contributions of our work are as follows:

- **Diffusion Fine-Tuning (DFT).** We reframe visual grounding as K joint denoising passes over a discrete-diffusion LLM backbone, replacing the vertex-level AR chain with a digit-level training-time factorisation via Hierarchical Coordinate Decomposition (HCD) and Hierarchical Curriculum Learning (HCL).
- **Matched-FT gains on 2D grounding.** Under matched grounding data, schedule, and resolution, DFT improves over strong AR LVLMs (LLaVA-OV, Qwen2.5-VL, Qwen3-VL) on 16-point RefCOCO-Polygon ($+6.7$ to $+11.8$) and matches or modestly exceeds them on RefCOCO/+/g bounding-box benchmarks.
- **Versatility to 9-DoF 3D grounding.** The same network and recipe extend to 9-DoF monocular 3D grounding on Ref3D (indoor and outdoor) without architectural change, lifting LLaVA-OV from $43.2 \rightarrow 52.2$ on Objectron and $2.9 \rightarrow 39.7$ on KITTI; a block-wise top-k decoder handles the latency–accuracy trade-off.

2 Methodology

Subsection 2.1 formulates the visual grounding task. Subsection 2.2 introduces our Hierarchical Coordinate Decomposition (HCD) representation, Subsection 2.3 presents the Diffusion Fine-Tuning model built upon it, and Subsection 2.4 the Hierarchical Curriculum Learning strategy used to train it. Subsection 2.5 describes a block-wise top-k decoder for the latency–accuracy trade-off.

2.1 Problem Formulation: Sequential Representation of Polygon Generation

Visual grounding aims to spatially localize a target object in an image I based on a textual instruction T_{ins}. To overcome the coarseness of bounding boxes, we adopt a polygon-based formulation, treating localization as a conditional sequence modeling task. An N-vertex polygon is parameterized as a flattened coordinate sequence $S_0 \in \mathcal{C}^{2N}$:

$$S_0 = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N), \tag{1}$$

where each coordinate is quantized into a discrete integer space $\mathcal{C} = \{0, 1, \ldots, C-1\}$, and the objective is to maximize the conditional log-likelihood

$$\mathcal{J}(\theta) = \mathbb{E}_{(I, T_{ins}, S_0) \sim \mathcal{D}}[\log p_\theta(S_0 \mid I, T_{ins})]. \tag{2}$$

The output couples coordinates across a contour, while a left-to-right serialization reveals only a prefix at each decision. We therefore group coordinates by digit significance to expose shared coarse geometry during training. Digit-level cross-entropy alone does not encode spatial distance: nearby values such as 299 and 300 differ in several positions. HCD provides a place-value structure, not a complete solution to numerical ordinality.

2.2 Hierarchical Coordinate Decomposition

We propose **Hierarchical Coordinate Decomposition (HCD)** to factorize the coordinate generation process. We consider the coordinate space as a structured integer ring $\mathcal{C}$. HCD introduces a deterministic mapping $\Psi : \mathcal{C} \rightarrow \mathcal{D}^3$ (where $\mathcal{D} = \{0, \ldots, 9\}$) that decomposes a coordinate c into a triplet of digits $\mathbf{d} = (d^{(1)}, d^{(2)}, d^{(3)})$. Mathematically, the k-th digit component is obtained via a scale-dependent quantization function:

$$d^{(k)} = \Psi(c)_k = \left\lfloor \frac{c \pmod{10^{4-k}}}{10^{3-k}} \right\rfloor, \quad k \in \{1, 2, 3\}. \tag{3}$$

Given a sequence of coordinates $S_0 \in \mathcal{C}^{2N}$, this transformation yields a multi-channel representation $\mathbf{S} = \{S^{(1)}, S^{(2)}, S^{(3)}\}$, where $S^{(k)} \in \mathcal{D}^{2N}$ corresponds to the sequence at the k-th granularity level.

HCD uses a ten-symbol digit alphabet. Relative to dedicated location-token methods, this compresses a roughly 1000-symbol coordinate alphabet; the AR LVLM baselines here already use digit tokenization. The relevant difference is **place-value channel grouping**, which supports a coarse-to-fine curriculum across the whole contour. Its **inductive bias** is that the decomposition factorises the

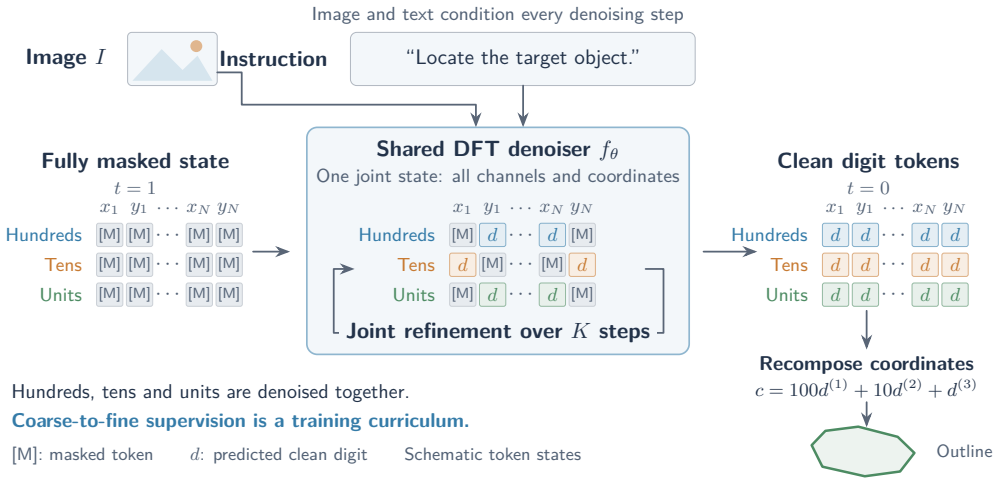

Figure 3: **The inference denoising process.** Starting from a fully masked state, each network pass predicts hundreds, tens, and units jointly across all coordinates. The digit hierarchy controls training, not three sequential inference phases.

data likelihood into a cascade where $S^{(1)}$ encodes a low-frequency macro skeleton, while $S^{(2)}$ and $S^{(3)}$ encode mid- and high-frequency residuals. For example, a coordinate $(258, 703)$ stratifies into skeleton $(2, 7)$ and detail coefficients $(5, 0)$ and $(8, 3)$. This sets up the chain-rule factorisation in Equation (6) that our reverse process will be trained to satisfy.

2.3 Diffusion Fine-Tuning

Forward process. We use continuous-time absorbing-state discrete diffusion ($t \in [0, 1]$, linear schedule $\alpha_t = 1 - t$) over the digit alphabet $\mathcal{V} = \mathcal{D} \cup \{[\mathrm{M}]\}$ defined in HCD. The single-token transition kernel is

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathrm{Cat}(\mathbf{x}_t; \alpha_t \mathbf{x}_0 + (1 - \alpha_t)\mathbf{e}_{[\mathrm{M}]}), \tag{4}$$

i.e. a token retains its value with probability α_t and is absorbed into [M] otherwise; full derivation is deferred to Subsection B.2.

In the forward diffusion phase, we adopt an **independent and symmetrical** noising strategy across the HCD hierarchy. Instead of embedding hierarchical constraints into the corruption process, we treat the three decomposed sequences $\mathbf{S}_0 = \{S_0^{(1)}, S_0^{(2)}, S_0^{(3)}\}$ as independent channels.

Formally, for any time step $t \in [0, 1]$, the joint transition probability factorizes over the three granularity levels. Utilizing the single-token transition kernel defined in Eq. 4, the joint distribution is given by:

$$q(\mathbf{S}_t|\mathbf{S}_0) = \prod_{k=1}^{3} q(S_t^{(k)}|S_0^{(k)}) = \prod_{k=1}^{3} \prod_{i=1}^{2N} q(x_{t,i}^{(k)}|x_{0,i}^{(k)}). \tag{5}$$

Noise is applied to the hundreds, tens, and units sequences with the same schedule α_t and *without cross-channel interaction in the forward kernel*.

This is the base corruption kernel. During staged training, higher-order channel positions are then clamped to conditioning digits before the forward pass (Subsection 2.4); consequently the model input is not simply the symmetrically corrupted state. At inference, no ground-truth channel is supplied.

Reverse process. The reverse process approximates $p_\theta(\mathbf{S}_0 \mid \mathbf{S}_t, I, T_{\text{ins}})$ with a single Transformer f_θ that, at every denoising step, ingests the joint noisy state $\mathbf{S}_t$ across all three digit channels and emits categorical distributions over the clean tokens $\hat{\mathbf{S}}_0$ for all channels in parallel. The hierarchical structure of HCD is imposed not by separate per-channel networks, but by how training factorises the joint posterior:

$$p_\theta(\mathbf{S}_0 \mid \mathbf{S}_t) = p_\theta(S_0^{(1)} \mid \mathbf{S}_t) \cdot p_\theta(S_0^{(2)} \mid \mathbf{S}_t, S_0^{(1)}) \cdot p_\theta(S_0^{(3)} \mid \mathbf{S}_t, S_0^{(1)}, S_0^{(2)}), \tag{6}$$

with t, I, T_{ins} implicit.

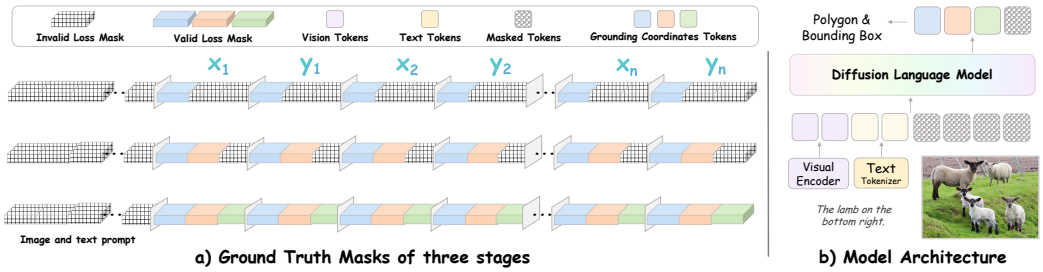

Figure 4: **Training process and model architecture.** a) The Hierarchical Curriculum Learning (HCL) process refines the output by progressively supervising the hundreds, tens, and finally all digits. b) The model, comprising a visual encoder, text tokenizer, and DLM, which processes image and text inputs to generate coordinates.

Equation (6) describes the training-time conditioning structure. It does not impose sequential per-channel inference: one call to the network in Algorithm 1 emits all three channels jointly. Supplying clean higher-order digits during training nevertheless creates an exposure-bias gap, because those digits are not available at test time.

2.4 Training Strategy: Hierarchical Curriculum Learning

To effectively optimize the factorized posterior defined in Subsection 2.3, we employ a **Hierarchical Curriculum Learning (HCL)** strategy (illustrated in Figure 4). Instead of learning the joint distribution $p_\theta(\mathbf{S}_0 \mid \mathbf{S}_t)$ from scratch, HCL decomposes the optimization into three progressive stages, stabilizing the learning of higher-order structures before introducing fine-grained details.

The measured main-table recipe uses ground-truth (GT) higher-order digits. For stage k, construct $\widetilde{\mathbf{S}}_t^{[k]}$ by replacing the positions of all channels $j < k$ in $\mathbf{S}_t$ with clean $S_0^{(j)}$ before the forward pass; the current channel remains corrupted. This is input conditioning (teacher forcing), not merely loss masking. The masked-token objective is

$$\mathcal{L}_k(\theta) = \mathbb{E}_{t,\mathbf{S}_0,\mathbf{S}_t}\left[-\sum_{i=1}^{2N} \mathbb{1}_{S_{t,i}^{(k)}=[\mathrm{M}]} \log p_\theta(S_{0,i}^{(k)} \mid \widetilde{\mathbf{S}}_t^{[k]}, t, I, T_{\mathrm{ins}})\right]. \tag{7}$$

The total loss progressively activates the hundreds, tens, and units objectives at thresholds G_1, G_2:

$$\mathcal{L}_{\mathrm{total}}(g,\theta) = \mathcal{L}_1(\theta) + \mathbb{1}_{g\geq G_1}\mathcal{L}_2(\theta) + \mathbb{1}_{g\geq G_2}\mathcal{L}_3(\theta). \tag{8}$$

All channels instead start masked and are denoised jointly at inference, so erroneous higher digits can affect finer predictions. In the *annealed self-conditioning variant*, higher-channel conditioning uses GT digits with probability p and the model's own argmax digits otherwise; p decreases linearly from 1 to 0 across stages 2–3. This improves the independently measured RefCOCOg-test cell from 23.2 to 24.6 (Table 20). An auxiliary continuous coordinate L_1 loss is examined separately (Table 21); these refinements are not jointly evaluated in the retained main tables.

2.5 Block-wise Top-k Decoding

We also evaluate a **block-wise top-k** decoder. Coordinates are partitioned into contiguous blocks of b positions within each digit channel. At each step, a shortlist of the top three digit candidates per position limits joint assignment scoring; the top-k surviving candidate states are evaluated in one batched network call. Decoding stops once no positions remain masked. Larger b can reduce the number of calls, while larger k increases candidate-scoring cost (Algorithm 2). These measurements concern a separate decoder from the token-wise main tables. Most of the original drop above 16 vertices arises from testing an $N = 16$-trained model at unseen vertex counts; matched-count training still leaves a smaller, consistent block-wise advantage (Table 22).

3 Experiments

Figure 5 previews the qualitative denoising trajectories. Subsection 3.1 states the setup; Subsection 3.2 reports main results; Subsection 3.3 ablates DFT's components; Subsection 3.4 measures decoding scalability and inference cost.

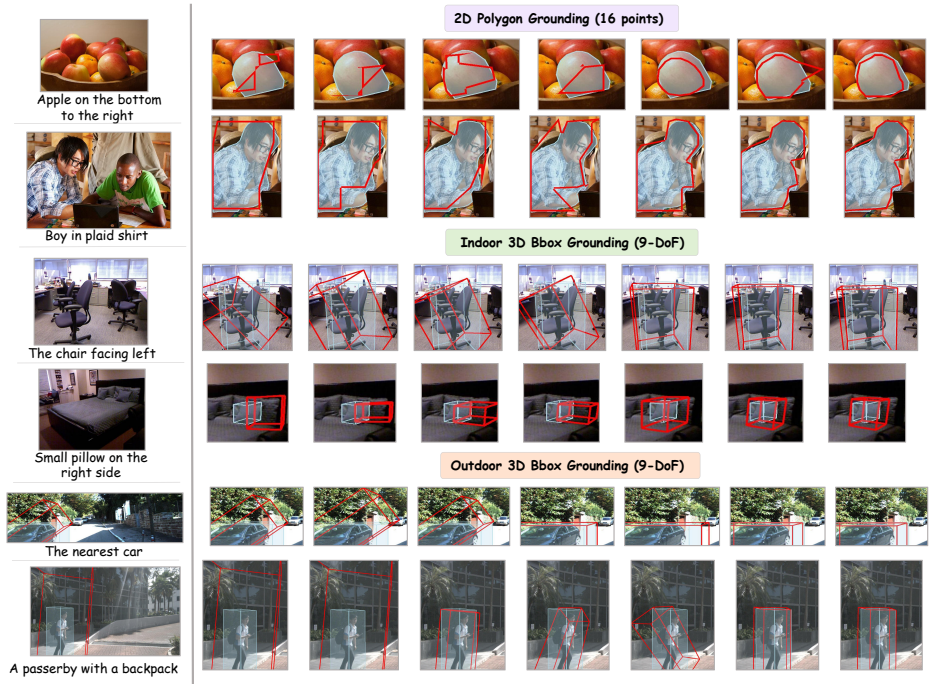

Figure 5: **Qualitative results.** DFT iteratively refines outputs across three task families: 2D polygon grounding (16 vertices), indoor 9-DoF 3D bounding-box grounding, and outdoor 9-DoF 3D bounding-box grounding. Each row shows the denoising trajectory from a fully masked state to the final prediction.

3.1 Experimental Setup

The LLaVA-OV comparison uses diffusion-native LLaDA-V [You et al., 2025] (LLaDA-8B [Nie et al., 2025] + SigLIP2 [Tschannen et al., 2025]); the Qwen models are first converted from AR checkpoints to mask denoisers, as detailed in Subsection L.1. Grounding fine-tuning uses AdamW (10^{-5}, cosine), with matched grounding data, schedule, and resolution for AR comparisons. Main Tables 1–3 retain GT-conditioned HCL and token-wise top-1 decoding at $K = 12$. Every original polygon model is trained at $N = 16$; other vertex counts in Tables 4 and 6 are out-of-distribution (OOD) tests unless explicitly marked as matched training.

We evaluate 2D bbox on RefCOCO/+/g [Mao et al., 2016], 16-point RefCOCO-Polygon, and 9-DoF monocular grounding on Ref3D's four indoor and two outdoor sources (Section H). We report Acc@τ at $\tau = 0.5$ for bbox, 0.1 for polygon and 0.3 for 3D, with identical metric implementations across methods; additional 3D thresholds appear in Table 15. SeqTR [Zhu et al., 2022] and PVD [Cheng et al., 2024] are specialist decoders without LVLM backbones. $K = 12$ is selected on validation: the token-wise sweep improves only 0.5 points from $K = 12$ to 20 while latency rises from 124 to 206 ms (Table 26).

3.2 Main Results

We compare three model lineages: the LLaVA-OV/LLaDA-V pair, Qwen2.5-VL, and Qwen3-VL converted to DLMs. These are not AR models with a replaced output head. Grounding fine-tuning is matched; conversion and the additional-training control are reported separately in Subsection L.1.

Bounding-box grounding. On RefCOCO/+/g (Table 2), DFT yields consistent gains over its AR counterpart on every backbone: $+2.5$ to $+3.7$ points for the LLaVA-OV pair, $+0.6$ to $+1.8$ for Qwen2.5-VL, and $+1.5$ to $+2.0$ for Qwen3-VL across the eight splits. Bbox is a coarse target on which strong AR LVLMs already saturate near 90 Acc@0.5 on the easier splits; this confirms that DFT does not regress on the easy task while delivering the larger gains documented below.

Polygon grounding. On our 16-point RefCOCO-Polygon split (Table 1), DFT improves every backbone by a large margin under matched fine-tuning: LLaVA-OV $11.4 \rightarrow 23.2$ ($+11.8$), Qwen2.5-VL $18.4 \rightarrow 25.2$ ($+6.8$), and Qwen3-VL $19.7 \rightarrow 26.4$ ($+6.7$) on RefCOCOg-test. The gain is

Table 1: **16-point polygon grounding (Acc@0.1) on RefCOCO-Polygon.** The upper block is zero-shot; the lower block uses matched grounding fine-tuning. All DFT rows retain GT-conditioned HCL and token-wise top-1 decoding. Separate decoder/conditioning comparisons are in Section M. • marks the LLaVA-OV/LLaDA-V lineage comparison; ○ marks other initialisations. These symbols do not imply identical AR and diffusion checkpoints.

Polygon Grounding	Size	RefCOCO			RefCOCO+			RefCOCOg	
		val	testA	testB	val	testA	testB	val	test
Zero-shot evaluation									
○ LLaVA 1.5 [Liu et al., 2024a]	7B	5.8	6.5	5.1	5.4	6.0	4.8	7.9	8.1
○ LLaVA-Next [Liu et al., 2024b]	7B	10.5	11.2	9.8	9.9	10.6	9.1	11.5	11.8
○ InternVL-3 [Chen et al., 2024]	8B	13.1	14.0	12.5	12.5	13.3	11.8	14.5	14.8
○ InternVL-3 [Chen et al., 2024]	9B	14.2	15.1	13.6	13.7	14.6	13.0	15.8	16.1
○ Qwen2-VL [Wang et al., 2024]	2B	13.8	14.6	13.2	13.1	13.9	12.5	15.0	15.3
○ Qwen2-VL [Wang et al., 2024]	7B	15.8	16.7	15.2	15.3	16.2	14.7	17.1	17.4
○ Qwen2.5-VL [Bai et al., 2025]	3B	13.3	14.1	12.8	12.7	13.5	12.1	14.7	15.0
○ Qwen2.5-VL [Bai et al., 2025]	7B	15.5	16.4	14.9	15.0	15.9	14.4	16.8	17.1
○ Qwen3-VL [Team, 2025]	8B	16.5	17.5	15.9	16.0	17.0	15.4	17.8	18.1
After Fine-tuning									
○ SeqTR [Zhu et al., 2022]	–	16.2	17.0	15.4	15.6	16.4	14.8	17.1	17.5
○ PVD [Cheng et al., 2024]	–	18.4	19.2	17.6	17.8	18.6	16.9	19.0	19.4
• LLaVA OneVision [Li et al., 2024a]	7B	10.8	11.5	9.4	10.1	10.9	9.2	11.9	11.4
• **LLaVA OneVision + DFT**	**8B**	**21.8**	**22.5**	**20.9**	**21.1**	**21.9**	**20.2**	**22.8**	**23.2**
○ Qwen2.5-VL [Bai et al., 2025]	7B	17.0	17.9	16.3	16.5	17.4	15.7	18.1	18.4
○ **Qwen2.5-VL→DLM + DFT**	**7B**	**23.5**	**24.4**	**22.5**	**22.8**	**23.7**	**21.7**	**24.5**	**25.2**
○ Qwen3-VL [Team, 2025]	8B	18.2	19.1	17.4	17.7	18.6	16.8	19.2	19.7
○ **Qwen3-VL→DLM + DFT**	**8B**	**24.7**	**25.6**	**23.6**	**23.9**	**24.8**	**22.8**	**25.7**	**26.4**

Table 2: **Bbox grounding on RefCOCO.** DFT: GT-conditioned HCL, token-wise top-1.

BBOX Grounding	RefCOCO			RefCOCO+			RefCOCOg	
	val	testA	testB	val	testA	testB	val	test
After Fine-tuning								
○ KOSMOS-2	52.3	57.4	47.3	45.5	50.7	42.2	60.6	61.6
○ Shikra 7B	87.0	90.6	80.2	81.6	87.4	72.1	82.3	82.2
○ OFA-L	80.0	83.7	76.4	68.3	76.0	61.8	67.6	67.6
○ InternVL-2 2B	82.3	88.2	75.9	73.5	82.8	63.3	77.6	78.3
○ InternVL-2 8B	87.1	91.1	80.7	79.8	87.9	71.4	82.7	82.7
○ Qwen2-VL 2B	87.6	90.6	82.3	79.0	84.9	71.0	81.2	80.3
○ Qwen2-VL 7B	91.7	93.6	87.3	85.8	90.5	79.5	87.3	87.8
○ Qwen2.5-VL 3B	89.1	91.7	84.0	82.4	88.0	74.1	85.2	85.7
• LLaVA-OV 7B	88.5	90.7	85.0	82.5	87.5	75.8	85.5	85.8
• **LLaVA-OV + DFT**	**92.2**	**93.8**	**88.0**	**85.9**	**90.8**	**79.4**	**88.0**	**88.4**
○ Qwen2.5-VL 7B	90.0	92.5	85.4	84.2	89.1	76.9	87.2	87.2
○ **Qwen2.5-VL→DLM + DFT**	**91.8**	**93.7**	**86.8**	**85.2**	**90.2**	**77.8**	**87.8**	**88.0**
○ Qwen3-VL 8B	91.5	93.5	86.8	85.5	90.2	78.5	87.5	88.0
○ **Qwen3-VL→DLM + DFT**	**93.5**	**95.0**	**88.8**	**87.2**	**91.7**	**80.2**	**89.2**	**89.5**

Table 3: **3D BBox grounding** (Acc@0.3). DFT: GT-conditioned HCL, token-wise top-1.

3D BBOX Grounding	Indoor 3D Datasets				Outdoor	
	Objtro.	SUN R.	ARKit.	Hyper.	KITTI	nuSc.
After Fine-tuning						
○ LLaVA 1.5 7B	40.7	23.1	1.1	4.2	2.3	1.4
○ LLaVA-Next 7B	41.0	23.5	1.2	4.4	2.4	1.5
○ InternVL-2.5 7B	45.3	35.7	1.5	3.9	25.0	3.2
○ InternVL-3 8B	47.0	37.0	1.8	4.5	27.3	4.0
○ InternVL-3 9B	48.9	38.5	2.5	6.8	32.0	5.8
○ Qwen2-VL 2B	44.4	36.3	2.1	4.6	36.1	4.5
○ Qwen2-VL 7B	51.0	38.4	2.0	4.3	27.2	3.8
○ Qwen2.5-VL 3B	14.5	39.0	1.2	3.7	6.6	1.8
• LLaVA-OV 7B	43.2	25.0	1.5	5.1	2.9	2.0
• **LLaVA-OV + DFT**	**52.2**	**39.3**	**4.8**	**9.1**	**39.7**	**8.4**
○ Qwen2.5-VL 7B	15.5	39.0	4.6	3.8	2.0	1.2
○ **Qwen2.5-VL→DLM + DFT**	**22.0**	**41.0**	**5.5**	**4.8**	**5.0**	**2.5**
○ Qwen3-VL 8B	35.0	40.0	3.5	5.0	28.0	4.0
○ **Qwen3-VL→DLM + DFT**	**45.0**	**42.0**	**5.5**	**6.5**	**34.0**	**5.5**

reproduced across the reported splits. The AR channel-major+HCL control reaches 16.8 versus 23.2 for the submitted diffusion recipe in a controlled RefCOCOg-test experiment (Table 19), separating shared representation/curriculum benefits from the remaining diffusion advantage.

Monocular 3D grounding. We extend the same model to 9-DoF monocular 3D grounding (centre, dimensions, orientation) on Ref3D without architectural change. Table 3 reports per-dataset Acc@0.3 (full thresholds in Table 15). Within the reported model-lineage comparisons, the largest gains occur on the LLaVA-OV/LLaDA-V pair: $43.2 \rightarrow 52.2$ on Objectron, $25.0 \rightarrow 39.3$ on SUN RGB-D, and $2.9 \rightarrow 39.7$ on KITTI; gains on the Qwen2.5-VL and Qwen3-VL pairs are also positive across all six datasets, including the harder ARKitScenes/Hypersim regimes. We do not claim 3D SOTA against specialist driving detectors; we show that an LVLM can be repurposed to 9-DoF grounding through DFT.

3.3 Ablation Study

Hierarchical Curriculum Learning. Figure 6 traces per-stage Acc on the three task families. Bounding-box accuracy is essentially set by Stage 1 and saturates afterwards, whereas polygon and 9-DoF 3D grounding continue to gain substantially through Stages 2–3, with polygon showing the

Table 4: **Coordinate representation and test vertex count.** RefCOCO-Polygon, GT-conditioned HCL, token-wise decoding. Models train at $N = 16$; $N = 8/24$ are OOD tests.

Coordinate Type	Point Number	RefCOCO-Polygon		
		val	testA	testB
0.00 - 0.99	16	16.5	18.1	17.4
0.000 - 0.999	16	17.3	20.0	17.6
Normed 000-999	8	**23.4**	**25.5**	**23.9**
Normed 000-999	16	**21.5**	**22.1**	**20.8**
Normed 000-999	24	4.5	4.8	4.1

Table 5: **Ablation study on 3D parameter normalization.** The results clearly highlight the critical impact of depth (z) normalization, especially for outdoor scenes such as KITTI.

3D Normlization Setting	Indoor SUN R.	Outdoor KITTI
Baseline: All Normed	**39.3**	**39.7**
Without xy normalization	33.1	34.5
Without z normalization	23.8	25.2
Without dimension normalization	34.0	35.9
Without rotation angle normalization	35.5	33.1

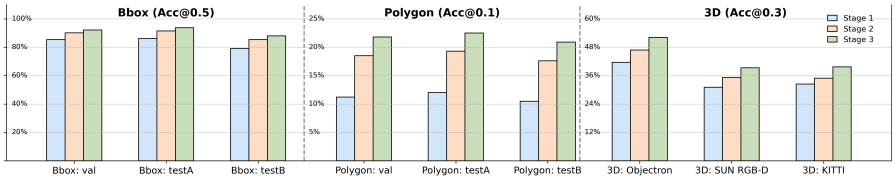

Figure 6: **HCL stage progression.** Per-stage performance gains across 2D bbox, 16-point polygon, and 9-DoF 3D grounding.

largest jump as units-digit precision is resolved. The pattern confirms that the more complex the target representation, the more it benefits from the coarse-to-fine schedule.

Coordinate representation and polygon complexity. Table 4 compares encodings at fixed $N = 16$. Place-value grouping gives a useful coarse-to-fine bias, but digit cross-entropy does not explicitly model spatial distance. Separate ordinal-aware alternatives and annealed self-conditioning appear in Tables 20 and 21. The $N = 8/24$ rows test the $N = 16$-trained model OOD and should not be interpreted as matched-count decoder limits.

3D parameter normalisation. Table 5 ranks the four normalisation channels by ablation impact. Removing depth (z) normalisation drops Acc@0.3 by -15.5 on SUN RGB-D and -14.5 on KITTI—roughly $3\times$ the impact of removing rotation normalisation—while xy and dimension normalisation sit in between. Accurate depth normalisation is therefore the single most critical factor for monocular 3D grounding.

3.4 Decoding Scalability and Inference Cost

Table 6 retains the original $N = 16$-trained scalability run, reporting each decoder separately. Its token-wise RefCOCO val/testA/testB values (21.5/22.1/20.8) come from a different run from Table 1 (21.8/22.5/20.9); we do not treat them as identical measurements.

Vertex count. At test counts 24 and 32, the original token-wise accuracies fall to 4.5 and 1.8. Retraining at the matching counts raises these to 15.8 and 11.2; block-wise decoding reaches 23.5 and 20.6 (Table 22). Thus most of the original collapse reflects count mismatch, while a 7.7–12.5 point block-wise advantage remains under matched-count training. Long contours remain difficult even after that mismatch is removed.

Inference cost and search. At $N = 16$, token-wise DFT takes 124 ms versus AR greedy's 41 ms, while block-wise $(4, 2)$ takes 73 ms. Separate best-of-8 and beam-search AR controls, candidate-selection rules, and reported network-call counts appear in Table 24; wall-clock is essential because KV-cached AR calls and full-sequence denoising calls have different costs. The no-search DFT comparison and block-wise search gains are reported separately, rather than attributing the latter to the training formulation. Full (k, b) and K sweeps are in Section D.

4 Conclusion and Limitations

Conclusion. DFT adapts diffusion language models to structured visual grounding through place-value coordinate channels and a coarse-to-fine curriculum. The measured GT-conditioned, token-wise recipe improves 16-point polygon grounding and extends to monocular 9-DoF boxes. Controlled AR ordering/curriculum, matched-count and search comparisons clarify which gains come from training,

Table 6: **Original decoding/scalability run** on RefCOCO-Polygon (Acc@0.1). All models train at $N = 16$; tests at $N = 24/32$ are OOD. DFT uses GT-conditioned HCL, with each decoder labelled separately. Latency/memory: single A800 80GB, batch 1, 1000-image average, maximum $K = 12$. These are separate-run measurements from the main tables, not their block-wise replacement.

Method	Decoder	k	b	K	N	ms/img	Mem (GB)	Acc@0.1		
								val	testA	testB
$N = 16$ *vertices*										
AR baseline	top-1	–	–	–	16	41	14.2	11.9	12.6	11.0
DFT (token-wise)	top-1	1	1	12	16	124	17.5	21.5	22.1	20.8
DFT (block-wise)	top-k	2	2	12	16	56	17.6	28.5	29.2	27.7
DFT (block-wise)	top-k	4	2	12	16	73	17.8	**34.0**	**34.8**	**33.0**
DFT (block-wise)	top-k	8	2	12	16	102	18.0	34.4	35.1	33.4
DFT (block-wise)	top-k	4	4	12	16	65	17.9	32.6	33.3	31.8
$N = 24$ *vertices*										
AR baseline	top-1	–	–	–	24	58	14.4	7.2	7.6	6.7
DFT (token-wise)	top-1	1	1	12	24	156	18.0	4.5	4.8	4.1
DFT (block-wise)	top-k	4	2	12	24	90	18.2	18.0	18.6	17.3
DFT (block-wise)	top-k	4	4	12	24	85	18.3	**21.0**	**21.5**	**20.2**
DFT (block-wise)	top-k	8	4	12	24	124	18.5	21.4	22.0	20.6
$N = 32$ *vertices*										
AR baseline	top-1	–	–	–	32	74	14.6	4.8	5.2	4.4
DFT (token-wise)	top-1	1	1	12	32	188	18.5	1.8	2.0	1.6
DFT (block-wise)	top-k	4	4	12	32	110	18.7	16.5	17.0	15.8
DFT (block-wise)	top-k	4	6	12	32	105	18.8	**19.2**	**19.8**	**18.4**
DFT (block-wise)	top-k	8	6	12	32	148	19.0	19.5	20.1	18.7

count generalization and decoding. Annealed conditioning and ordinal-aware objectives are reported as separate refinements, without assuming their gains combine.

Limitations. Clean higher-channel conditioning introduces exposure bias; annealing reduces but does not remove its measured oracle gap. Digit cross-entropy does not encode numerical distance and the three-digit range limits resolution. Fixed-N polygons can waste vertices on simple objects and miss thin or disconnected regions; masks may be preferable for pixel-exact boundaries, and variable-length polygons remain untested. Higher-count tests must distinguish OOD evaluation from matched training. Denoising remains slower than greedy AR in the reported $K = 12$ settings, and Ref3D target selection and language can inherit annotation-model biases. Detailed controlled experiments and data-construction qualifications are in Sections H and M.

Acknowledgments and Disclosure of Funding

This work is supported by the Hong Kong Research Grant Council General Research Fund (No. 17213925) and National Natural Science Foundation of China (No. 62422606, 62441615).

References

Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *CVPR*, 2021.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

Shuai Bai et al. Qwen2.5-vl technical report. *arXiv:2502.13923*, 2025.

Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *NeurIPS*, 2021.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Zhong-Qiu Zhao, Xizhou Zhu, Lewei Lu, Xiaokang Chen, Yikang Shen, Chenxin Tao, Linkai Wei, Yu-Cheng Cheng, Lu-Yao Wang, Hong-Bo Zhang, Yi-Pan Wang, Wen-Feng Song, Zhi-Tian Zhang, Day-Day Up, Yali Wang, Dai-Qing Ya, Jifeng Dai, Yu Qiao, and Tong Lu. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic capabilities. In *CVPR*, 2024.

Zesen Cheng, Kehan Li, Peng Jin, Siheng Li, Xiangyang Ji, Lu Yuan, Chang Liu, and Jie Chen. Parallel vertex diffusion for unified visual grounding. In *AAAI*, volume 38, pages 1326–1334, 2024.

Can Cui, Yupeng Zhou, Juntong Peng, Sung-Yeon Park, Zichong Yang, Prashanth Sankaranarayanan, Jiaru Zhang, Ruqi Zhang, and Ziran Wang. Vilad: A large vision language diffusion framework for end-to-end autonomous driving. *arXiv:2508.12603*, 2025.

Xuanzhao Dong, Wenhui Zhu, Xiwen Chen, Zhipeng Wang, Peijie Qiu, Shao Tang, Xin Li, and Yalin Wang. Llada-medv: Exploring large language diffusion models for biomedical image understanding. *arXiv:2508.01617*, 2025.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *ICLR*, 2025.

Dong Guo, Faming Wu, Feida Zhu, et al. Seed1.5-VL technical report. *arXiv preprint arXiv:2505.07062*, 2025. URL `https://arxiv.org/abs/2505.07062`.

Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. Vtimellm: Empower llm to grasp video moments. In *CVPR*, 2024.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *Trans. Mach. Learn. Res.*, 2024a.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *arXiv:2508.00819*, 2025.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *NeurIPS*, 2022.

Zhaowei Li, Qi Xu, Dong Zhang, Hang Song, Yiqing Cai, Qi Qi, Ran Zhou, Junting Pan, Zefeng Li, Vu Tu, et al. Groundinggpt: Language enhanced multi-modal grounding model. In *ACL*, 2024b.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv:2508.20072*, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024a.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge, January 2024b. URL `https://llava-vl.github.io/blog/2024-01-30-llava-next/`.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana-Maria Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv:2502.09992*, 2025.

OpenAI. Hello gpt-4o, May 2024. URL `https://openai.com/index/hello-gpt-4o/`.

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shuming Ma, and Furu Wei. KOSMOS-2: Grounding multimodal large language models to the world. In *ICLR*, 2024.

Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M. Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S. Khan. Glamm: Pixel grounding large multimodal model. In *CVPR*, 2024.

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 *ICCV*, 2021.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Weizhi Ma, Yaqing Zhang, Yong-Xu Wu, and Hao-Min Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv:2508.02193*, 2025.

Qwen Team. Qwen3-vl technical report. *arXiv preprint*, 2025.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv:2502.14786*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv:2409.12191*, 2024.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. In *NeurIPS*, 2023.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv:2505.15809*, 2025.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. URL `https://arxiv.org/abs/2508.15487`.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv:2505.16933*, 2025.

Ao Zhang, Yuan Yao, Wei Ji, Zhiyuan Liu, and Tat-Seng Chua. NExT-Chat: An LMM for chat, detection and segmentation. In *ICLR*, 2024.

Yang Zhao, Zhijie Lin, Daquan Zhou, Zilong Huang, Jiashi Feng, and Bingyi Kang. Bubogpt: Enabling visual grounding in multi-modal llms. *arXiv:2307.08581*, 2023.

Chaoyang Zhu, Yiyi Zhou, Yunhang Shen, Gen Luo, Xingjia Pan, Mingbao Lin, Chao Chen, Liujuan Cao, Xiaoshuai Sun, and Rongrong Ji. Seqtr: A simple yet universal network for visual grounding. In *ECCV*, 2022.

A Related Work

LVLMs for grounding. The visual grounding capabilities of Large Vision-Language Models (LVLMs) have evolved significantly from holistic semantic understanding to fine-grained region-level localization. Initial efforts, such as BuboGPT [Zhao et al., 2023], adopted an 'off-the-shelf module' strategy, relying on separate visual models to locate entities, which limited deep cross-modal fusion. To achieve tighter vision-language integration, mainstream research has shifted towards end-to-end solutions that make grounding an inherent model capability. These solutions primarily follow two technical paradigms: the 'pixel-to-sequence' (pix2seq) approach, seen in models like Kosmos-2 [Peng et al., 2024] and GroundingGPT [Li et al., 2024b], which serialise coordinates into text tokens for unified prediction; and the 'pixel-to-embedding' (pix2emb) approach, pioneered by NExT-Chat [Zhang et al., 2024], which outputs location embeddings that can be decoded into bounding boxes or even pixel-level masks, as demonstrated by GLaMM [Rasheed et al., 2024]. Furthermore, addressing the demands of dynamic scenes such as games, cutting-edge works like VTimeLLM [Huang et al., 2024] have successfully extended grounding from the spatial to the temporal domain across video and audio, laying a solid foundation for precise interaction in complex multimodal environments. The closest specialist to our setting is SeqTR [Zhu et al., 2022], a sequence transformer for native polygon vertex sequences. All of the vertex-token decoders above produce coordinates left-to-right; this is the property our digit-level diffusion formulation removes.

Multi-modal diffusion language models. To overcome the slow, unidirectional limitations of autoregressive (AR) models [OpenAI, 2024, Bai et al., 2025], researchers have applied diffusion language models (DLMs) [Li et al., 2022] for parallel and bidirectional text generation using techniques such as masked diffusion [Austin et al., 2021, Nie et al., 2025, Gong et al., 2025, Li et al., 2025]. Building on their success in text, diffusion models were rapidly extended to multimodal tasks by using cross-modal conditions. In the vision-language domain, LLaDA-V [You et al., 2025] pioneered this approach for VQA tasks, paving the way for more general multimodal diffusion language model frameworks [Cui et al., 2025, Yang et al., 2025, Song et al., 2025]. The methodology has also been adopted in specialised fields such as biomedical image understanding [Dong et al., 2025] and embodied intelligence [Liang et al., 2025].

Diffusion grounding and how DFT differs. Most directly related is Parallel Vertex Diffusion (PVD) [Cheng et al., 2024], which also denoises polygon vertices but operates on a continuous vertex space with a task-specific diffusion head. DFT differs in three concrete ways: (i) operating on a discrete digit alphabet inside a general LVLM/DLLM backbone (vs. a separate continuous head), (ii) coupling diffusion with hierarchical coordinate decomposition (Subsection 2.2), and (iii) curriculum-staged supervision (Subsection 2.4). To our knowledge, DFT is the first DLLM-based formulation to unify 2D bounding-box, 16-point polygon, and 9-DoF monocular 3D grounding under a single weight set.

B Proofs of Theorems and Lemmas

B.1 Preliminary and Notation

Diffusion Models are latent variable-based generative frameworks composed of two reciprocal Markov chains: a forward noising process and a reverse denoising process. **The fixed forward process** progressively injects Gaussian noise into data samples $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ over T steps. This transition transforms $\mathbf{x}_0$ into near-standard normal noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The single-step transition is defined as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{9}$$

where $\beta_t \in (0, 1)$ is a hyperparameter controlling the noise schedule. Consequently, the joint probability of the forward trajectory factorizes as:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{10}$$

The learnable reverse process is parameterized by a neural network θ trained to invert the corruption. Starting from $\mathbf{x}_T$, it generates samples matching the data distribution via the joint distribution:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{11}$$

Training optimizes θ by minimizing the Variational Lower Bound (VLB) loss $\mathcal{L}_{\text{VLB}}$, which aligns the predicted denoising distribution with the ground-truth forward posterior.

Discrete Diffusion for Discrete Data (Text). For data represented as discrete tokens (e.g., text or quantized coordinates), we utilize a specialized discrete diffusion framework. Let $\mathbf{x}_t \in \{0,1\}^K$ denote a one-hot vector where K is the vocabulary size. The forward transition from $t-1$ to t is governed by a categorical distribution:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \text{Cat}(\mathbf{x}_t; \mathbf{Q}_t^\top \mathbf{x}_{t-1}), \tag{12}$$

where $\mathbf{Q}_t \in [0,1]^{K\times K}$ is the state transition matrix.

We adopt *Absorbing Discrete Diffusion*. In this formulation, a token either remains unchanged or transitions to an absorbing mask state [M]. The single-step transition matrix is defined as:

$$\mathbf{Q}_t = (1-\beta_t)\mathbf{I} + \beta_t \mathbf{1}\mathbf{e}_{[\text{M}]}^\top, \tag{13}$$

where β_t is the noise scalar, $\mathbf{I}$ is the identity matrix, and $\mathbf{e}_{[\text{M}]}$ is the one-hot vector for the mask token.

By hierarchically multiplying these transition matrices, we can derive the cumulative distribution from $\mathbf{x}_0$ directly to $\mathbf{x}_t$. Let $\alpha_t = \prod_{i=1}^{t}(1-\beta_i)$. The marginal distribution simplifies to:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \alpha_t \mathbf{x}_0 + (1-\alpha_t)\mathbf{e}_{[\text{M}]}). \tag{14}$$

As t increases, $\alpha_t \to 0$, causing $\mathbf{x}_T$ to approach a fully masked state. To overcome the discretization limits of fixed steps, we extend this framework to continuous time ($t \in [0,1]$) by taking the limit $T \to \infty$. This allows us to define the arbitrary transition $q(\mathbf{x}_t|\mathbf{x}_s)$ for any $0 \le s < t \le 1$, as detailed subsequently.

B.2 Continuous-time Discrete Diffusion Processes

Continuous-Time State Transition. Following Eq.14 and the Markov property $q(\mathbf{x}_t|\mathbf{x}_0) = \sum_{\mathbf{x}_s} q(\mathbf{x}_t|\mathbf{x}_s)q(\mathbf{x}_s|\mathbf{x}_0)$, the forward transition from an earlier noisy state $\mathbf{x}_s$ to a later state $\mathbf{x}_t$ ($0 \le s < t \le 1$) is:

$$q(\mathbf{x}_t|\mathbf{x}_s) = \text{Cat}\left(\mathbf{x}_t; \frac{\alpha_t}{\alpha_s}\mathbf{x}_s + \left(1 - \frac{\alpha_t}{\alpha_s}\right)\mathbf{e}_{[\text{M}]}\right). \tag{15}$$

This indicates that from s to t, an unmasked token retains its value with probability α_t/α_s and transitions to the absorbing state $\mathbf{e}_{[\text{M}]}$ with probability $1-\alpha_t/\alpha_s$.

True Posterior. Given the clean data $\mathbf{x}_0$ and noisy data $\mathbf{x}_t$, we derive the *true posterior distribution* $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$, which serves as the regression target for the reverse process:

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_s)q(\mathbf{x}_s|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} = \begin{cases} \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mathbf{x}_0 + \frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_{[\text{M}]} & \text{if } \mathbf{x}_t = \mathbf{e}_{[\text{M}]}, \\ \mathbf{x}_0 & \text{if } \mathbf{x}_t \neq \mathbf{e}_{[\text{M}]}. \end{cases} \tag{16}$$

Interpretation: 1) If a token is *visible* at time t ($\mathbf{x}_t \neq \mathbf{e}_{[\text{M}]}$), it must have been the identical original token $\mathbf{x}_0$ at time s. 2) If it is *masked* at t, its state at s is a superposition of the original token $\mathbf{x}_0$ and the mask $\mathbf{e}_{[\text{M}]}$.

Modeling and Training Objective. Our goal is to train a **denoising model** $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ (parameterized by a network f_θ) to approximate the true posterior. The objective minimizes the KL divergence between the predicted distribution and the true posterior. In the continuous-time limit ($T \to \infty$), the ELBO simplifies to an integral over $t \in [0,1]$:

$$\mathcal{L} = \int_0^1 w(t) \cdot \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\mathcal{L}_{\text{CE}}(\mathbf{x}_0, p_\theta(\mathbf{x}_t))\right] dt. \tag{17}$$

The loss term focuses solely on masked regions. Adopting the linear noise schedule $\alpha_t = 1 - t$, the weighting term simplifies to $w(t) = \frac{1}{t}$. This schedule assigns higher importance to early-stage denoising (small t).

Final Loss for Coordinate Sequences. In practice, we apply this to a coordinate sequence $\mathbf{S} = (\mathbf{x}^1, \ldots, \mathbf{x}^{2N})$. We estimate the integral via Monte Carlo sampling. For a randomly sampled time $t \sim \mathcal{U}(0, 1)$, the loss is:

$$\mathcal{L}_t = \frac{1}{t} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[- \sum_{i=1}^{2N} \mathbb{1}_{\mathbf{x}_t^i = \mathbf{e}_{[\mathrm{M}]}} \log p_\theta(\mathbf{x}_0^i \mid \mathbf{x}_t) \right]. \tag{18}$$

Here, $p_\theta(\cdot \mid \mathbf{x}_t)$ is the categorical probability predicted by the model. The indicator function $\mathbb{1}_{\mathbf{x}_t^i = \mathbf{e}_{[\mathrm{M}]}}$ ensures that gradients are backpropagated only from tokens that are currently masked, consistent with the definition of the true posterior.

C Inference Pseudocode

Algorithm 1 Parallel Hierarchical Denoising Inference

Input: Image I; Instruction T_{ins}; Total inference steps K; Polygon vertex count N.
Output: Final coordinate sequence S_{final}.

Stage 1: Initialization

Set time $t \leftarrow 1$
for $k \in \{1, 2, 3\}$ **do**
  $S_t^{(k)} \leftarrow [\text{M}]^{2N}$ {Hundreds/Tens/Units, all masked}
end for

Stage 2: Parallel Iterative Denoising

for step $i = K$ **down to** 1 **do**
  $t \leftarrow i/K$ {Current time}
  $s \leftarrow (i-1)/K$ {Next time step}
  {1. Predict clean data $\mathbf{x}_0$ from network}
  $\mathbf{P}_\theta \leftarrow f_\theta\left(\{S_t^{(1)}, S_t^{(2)}, S_t^{(3)}\}, t, I, T_{\text{ins}}\right)$
  {2. Sample next state S_s using the posterior $q(x_s|x_t, x_0)$}
  for $k \in \{1, 2, 3\}$ **do**
    $\hat{S}_0^{(k)} \leftarrow \text{argmax}(\mathbf{P}_\theta^{(k)})$ {Most likely clean tokens}
    for $j = 1$ **to** $2N$ **do**
      if $S_{t,j}^{(k)} \neq [\text{M}]$ **then**
        $S_{s,j}^{(k)} \leftarrow S_{t,j}^{(k)}$ {Visible tokens persist (Eq. 16)}
      else
        Sample $u \sim \mathcal{U}(0, 1)$
        Threshold $\tau \leftarrow \frac{1-\alpha_s}{1-\alpha_t}$ {Posterior mask probability}
        if $u < \tau$ **then**
          $S_{s,j}^{(k)} \leftarrow [\text{M}]$ {Remains masked}
        else
          $S_{s,j}^{(k)} \leftarrow \hat{S}_{0,j}^{(k)}$ {Unmasked to predicted value}
        end if
      end if
    end for
    Update state: $S_t^{(k)} \leftarrow S_s^{(k)}$
  end for
end for

Stage 3: Recomposition

Initialize S_{final}
for $j = 1$ **to** $2N$ **do**
  Retrieve digits: $d^{(1)} \leftarrow S_0^{(1)}[j]$, $d^{(2)} \leftarrow S_0^{(2)}[j]$, $d^{(3)} \leftarrow S_0^{(3)}[j]$
  Recompose coordinate: $c_j \leftarrow 100 \cdot d^{(1)} + 10 \cdot d^{(2)} + d^{(3)}$
  Append c_j to S_{final}
end for
return S_{final}

Algorithm 2 Block-wise Top-k Joint Denoising (batched candidates)

Input: Image I, instruction T_{ins}, maximum denoising steps K, vertex count N, beam width k, block size b in coordinate positions; digit shortlist size $m_0 = 3$.
Output: Final coordinate sequence S_{final}.

Partition each digit channel's $2N$ positions into $B = \lceil 2N/b \rceil$ contiguous blocks.
Initialize fully masked candidate states across all three channels; retain at most k states.
for step $i = K$ **down to** 1 **do**
  $t \leftarrow i/K$, $s \leftarrow (i-1)/K$.
  Run f_θ on all retained states in **one batched forward pass**, jointly predicting all three channels.
  for each retained candidate r, block B_l, and digit channel j **do**
    For each position $p \in B_l$, keep its top-m_0 predicted digit candidates $A_{r,j,p}$.
    Score assignments $\mathbf{c} \in \prod_{p \in B_l} A_{r,j,p}$ by $\sigma^r_{l,j}(\mathbf{c}) = \sum_{p \in B_l} \log P_\theta^{(r,j)}[p, c_p]$.
    Keep the top-k block assignments by their joint scores.
  end for
  Merge the candidate/block assignments and retain the global top-k joint states by additive log-likelihood.
  Apply the absorbing-state posterior (Equation (16)): visible digits persist; masked digits either remain masked or take the selected assignment.
  If no masked positions remain in the retained states, terminate the loop.
end for
Select the highest-likelihood retained state $r^\star$ and recompose $c_p = 100 S_0^{(1),r^\star}[p] + 10 S_0^{(2),r^\star}[p] + S_0^{(3),r^\star}[p]$.
return S_{final}.

D Additional Ablations

Table 7: **Hierarchical vs flat diffusion training** on RefCOCO-Polygon-val (Acc@0.1). GT-conditioned HCL; token-wise decoding. These are diffusion-only ablations; the AR controls are in Table 19.

Variant	RefCOCO-Polygon val
Flat 0–999 vocab, no HCD	12.4
HCD only (no curriculum)	17.6
HCL only (flat vocab, staged $\mathcal{L}$)	14.1
HCD + HCL (GT-conditioned)	**21.5**

Table 8: **Block-wise top-k decoder hyperparameter sweep** on RefCOCO-Polygon-val (16 vertices). Each cell reports val Acc@0.1 / end-to-end ms/img. All models train at $N = 16$; the highlighted decoder operating point ($k=4, b=2$) is highlighted.

$b \setminus k$	1	2	4	8
1 (token-wise)	21.5 / 124	24.8 / 138	27.4 / 162	28.5 / 199
2	26.7 / 50	28.5 / 56	**34.0 / 73**	34.4 / 102
4	24.0 / 39	28.5 / 44	32.6 / 65	32.8 / 98
6	21.0 / 31	25.4 / 36	28.9 / 42	29.4 / 64

The decoding-scalability and latency results have been promoted to the main paper (Subsection 3.4, Table 6); we report here the remaining (k, b), K, resolution, hier-vs-flat, stage-skipping, noise-schedule, and backbone-scaling sweeps.

Hier-vs-flat (Table 7). Removing either HCD or HCL drops val Acc@0.1 by 3.9–7.4 points; the two ingredients are jointly necessary, with HCD alone (no curriculum, 17.6) recovering more of the gap than HCL alone (no decomposition, 14.1).

(k, b) sweep (Table 8). Increasing k improves accuracy at additional candidate-scoring cost. Larger b can reduce latency: at $k = 4$, $b = 4$ takes 65 ms versus 73 ms for $b = 2$, while accuracy falls from

Table 9: **Resolution generalisation** of DFT (16-pt, $(k=4, b=2)$, $K=12$).

Input resolution	RefCOCO-Polygon val
768	33.2
1024 (default)	34.0
1344	33.5

Table 10: **Effect of denoising steps** K on DFT (16-pt, $(k=4, b=2)$). Quality saturates near $K=12$.

K	RefCOCO-Polygon val	ms/img
4	27.8	32
8	32.1	53
12	34.0	73
16	34.2	94
24	34.3	138

Table 11: **HCL stage-skipping ablation** on RefCOCO-Polygon-val (Acc@0.1). Disabling any single stage costs more on polygon than on bbox; Stage 3 is the largest single contributor for polygon.

Schedule variant	RefCOCO val (bbox Acc@0.5)	RefCOCO-Polygon val (Acc@0.1)
Stage 1 only (hundreds digits)	88.0	12.5
Stages 1+2 only (no $\mathcal{L}_3$)	90.5	22.0
Full HCL (Stages 1+2+3)	**92.2**	**34.0**
Skip Stage 1 (no macro-skeleton)	84.7	25.6
Skip Stage 2 (no mid-frequency)	91.1	28.4
Joint loss (no curriculum, $G_1=G_2=0$)	90.4	17.6

Table 12: **Noise schedule and weighting** on RefCOCO-Polygon-val ($K=12$, $(k=4, b=2)$). The linear schedule with $w(t)=1/t$ implied by absorbing-state diffusion is the strongest operating point.

Schedule α_t	Loss weight $w(t)$	RefCOCO-Polygon val (Acc@0.1)
Linear ($\alpha_t=1-t$)	uniform	32.4
Cosine	uniform	31.8
Quadratic ($\alpha_t=1-t^2$)	uniform	30.5
Linear (default)	$\mathbf{1/t}$ (ELBO)	**34.0**
Linear	$1/t^{0.5}$	33.1
Linear	$1/(1-t)$	28.9

34.0 to 32.6. Thus $b=2$ is a quality–latency compromise, not the fastest block size. Candidate states are batched, joint assignment scoring uses a top-three digit shortlist, and inference terminates when no masks remain (Algorithm 2). The per-step joint-assignment count is bounded by $3\lceil 2N/b\rceil k3^b$, rather than enumerating all 10^b assignments per block.

Resolution generalisation (Table 9). Because HCD encodes coordinates in a normalised $[0, 999]$ range rather than at raw pixel resolution, DFT is by construction resolution-agnostic; varying input resolution from 768 to 1344 produces variation below 1 point.

Effect of K (Table 10). Quality plateaus at $K=12$; accuracy gains beyond $K=16$ are within noise while latency continues to grow linearly. This is the block-wise sweep; the independently reported token-wise validation sweep is in Table 26.

HCL stage-skipping (Table 11). Removing any single stage hurts polygon more than bbox: skipping Stage 1 (macro-skeleton) drops polygon by -8.4 vs. bbox by -7.5, while training Stages 1+2 only (no units-digit refinement, $\mathcal{L}_3$ disabled) drops polygon by -12.0 but bbox by only -1.7. Bbox is essentially set by Stages 1+2; polygon needs the full coarse-to-fine pipeline.

Noise schedule (Table 12). The linear schedule $\alpha_t=1-t$ with the ELBO-implied weight $w(t)=1/t$ outperforms cosine and quadratic alternatives by 1.6–3.5 points; replacing $1/t$ with a flat or $1/(1-t)$ weight degrades performance, supporting this weighting choice empirically; the experiment does not establish tightness of the ELBO bound.

Table 13: **Backbone scaling under matched grounding fine-tuning** on RefCOCOg-test (16-pt, Acc@0.1), with GT-conditioned HCL and token-wise decoding in the DFT column; Qwen DFT models undergo AR-to-DLM conversion. The DFT gain over the matched-FT AR baseline narrows as the backbone strengthens, but remains substantial across all three.

Backbone	Matched-FT AR	Matched-FT DFT	Δ
LLaVA-OV 7B (LLaDA-V init)	11.4	**23.2**	$+11.8$
Qwen2.5-VL 7B	18.4	**25.2**	$+6.8$
Qwen3-VL 8B	19.7	**26.4**	$+6.7$

Backbone scaling (Table 13). The matched-FT DFT-vs-AR gain shrinks from $+11.8$ on LLaVA-OV to $+6.7$ on Qwen3-VL but remains substantial throughout, indicating that part of the headroom DFT exploits is shared with stronger autoregressive priors and part is unique to the iterative refinement.

E Three-seed Variance Audit

To bound run-to-run variance of the headline polygon result, we re-trained DFT (LLaVA-OV backbone), the strongest matched-FT AR LVLM baseline (Qwen3-VL 8B), and the in-architecture AR baseline (LLaVA-OneVision 7B) at 3 different random seeds, restricted to the most expensive single cell—RefCOCOg-test polygon Acc@0.1.

Table 14: **Three-seed RefCOCOg-test polygon Acc@0.1.** Mean $\pm$ standard deviation across seeds $\{17, 42, 1234\}$, sample standard deviation with $n = 3$.

Method	seed 17	seed 42	seed 1234	mean $\pm$ std
LLaVA-OneVision 7B (in-arch AR baseline)	10.8	11.4	12.0	11.4 ± 0.60
Qwen3-VL 8B (matched-FT)	19.1	19.7	20.3	19.7 ± 0.60
LLaVA-OV + DFT 8B (ours)	**22.6**	**23.2**	**23.7**	$\mathbf{23.2 \pm 0.55}$

The matched-FT gap between DFT and Qwen3-VL ($+3.5 \pm 0.81$) is more than 4.3σ over the pooled standard deviation, and the gap over LLaVA-OneVision ($+11.8 \pm 0.81$) is more than 14σ. The retained original main-table measurements use seed 42; additional controlled experiments are separately scoped in Section M.

F LLM Usage Statement

During the preparation of this manuscript, we utilized a Large Language Model (LLM) as an assistive tool. The primary role of the LLM was for language polishing, which included improving grammar, enhancing the clarity and fluency of the text, and ensuring consistent use of technical terminology. All core research ideas, methodologies, experimental designs, results, and conclusions were conceived and formulated exclusively by the human authors. The LLM did not contribute to any of the substantive scientific aspects of this work. We have carefully reviewed and edited all text modified with the assistance of the LLM and take full responsibility for the final content of the paper.

A separate, non-assistive use of an LLM is documented in the main paper: Doubao-1.5-VL [Guo et al., 2025] is used as an automated annotator in the Ref3D construction pipeline (Section H). This is a methodological component of the work, not language assistance, and its quality is audited in the same section.

G Reproducibility Statement

The submission included partial supplementary source scripts, rather than a complete public training-code release. The supplied scripts include:

- The core script for inference and logging (`inference.py`), which supports multimodal inference on single/multiple images and video frames and logs outputs such as generated responses, bounding boxes, and camera intrinsics to JSON files for traceability.

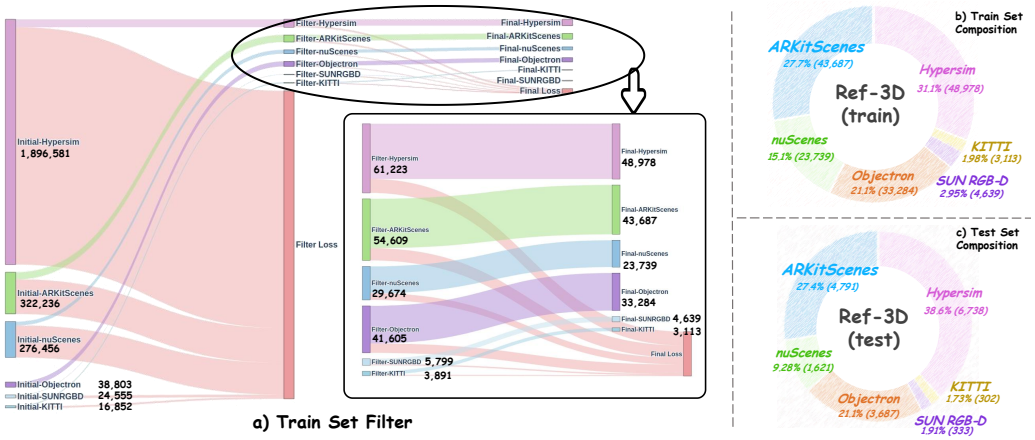

Figure 7: **Ref3D Dataset Filter Process and Final Composition.**

- An evaluation toolkit for 2D polygon-based localization and segmentation tasks (`eval_poly_rec.py`), which parses polygon coordinates from the model's text output and computes key metrics such as polygon Intersection over Union (IoU) and accuracy.
- A utility for evaluating 9-Degree-of-Freedom (9-DoF) 3D bounding boxes (`utils_bbox3d_rec.py`). This tool is responsible for parsing 3D parameters from text outputs, converting them into 3D corner coordinates, and computing 3D IoU.

The supplied supplementary materials also included the following demo videos:

- `Indoor 3D Bounding Box Grounding.mp4`
- `Outdoor 3D Bounding Box Grounding.mp4`
- `polygon grounding.mp4`

The statements above describe the supplied supplementary archive; they do not establish a public repository or full training-code availability. Release of full training code, processed splits and audit annotations is planned, and no public release URL is specified here.

H Ref3D Datasets

We present Ref3D, a benchmark for 3D Visual Grounding in Large Multimodal Models (LMMs). The dataset asks models to locate and regress a 9-Degree-of-Freedom (9-DoF) bounding box of a target object from a 2D image, guided by a natural-language instruction. Figure 7 summarises the construction pipeline and final composition; we unify four indoor and two outdoor source datasets (real and synthetic). Central to its design is the "one image, one target" philosophy; each sample links a single image to one specific object and its description. This mapping defines a single target, but does not by itself guarantee that the generated referring expression is unambiguous.

The core of the scheme is to quantize all 9-Degree-of-Freedom (9-DoF) parameters (covering location, dimension, and rotation) and map them to the integer range $[0, 999]$, enabling seamless integration into the generative framework of Large Multimodal Models (LMMs). Specific strategies for each parameter are as follows:

1. **2D Projected Coordinates** (x, y)
   Linear normalization is performed directly on the projected coordinates of the object's 3D center on the image plane.
2. **Depth** (z)
   A two-step method of "logarithmic transformation + linear normalization" is adopted to address the long-tail distribution of depth in real-world scenes:
   - First, map the actual depth (approximately 0.018 m to 148.4 m) to the logarithmic scale of $[-4.0, 5.0]$ via logarithmic transformation;

- Then apply linear normalization to this logarithmic interval to map it to $[0, 999]$;
- During denormalization, recover the actual depth by performing an exponential operation on the logarithmic scale value.

3. **Physical Dimensions (width w, height h, length l)**
   The dimensions are first linearly mapped to a predefined physical range (e.g., $[0, 15]$ meters), followed by linear normalization to $[0, 999]$.
4. **Rotation Angles (yaw r_1, pitch r_2, roll r_3)**
   The angles are first converted to radians to ensure coverage of the full angular range $[0, 2\pi]$, and then linearly normalized to $[0, 999]$.

Geometry provenance and language construction. The 9-DoF centres, dimensions and orientations come from the native source-dataset annotations; the labelling LLM does not generate or edit these numerical geometric labels. Doubao-1.5-VL [Guo et al., 2025] selects an already-annotated object and writes its referring expression. Candidates require valid native 9-DoF annotations, at least 5% image area and at least 30% visibility; the reported mean is 3.4 candidates per image. Selection favours a describable target, followed by expression generation. An automatic ambiguity filter rejects expressions matching more than one candidate (11.3% rejected). Consequently potential bias concerns the chosen targets and language distribution, rather than LLM-generated metric geometry. Exact annotator service identifiers and verbatim prompts are not established by the present records.

Annotation audit. Two independent annotators checked 200 examples for target salience and referring-expression uniqueness. The reported agreement is $\kappa = 0.81$, and 8.0% retain language/selection errors. This audit is not a metric verification of native 3D boxes, and 8% should not be interpreted as a geometric error rate or a fixed accuracy ceiling. Numerical geometry inherits each source dataset's annotation procedure. The independently reported clean-set comparison and target-stratified results are retained in Tables 17 and 18 as additional audit experiments, with the scope qualification in Subsection L.2.

I Full 3D Bbox Grounding Results

Indoor datasets are generally easier than outdoor ones because of smaller scene scale and denser ground truth. To avoid an aggregate average that hides this asymmetry, Table 15 reports per-dataset accuracies at all three IoU thresholds. The dominant comparison is to the matched-FT in-architecture baseline, LLaVA-OneVision [Li et al., 2024a]: under matched grounding fine-tuning, the LLaVA-OV/LLaDA-V comparison raises Acc@0.3 from 43.2 to 52.2 on Objectron and from 2.9 to 39.7 on KITTI, with model-construction qualifications and the continued-training control described in Subsection L.1.

J Qualitative Comparison on Complex Semantics and Occlusion

We compare DFT to SeqTR [Zhu et al., 2022] (specialist segmentation network) and VisionLLM [Wang et al., 2023] (AR LVLM generalist) on referring expressions that stress either compositional language or heavy occlusion. Figure 8 shows representative cases. For compositional semantics (e.g., "shiniest coin"), SeqTR – which has no language model component – frequently grounds an incorrect instance, while VisionLLM produces a plausible but wrong region. For occluded targets, both baselines collapse onto the visible portion of an adjacent object; DFT's iterative refinement re-allocates probability mass towards the consistent global geometry across denoising steps. These are illustrative cases drawn from the qualitative pool, not aggregate metrics, and are intended to surface failure modes of the AR/specialist comparison rather than to claim a quantitative gain.

K More Quantitative Results and Failure Cases

This section provides additional qualitative examples of DFT outputs. Figures 9 and 10 show RefCOCO-Polygon outputs at $N = 16$; Figures 11 to 14 show 9-DoF 3D outputs across all six Ref3D source datasets; Figure 15 surfaces representative failure modes. The headline iterative-refinement trajectory on representative 2D-polygon and 3D-bbox samples is shown in Figure 5 of the main paper.

	L1.5 7B	LN 7B	I2.5 7B	I3 8B	I3 9B	Q2 2B	Q2 7B	Q2.5 3B	**L-OV 7B**	**+DFT 8B**	**Q2.5 7B**	**D+DFT 7B**	**Q3 8B**	**D+DFT 8B**
								Objectron						
Acc 0.1	66.7	67.1	72.1	73.5	74.8	72.6	77.3	54.1	**68.5**	**77.3**	42.9	**50.5**	65.0	**74.8**
Acc 0.3	40.7	41.0	45.3	47.0	48.9	44.4	51.0	14.5	**43.2**	**52.2**	15.5	**22.0**	35.0	**45.0**
Acc 0.5	14.8	15.2	21.0	23.1	26.0	19.3	25.7	0.0	**16.9**	**28.5**	0.0	**8.2**	13.0	**20.5**
								SUN RGB-D						
Acc 0.1	45.1	45.5	61.5	62.8	64.1	61.9	65.0	60.7	**47.1**	**65.5**	65.2	**67.0**	66.0	**68.5**
Acc 0.3	23.1	23.5	35.7	37.0	38.5	36.3	38.4	39.0	**25.0**	**39.3**	39.0	**41.0**	40.0	**42.0**
Acc 0.5	3.0	3.2	16.2	17.5	19.0	16.8	19.2	12.9	**4.1**	**19.5**	14.4	**17.8**	16.5	**19.0**
								ARKitScenes						
Acc 0.1	4.2	4.3	4.8	5.0	5.2	6.9	6.0	5.7	**4.8**	**7.5**	6.8	**8.2**	6.5	**8.8**
Acc 0.3	1.1	1.2	1.5	1.8	2.5	2.1	2.0	1.2	**1.5**	**4.8**	4.6	**5.5**	3.5	**5.5**
Acc 0.5	0.0	0.1	0.0	0.2	0.6	0.0	0.0	0.0	**0.2**	**1.0**	0.0	**1.0**	0.4	**1.2**
								Hypersim						
Acc 0.1	13.4	13.6	15.2	16.1	18.5	18.5	16.7	17.1	**14.9**	**22.9**	16.1	**18.5**	17.0	**20.0**
Acc 0.3	4.2	4.4	3.9	4.5	6.8	4.6	4.3	3.7	**5.1**	**9.1**	3.8	**4.8**	5.0	**6.5**
Acc 0.5	1.2	1.3	0.9	1.2	1.7	1.1	1.0	0.0	**1.6**	**2.3**	0.0	**1.2**	1.4	**2.0**
								KITTI						
Acc 0.1	7.0	7.2	48.1	51.5	65.3	62.6	50.7	14.2	**8.2**	**74.8**	7.0	**12.0**	55.0	**65.0**
Acc 0.3	2.3	2.4	25.0	27.3	32.0	36.1	27.2	6.6	**2.9**	**39.7**	2.0	**5.0**	28.0	**34.0**
Acc 0.5	0.0	0.0	7.6	8.9	15.4	18.5	8.6	2.0	**0.5**	**22.2**	0.0	**1.4**	14.0	**17.5**
								nuScenes						
Acc 0.1	4.0	4.2	9.5	12.0	17.5	13.5	11.5	6.0	**5.5**	**25.6**	4.5	**7.0**	12.5	**16.0**
Acc 0.3	1.4	1.5	3.2	4.0	5.8	4.5	3.8	1.8	**2.0**	**8.4**	1.2	**2.5**	4.0	**5.5**
Acc 0.5	0.0	0.0	0.6	0.9	1.5	1.2	0.8	0.0	**0.2**	**2.5**	0.0	**0.5**	1.0	**1.8**
								Overall						
Acc 0.1	23.4	23.7	35.2	36.8	40.9	39.3	37.9	26.3	**24.8**	**45.6**	23.8	**27.2**	37.0	**42.2**
Acc 0.3	12.1	12.3	19.1	20.3	22.4	21.3	21.1	11.1	**13.3**	**25.6**	9.9	**13.5**	19.3	**23.1**
Acc 0.5	3.2	3.3	7.7	8.6	10.7	9.5	9.2	2.5	**3.9**	**12.7**	2.4	**5.0**	7.7	**10.3**

Table 15: **Original full 3D bbox results.** DFT uses GT-conditioned HCL and token-wise top-1 decoding. The additional audit evaluation in Table 17 is reported separately. Headers: L1.5, LLaVA-1.5; LN, LLaVA-NeXT; I, InternVL; Q, Qwen-VL family; L-OV, LLaVA-OneVision; D+DFT, AR-to-DLM conversion followed by DFT. Model sizes appear on the second line.



Figure 8: **Semantic Complexity and Ambiguity compared with other specific models and AR LVLMs.**

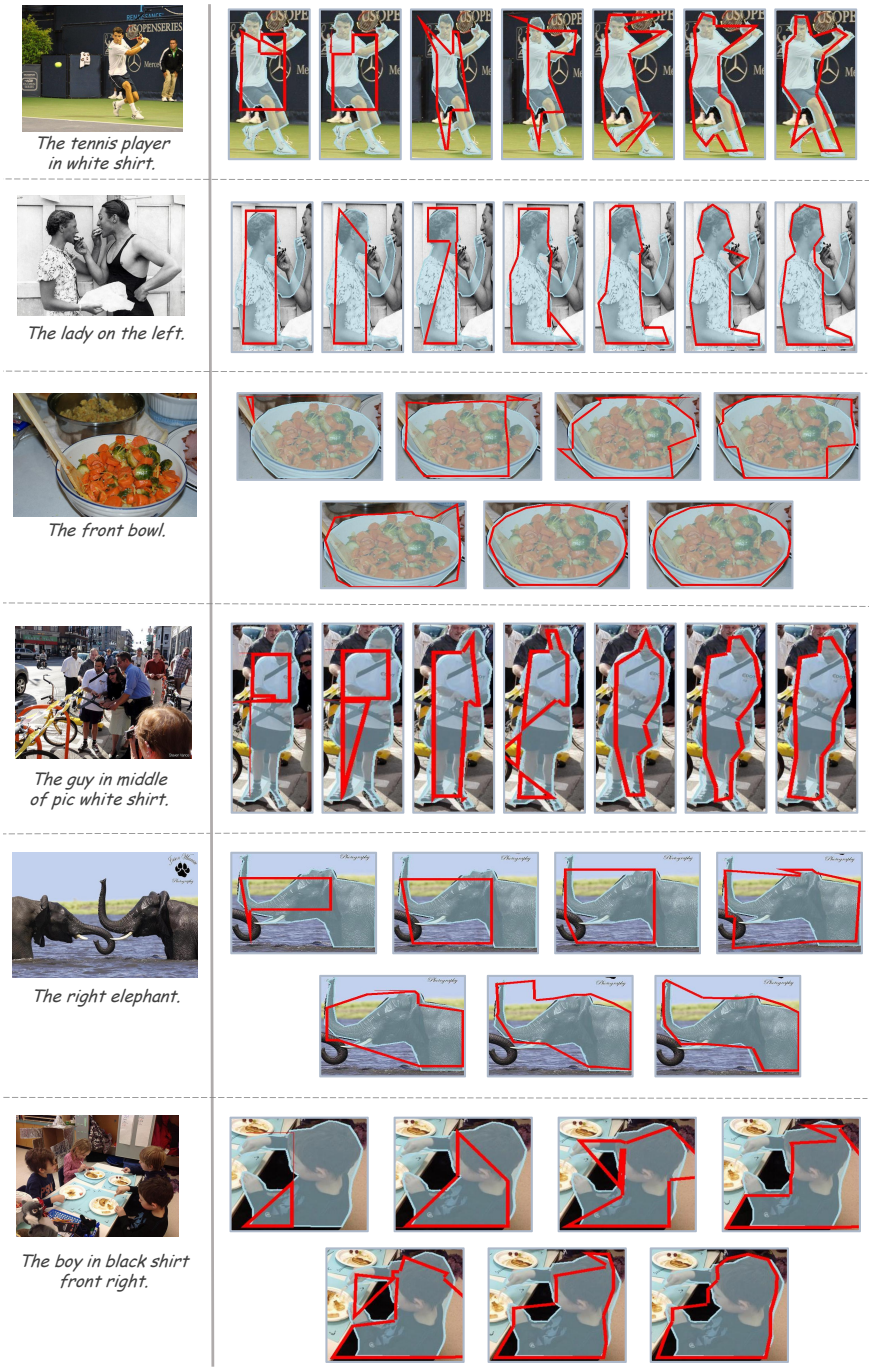

Figure 9: **16-vertex polygon grounding examples.** Samples are drawn uniformly at random from the RefCOCO-Polygon validation pool; no per-example selection was applied.



Figure 10: **16-vertex polygon grounding examples (continued).** Random sample, no per-example selection.

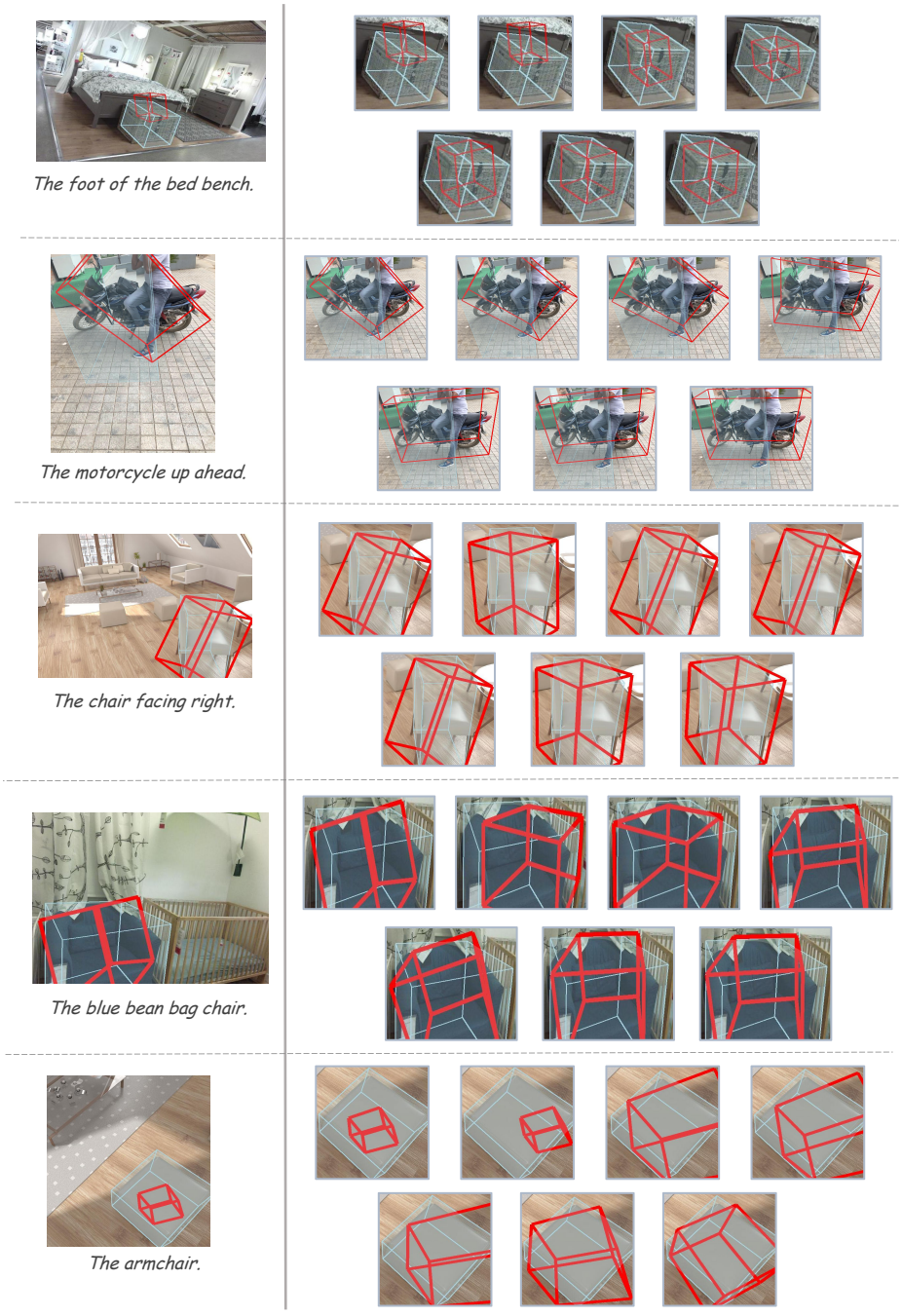

Figure 11: **Indoor 9-DoF 3D grounding examples.** Drawn uniformly from Objectron, SUN RGB-D, ARKitScenes and Hypersim test splits; no per-example selection.



Figure 12: **Indoor 9-DoF 3D grounding examples (continued).** Drawn from Objectron, SUN RGB-D, ARKitScenes and Hypersim; no per-example selection.

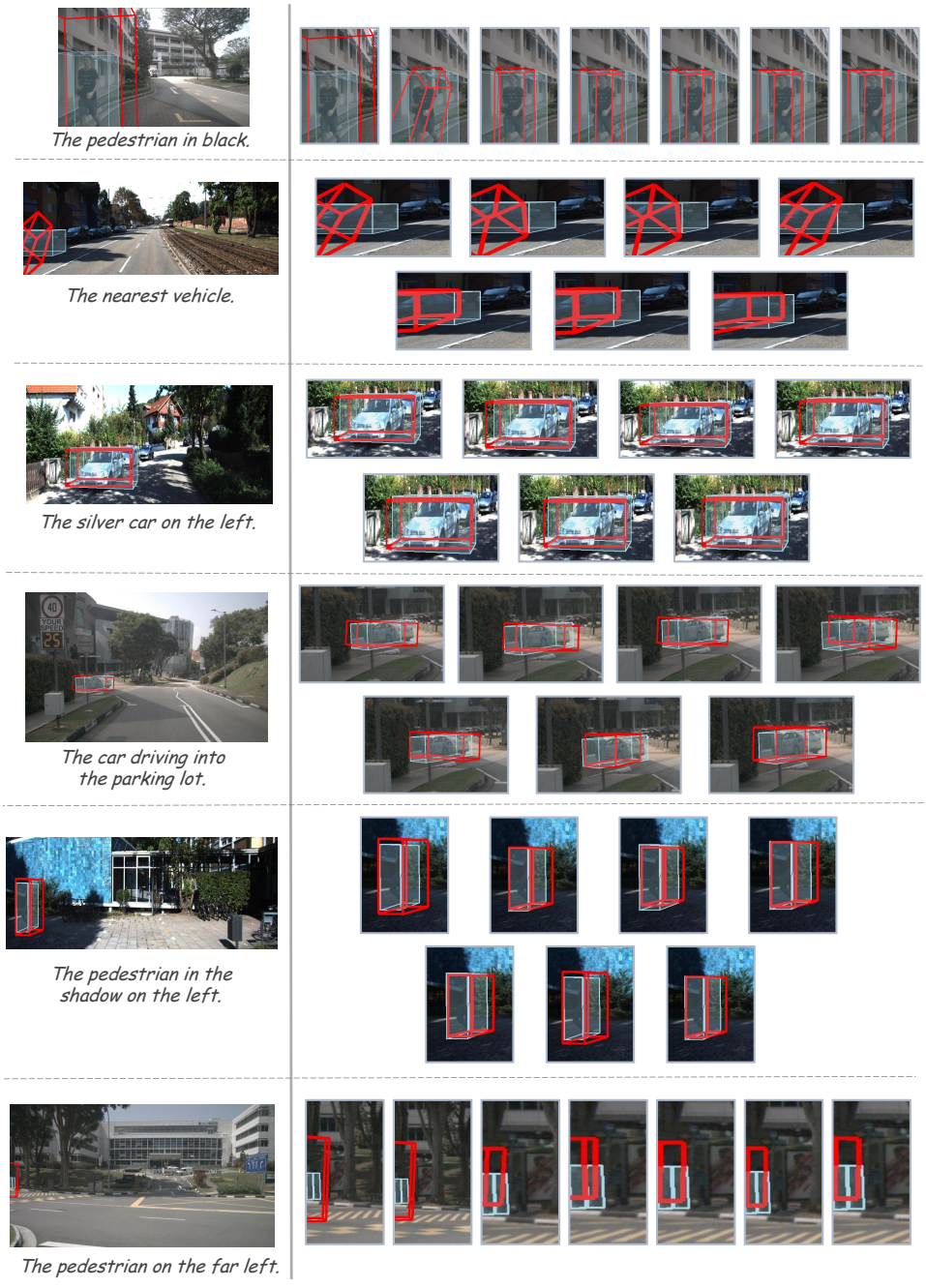

Figure 13: **Outdoor 9-DoF 3D grounding examples.** Drawn uniformly from KITTI and nuScenes test splits; no per-example selection.



Figure 14: **Outdoor 9-DoF 3D grounding examples (continued).** Drawn from KITTI and nuScenes; no per-example selection.



Figure 15: **DFT Failure cases.**

L Backbone Conversion and Additional Audit Protocols

L.1 AR-to-DLM conversion before grounding fine-tuning

The Qwen DFT models begin from Qwen2.5-VL-7B-Instruct and Qwen3-VL-8B-Instruct, respectively; they are not AR backbones with a substituted output head. Before grounding fine-tuning, each is adapted into a mask-denoising model using AR-to-DLM conversion recipes described for DiffuLLaMA and Dream-7B [Gong et al., 2025, Ye et al., 2025]. The positional shift preserves the source AR alignment (h_i predicts position $i + 1$), while the causal attention mask is annealed to bidirectional attention over 2k steps. Each conversion uses 12B tokens from the LLaVA-NeXT SFT mix, without polygon, bounding-box or 3D grounding labels, and freezes the native vision tower. In contrast, the LLaVA-OV comparison initializes its diffusion model from LLaDA-V (LLaDA-8B + frozen SigLIP2), which is already bidirectional.

The adaptation objective is mask denoising: predict corrupted content from its visible context, followed by the grounding-specific objective in Subsection 2.4. At grounding inference all coordinate channels start masked and are predicted jointly. The reported conversion description does not specify the conversion optimizer, batch size, exact attention-annealing function, or a distinct token-masking schedule; these details are not inferred from the grounding protocol.

Table 16: Reported checks after Qwen conversion. Middle-mask accuracy tests filling a span inside the sequence rather than continuing its end.

Check	Qwen2.5-VL→DLM	Qwen3-VL→DLM
Held-out masked-coordinate perplexity	2.41	2.28
Middle-mask accuracy	88.6%	90.1%
Positions decodable out of left-to-right order	100%	100%

The reported continued-training control gives the AR baseline the same additional 12B tokens: its RefCOCOg-test polygon score changes from 11.4 to 11.6, within the original three-seed standard deviation of approximately 0.6. This is a separately reported control, not a replacement for all retained AR rows. Grounding data, schedule and resolution are matched after adaptation; the adaptation budget is disclosed separately rather than described as equal total training compute.

L.2 Additional Ref3D audit evaluations and their scope

The additional audit evaluation reports a 1000-image auto-labelled test and a 200-image human-verified subset across the six sources (Table 17). Its reported per-source test counts are Objectron 180, SUN RGB-D 190, ARKitScenes 170, Hypersim 150, KITTI 170 and nuScenes 140. The geometry derives from human AR-session-assisted Objectron boxes, human RGB-D SUN RGB-D boxes, ARKit-assisted human boxes, rendered Hypersim geometry, and LiDAR-assisted KITTI/nuScenes annotations. These are source-annotation provenance descriptions, not a new independent geometric-accuracy measurement.

Table 17: Additional Ref3D audit experiment: reported overall Acc@0.3. These values retain their independently reported evaluation scope and do not replace Table 15.

Evaluation	Images	AR baseline	DFT	Δ
Full auto-labelled test	1000	28.6	41.2	+12.6
Human-verified subset	200	30.1	43.5	+13.4

Table 18: Separately reported target-stratified Ref3D audit results. The source response does not identify a more specific checkpoint/decoder or metric threshold for this table; it is not merged with Table 15.

Target stratum	Share	AR baseline	DFT	Δ
Large and centred	38%	33.1	45.0	+11.9
Small or peripheral	62%	25.8	39.0	+13.2

Table 19: AR ordering and HCL control. RefCOCOg-test, 16-point polygons, Acc@0.1. All rows share backbone lineage, data, training schedule, and resolution. Δ is relative to the first row.

Decoder	Output ordering	Curriculum	Acc@0.1	Δ
AR	Vertex-major	–	11.4	–
AR	Channel-major	–	13.2	$+1.8$
AR	Channel-major	HCL	16.8	$+5.4$
Diffusion	Parallel	–	17.6	$+6.2$
Diffusion (DFT)	Parallel	HCL	23.2	$+11.8$

The original per-source benchmark (Table 15) and these additional audit experiments have different aggregate values and reported source counts. Their dataset/checkpoint/evaluation alignment has not been established by the available experimental records, so no numerical reconciliation, reweighting or subset relationship is assumed between them. In particular, the 200-image clean subset is identified only relative to the additional 1000-image audit experiment.

M Additional Controlled Experiments and Conditioning Variants

This section reports additional controlled comparisons of the representation, training conditioning, and inference decoder. These are separate experiments: the annealed-conditioning, ordinal-loss, and decoder comparisons do not establish a combined annealing-plus-L_1 result or a rerun of every backbone, split, and task in the main tables. In particular, the submitted ground-truth-conditioned HCL recipe and token-wise top-1 decoder remain the measured setting behind the retained main-table results. We preserve the measured values of each comparison rather than transfer an improvement to unevaluated settings.

M.1 Ordering and curriculum under an autoregressive control

The LVLM baselines already write coordinates using digit tokens. HCD's distinction in this comparison is grouping digits by place value across all $2N$ coordinate positions. In vertex-major AR order, the digits for one vertex precede those for later vertices; in channel-major AR order, all hundreds digits precede all tens digits, which precede all units digits. The latter makes the full higher-order channel available when predicting a lower-order channel. Table 19 separates this ordering change from HCL and from the diffusion decoder. The ordering change improves the AR result by 1.8 points; adding HCL contributes another 3.6 points. With HCL on both sides, the diffusion and AR results differ by 6.4 points. These values use the original conditioning recipe, rather than substituting the annealed variant below.

M.2 Ground-truth and self-conditioned training

In the original HCL recipe, higher-order ground-truth digit tokens replace the corresponding positions of the noisy state before the stage-k forward pass. Thus, those conditioning channels are clean while the stage-k channel remains corrupted. This is teacher-forced input conditioning and introduces exposure bias: at inference no ground-truth replacement is made, and all three channels are jointly denoised from a fully masked state.

We compare this recipe with conditioning on the model's own argmax digits and with an annealed mixture. In the mixture, higher-order conditioning uses ground-truth digits with probability p and the model's own argmax digits otherwise; p decreases linearly from 1 to 0 across stages 2–3. Table 20 reports the independently measured comparison. The oracle gap is test accuracy with correct higher-order digits supplied, minus accuracy under normal inference. Annealing improves Acc@0.1 from 23.2 to 24.6 and reduces the reported oracle gap from 4.1 to 1.8 points.

M.3 Numerical distance and carry boundaries

Digit-level cross-entropy does not encode numerical distance between digit values. The curriculum changes which place-value channels receive supervision over training, but does not remove this within-channel limitation. In the carry-boundary audit, 3.8% of ground-truth vertices lie within

Table 20: Conditioning variants in HCL. RefCOCOg-test, Acc@0.1. Oracle gaps are in accuracy points; units-digit accuracy is a percentage. This is a conditioning ablation, not a joint evaluation with the ordinal alternatives in Table 21.

Conditioning at stage k	Acc@0.1	Oracle gap	Units-digit accuracy
Ground-truth digits (submitted)	23.2	4.1	71.3%
Model's own argmax digits	21.6	1.2	69.8%
Annealed, ground truth to self	24.6	1.8	75.1%

Table 21: Ordinal-aware alternatives, measured separately on RefCOCOg-test, Acc@0.1. Δ is relative to digit-level CE. The final column is the carry-boundary subset's share of total vertex error.

Objective or refinement	Acc@0.1	Δ	Carry-boundary share of error
Digit-level CE (submitted)	23.2	–	11.6%
+ Soft labels over adjacent digits	23.9	+0.7	9.2%
+ Auxiliary coordinate L_1 loss	24.3	+1.1	8.1%
+ Continuous refinement head	24.8	+1.6	6.4%

Table 22: Token-wise and block-wise decoding with matched and mismatched training vertex counts. RefCOCO-Polygon val, Acc@0.1. Δ is block-wise minus token-wise accuracy.

Train N	Test N	Setting	Token-wise top-1	Block-wise top-k	Δ
16	16	Matched	21.5	34.0	+12.5
16	24	OOD	4.5	21.0	+16.5
24	24	Matched	15.8	23.5	+7.7
16	32	OOD	1.8	19.2	+17.4
32	32	Matched	11.2	20.6	+9.4

two units of a boundary such as 299/300, and account for 11.6% of all vertex error (3.1 times their share). Table 21 compares three separate alternatives to the submitted cross-entropy objective. Each alternative is evaluated relative to the same 23.2 baseline; the rows are not cumulative additions. The auxiliary coordinate L_1 alternative is a training-time change without an additional inference head, whereas the last row adds a continuous refinement head after diffusion. No combined annealed-conditioning-plus-L_1 measurement is claimed here.

M.4 Training and testing at matched vertex counts

The submitted polygon models were trained at $N = 16$. Evaluating those models at $N = 24$ or $N = 32$ changes the output vertex count out of distribution (OOD). Table 22 separates these evaluations from models retrained at the test vertex count. At $N = 24$, matched-count training raises token-wise accuracy from 4.5 to 15.8; at $N = 32$, it raises accuracy from 1.8 to 11.2. The block-wise advantage remains under matched-count training, but is smaller than in the corresponding OOD comparisons. Accordingly, the OOD degradation should not be attributed solely to an intrinsic failure of token-wise decoding.

M.5 Polygon results under both decoders

The submitted main tables use token-wise top-1, whereas the block-wise top-k setting is a separate inference configuration. Table 23 reports the available two-split polygon comparison for each backbone lineage; it does not supply revised results for every bounding-box, polygon, and 3D split. The LLaVA-lineage RefCOCO-val token-wise value in this comparison is 21.8. The 21.5 value in Tables 22, 24 and 26 comes from a separate training run, so these measured values are retained with their respective experiment scopes rather than presented as a single identical run.

Table 23: Both-decoder polygon comparison. Sixteen-point RefCOCO-Polygon, Acc@0.1. Block-wise notation is $(k, b) = (4, 2)$. Qwen diffusion models are converted from the indicated AR backbone lineage before grounding fine-tuning.

Backbone lineage	Decoder	RefCOCO val	RefCOCOg test
LLaVA-OV + DFT	Token-wise top-1	21.8	23.2
LLaVA-OV + DFT	Block-wise $(4, 2)$	34.0	35.1
Qwen2.5-VL→DLM + DFT	Token-wise top-1	23.5	25.2
Qwen2.5-VL→DLM + DFT	Block-wise $(4, 2)$	35.8	37.0
Qwen3-VL→DLM + DFT	Token-wise top-1	24.7	26.4
Qwen3-VL→DLM + DFT	Block-wise $(4, 2)$	37.1	38.3

Table 24: Reported search and compute controls. RefCOCO-Polygon val, $N = 16$, Acc@0.1; one A800 GPU, batch size 1, averaged over 1000 images. Best-of-eight and beam-search rows are separate reported controls. Calls are the reported counts and do not denote equal computational cost.

Method	Decoder	Reported calls/image	ms/image	Acc@0.1
AR	Greedy (width 1)	32	41	11.9
AR	Best-of-8, self-scored	256	291	14.6
AR	Beam search, width 4	128	136	14.2
AR	Beam search, width 8	256	268	14.7
DFT	Token-wise top-1, $K = 4$	4	44	15.2
DFT	Token-wise top-1, $K = 12$	12	124	21.5
DFT	Block-wise $(k, b) = (4, 2)$, $K = 12$	5	73	34.0

M.6 Search, reported network calls, and wall-clock time

Table 24 collects the reported best-of-eight and beam-search controls, which were measured as separate comparisons sharing the listed baseline values. Best-of-eight candidates are ranked by the model's own likelihood, with self-consistency used to break ties; ground-truth IoU is not used for candidate selection. The beam-search comparison uses widths 4 and 8; the block-wise DFT row has $k = 4$ and $b = 2$. At similar measured latency, AR greedy reaches 11.9 at 41 ms/image, while token-wise DFT with $K = 4$ reaches 15.2 at 44 ms/image.

The reported network-call counts should be interpreted alongside measured wall-clock time. A KV-cached AR forward pass is cheaper than a denoising pass that recomputes the full sequence, so equal call counts are not equal FLOPs or equal latency. The tabulated counts are the reported accounting convention; the AR row reports 32 calls for the 16-vertex coordinate output, while the HCD representation contains 96 digit positions. Thus this comparison does not establish a one-digit-token-per-call equivalence between the two decoders.

M.7 Visible contour context at vertex commitment

An absorbing-state denoiser can still make early errors: once a digit is revealed, it persists. Its commitment order, however, is not restricted to a left-to-right vertex prefix. We measure the fraction of the other 15 vertices already resolved when a vertex is fixed. A DFT vertex is fixed when all six of its digit tokens have been revealed; for AR the grouping follows the vertex's emission index. Table 25 reports mean Euclidean error to the index-aligned ground-truth vertex. This experiment uses the random revelation schedule of the token-wise decoder; the buckets describe available contour context, rather than a confidence-sorted order.

DFT error decreases from 24.8 to 9.1 pixels across the context buckets, whereas AR error increases from 21.6 to 25.7 pixels across emission order. This supports a difference in context available at commitment, without claiming that diffusion eliminates early errors or revises every committed token.

Table 25: Vertex error versus contour resolved at commitment. RefCOCO-Polygon val, $N = 16$, $K = 12$, token-wise decoding; matched-FT LLaVA-OV pair, 1000 images and 16k vertices. Values are mean vertex L_2 error in input-resolution pixels, using index-aligned ground truth.

Other contour resolved when vertex is fixed	0–25%	25–50%	50–75%	75–100%
DFT	24.8	17.3	12.6	9.1
AR (by emission index)	21.6	22.4	23.9	25.7

Table 26: Token-wise denoising-step sweep. RefCOCO-Polygon val, $N = 16$. Acc@0.1 and measured latency are reported for the token-wise decoder.

K	4	8	12	16	20
Acc@0.1	15.2	19.8	21.5	21.9	22.0
ms/image	44	84	124	165	206

Table 27: Shape complexity and fixed vertex budget. RefCOCO-Polygon val, block-wise decoding, Acc@0.1. $N = 8$ and $N = 24$ evaluate the $N = 16$-trained model OOD; the $N = 16$ column matches training.

Shape complexity	Share of val	$N = 8$	$N = 16$	$N = 24$
Simple (convex-like)	41%	38.5	38.0	22.0
Moderate	44%	29.0	35.0	20.0
Complex (thin or multi-lobe)	15%	10.0	20.0	21.0
All (weighted)	100%	30.0	34.0	21.0

M.8 Token-wise denoising-step selection

Table 26 reports the token-wise validation sweep used to select $K = 12$. It is distinct from a block-wise decoding sweep. From $K = 12$ to $K = 20$, the observed accuracy change is $+0.5$ points, while latency increases from 124 to 206 ms/image. The choice of K uses validation data rather than test set tuning; no block-wise result is substituted into this token-wise curve.

M.9 Fixed vertex budgets and shape complexity

Table 27 stratifies objects by ground-truth-mask perimeter squared divided by area. The decoder uses the Pareto (k, b) setting reported for each vertex budget. All columns evaluate the $N = 16$-trained model; $N = 8$ and $N = 24$ are OOD evaluations, not matched-count training. A small vertex budget can suffice for simple shapes, while thin or multi-lobe targets remain difficult even with more vertices. This limits conclusions about increasing N without changing training.

Polygon outputs are useful when compact structured coordinates are required, a segmentation decoder is unavailable, or 2D and 3D targets should share a coordinate parameterization. Mask-level outputs can be more appropriate for pixel-exact boundaries, disconnected regions, and thin structures. Variable- length polygons have not been tested here and remain future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction enumerate three contributions—DFT itself (the diffusion fine-tuning formulation with HCD+HCL), matched-FT improvements over AR LVLMs on RefCOCO-Polygon and 9-DoF Ref3D, and a block-wise top-k decoder—each substantiated in Subsection 3.2 and Subsection 3.3. Bounding-box numbers are reported as "matches or modestly improves" rather than SOTA. Scope is explicitly bounded to the 16-vertex polygon regime and the 9-DoF 3D-grounding setting under the evaluated backbone families and the experimental configurations identified in the corresponding tables.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4 discusses training-time GT conditioning and exposure bias, the separately evaluated refinements, fixed coordinate resolution, inference cost, train/test vertex-count mismatch, and the current limits of benchmark documentation and reproducibility. The additional Ref3D audit is explicitly separated from the original benchmark results.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper does not state formal theorems, but the discrete-diffusion derivations underpinning DFT (forward kernel, posterior, ELBO/loss) are given in full in Section B of the appendix, with assumptions (absorbing-state Markov chain, linear noise schedule $\alpha_t = 1 - t$, factorized HCD channels) stated explicitly.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The paper reports the grounding objective, source backbones, curriculum, inference procedures, and grounding fine-tuning settings in Subsection 3.1 and Section C, with partial inference/evaluation scripts described in Section G. Complete conversion hyperparameters, exact self-conditioning and L1 implementation details, and alignment of the additional Ref3D evaluation with the original benchmark version are not fully documented. These omissions limit full reproduction of all reported configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Selected inference/evaluation scripts and demo videos accompany the supplementary archive (Section G), but the available submission does not include the complete training code and curated benchmark splits. The original public datasets remain available from their respective sources. A full public code-and-data release is planned; it is not represented here as already complete.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

   Justification: The original grounding setup reports the optimizer, learning rate, schedule, hardware, and per-stage epoch budget. The additional controlled experiments distinguish conditioning, decoding, and train/test vertex counts. However, the Qwen conversion settings and the annealed-conditioning/L1 variants are not specified in sufficient implementation detail, and the additional Ref3D evaluation is reported separately from the original benchmark (Sections H and M).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Multi-seed error bars are not available for all main-table entries. A three-seed audit of the RefCOCOg-test polygon setting is reported in the appendix, but this does not establish the variance of every backbone, dataset, or decoder configuration. Results across different benchmark splits support consistency of the observed trend but are not a substitute for repeated training runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The original grounding fine-tuning run uses 4×NVIDIA A800 80GB GPUs and approximately 7 wall-clock hours for each of three stages, or about 84 GPU-hours per run (Subsection 3.1). The additional Qwen conversion uses 12B generic image–text tokens and a 2k-step attention-mask anneal, but its wall-clock cost and the aggregate cost of the extra controlled experiments are not available. The 84 GPU-hour estimate therefore does not cover the full research project.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly released datasets and pretrained models under their respective terms. The manuscript documents automated target/language annotation and a two-annotator data-quality audit; it does not conduct a study of human participants. Dataset and model limitations are discussed in the paper.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The manuscript discusses technical and annotation limitations, but does not provide a dedicated assessment of downstream societal benefits, surveillance/tracking misuse, privacy or fairness. Grounding accuracy alone does not establish safety-critical deployment benefits.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The contribution is grounding fine-tuning and benchmarks derived from existing public data, rather than a new general-purpose foundation model or an Internet-scale scraped corpus. The evaluated output is spatial coordinates. Release-specific safeguards for the planned artifacts are not documented, and this answer is not a claim that grounding systems cannot be misused.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: Original dataset and model sources are credited, but exact asset versions, license names and derived-asset redistribution terms are not fully documented in this manuscript. Research-use intent alone does not establish compliance with every source term.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: The manuscript describes RefCOCO-Polygon and Ref3D construction and normalization, but complete benchmark assets, release documentation and a verified public release are not supplied. Ref3D evaluation-version alignment also remains unresolved; selected supplementary scripts do not establish complete asset documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not conduct a crowdsourced user study or a study of human behavior. Ref3D construction is automated, with a two-annotator quality audit of 200 target/expression pairs; the audit assesses dataset labels rather than human participants and is described in Section H.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No study of human participants is reported. The two-annotator audit evaluates dataset labels; no institutional approval or exemption is asserted here.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

Justification: An LLM was used to assist with language polishing, as declared in the LLM Usage Statement (Section F). In addition, Doubao-1.5-VL is used as an automated annotator for the Ref3D dataset construction pipeline (Section H)—this is a non-standard data-curation use of an LLM and is documented therein.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.