Imaginative Perception Tokens Enhance Spatial Reasoning in Multimodal Language Models

Supplementary Material

7. Training Details and Hyperparameters

7.1. Training Setup

We fine-tune BAGEL-7B-MoT [12] using PyTorch FSDP (Fully Sharded Data Parallel) with `bf16` mixed precision on 8 NVIDIA A100 80 GB GPUs. Table 7 summarizes the key hyperparameters.

System prompts. All training modes share one of two system prompts prepended to every input:

- **Thinking prompt** (used for IPT, Text CoT, and label-only):

  > *Let's think step by step to answer the question. For text-based thinking, enclose the process within <think> </think>. For visual thinking, enclose the content within <image_start> </image_end>. Finally conclude with the final answer wrapped in <answer></answer> tags.*

- **Answer-only prompt** (used for the answer-only portion of mixed training):

  > *Answer the question directly. Wrap your answer in <answer></answer> tags. Do not think or generate any images.*

Training modes. Table 8 summarizes the five training configurations evaluated in this work. Each mode differs in the output format the model is trained to produce, and correspondingly in which loss terms are active. In IPT mode, both $\mathcal{L}_{fm}$ and $\mathcal{L}_{lm}$ are active; in Text CoT and label-only, only $\mathcal{L}_{lm}$ is used.

Mixed training. Mixed training combines 50% IPT examples (with the thinking prompt and visual generation targets) and 50% answer-only examples (with the answer-only prompt and direct answers). The two data subsets are mixed at the dataloader level: both dataset names and sample counts are specified in the training configuration, and the dataloader interleaves batches from both sources. The model learns to switch between generating imaginative perception tokens and producing direct answers based on which system prompt is provided, enabling a single checkpoint to operate in either mode at inference time.

Text CoT generation. Text chain-of-thought training targets are generated by GPT-5.1 using ground-truth scene metadata from the simulator. For each training example, GPT-5.1 receives the input image, the ground-truth answer, and a task-specific instruction, and produces a step-by-step textual reasoning trace (100–300 words). Below we describe the task-specific prompts.

Path tracing CoT prompt. The system prompt instructs the model to act as a spatial reasoning AI that solves indoor navigation questions by analyzing top-down views step by step. The task instruction is:

> *You are navigating along a numbered path through an indoor scene. The top-down view shows the path with waypoints and midpoints. You need to determine what object is visible from a specific side (left or right) at a midpoint.*
>
> *Reason step by step: (1) Identify the path direction (which waypoint to which). (2) Determine your orientation at the midpoint. (3) Figure out what "left" or "right" means given that orientation. (4) Analyze the top-down layout to identify objects on that side. (5) Compare against the answer choices and eliminate wrong ones.*

Figure 3 shows three Text CoT training examples for path tracing in the EgoDir setting, illustrating how the generated reasoning traces analyze the top-down layout, determine the agent's orientation at the midpoint, and systematically eliminate incorrect answer choices.

Perspective Taking CoT prompt. For Perspective Taking, the model receives the input image together with privileged hidden information, the ground-truth answer and a second image showing the target view after the camera motion, to ensure correctness. The generated chain-of-thought must be written as if only the original image and question were available, without referencing any hidden information. The prompt is:

> *You generate student-facing chain-of-thought explanations for visual navigation/spatial reasoning from a single image. You may receive hidden privileged information (metadata, ground-truth answer, and a second image) to ensure correctness, but your explanation must be written as if you only saw the original image and question. Do not mention, quote, paraphrase, or allude to any hidden information.*
>
> *Student-visible input (ONLY what the explanation may reference): (1) Image A (original scene with a red "X" on the floor). (2) Question:* $\{Q\}$
>
> *Hidden privileged information (FOR CORRECTNESS ONLY — MUST NOT APPEAR IN THE EXPLANATION): (1) Correct answer:* $\{A\}$ *(2) Image B (ground-truth final view after moving and turning)*

Table 7. Training hyperparameters.

Parameter	**Value**
Optimizer	AdamW (β_1=0.9, β_2=0.95, ϵ=10^{-15})
Learning rate	1×10^{-5} (constant after warmup)
Warmup steps	2,000
Gradient clipping	max norm = 1.0
EMA decay	0.9999
Max tokens per batch	32,768
Max tokens per sample	24,576
Input image resolution	1024×1024 (PET) / 512×512 (PT, MVC)
IPT latent resolution	64×64 (Latent-64)
Flow-matching loss weight (λ_{fm})	1.0
Language modeling loss weight (λ_{lm})	1.0
Frozen modules	VAE encoder & decoder
Fine-tuned modules	LLM (all layers), ViT encoder, connector

Table 8. Training modes and their output formats. `[IMG]` denotes the generated intermediate image tokens.

Mode	**Prompt**	**Training target**
Label-only	Think	`<answer>A</answer>`
Text CoT	Think	`<think>`*reasoning*`</think>` `<answer>A</answer>`
IPT	Think	`<think>`*task prompt*`</think>` `<image_start>[IMG]<image_end>` `<answer>A</answer>`
Mixed	50% Think / 50% Ans-only	50% IPT + 50% answer-only

Task: Write a chain-of-thought that a student could produce from Image A and the question alone (not more than 10 sentences).

Hard constraints: Do NOT mention Image B, metadata, coordinates, angles, yaw/pitch/roll, agent states, movement info, IDs, or any numbers from the hidden block. The explanation must be plausible using only Image A: talk only about visible objects, their relative positions, and the effect of turning left/right. Keep it short (3–6 steps). No extra commentary. End with the correct final answer: My answer is [A]/[B]. *Do not answer based on your own perspective; stick to the ground-truth answer provided.*

Output format:
Step 1: Identify the object in the question and describe its relative location in the original view.
Step 2: Describe the walk-to-X action (translation without changing facing direction).
Step 3: Describe the turn action (update egocentric left/right).
Step 4: Conclude whether the object is on the left or right relative to the agent.

An example of a generated thought:

Step 1: The oven is on the right wall of the kitchen, a bit ahead of the current viewpoint, and the red X is on the floor between the viewer and that oven wall. Step 2: Walking straight to the X without turning keeps me facing the same direction, so I move forward toward the area in front of the oven and end up nearer that wall. Step 3: From the X, turning right 90 degrees means I would now face the wall with the oven and cabinets. Step 4: Since I walked closer to that wall before turning, I end up standing nearer to the oven than I was at the start. My answer is [A].

Multiview Counting CoT prompt. For Multiview Counting, the model receives per-frame metadata listing which object instances (by ID) are visible in each frame, and is asked to produce a frame-by-frame reasoning trace without directly referencing the hidden instance IDs. The prompt is:

You are given an object-counting question based on multiple images from the same scene. You will be provided with: the target object, the correct total count, and frame-level information listing which object instances are visible in each frame (this information is hidden and should not be referenced directly).

Text CoT Training Examples (Path Tracing, EgoDir)

Question: You are navigating from waypoint 1 to 2, passing through M1, and viewing the space from a top-down perspective as shown in <image_1>. At point 1, I'm facing forward and see: <image_2>. At point 2, facing forward: <image_3>. Which object can you see on your **left** side at waypoint M1?

A: the countertop ✓ | B: the house plant | C: the dining table | D: the dresser

<image_1> <image_2> <image_3>

Generated Text CoT:

From the top-down map, the path goes from waypoint 1 (to the right) toward waypoint 2 (down-left), with M1 located slightly left of 1 along that diagonal segment. So at M1, "forward" is oriented down-left (southwest on the map). Facing southwest, the "left side" corresponds to the southeast side of the path (down-right relative to the direction of travel).

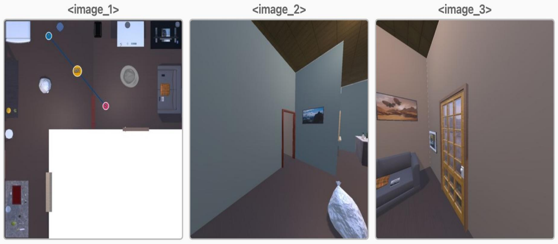
Looking at the top-down layout around M1: a long white counter run is positioned just to the right (east/southeast) of M1, aligned roughly horizontally near waypoint 1. The house plant is up-left (northwest) of M1, which would be on the right side when facing southwest. The dining table is far up-right, not adjacent to M1's left side. The dresser/cabinet block is also up-left of M1, again on the right side for this heading.

Therefore, the object visible on the left side at waypoint M1 is **the countertop (A)**.

Question: You are navigating from waypoint 1 to 2, passing through M1, and viewing the path from a top-down perspective as shown in <image_1>. At point 1, I'm facing forward and see: <image_2>. At point 2, facing forward: <image_3>. Which object can you see on your **left** side at waypoint M1?

A: a book ✓ | B: a shelving unit | C: a garbage bag | D: a stool

<image_1> <image_2> <image_3>



Generated Text CoT:

From the top-down map, the path goes diagonally from waypoint 1 (upper-left) to waypoint 2 (lower-right), with M1 in between. So at M1, "forward" means continuing along that same diagonal toward waypoint 2 (southeast). Facing southeast, your left side points toward the northeast (up-right on the map).

Using the egocentric views to anchor objects: at waypoint 2 we see a couch area with a small rectangular object on the couch that looks like a book. On the top-down view, that couch is on the right side of the room, above and to the right of waypoint 2, i.e., on the northeast side relative to M1.

Now check the choices against the left (northeast) side at M1: the garbage bag is positioned left-of-center (more west), not on the northeast side; the shelving unit appears in the far top-right but is farther and not the prominent item; the stool is near the center-right but not as clearly on the left-of-forward line at M1. The book on/near the couch lies on M1's left side when moving toward 2.

Therefore, the object visible on the left at M1 is **the book (A)**.

Question: You are following the path 1→2, looking at a top-down view at waypoint M1 as shown in <image_1>. This is what you see looking forward from point 1: <image_2>. This is what you see looking forward from point 2: <image_3>. Which object can you see on your **left** side at waypoint M1?

A: the floor lamp | **B: the armchair ✓** | C: the sink | D: the bed

<image_1> <image_2> <image_3>

Generated Text CoT:

From the top-down map, the path goes from waypoint 1 on the right side of the blue-floored bedroom area toward waypoint 2 in the brown-floored bathroom area on the left. So at the midpoint M1, our forward travel direction is generally leftward (west) toward the bathroom doorway.

To interpret "left side" at M1, take that forward direction (toward 2); left of a westward heading is south (down on the map). Looking at the layout around M1, the large floral armchair sits in the lower (southern) part of the blue bedroom area, just below the path near waypoint 1/M1. The bed is far up at the top of the bedroom, so it would be on the right side (north) relative to the westward heading, not the left. The sink is inside the bathroom area to the left of the doorway, ahead/left but separated by the doorway and not on the immediate left side at M1. A floor lamp is not positioned on the south side near M1.

Therefore, the object visible on the left at M1 is **the armchair (B)**.

Figure 3. **Text CoT training examples for path tracing (EgoDir setting).** Each example shows the input images (top-down map and egocentric views at endpoints), the question with answer choices, and the GPT-5.1-generated reasoning trace. The reasoning follows the structured prompt: identifying path direction, determining orientation at M_1, interpreting left/right relative to that orientation, and eliminating distractors.

IMPORTANT: An empty list for a frame means that no objects of the target type are visible in that frame.

Write a brief, frame-by-frame explanation describing what is visible in each frame. Do not mention object IDs or refer to them explicitly. When explaining each frame, do not count objects that were already visible in previous frames. After the frame-by-frame explanation, conclude with: "The total number is X."

> *Object:* {O}
> *Correct total count:* {*answer*}
> *Frames:* {*frames_text*}

An example of a generated thought:

> *Frame 1: No bowls are visible. Frame 2: A bowl appears and is visible for the first time. Frame 3: No new bowls appear; the same bowl from before may still be present. The total number is 1.*

7.2. Evaluation Setup

Inference parameters. Table 9 lists the inference hyperparameters used across all evaluations.

Evaluation modes. At inference, each model variant is evaluated in the mode that matches its training configuration. IPT models are evaluated in two settings: (1) *imagination mode*, where the model generates an intermediate image before answering, and (2) *answer-only mode*, where the model produces only a text answer without generating any image. For models trained with visual generation (IPT, Mixed), the VAE weights are always loaded (`visual_gen=True`), and input images are encoded through both the ViT and VAE pathways (`vae_input=True`) to match the training-time encoding. Without setting `vae_input=True`, a train–eval mismatch would occur: during training, input images pass through both VAE and ViT, but the default evaluation behavior sends inputs through ViT only.[1]

Answer extraction. We extract the predicted answer letter from model outputs using a cascading rule-based procedure: (1) parse <answer>X</answer> tags; (2) extract from `\boxed{X}` format; (3) match patterns such as "the answer is X"; (4) detect bold letter formatting (`**X**`); (5) fall back to the last single letter in the response. All benchmarks use the same unified scoring function that compares the extracted letter against the ground-truth answer.

8. Data Curation Details

8.1. Path Tracing

We generate path tracing data from two sources: AI2-THOR (synthetic) and Matterport3D (real-world).

8.1.1. AI2-THOR

Scene selection. We use 120 standard iTHOR [20] scenes spanning four room types (kitchens, living rooms, bedrooms, bathrooms), with 30 scenes per type, split into train (20), val (5), and test (5) per type. The training set additionally incorporates procedurally generated houses from ProcTHOR-10k [10], which include a fifth room type (hallways, offices, dining rooms).

[1] For Path-Tracing, we found that setting `vae_input=False` actually improves generalization to real environments.

Path sampling. For each scene, we sample feasible two-waypoint paths on the navigation mesh, balanced across room types and three distance bins: short (1–2 m), medium (2–4 m), and long ($\geq$4 m). Grid-based path sampling uses a spacing of 0.5 m with a minimum waypoint separation of 1.0 m.

Camera configuration. All views are rendered at 1024×1024 resolution. To increase viewpoint diversity, we randomize the camera height (sampled from seven values between 1.4 and 1.8 m), field of view (75–120), and pitch (-5 to 5).

Rendering. At each sample we render top-down views, egocentric forward views at both endpoints, and a sweep of candidate sideviews at the midpoint M_1. The sideview sweep covers 7 yaw angles $\times$ 7 horizontal offsets $\times$ 3 pitch values, yielding 147 candidate views per midpoint. We select the sideview that best exposes the queried object using simulator segmentation masks, requiring a minimum object coverage of 0.15% of the image area and a maximum view angle of 90 relative to the path direction.

Question generation. Questions are generated from templates ("Which object can you see on your {side} at waypoint M1?") with four choices: the correct answer drawn from verified visible objects and three distractors drawn from the opposite side or a global object pool. Each base MCQ is expanded into eight input variants by combining different image types (top-down path, top-down with arrow, top-down with midpoint marker, dollhouse view) and egocentric cue availability (with/without endpoint views).

TIFA filtering. We apply TIFA-style filtering [16] to ensure question quality. Each candidate is decomposed into binary visibility queries and verified by GPT-4.1 with three-round majority voting, using early exit after round 2 when unanimous. Samples are dropped if (1) the correct answer is not visible in the sideview, (2) a distractor is also visible in the sideview, or (3) the model answers incorrectly even when provided the sideview. We further remove samples where GPT-4.1 answers correctly from the top-down view and egocentric endpoint views alone, ensuring the benchmark requires genuine spatial imagination.

Debiasing. Answer choices are reshuffled with per-sample deterministic seeds to remove positional bias. We additionally verify that per-object and per-room answer distributions remain approximately uniform.

Statistics. The synthetic training set contains 11,204 examples.

Table 9. Inference hyperparameters for evaluation.

Parameter	**Value**
Text temperature	0.3
Sampling	do_sample = True
Max thinking tokens	4,096
Diffusion timesteps	50
Timestep shift	3.0
Text CFG scale	4.0
Image CFG scale	2.0
CFG interval	$[0.0, 1.0]$
Generated image resolution	1024×1024
Max generation rounds	1
GPU allocation	2× A100 80 GB (model parallelism)

8.1.2. Real-World Data (Matterport3D)

To evaluate cross-domain transfer, we construct a real-world test set from Matterport3D [6] top-down views.

Image collection. We collect top-down screenshots from Matterport 3D indoor tours, capturing per-floor views with UI elements removed and dark borders cropped.

Auto-annotation. We annotate walking paths on each image using a two-pass GPT pipeline. In the first pass, GPT proposes N candidate walking paths, each defined by three waypoints (start "1", midpoint "M1", end "2") placed on open floor areas. In the second pass, the proposed waypoints are drawn on the image and GPT is asked to (a) verify that M_1 lies on walkable floor, adjusting its position if necessary, and (b) identify 2–5 visible furniture items or objects on each side (left/right) of the path at M_1.

Post-processing. Several geometric filters are applied to ensure path quality. Waypoints are clamped to image bounds, and paths shorter than 30% of the shorter image dimension are rejected. M_1 is snapped onto the line segment between waypoints 1 and 2 and constrained to the $[0.2, 0.8]$ interval to avoid proximity to endpoints. Paths that traverse dark or background regions (more than 20% dark pixels along the path) are discarded.

TIFA filtering. Because no side-view images exist for real environments, TIFA verification operates on the top-down image only. We verify three properties: (1) all three waypoints lie on walkable floor (not on furniture, walls, or background); (2) each annotated object is visible in the image; and (3) each object is on the correct side of the path. Paths with fewer than 2 verified objects on either side are dropped. Majority voting across up to 3 rounds is used, with early exit when the first two rounds agree.

Human review. After automated filtering, all surviving annotations undergo human review, where annotators can approve, delete, or edit individual paths and their associated object lists.

Question generation. Each verified path yields eight question variants (forward/reverse × left/right × arrow/no-arrow). When the walking direction is reversed ($2 \rightarrow 1$), left and right swap because the agent faces the opposite direction. Distractors are drawn from opposite-side objects first, then supplemented from a global pool of 50 common indoor objects. Answer choices are shuffled with per-sample deterministic seeds to eliminate positional bias. Because the real environments lack egocentric viewpoints, we evaluate on the Path and PathArr settings only. The real-world benchmark contains 332 human-verified questions.

8.2. Perspective Taking

We generate perspective taking data from three sources: AI2-THOR (synthetic), Habitat (photorealistic scans), and Visual Spatial Tuning (real-world images). All sources share the same task structure—given a first-person view with a marked target position, answer a spatial question about the scene from the new viewpoint—but differ in visual domain and 3D engine.

8.2.1. AI2-THOR

Scene selection. We use procedurally generated indoor scenes from ProcTHOR [10], which provides diverse house layouts with varied room configurations. For each scene, we sample multiple camera positions from the navigable area and generate perspective-taking examples at each position.

Target position placement. The target position (marked with a red "X" on the input image) is determined by ray-casting from the camera into the scene. To ensure physically plausible movement targets, we filter ray hits to *ground-only* surfaces: floors, carpets, rugs, and tiles. Hits on furniture, tabletops, or other elevated surfaces are rejected. The hit point must lie within ± 0.2 m of the ground-level height. Up to 40 raycasting attempts are made per sample; if no valid ground hit is found, the sample is skipped.

Viewpoint transformation. The agent is teleported to the target position and rotated 90 in a randomly chosen direction (left or right with equal probability), simulating a realistic movement-and-turn action. A ground-truth novel-viewpoint image is rendered at this new pose to serve as the imaginative perception target.

Object filtering. To ensure well-defined questions, target objects must satisfy several criteria:

- Visible in *both* the original and new viewpoint (dual-view visibility).
- Occupy at least 0.4% of the image area.
- Lie within 5 m of the camera.
- Fall at least 150 px from the image edge (to avoid partially visible objects).
- For relative position questions: the object must be unambiguously on one side of the image center, with a 150 px margin from the center line.

Additionally, we enforce a *left-right eligibility* constraint: an object is only used for relative position questions if it appears on at most one side of the image (not straddling the center), ensuring that the left/right answer is unambiguous.

Question generation. We generate two types of questions with 10 template variants each, varying in person perspective (first, second, third person) and formality level:

- **Distance change**: "After moving to 'X' and turning {direction} 90, will the {object} get closer or further?" Requires a minimum distance change of ± 0.5 m between the old and new camera positions.
- **Relative position**: "After moving to 'X' and turning {direction} 90, will the {object} be on your left or right?" Left/right is determined by the object's 2D position in the new-viewpoint image.

Sub-categories. As described in Sec. 3.1, the six balanced sub-categories arise from the combination of question type and answer: two for distance change (*closer*, *further*) and four for relative position (*left→left*, *left→right*, *right→left*, *right→right*), where the notation indicates the object's lateral position before and after the viewpoint transformation. The training set is balanced across all six sub-categories.

Image annotation. Each input image is annotated in two versions: (1) with only a red "X" marking the target position, and (2) with the "X" plus a blue directional arrow indicating the agent's facing direction after rotation. The arrow version provides an additional spatial cue at evaluation time.

Statistics. The AI2-THOR training set contains 20,531 examples across 98 scenes, with an average of $\sim$210 questions per scene.

8.2.2. Habitat

Scene source. We use photorealistic 3D scans from HM3D (Habitat-Matterport 3D) [6] with semantic annotations. Only single-floor scenes are selected (floor-level Y-variance < 2.0 m) to avoid cross-level ambiguities.

Camera configuration. Images are rendered at 1024×1024 resolution with a horizontal field of view of 90. The sensor height is set to 1.25 m above the navigable floor surface, matching a standing human eye level.

Target position and viewpoint. Camera A (original viewpoint) is placed at a random navigable point with a random yaw. The target position (Camera B) is determined by selecting a visible object as an anchor: Camera B is placed at the object's XZ coordinates, offset slightly along the facing direction to avoid clipping into geometry, and snapped to the nearest navigable point on the mesh (within a 1.0 m snap radius). The ground-truth novel-viewpoint image is rendered at Camera B's position and orientation.

Object filtering. We apply a strict whitelist of $\sim$70 mainstream furniture categories (seating, tables, beds, storage, appliances, bathroom fixtures) and exclude structural elements (walls, floors, doors). Objects must occupy at least 0.8% of the image area in the original frame and at least 0.5% in the imagined frame. To avoid ambiguous references, only objects whose category is *unique* in the frame are used (*e.g.*, if two chairs are visible, neither is selected as a question target). An edge margin of 200 px is applied.

Left/right determination. Object laterality is determined by the mean x-coordinate of the object's semantic segmentation mask in the rendered image, with a 180 px margin from the image center (512 px). Objects falling in the center zone ($332 < x < 692$) are excluded as ambiguous.

Statistics. The Habitat training set contains 19,998 examples balanced across the six sub-categories.

8.2.3. Real-World Data (VST)

To bridge the synthetic-to-real domain gap, the *mixed* training variant incorporates 15,000 real-world examples drawn from the camera motion subset of the Visual Spatial Tuning (VST) dataset [37]. Each example contains a pair of multi-view images captured from different viewpoints in real indoor scenes, along with a question about the camera motion between them and a corresponding answer.

Filtering uncertain answers. We first filter out examples whose answers are uncertain or underspecified using GPT-5.1, prompting it as a binary classifier. Any example whose answer contains phrases such as "cannot be determined," "unknown," or "insufficient information" is removed. The filtering prompt is:

You are a binary classifier.
Proposed Answer:
<start_answer>{A}<end_answer>
If the answer contains ANY of the following phrases or meanings, output "NO":

- *cannot be determined*
- *insufficient information*
- *not enough information*
- *unknown*
- *unclear*
- *cannot tell*
- *impossible to determine*
- *indeterminate*

Only output "YES" if the answer clearly states a specific, determined choice (e.g., a direction, location, label, or concrete option).
Output exactly one token: YES or NO.

Rewriting into generation prompts. After filtering, we use GPT-5.1 to rewrite each question, answer pair into a generation prompt describing the camera motion *from the first image to the second*. This requires careful handling of reference frame direction: if the original question asks where the first camera is relative to the second image, the motion direction must be inverted before constructing the prompt. The rewriting prompt is:

You are given a question about camera motion between two images and its correct answer.

IMPORTANT INVERSION RULE:

- *If the question asks "where is the FIRST camera relative to the SECOND image" (or uses the second image as reference), then the answer describes motion FROM second TO first.*
- *In this case, you MUST invert the direction to get motion FROM first TO second.*
- *If the question asks "where is the SECOND camera relative to the FIRST image" (or uses the first image as reference), NO inversion is needed.*

STEPS:

1. *Determine which image is the reference point in the question.*
2. *If the reference is the second image, invert the direction in the answer.*
3. *Using the final motion FROM first TO second, create a generation prompt.*

INVERSION EXAMPLES:

- *"right" → "left"*
- *"left" → "right"*
- *"front" → "back"*
- *"back" → "front"*
- *"front left" → "back right"*
- *"back right" → "front left"*

Your output should start with "generate" and describe viewing the scene from the new camera position after applying the motion from the first image.

Question: <start_question>{Q}<end_question>
Answer: <start_answer>{A}<end_answer>

First, identify the reference image. Then apply inversion if needed. Then generate your output.

The resulting generation prompts condition the model on the first view and the inferred camera motion, with the second view serving as the imaginative perception target. Because no programmatic 3D annotation is available for these real scenes, this data serves as a domain bridge rather than a source of the full six-sub-category question format.

Mixed training composition. The mixed PET training variant combines AI2-THOR (20,531), Habitat (19,998), and VST (15,000) examples, totaling 55,529 samples.

8.3. Multiview Counting

We construct multiview counting data from both synthetic and real-image sources. Our main training set is generated from ProcTHOR/AI2-THOR environments, which provide full 3D supervision for both egocentric observations and top-down bird's-eye-view (BEV) targets. To complement this synthetic source, we additionally curate two real-image multiview counting sets from MessyTable and ScanNet++, which expose the model to real visual appearance and partial observability under natural image statistics.

8.3.1. ProcTHOR / AI2-THOR

We generate the main multiview counting training set from AI2-THOR environments, using two trajectory types that capture complementary modes of partial observability.

Trajectory types.

- **Rotation**: The agent remains at a fixed position and rotates in $90°$ increments through four cardinal directions ($0°, 90°, 180°, 270°$), producing four frames that together cover a $360°$ panorama of the surrounding area.
- **Multi-camera**: The agent traverses a square path, capturing one frame at each of four corners. This setup simulates a multi-camera rig where viewpoints are spatially distributed around the scene.

Both trajectory types produce exactly four input frames per sample.

Bird's-eye view (BEV) generation. The ground-truth intermediate image is a top-down BEV map rendered from an overhead camera in the 3D scene. To ensure the BEV only covers the *explored* area (the region visible from the input frames), we crop the map with trajectory-aware padding: 5 m around the agent position for rotation trajectories and 4 m around the traversed path for multi-camera trajectories. Object counts in the cropped BEV are validated against the segmentation maps to ensure consistency.

Object filtering and category balancing. Structural elements (walls, floors, ceilings, doorways) are excluded from counting targets. Target objects must be visible in both the first-person frames and the cropped top-down segmentation map (with a minimum coverage of 0.1%). Because initial generation heavily favors count$=1$ questions ($\sim$82%), we apply iterative rebalancing: high-frequency categories are capped at 9.9% of the dataset, and count$=1$ samples are downsampled. This produces a more uniform distribution across object categories and count values.

Question format and distractor generation. Questions follow the template: "How many {category}(s) are in this area?" with four answer choices (A–D). Distractors are sampled from ±1 and ±2 of the correct count, producing plausible alternatives. Negative counts are removed, and the four options are shuffled with a per-sample deterministic seed to eliminate positional bias.

Statistics. The synthetic training set contains 17,079 examples generated from ProcTHOR [10] scenes, covering both trajectory types.

8.3.2. MessyTable

To expose the model to real tabletop imagery with severe clutter and occlusion, we construct an additional multiview counting set from MessyTable [4]. Each scene contains multiple camera views of the same tabletop arrangement together with instance-level annotations.

Scene-level counting targets. For each scene, we aggregate annotations across all cameras and de-duplicate instance IDs across views, so that the ground-truth answer corresponds to the number of unique physical objects rather than the sum of per-view detections. Counting targets are defined at the subclass level and mapped to readable category names.

Target and view sampling. For each scene, we sample one target category from the categories present in the scene. To reduce the dominance of trivial singleton cases, sampling is biased toward categories with count ≥ 2: when both singleton and multi-instance categories are available, 90% of samples are drawn from the multi-instance bucket and 10% from the singleton bucket. Input images are selected from the eight surrounding non-top cameras, with priority given to views in which the target category is absent. When too few such views exist, they are supplemented with additional non-adjacent views to maintain viewpoint diversity. This makes the final count require aggregation across multiple views rather than inspection of a single image.

Top-view supervision and question generation. Each sample also stores the canonical top-view image as the reasoning target. In the exported JSONL format, image inputs use centered crops derived from the union of all annotated object boxes in each selected camera view, which reduces empty borders while preserving the visible object layout. Questions are instantiated from a diverse pool of natural-language counting templates such as "How many {object} are in this scene?"

8.3.3. ScanNet++

We further construct a real indoor multiview counting set from ScanNet++ [41], using iPhone image trajectories paired with labeled 3D scene reconstructions. Compared with MessyTable, this set covers larger indoor spaces, more varied viewpoints, and more realistic household layouts.

Top-down map and candidate view generation. For each scene, we first generate a top-down map from the labeled 3D reconstruction. Because raw point-cloud renderings are visually sparse and unrealistic, we further use Qwen-Image-Edit [32] to transform the rendered top-down visualization into a more realistic top-down image while preserving the scene layout. We then build a candidate egocentric view pool by combining a small set of canonical iPhone views with additional randomly sampled frames, requiring each extra frame to differ from the canonical views by at least a minimum yaw angle.

Visibility estimation and target selection. We estimate which object instances are visible in each candidate frame by projecting the labeled 3D scene into the camera views using mesh ray-casting, and use the semantic annotations to obtain scene-level category counts. Top-down maps are filtered by automatic quality rules to remove blurry, blank, or low-texture renderings. Candidate counting targets are restricted to non-structural object categories with bounded scene-level counts and sufficient visible support in the candidate views. To avoid metadata leakage, target selection is performed in a blind setting: the model is shown the top-down image and candidate labels, but not their annotated counts, and we keep only categories whose visually predicted count matches the ground truth. When multiple valid categories remain, we preferentially sample categories with counts greater than one.

Final evidence image selection. The final evidence set is selected in two stages. We first greedily choose images that jointly cover all instances of the target category while enforcing a minimum yaw-separation constraint. We then fill the remaining slots with views that add new foreground content and viewpoint diversity. Each final sample contains 5–8 egocentric images together with the top-down map as the reasoning image, and the question is rewritten into a natural counting form.

9. Additional Results

Table 2 in the main paper reports path tracing accuracy averaged across input settings. Table 10 provides the full per-split breakdown for both AI2-THOR (EgoDir, Path, PathArr) and different-environment (Real, Real+Arr) benchmarks.

10. Visualizations

10.1. Path Tracing

Inference with imaginative perception. Figure 4 shows examples of path tracing inference with imaginative perception tokens in the EgoDir setting. Path tracing presents a particularly challenging imagination target: the model must synthesize a first-person sideview at midpoint M_1 from a top-down map and two egocentric endpoint views, requiring accurate reasoning about camera height, 3D object layout, and occlusion from a bird's-eye representation. As shown in the figure, the generated visual thoughts are often spatially imprecise, with noticeable artifacts and layout errors compared to ground-truth sideviews. Despite this, the model frequently arrives at the correct answer (rows 1 and 2). This observation suggests that the value of imaginative perception training lies not in producing pixel-accurate intermediate images, but in encouraging the model to internalize spatial reasoning during training. The imagination supervision acts as an auxiliary signal that shapes the model's internal spatial representations, enabling it to reason about 3D visibility even when the externalized image is imperfect. This is further supported by the finding in our main experiments that IPT-trained models achieve strong performance in answer-only mode, where no image is generated at inference time.

Dataset examples. Figure 5 shows representative examples from both the real-world (Matterport3D) and synthetic (AI2-THOR/ProcTHOR) path tracing datasets across all input settings. The real-world examples (top) use photographic top-down views and are evaluated in the Path and PathArr settings only, since real environments do not provide egocentric viewpoints at arbitrary positions. The synthetic examples (bottom) include ground-truth sideview images rendered at the midpoint M_1, which serve as the imaginative perception target during training. The PathArr setting provides a directional arrow at M_1 indicating the query direction, while the EgoDir setting provides egocentric forward views at both endpoints in addition to the top-down map.

10.2. Perspective Taking

Inference with imaginative perception. Figure 6 shows examples of perspective taking inference with imaginative perception tokens. The model receives a first-person view with an "X" mark indicating the target position, generates an imagined novel viewpoint as a visual thought, and predicts whether an object is closer/further or on the left/right. As with path tracing, the generated visual thoughts are not pixel-perfect but capture the essential spatial layout. The first two rows show correct predictions where the model successfully imagines the scene from the new viewpoint. The third row shows a failure case where the model incorrectly predicts the relative position.

Dataset examples. Figure 7 shows representative examples from the AI2-THOR/ProcTHOR perspective taking dataset across all four sub-categories: distance change (closer/further) and relative position (left→right / right→left). Each example shows the input image with the target position marked by "X" and the ground-truth novel viewpoint after moving to "X" and turning.

10.3. Multiview Counting

Inference with imaginative perception. Figure 8 shows examples of multiview counting inference with imaginative perception tokens. The model receives four egocentric views from a rotation or multi-camera trajectory, generates a top-down BEV map as its visual thought, and counts the target objects. The generated top-down maps show that the model learns to synthesize a bird's-eye view from multiple perspectives, capturing the approximate room layout and object placements. The first two rows show correct predictions; the third row shows a failure case.

Dataset examples. Figure 9 shows representative examples from the AI2-THOR/ProcTHOR multiview counting dataset for both trajectory types: rotation (four cardinal directions from a fixed position) and multi-camera (four cameras placed at the corners of a square path). Each example shows the four input views and the ground-truth top-down map with agent/camera positions annotated.

11. Imaginative Token Exploration with Different VLMs

Prior to adopting unified models like BAGEL for imaginative perception token generation, we investigated adding

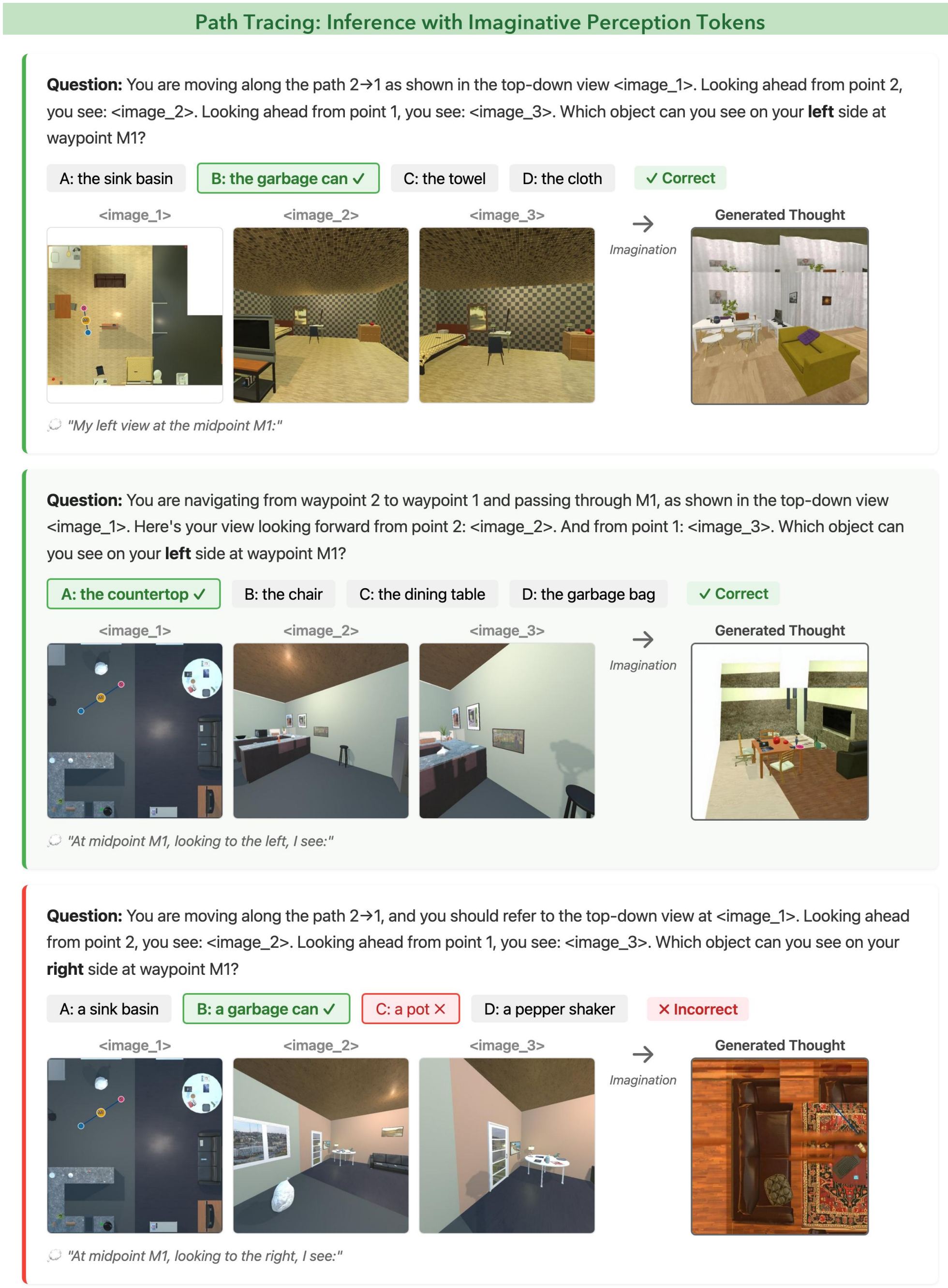


Figure 4. **Path tracing with imaginative perception tokens (EgoDir setting).** The model receives a top-down map (`<image_1>`) and egocentric views at the two endpoints (`<image_2>`, `<image_3>`), generates a visual thought (imagined sideview at M_1), and predicts an answer. Although the generated thoughts exhibit spatial imprecision and artifacts, the model still arrives at the correct answer in the first two examples, suggesting that imagination training encourages internalized spatial reasoning rather than reliance on pixel-accurate intermediate outputs. The third row shows a failure case. Correct answers are highlighted in green; incorrect predictions in red.

discrete imaginative perception tokens directly to the language model vocabulary of state-of-the-art vision-language models. Inspired by Aurora [2], we first trained a VQ-VAE from scratch on the intermediate RGB images in our datasets, including novel viewpoint renders, top-down BEV maps, and sideview images. However, the reconstruction

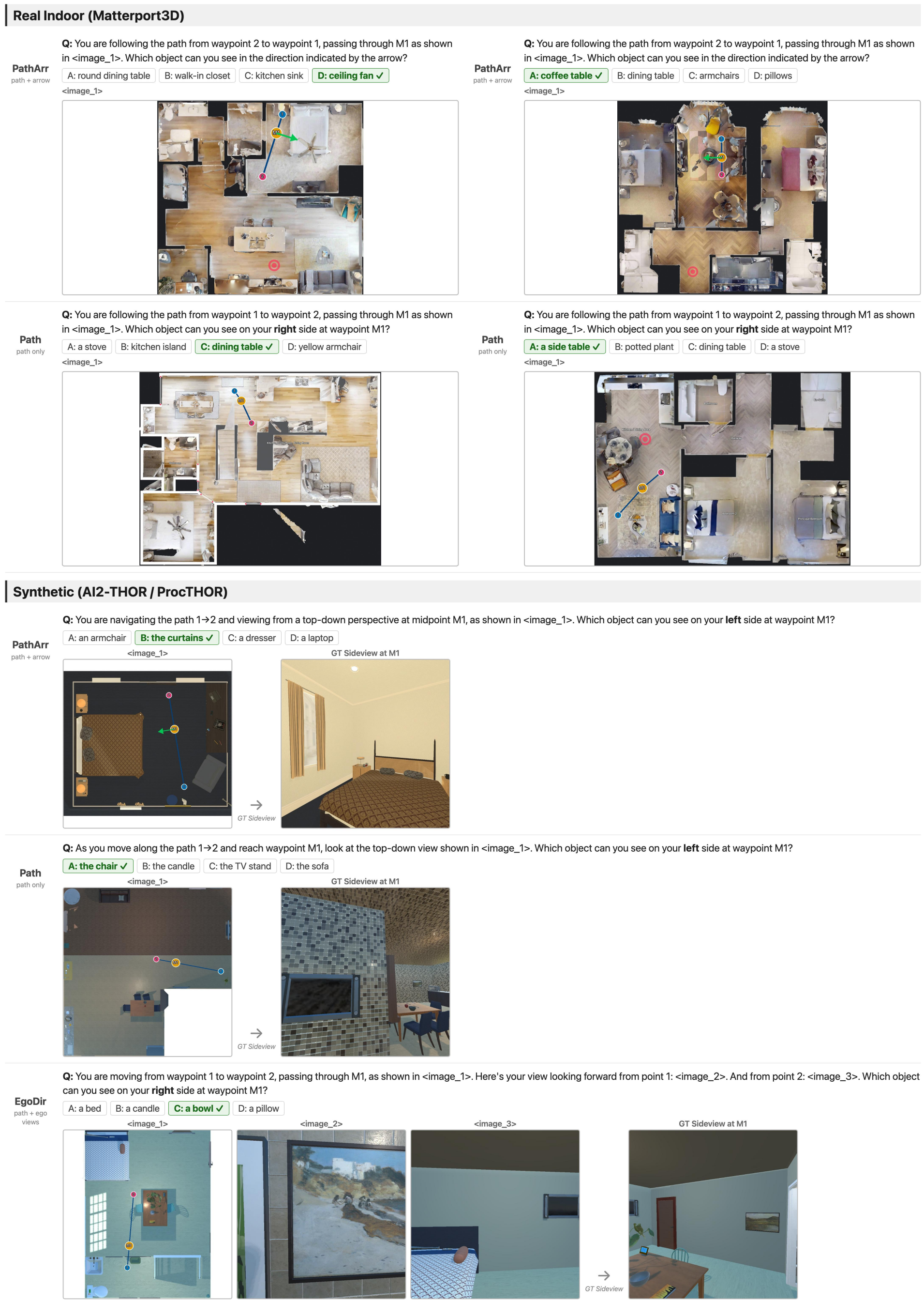


Figure 5. **Path tracing dataset examples.** Top: real-world examples from Matterport3D in the PathArr and Path settings. Bottom: synthetic examples from AI2-THOR/ProcTHOR in the PathArr, Path, and EgoDir settings, with ground-truth sideviews at midpoint M_1 shown on the right. Each example shows the input image(s), question, and four answer choices with the correct answer highlighted.

quality of these simple VQ-VAEs was insufficient for supervising models on their intermediate token sequences as image outputs.

We therefore switched to off-the-shelf pretrained VQ-GANs [14] with varying configurations. These configurations differ along two axes: codebook size (e.g., 1K, 8K, and 16K entries) and spatial downsampling ratio ($f = 8$ vs. $f = 16$). Each choice involves a tradeoff: a larger codebook improves representational fidelity but inflates the model vocabulary, while a smaller downsampling ratio yields higher

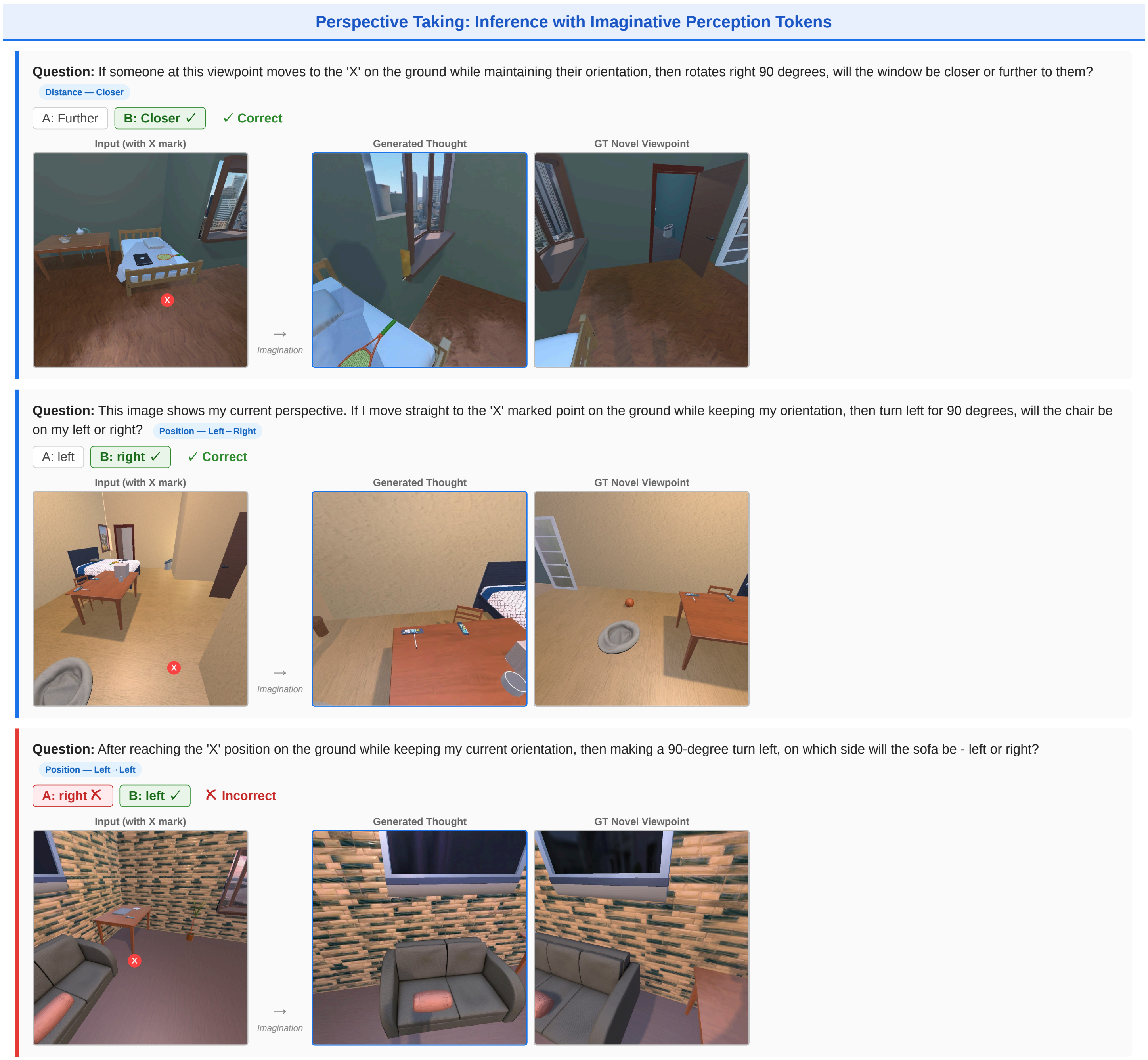


Figure 6. **Perspective taking with imaginative perception tokens.** The model receives an input view with an "X" mark on the ground, imagines the scene from the target viewpoint, and predicts spatial relationships. Generated thoughts are compared against ground-truth novel viewpoints. Correct answers are highlighted in green; incorrect predictions in red.

reconstruction quality at the cost of a longer token sequence per image, increasing context length. Figure 10 illustrates reconstruction quality across these settings.

We selected Qwen2.5-VL [32] in two sizes (3B and 7B) as our backbone for discrete token finetuning experiments. We note that the training data at this stage was of lower quality than our final datasets described in main text; these experiments were intended solely to probe whether discrete imaginative perception tokens can serve as a useful intermediate, not to achieve peak performance. We focused on

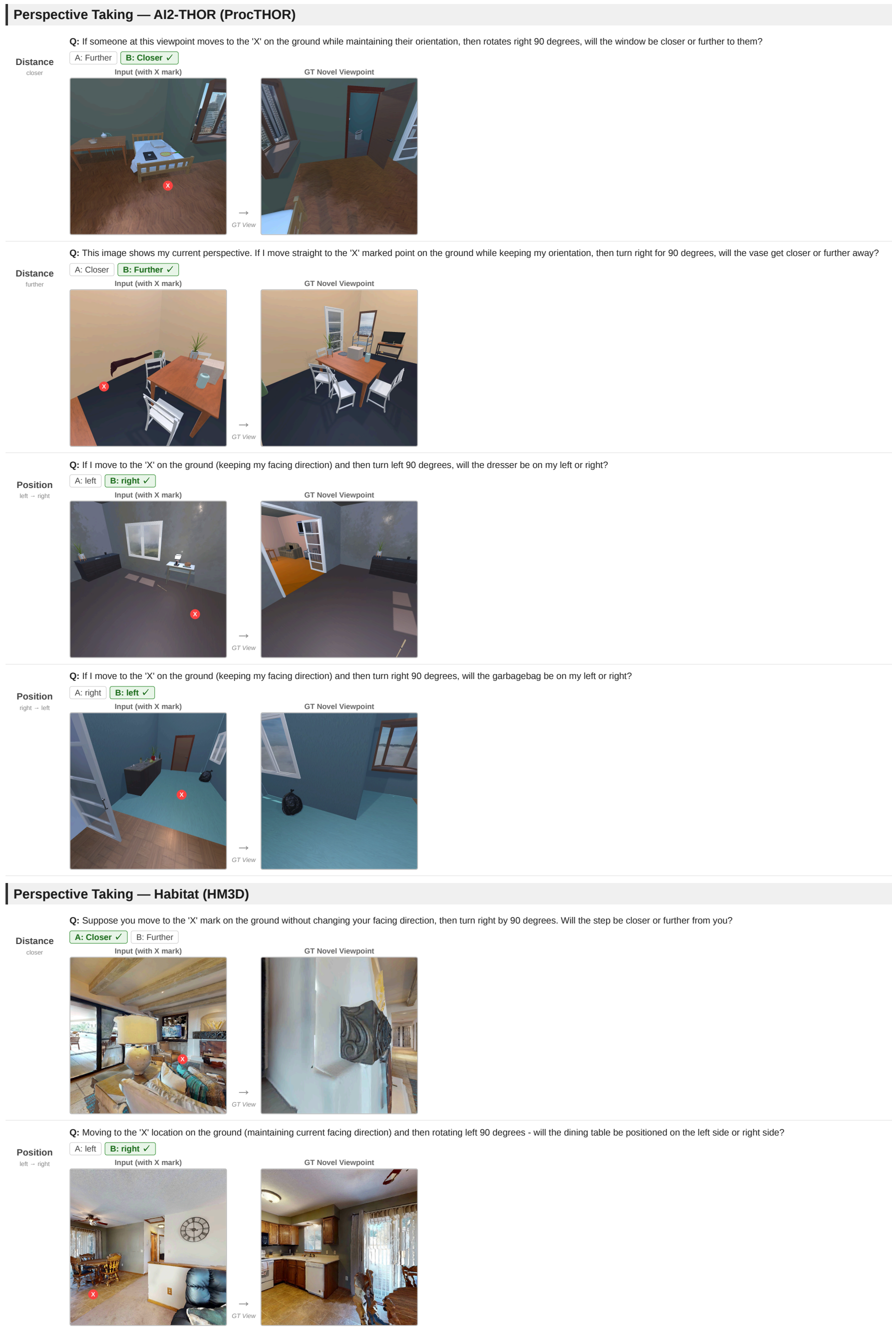


Figure 7. **Perspective taking dataset examples.** Examples from AI2-THOR/ProcTHOR across the four sub-categories. Each example shows the input view with "X" mark and the ground-truth novel viewpoint. The correct answer is highlighted in green.

Path Tracing (PT) and Perspective Taking (PET).

For each model we trained three variants: answer-only finetuning, Text CoT, and image chain-of-thought with discrete IPTs, where the VQGAN codebook tokens are appended to the model vocabulary and the model first autoregressively generates the imaginative perception token sequence before predicting the final answer plus a zeroshot baseline. For IPT variants we tested two VQGAN settings that keep sequence length manageable: CB 16K f=16 and CB 1K f=16. Results are shown in Table 11.

IPT consistently outperforms both answer-only finetuning and Text CoT on Path Tracing, with CB 1K f=16 yielding the best results for both model sizes (55.0 for 3B, 55.9 for 7B). On Perspective Taking, gains are modest and near the zero-shot baseline, suggesting that lower data quality and the representational limitations of discrete token reconstruction are a bottleneck for this more visually demanding task. A substantial gap remains relative to our final BAGEL-based results, motivating the move to a unified model.

Figure 11 shows ground-truth imagination images alongside the corresponding IPTs decoded from the Qwen2.5-VL 3B model. The decoded outputs are visually degraded, lacking the spatial structure and object detail present in the ground truth, which helps explain the remaining performance gap and further motivated our switch to continuous latent representations.

We further investigated alternative intermediate image representations. Instead of training the model to generate imaginative perception tokens for RGB thought images, we replaced them with tokens for grayscale images and tokens for pseudo depth maps obtained from the DepthAnything model [36]. The intuition is that simplifying the generation target, from full RGB to grayscale, reduces the difficulty of the token prediction task and may improve spatial reasoning downstream. Results in Table 12 show that switching from RGB to grayscale does boost performance (55.0→59.6 on PT, 50.0→55.5 on PET), while depth tokens perform comparably to RGB. Nevertheless, a substantial gap remains, and Figure 12 shows that the decoded grayscale outputs are still visually degraded, indicating that generation quality rather than representation type is the primary bottleneck.

Figure 12 shows ground-truth imagination images alongside the corresponding IPTs decoded from the Qwen2.5-VL 3B model. The decoded outputs are visually degraded, lacking the spatial structure and object detail present in the ground truth, which helps explain the remaining performance gap and further motivated our switch to continuous latent representations.

These findings collectively motivated us to move away from discrete token generation in non-unified VLMs and instead adopt a unified model: BAGEL, that natively supports interleaved image understanding and generation through continuous latent representations.

Table 10. **Path tracing per-split results.** Accuracy (%) broken down by input setting. The main paper reports the average across these splits. For our models, accuracy reports the maximum between answer-only and free-generation inference. Best per group in **bold**.

	AI2-THOR			Different Env.	
Model	EgoDir	Path	PathArr	Real	Real+Arr
VQA Models					
GPT-5	**61.1**	**56.8**	**62.6**	**74.5**	87.3
GPT-5.2	22.1	40.2	36.3	57.5	68.4
Gemini 2.5 Flash	45.1	37.9	41.5	58.0	84.8
Gemini 3 Flash	48.7	39.1	39.2	70.1	**96.2**
InternVL3.5-8B	43.4	29.0	35.1	45.4	49.4
Qwen2.5-VL-7B	44.2	32.5	35.1	47.1	42.4
Qwen3-VL-8B	31.9	33.7	42.1	52.9	75.3
Unified Models					
Janus-Pro-7B	36.3	30.2	33.9	34.5	36.1
Chameleon 7B	5.3	23.7	19.9	23.0	25.9
Ours (fine-tuned BAGEL)					
Bagel (base)	36.3	26.0	27.5	39.7	45.6
Bagel (label-only)	**73.5**	61.5	62.0	46.6	62.7
+ Text CoT	53.1	47.9	48.0	**52.3**	51.3
+ IPT	61.1	43.2	42.7	46.6	**68.4**
+ Mixed Training	71.7	**65.1**	**63.2**	50.6	66.5

Table 11. **Discrete IPT experiments on Qwen2.5-VL.** Accuracy (%) on Path Tracing (PT) and Perspective Taking (PET). Training data at this stage was of lower quality than the final datasets. "–" denotes experiments not conducted.

Model	**Method**	**PT**	**PET**
Qwen2.5-VL 3B	Zero-shot	33.0	50.0
	Answer-only	48.6	50.5
	Text CoT	43.0	–
	IPT (CB 16K, f=16)	50.5	48.5
	IPT (CB 1K, f=16)	**55.0**	50.0
Qwen2.5-VL 7B	Zero-shot	38.5	47.2
	Answer-only	37.6	–
	Text CoT	35.7	–
	IPT (CB 16K, f=16)	55.0	–
	IPT (CB 1K, f=16)	**55.9**	–

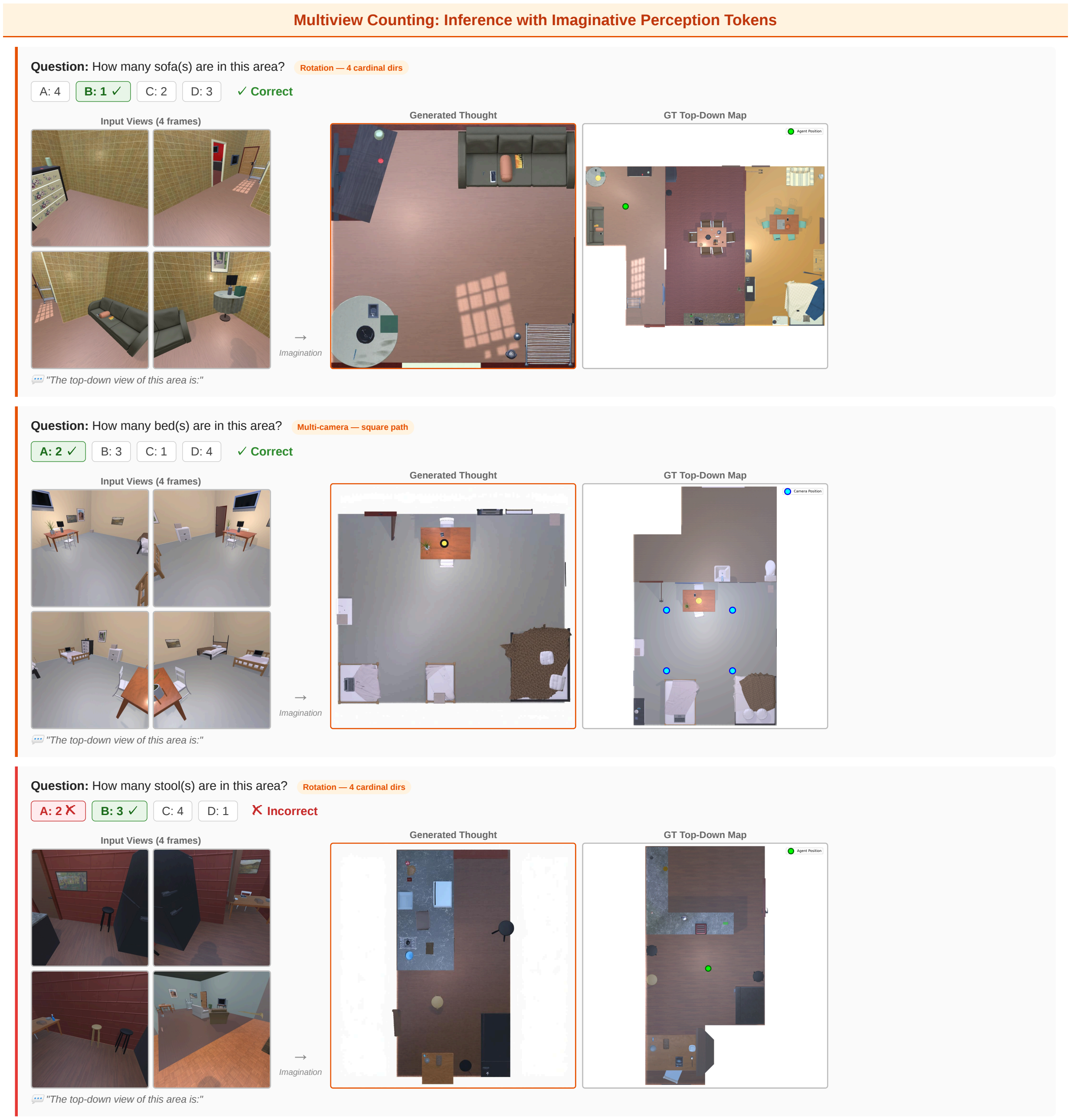


Figure 8. **Multiview counting with imaginative perception tokens.** The model receives four egocentric views, imagines a top-down BEV map, and counts target objects. Generated thoughts are compared against ground-truth top-down maps. Correct answers are highlighted in green; incorrect predictions in red.

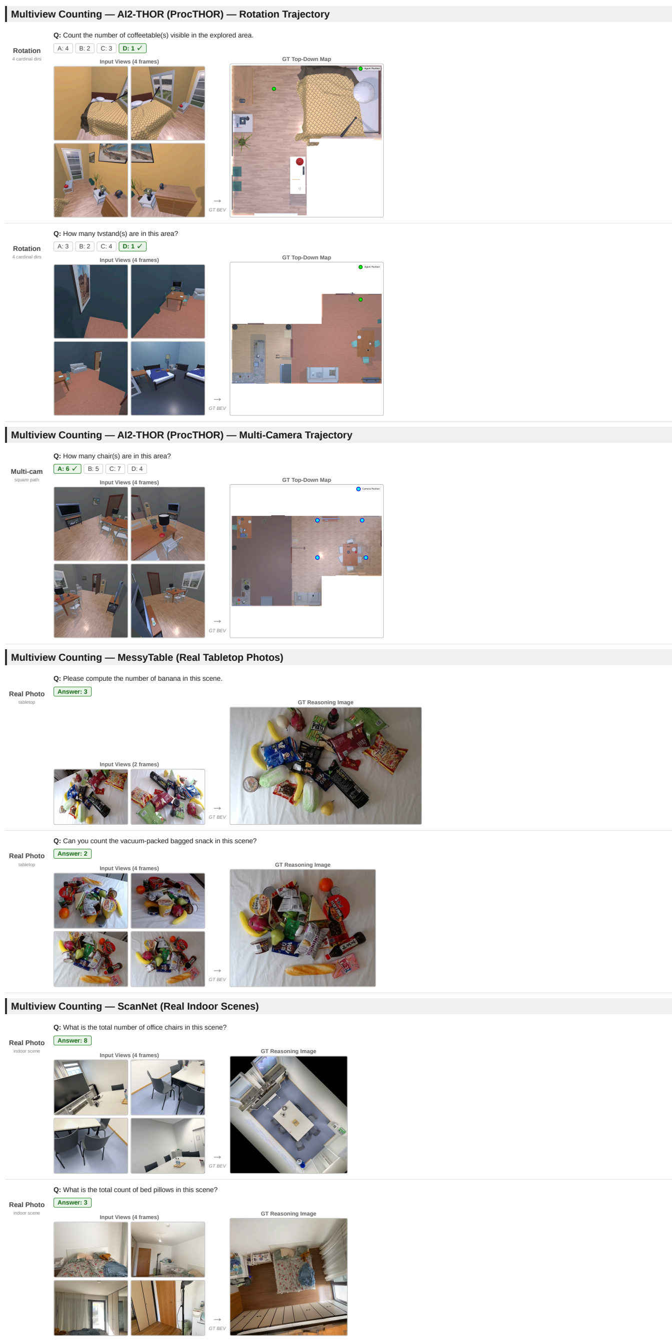


Figure 9. **Multiview counting dataset examples.** Examples from AI2-THOR/ProcTHOR showing both rotation (top) and multi-camera (bottom) trajectories. Each example shows four input views in a 2 × 2 grid and the ground-truth top-down map. The correct answer is highlighted in green.

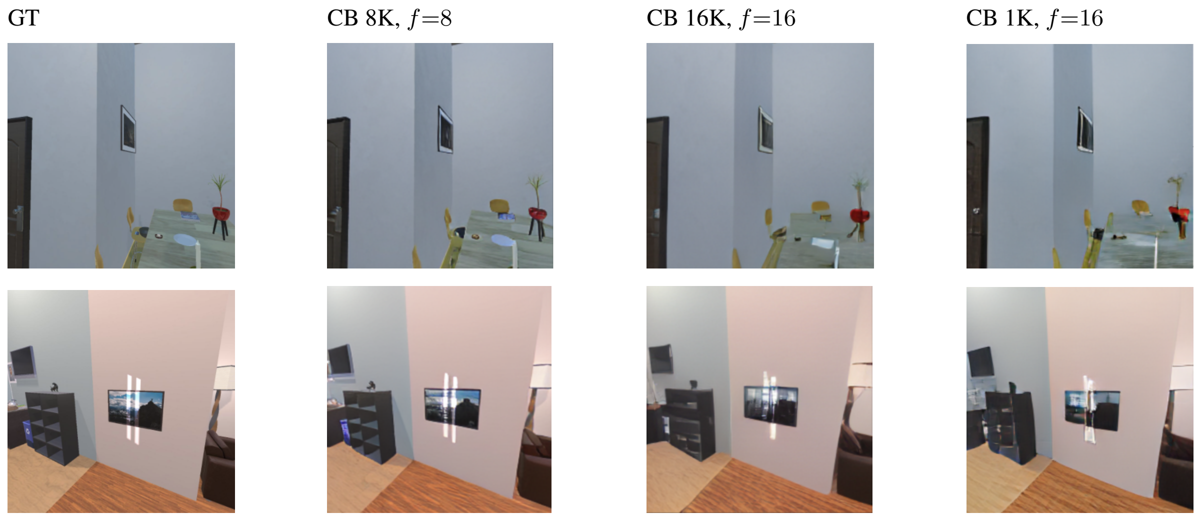


Figure 10. **VQGAN reconstruction quality across codebook and downsampling settings.** Larger codebooks and smaller f improve fidelity but increase vocabulary size and sequence length respectively.

Figure 11. **Ground-truth vs. decoded IPTs from Qwen2.5-VL 3B.** The model-generated imagination tokens decode into visually degraded images that fail to preserve the spatial structure of the ground truth, highlighting the limitations of discrete token generation in non-unified VLMs.

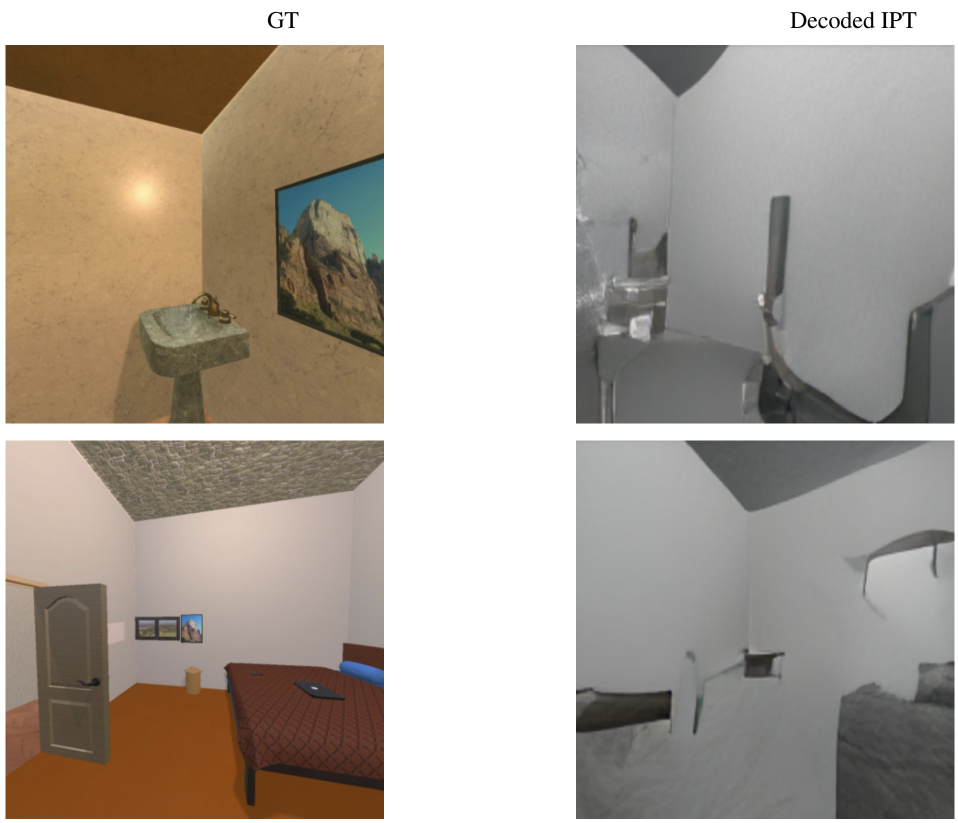


Figure 12. **Decoded IPTs from Qwen2.5-VL 3B for grayscale representations.** Despite the simpler generation target, grayscale decoded outputs remain visually degraded and fail to preserve spatial structure.

Table 12. **Effect of intermediate image representation on Qwen2.5-VL 3B.** Accuracy (%) on Path Tracing (PT) and Perspective Taking (PET).

Method	**PT**	**PET**
IPT RGB (CB 1K, f=16)	55.0	50.0
IPT Grayscale (CB 1K, f=16)	**59.6**	**55.5**
IPT Depth (Aurora VQVAE)	55.0	54.7