A. Implementation Details

A.1. Room-Instance Tokenization

We employ two lightweight VQ-VAE branches (Fig. A.1) to discretize floor plan geometry into spatial tokens: an **outline branch** that encodes the global building boundary, and a **conditional room branch** that encodes each room conditioned on its corresponding outline. Specifically, the room encoder concatenates the outline mask as an additional input channel to preserve adjacency and boundary consistency. Together, these branches produce a hierarchical spatial vocabulary serving as the foundation for multimodal alignment and instruction tuning.

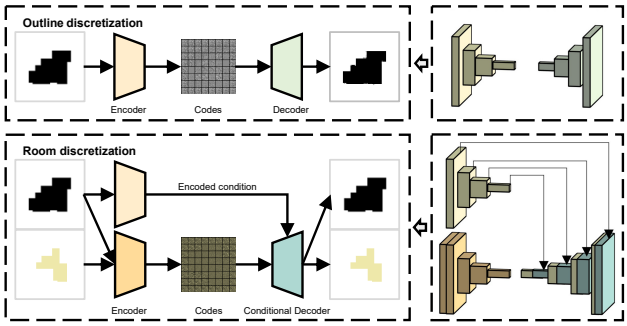

Figure A.1. **VQ-VAE tokenization framework.** The outline branch encodes the global boundary, while the conditional room branch encodes each room with outline context to capture spatial relations.

We study reconstruction fidelity with respect to token granularity and codebook capacity. As shown in Tables A.1 and A.2, token count strongly influences downstream sequence length and contextual reasoning ability: 2×2 tokens underfit geometry, while 16×16 tokens generate unnecessarily long sequences. Hence, we focus on 4×4 and 8×8 configurations for practical balance.

Table A.1. Outline branch: PSNR (dB) and SSIM under different token grids and codebook sizes.

N_{tokens}	Codebook	PSNR	SSIM
4×4	256	20.003	0.911
4×4	512	22.188	0.946
4×4	1024	23.202	0.957
8×8	256	35.898	0.998
8×8	512	36.657	0.998
8×8	1024	37.053	0.998

Since each outline corresponds to multiple room instances, the room-level data volume is substantially larger, necessitating a smaller learning rate and fewer epochs for stable convergence. Considering the trade-off between fidelity and efficiency, 8×8 tokens deliver clearly superior reconstruction to 4×4 while avoiding the context-length overhead of 16×16. Although increasing the codebook size yields modest gains, a compact codebook of 256 effectively balances information density and vocabulary size.

Table A.2. Room branch: PSNR (dB) and SSIM under different token grids and codebook sizes.

N_{tokens}	Codebook	PSNR	SSIM
4×4	256	31.896	0.994
4×4	512	33.106	0.995
4×4	1024	34.506	0.997
8×8	256	44.624	1.000
8×8	512	48.467	1.000
8×8	1024	51.769	1.000

Concretely, both branches adopt a latent dimension of 256 with three downsampling layers. The outline VQ-VAE is trained for 50 epochs with batch size 256 and learning rate 3×10^{-4}, while the conditional room VQ-VAE is trained for 30 epochs with batch size 256 and learning rate 1×10^{-4}.

A potential concern is whether the VQ-VAE discretization introduces an information bottleneck that limits downstream generation quality. To assess this, we first examine reconstruction fidelity. As the reconstruction PSNR is controlled around 40 dB, it is already close to a near-lossless level for layout geometry. Qualitative comparisons in Fig. A.1 show that reconstructed layouts are almost indistinguishable from the original inputs, indicating minimal information loss during tokenization.

We further conduct a codebook size ablation study to evaluate whether generation performance is sensitive to token vocabulary capacity. Results are summarized in Table A.3. Across codebook sizes ranging from 256 to 1024, generation metrics (Macro IoU, FID, GED, Node F1, and Edge Overlap) remain largely stable. The performance variation is marginal, suggesting that the discretized latent space preserves sufficient structural information for downstream reasoning.

These results indicate that the VQ-VAE module does not constitute a performance bottleneck in our pipeline.

Table A.3. Codebook size ablation study

Size	Ma. IoU	FID↓	GED↓	Node F1	Edge Ovl.
256	0.654	**1.89**	1.03	0.994	0.880
512	**0.657**	1.95	1.04	**0.995**	0.877
1024	**0.657**	1.97	**1.00**	0.994	**0.881**

A.2. Multimodal Alignment and Instruction Tuning

LLM training is conducted on Ubuntu 22.04.5 LTS with an AMD EPYC 7K62 48-Core CPU (96 threads), 256 GB RAM, and an NVIDIA GeForce RTX 5090 GPU (32 GB VRAM). All experiments are implemented using the LLaMA-Factory framework with the Qwen3-0.6B backbone and FlashAttention-2 acceleration. To integrate spatial information, we extend the original tokenizer by manually appending a set of discrete outline and room tokens derived from the VQ-VAE codebooks, enabling the LLM to process geometric and semantic content within a unified vocabulary. Each multimodal sample contains up to 2048 tokens, including spatial tokens and textual instructions. The model is trained in three sequential stages following the pipeline described in Sec. 4.2: embedding initialization, multimodal pre-training, and instruction tuning. A cosine learning-rate schedule with 10% warm-up is adopted throughout. Early stages employ a small effective batch size (4 with gradient accumulation) to stabilize optimization under long-sequence multimodal inputs, while Stage 3 uses a larger batch size (16) for instruction-level generalization. Learning rates are set to 1×10^{-5} for Stages 1–2 and 2×10^{-5} for Stage 3, with 1, 2, and 3 epochs respectively.

A.3. Baseline Methods

To ensure a fair comparison, all baseline methods are reproduced or standardized under a unified training and inference pipeline. All models are evaluated at 256×256 resolution with wall-boundary restoration, and inference is conducted on a single RTX 3090 GPU (an Intel Xeon E5-2682 v4 CPU and 32 GB RAM).

Understanding. For multimodal reasoning baselines, including LLaVA-v1.6-Mistral-7B-HF, Qwen3-VL-8B-Instruct, InternVL3.5-8B, and MiniCPM-V 4.5, we use their official checkpoints and serve them through vLLM for efficient inference. Each model receives the color-mapped layout and text instruction, producing structured descriptions for evaluating room-type classification, localization accuracy, and relational reasoning. The prompt template used for understanding is as follows:

Prompt Template for Understanding:

```
You are an expert in architectural layout
understanding.
The input is a color-coded floor plan image
with an overall size of 18m × 18m.
Each color approximately represents a
functional area, and the full layout fits
within this boundary.
Color-to-room mapping:
Light khaki or pale yellow → Livingroom
(large open space)
Orange → Masterroom (private sleeping area)
Light salmon or red → Kitchen (food
preparation area)
Light cyan → Bathroom (toilet or shower
area)
Violet → Diningroom (area for meals)
Plum → Storage (small enclosed area)
Bright yellow → Commonroom (Secondroom /
Studyroom / Childroom / Guestroom)
Olive green → Balcony (semi-outdoor area)
Black → Exterior wall or building boundary
White → Front door, main entrance, interior
walls, interior doors, and external
background
Example:
Below is an example floor plan analysis.
Input image: [example_floorplan.png]
Output JSON:
{
'rooms': [
{'idx': 0, 'type': 'LivingRoom', 'area':
33, 'width': 6, 'height': 9, 'position':
'east'},
{'idx': 1, 'type': 'SecondRoom', 'area':
12, 'width': 3, 'height': 4, 'position':
'northwest'},
{'idx': 2, 'type': 'MasterRoom', 'area':
12, 'width': 3, 'height': 5, 'position':
'southwest'},
{'idx': 3, 'type': 'StudyRoom', 'area':
11, 'width': 3, 'height': 4, 'position':
'north'},
{'idx': 4, 'type': 'Bathroom', 'area':
4, 'width': 2, 'height': 2, 'position':
'west'},
{'idx': 5, 'type': 'Kitchen', 'area':
4, 'width': 2, 'height': 2, 'position':
'northeast'}
],
'edges': [
{'room1': 5, 'room2': 3, 'relation':
'right-of', 'text': 'Kitchen is right-of
StudyRoom'},
{'room1': 5, 'room2': 0, 'relation':
'above', 'text': 'Kitchen is above
LivingRoom'},
{'room1': 1, 'room2': 3, 'relation':
'left-of', 'text': 'SecondRoom is left-of
StudyRoom'},
{'room1': 1, 'room2': 4, 'relation':
'above', 'text': 'SecondRoom is above
Bathroom'},
{'room1': 4, 'room2': 2, 'relation':
'above', 'text': 'Bathroom is above
MasterRoom'}
],
'description': 'The floor plan centers
around the spacious Living Room, with
surrounding rooms arranged by clear spatial
logic.'
}
Now follow this example format for the next
images.
Instruction: Provide both room (node)
attributes and spatial (edge) relations in
JSON format.
Your task:
- Identify all rooms (nodes) within the 18m ×
```

```
18m plan.
- For each room, estimate attributes:
- idx (int): unique ID
- type (str): functional category (e.g.,
LivingRoom, Kitchen)
- area (float): area in square meters
- width, height (float): dimensions in
meters
- position (str): coarse location in the 18m
× 18m layout (e.g., north, center, southeast)
- Infer spatial relations (edges) between
rooms using ONLY the following edge types:
['left-above', 'left-below', 'left-of',
'above', 'inside', 'surrounding', 'below',
'right-of', 'right-above', 'right-below']
- Each edge must:
- reference valid room indices (room1, room2)
- use exactly one relation from the list
above as "relation"
- include a short human-readable "text"
Semantics (guidance):
- "left-of"/"right-of"/"above"/"below":
strict axis-aligned relations.
- "left-above", "left-below", "right-above",
"right-below": diagonal/oblique relations
combining horizontal and vertical.
- "inside": room1 is fully inside room2
(room2 acts as container).
- "surrounding": room1 encloses or wraps
around room2 (the inverse of "inside").
Output format (must be valid JSON; use double
quotes):
{
"rooms": [{"idx": 0, "type": "RoomType",
"area": 0.0, "width": 0.0, "height": 0.0,
"position": "center"}],
"edges": [{"room1": 0, "room2": 1,
"relation": "left-of", "text": "Room0 is
left-of Room1"}],
"description": "One-sentence summary of the
layout."
}
Requirements:
- Use only the allowed edge types listed
above.
- All dimensions/positions are interpreted
relative to the 18m × 18m plan.
- Output a single valid JSON object and
nothing else.
```

Generation. Tell2Design and ChatHouseDiffusion are reproduced using their released implementations. Tell2Design performs direct text-to-layout generation, while ChatHouseDiffusion employs diffusion sampling with language conditioning. Since the official FloorPlan-LLaMA is not publicly available, we reproduce it using a VQ-VAE tokenizer and Qwen backbone. Floor plans are converted to grayscale and tokenized with 16×16 latent grids, codebook size 1024, latent dimension 256. The tokenizer reaches approximately 38.1 dB PSNR and 0.99 SSIM, confirming high geometric fidelity before language alignment. Qwen-Image-Edit-2509 is also included as a generation baseline. The prompt template used for generation is:

Prompt Template for Generation:

```
You are an expert in architectural layout
generation.
The input image is a black building outline.
Please generate a complete flat color-blocked
floor plan within the provided outline.
- Keep the wall boundaries fixed and fill
enclosed regions with appropriate colors.
- Each room or functional area should be
represented by a solid color block according
to the color legend.
- The generated layout must align with the
spatial description below.
- Avoid adding any text, labels, furniture,
shadows, or perspective effects.
Spatial Description:
[SPATIAL_DESCRIPTION]
Color Legend:
Light khaki (RGB 238 232 170) → Livingroom
(large open space), Entrance, Wall-in
Orange (RGB 255 165 0) → Masterroom (private
sleeping area)
Light salmon (RGB 240 128 128) → Kitchen
(food preparation area)
Light cyan (RGB 173 216 210) → Bathroom
(toilet or shower area)
Olive green (RGB 107 142 35) → Balcony
(semi-outdoor area)
Violet (RGB 218 112 214) → Diningroom (area
for meals)
Plum (RGB 221 160 221) → Storage (small
enclosed area)
Bright yellow (RGB 255 215 0) → Commonroom
/ Secondroom / Studyroom / Childroom /
Guestroom
Black (RGB 0 0 0) → Exterior wall or
building boundary
White (RGB 255 255 255) → Front door / main
entrance / interior walls / interior doors /
external background
Output Goal:
Produce a visually clean and semantically
accurate colored floor plan image that fits
exactly within the input outline.
```

Editing. For structure-aware layout editing, we evaluate FLUX.1-Kontext-dev and Qwen-Image-Edit-2509 using the original layout and a textual edit command. The prompt template for editing is:

Prompt Template for Editing:

```
You are an expert in architectural floor plan
editing.
The input image is a color-coded floor plan
representing functional areas with flat color
blocks.
Your task is to edit the given layout based
on the following instruction,
while maintaining architectural coherence and
color consistency.
```

```
Editing Guidelines:
- Follow the edit instruction carefully
(e.g., add, remove, move, merge, resize
rooms).
- Preserve unrelated regions and overall wall
boundaries.
- Use the same color scheme for all room
types as specified in the legend.
- Maintain clean edges and closed regions
without overlapping or blending.
- Do not add any text, furniture, or
decorations.
- Ensure that all colors remain consistent
with the color legend below.
Edit Instruction:
[EDIT_INSTRUCTION]
Color Legend:
Light khaki (RGB 238 232 170) → Livingroom
(large open space), Entrance, Wall-in
Orange (RGB 255 165 0) → Masterroom (private
sleeping area)
Light salmon (RGB 240 128 128) → Kitchen
(food preparation area)
Light cyan (RGB 173 216 210) → Bathroom
(toilet or shower area)
Olive green (RGB 107 142 35) → Balcony
(semi-outdoor area)
Violet (RGB 218 112 214) → Diningroom (area
for meals)
Plum (RGB 221 160 221) → Storage (small
enclosed area)
Bright yellow (RGB 255 215 0) → Commonroom
/ Secondroom / Studyroom / Childroom /
Guestroom
Black (RGB 0 0 0) → Exterior wall or
building boundary
White (RGB 255 255 255) → Front door / main
entrance / interior walls / interior doors /
external background
Output Goal:
Return an updated floor plan image with only
the described edits applied, preserving the
rest of the layout unchanged.
```

B. Evaluation Metrics

We evaluate model performance for three tasks using unified geometric and structural metrics. Each floor plan is a 256×256 color-coded layout corresponding to an $18\times18\,\mathrm{m}$ area.

B.1. Understanding Metrics

Success Rate (Success).

$$\mathrm{Success} = \frac{N_{\mathrm{success}}}{N_{\mathrm{total}}}, \tag{B.1}$$

where N_{success} and N_{total} are the numbers of successful and total samples.

Room Match Rate (RMR).

$$\mathrm{RMR} = \frac{1}{N}\sum_{i=1}^{N} I(t_i = \hat{t}_i), \tag{B.2}$$

where N is the number of rooms, t_i and $\hat{t}_i$ are the ground-truth and predicted room types, and $I(\cdot)$ equals 1 if the condition is true and 0 otherwise.

Room Location Accuracy (LocAcc).

$$\mathrm{LocAcc} = \frac{1}{N}\sum_{i=1}^{N} I(p_i = \hat{p}_i), \tag{B.3}$$

where p_i and $\hat{p}_i$ denote the coarse positions of room i (e.g., north, south, east, west, or center).

Room Area Difference (AreaDiff).

$$\mathrm{AreaDiff} = \frac{1}{N}\sum_{i=1}^{N} |A_i - \hat{A}_i|, \tag{B.4}$$

where A_i and $\hat{A}_i$ are the ground-truth and predicted areas (in m^2).

Room Adjacency Accuracy (AdjAcc).

$$\mathrm{AdjAcc} = \frac{|E \cap \hat{E}|}{|E \cup \hat{E}|} \tag{B.5}$$

where E and $\hat{E}$ denote the ground-truth and predicted adjacency sets.

Spatial Relation Accuracy (RelAcc).

$$\mathrm{RelAcc} = \frac{1}{|R|}\sum_{(i,j)\in R} I(r_{ij} = \hat{r}_{ij}), \tag{B.6}$$

where R is the set of all adjacent ordered pairs with directional relations (e.g., left-of, above, below), and r_{ij} and $\hat{r}_{ij}$ represent their ground-truth and predicted relation types.

B.2. Generation Metrics

Micro IoU.

$$\mathrm{IoU}_{\mathrm{micro}} = \frac{\sum_{c=1}^{C} |M_c^{\mathrm{pred}} \cap M_c^{\mathrm{gt}}|}{\sum_{c=1}^{C} |M_c^{\mathrm{pred}} \cup M_c^{\mathrm{gt}}|}, \tag{B.7}$$

where M_c^{pred} and M_c^{gt} are binary masks for category c.

Macro IoU.

$$\mathrm{IoU}_{\mathrm{macro}} = \frac{1}{C}\sum_{c=1}^{C} \frac{|M_c^{\mathrm{pred}} \cap M_c^{\mathrm{gt}}|}{|M_c^{\mathrm{pred}} \cup M_c^{\mathrm{gt}}|}. \tag{B.8}$$

Macro-IoU balances room frequencies by averaging across categories.

Structural Similarity (SSIM).

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{B.9}$$

where μ_x, μ_y are mean intensities, σ_x, σ_y variances, and σ_{xy} covariance. C_1, C_2 are small constants for stability.

Peak Signal-to-Noise Ratio (PSNR).

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{L^2}{\mathrm{MSE}}\right), \tag{B.10}$$

where L is the maximum intensity (255) and MSE is the mean squared error.

Fréchet Inception Distance (FID).

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}), \tag{B.11}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the feature means and covariances of real and generated layouts.

Graph Edit Distance (GED).

$$\mathrm{GED}(G_r, G_g) = \min_{\pi} \sum_{o \in \pi} c(o), \tag{B.12}$$

where G_r and G_g are room–adjacency graphs, π is a sequence of edit operations (node/edge insertion, deletion, or substitution), and $c(o)$ is the edit cost. We use the `graph_edit_distance()` function in NetworkX.

Node F1.

$$P = \frac{|T_{\mathrm{pred}} \cap T_{\mathrm{gt}}|}{|T_{\mathrm{pred}}|}, \quad R = \frac{|T_{\mathrm{pred}} \cap T_{\mathrm{gt}}|}{|T_{\mathrm{gt}}|}, \tag{B.13}$$

$$\text{Node F1} = \frac{2PR}{P+R}, \tag{B.14}$$

where T_{pred} and T_{gt} are the sets of predicted and ground-truth room types, and P, R are precision and recall.

Edge Overlap.

$$\text{Edge Overlap} = \frac{|E_{\mathrm{pred}} \cap E_{\mathrm{gt}}|}{|E_{\mathrm{pred}} \cup E_{\mathrm{gt}}|}, \tag{B.15}$$

which measures the proportion of correctly predicted adjacency relations, reflecting structural consistency.

B.3. Editing Metrics

Edit-Detection IoU (ΔIoU).

$$M_{\mathrm{gt}}^{\Delta} = T(M_{\mathrm{after}}^{\mathrm{gt}} - M_{\mathrm{before}}), \quad M_{\mathrm{pred}}^{\Delta} = T(M_{\mathrm{after}}^{\mathrm{pred}} - M_{\mathrm{before}}), \tag{B.16}$$

$$\Delta\mathrm{IoU} = \frac{|M_{\mathrm{gt}}^{\Delta} \cap M_{\mathrm{pred}}^{\Delta}|}{|M_{\mathrm{gt}}^{\Delta} \cup M_{\mathrm{pred}}^{\Delta}|}, \tag{B.17}$$

where $T(\cdot)$ denotes thresholding of the difference map to obtain the changed region. During computation, pixels corresponding to wall-line areas are excluded to avoid non-editable regions affecting the metric.

Diff-MSE (ΔMSE).

$$\Delta\mathrm{MSE} = \frac{1}{N} \sum_{p=1}^{N} \left(M_{\mathrm{gt}}^{\Delta}(p) - M_{\mathrm{pred}}^{\Delta}(p)\right)^2, \tag{B.18}$$

where p indexes pixels and N is the total number of pixels.

Structural Consistency. GED, Node F1, and Edge Overlap are reused to measure whether the edited layout maintains correct topology.

B.4. Metric Significance

Structural metrics (GED, Node F1, Edge Overlap) and class-balanced Macro-IoU capture spatial topology and functional balance. Perceptual metrics (SSIM, PSNR, FID) assess visual realism, while edit-specific metrics (ΔIoU, ΔMSE) evaluate the controllability and precision of modifications. For floor plan tasks, structural metrics provide more meaningful insights into spatial reasoning and design consistency, while pixel-level metrics serve primarily as complementary references.

C. Post-processing and Effect Analysis

C.1. Post-processing Pipeline

To ensure clean topology, structural consistency, and standardized color encoding, all generated floor plans are refined through a unified three-stage pipeline implemented with PyTorch, OpenCV, and NumPy. This end-to-end process converts discrete token predictions into visually coherent, evaluation-ready layouts.

Floor Plan Rendering. During the generation stage, model predictions in tokenized form are converted into complete color layouts through a rendering routine implemented in PyTorch. Conditioned on the building outline, each room is decoded into a binary occupancy map and overlaid in descending order of area to form the full layout. Each functional region is assigned a base color from the predefined legend, with slight random jitter to enhance visual distinction. The resulting layout contains black outer walls, white inner separators, and solid-colored functional areas, serving as the standardized input for the subsequent structural correction and standardization steps.

Structural Correction. Ambiguous gray zones and small irregularities are first removed by replacing undefined pixels with the dominant color of neighboring regions. Morphological open–close operations are then applied within each connected component to smooth protruding edges while preserving room boundaries and overall spatial integrity.

Standardization and Contour Refinement. After correction, each pixel is reassigned to the nearest entry in the predefined color map to ensure consistent semantic encoding across samples. Contours are redrawn with black outer

walls and white inner separators to enhance visual clarity, and all layouts are resized to 256×256 pixels using nearest-neighbor interpolation to maintain discrete color values.

This three-stage pipeline is uniformly applied across all generation and editing experiments for every compared method, ensuring consistent color alignment, structural fidelity, and fair quantitative evaluation under a unified protocol, as illustrated in Fig. C.1.

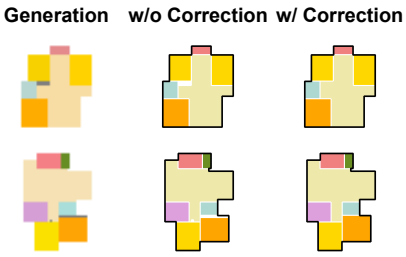

Figure C.1. Post-processing examples.

C.2. Effect of Structural Correction

To evaluate the impact of the structural correction step, we compare generation metrics before and after applying it, while keeping the standardization process fixed for both settings. As shown in Table C.1, structural correction further improves geometric consistency and perceptual clarity by smoothing irregular edges and resolving ambiguous regions near boundaries.

Table C.1. Impact of Structural Correction in **HouseMind-O**.

Metric	w/o Correction	w/ Correction
Micro IoU	0.710	0.710
Macro IoU	0.654	0.654
SSIM	0.887	0.887
PSNR	16.0	16.1
FID↓	1.87	1.89
GED↓	1.10	1.03
Node F1	0.994	0.994
Edge Overlap	0.873	0.880

The corrected results exhibit more coherent room boundaries and reduced local noise, as reflected by lower GED and higher Edge Overlap.

D. Result Analysis and Discussion

D.1. Stability Analysis

To evaluate the robustness and consistency of different models, we visualize the distribution of evaluation metrics across the entire test set using **HouseMind-O**.

In the **understanding task**, **HouseMind-O** demonstrates accurate prediction of room types, room locations, and room adjacency relations. The Room Area Difference is generally controlled within $2\,\mathrm{m}^2$, indicating stable quantitative reasoning of room size. However, the prediction of spatial relations between rooms remains relatively weaker, mainly due to minor inconsistencies in the definition of inter-room positional relationships within the training dataset.

In the **generation task**, pixel-level metrics exhibit more uniform distributions while maintaining consistently high values without significant outliers. For structural metrics, most generated floor plans achieve Node F1 $= 1$, demonstrating nearly perfect alignment between predicted and target room types. More than half of the samples also reach Edge Overlap $= 1$ and GED $= 0$, indicating precise prediction of room topological relations. Only a few cases present slight deviations, which can be further improved in subsequent training stages.

In the **editing task**, the distribution of ΔIoU shows a distinct bimodal pattern. This arises from a subset of editing prompts that do not explicitly specify the insertion position of the new room, leading to ambiguous geometric modifications. When the editing instruction clearly describes both the target room location and size, the predicted results are generally accurate and consistent with the ground truth.

D.2. Pixel-Structure Coupling Analysis

We further analyze the relationship between pixel-level accuracy (Macro IoU) and structural consistency (Edge Overlap) across three generation methods.

As shown in Fig. D.2, FloorPlanLLaMA (our re-implementation), which encodes the entire layout as a single image sequence, and ChatHouseDiffusion, which performs global image-to-layout diffusion generation, both exhibit relatively strong correlations ($r = 0.70$ and $r = 0.63$, respectively) between the two metrics. This indicates that pixel-level alignment and structural accuracy are inherently interdependent in holistic generation. As models pursue higher pixel fidelity, their topological precision also increases; however, this often comes at the cost of reduced flexibility in structural reasoning, as they overfit to visual similarity.

In contrast, **HouseMind-O**, equipped with room tokenization, shows a lower correlation ($r = 0.57$), indicating that the pixel and topological aspects are partially decoupled. Most data points lie in the upper region of the plot, where structural consistency remains high, even when pixel IoU slightly decreases. This reflects the model's priority in maintaining accurate topological relations across rooms, which better aligns with practical architectural requirements.

Overall, the reduced coupling in **HouseMind-O** demon-

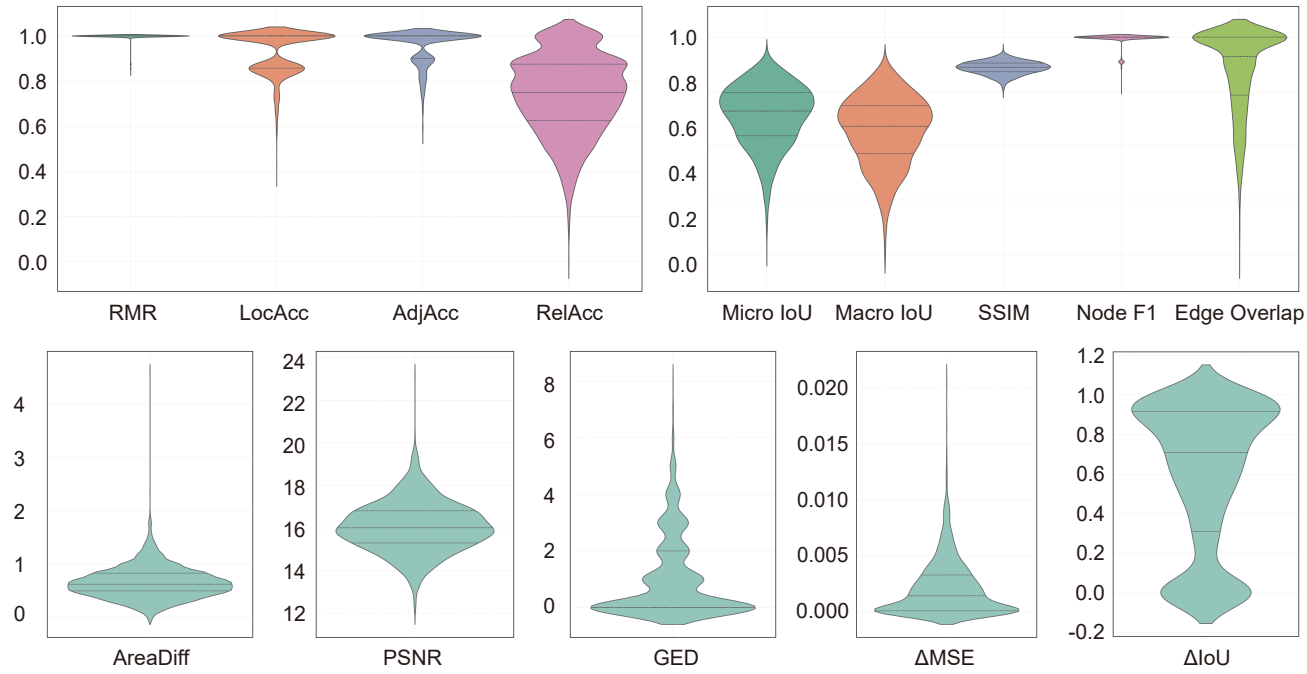

Figure D.1. **Result distribution across tasks. HouseMind-O** maintains stable, high-performance distributions in understanding and generation, with the bimodal ΔIoU in editing mainly caused by unclear spatial instructions.

strates that room-wise tokenization enables the model to reason about spatial topology independently from pixel reconstruction, achieving structure-aware generation that is both controllable and semantically consistent.

D.3. Methodological Comparison and Discussion

Beyond quantitative metrics, we compare the underlying paradigms adopted by representative model families to clarify the advantages of the proposed **HouseMind**.

General-purpose multimodal models such as GPT-5 integrate diffusion-based visual modules for open-domain image generation and editing. While these models excel in semantic reasoning and zero-shot transfer, they lack geometric and topological priors, leading to spatially invalid or unstructured layouts.

Autoregressive text-to-layout generators like FloorPlan-LLaMA encode entire floor plans as sequential token streams. This paradigm ensures high pixel fidelity and global semantic coherence but suffers from limited controllability and no editing capability, as the generation order constrains spatial reasoning.

Diffusion-based holistic generators, exemplified by ChatHouseDiffusion, produce visually realistic and smooth layouts through iterative denoising. However, their reliance on pixel-level reconstruction often causes inconsistent topology and unstable room adjacency, which undermines functional correctness.

In contrast, the proposed **HouseMind** adopts a room-tokenized multimodal reasoning paradigm, jointly modeling text, geometry, and topology within a unified LLM framework. This design enables room-wise controllability and structural preservation, achieving consistent topology even when visual pixel alignment fluctuates. Unlike holistic approaches, **HouseMind** can independently reason about spatial structure while maintaining semantic coherence across understanding, generation, and editing tasks.

Overall, this comparison highlights a paradigm shift from visually driven diffusion or autoregressive generation toward structure-aware multimodal reasoning, representing a critical step toward advancing controllable architectural layout generation.

E. Prompts for Generation Cases

The following prompts correspond to the generation results illustrated in Fig. 4. Each prompt describes the intended spatial relationships and functional layout used as textual input for the generation task.

Prompt G1

```
The layout centers around a spacious Living
Room, occupying the central area of the floor
plan with an area of 26 square meters.
To the west of the Living Room lies the
Master Room, which connects directly to it
and also extends slightly above the Balcony
```

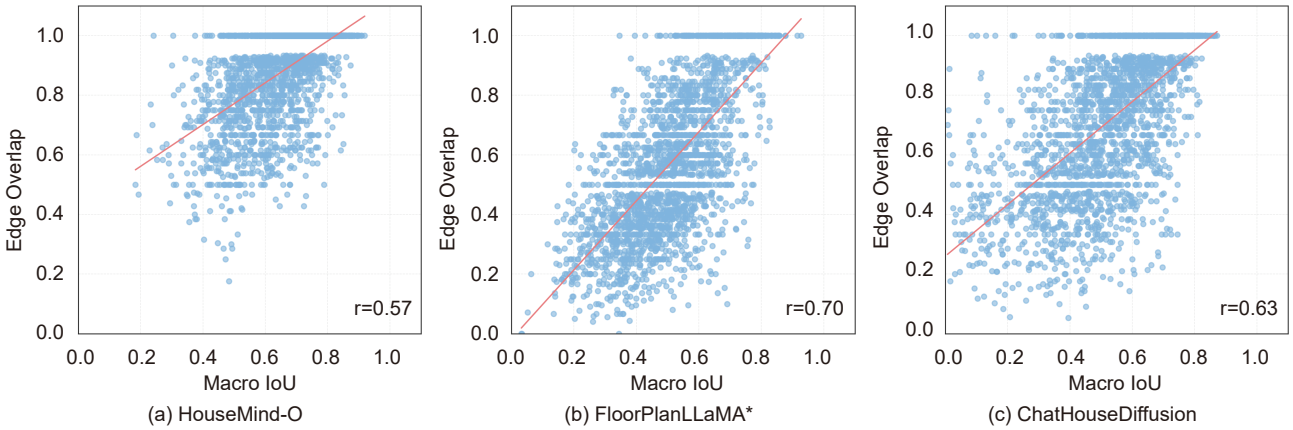

Figure D.2. **Pixel–structure coupling analysis.** Scatter plots show the correlation between Macro IoU and Edge Overlap for three generation paradigms.

situated to the south.
The Kitchen, positioned in the northwest corner, is smaller in size at 6 square meters and connects to both the Living Room on its right and the Bathroom directly below it.
The Bathroom, located just northwest of center and slightly west of the Living Room, is adjacent to both the Kitchen above and the Master Room below, forming a vertical sequence of rooms along the western side.
To the northeast, the Second Room sits comfortably next to the Living Room, sharing a direct right-of relationship with it.
Finally, the Balcony, positioned directly south of the Living Room, spans the southern edge of the plan and lies below both the Living Room and the Master Room, which projects slightly over it from the northwest.
All rooms are efficiently arranged, with clear adjacency and vertical alignment enhancing the flow between functional zones.

Prompt G2

Create a house layout that reflects the described spatial logic. The layout centers on a spacious LivingRoom, which is flanked to the left by the SecondRoom and to the right by the MasterRoom. Above the LivingRoom, toward the northeast, sit the Kitchen and Bathroom side by side, with the Kitchen positioned to the left of and slightly above the Bathroom. The Bathroom lies directly above the MasterRoom, while the Kitchen extends slightly northeastward, with a narrow Balcony positioned to its right and above the MasterRoom. A larger Balcony stretches along the southern edge of the LivingRoom, connecting to both the MasterRoom and Kitchen via diagonal relationships. The overall arrangement creates a functional, tiered flow from the central living area to private and service spaces.

Prompt G3

The layout centers on a spacious Living Room, occupying 36 square meters and positioned at the heart of the floor plan.
To the north, the Kitchen (4 sq. meters) sits directly above the Living Room and is connected to a smaller Balcony (2 sq. meters) that extends above it, with the Kitchen positioned just below this northern balcony.
To the northwest, the Second Room (10 sq. meters) lies above the Living Room and to the left of the Kitchen, with a diagonal connection beneath it to the northern balcony.
Directly south of the Living Room is the Master Room (8 sq. meters), which also connects to a larger Balcony (6 sq. meters) situated below it.
To the east of the Living Room lies the Bathroom (8 sq. meters), positioned to its right, with the Master Room lying diagonally below and to the left of it.
The two balconies|smaller to the north and larger to the south|frame the upper and lower edges of the layout, with the northern balcony adjacent to both the Kitchen and Second Room, and the southern balcony extending beneath the Master Room and aligned with the Living Room's southern edge.
All rooms are arranged in a cohesive, vertically and horizontally connected sequence, with the Living Room serving as the central hub linking all other spaces.

F. Generalization and Real-World Deployment

This section provides a more detailed analysis of the generalization ability and robustness of **HouseMind** beyond the standard in-distribution evaluation.

F.1. Modeling Design for Generalization

From a modeling perspective, **HouseMind** does not directly learn raw room distributions in pixel or coordinate space. Instead, spatial layouts are first discretized through a VQ-VAE encoder, which transforms continuous geometric configurations into structured spatial tokens. This representation abstracts away low-level coordinate noise and encourages the model to learn higher-level spatial relationships (e.g., adjacency, functional grouping, alignment patterns).

By operating in this tokenized latent space, the LLM captures relational and compositional regularities rather than memorizing specific layout instances. This representation-driven learning mechanism enables the model to generalize beyond layouts that strictly follow the empirical training distribution.

F.2. Out-of-Distribution Evaluation

To further evaluate generalization, we construct a deliberately more challenging out-of-distribution (OOD) test set for generation. Compared with the standard test split, this OOD set includes:

- Studio layouts (no separate bedroom);
- Layouts with two living rooms;
- Layouts including uncommon room types (e.g., wine cellar treated as storage);
- Logically invalid configurations (e.g., centrally located balcony);
- Irregular building outlines;
- Prompts without explicitly specified room types.

Representative examples are shown in Fig. F.1.

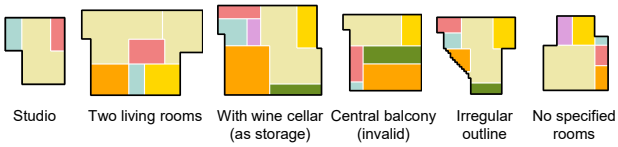

Figure F.1. Results of **HouseMind** on out-of-distribution cases.

Results indicate that **HouseMind** maintains structurally consistent generation across most uncommon layouts and room-type configurations. Even when room combinations deviate from typical training patterns, the model preserves global structural coherence and largely respects adjacency as well as functional grouping constraints. In addition, prompt paraphrasing does not significantly degrade structural validity, suggesting robustness to linguistic variations. Failure cases primarily arise under highly complex or mutually conflicting constraints. In such scenarios, the model tends to favor layouts that align with realistic architectural priors rather than strictly following logically inconsistent or physically implausible instructions (e.g., placing a balcony at the geometric center of the building). This behavior indicates that the model internalizes real-world spatial distributions and architectural regularities, which function as an implicit prior during generation.

F.3. Robustness Enhancement in Deployment

To further enhance robustness in real-world deployment, we implement several additional strategies in our public platform, available at https://housemind.ai-structure.com/:

1. **Structured prompt guidance**: predefined room type selections and prompt templates are provided to reduce ambiguity and improve the clarity of user inputs.
2. **Intermediate prompt reformulation**: an auxiliary LLM reformulates raw user instructions into structured representations that are better aligned with **HouseMind**'s token space.
3. **Post-generation validation**: automatic validation of room types, counts, and adjacency relations is performed, with regeneration triggered when inconsistency with the input prompt is detected.

These mechanisms collectively improve robustness under noisy, incomplete, or partially inconsistent user inputs, while preserving structural validity and generative diversity.