Ray-Voxel Transformer for Structured 3D Reconstruction

Kunal Chelani
Ericsson Research

Abstract

We present a multi-stage method for structured 3D reconstruction of building wireframes from multi-view imagery, developed for the S23DR 2026 challenge. Our core idea is to produce highly accurate 3D vertex predictions and use these to guide a learned baseline toward a more precise and complete wireframe. The pipeline consists of three stages: (1) a 2D corner predictor that detects candidate vertex locations in each view and lifts them to 3D rays, (2) a RayVoxelTransformer that processes a 6-channel voxel volume encoding ray statistics and structure-from-motion occupancy through dual dense and sparse pathways with a learned merge gate, and (3) a hybrid non-maximum suppression scheme that extracts precise vertex positions. The predicted vertices are snapped onto the organizers' learned baseline wireframe via a geometric fusion strategy to produce the final reconstruction. We describe the design of each component along with the loss functions used for training and evaluation.

1. Introduction

Recovering the 3D structure of buildings from images is a fundamental problem in computer vision with applications in urban planning, architectural modeling, and autonomous navigation. While dense 3D reconstruction methods [5, 3] have achieved impressive results for general scenes, the *structured* reconstruction of buildings—predicting a compact wireframe of vertices and edges that captures the geometric skeleton—remains challenging due to the combinatorial nature of the output and the need for precise localization of structural keypoints.

The Structured 3D Reconstruction (S23DR) challenge [10] provides a benchmark for this task, requiring participants to predict 3D wireframes of building rooftops from multi-view observations. The 2026 edition of the challenge uses the hoho22k_2026 dataset, which provides for each scene: multi-view RGB images, metric depth maps, semantic segmentation (ADE20k [9]), gestalt structural annotations identifying roof components (apex, ridge, eave, etc.), and COLMAP [6] sparse reconstructions with calibrated camera poses. The primary evaluation metric is the Hausdorff-based Structural Similarity (HSS), which measures the geometric fidelity of predicted wireframes against ground truth, supplemented by Vertex F1 scores at 0.5m and 1.0m thresholds.

A key difficulty of this task is bridging the gap between 2D image evidence and 3D structural output. Individual views provide strong cues about vertex locations through structural edges and junctions, but these must be aggregated across views and resolved in 3D space. Classical structure-from-motion provides sparse 3D points but lacks semantic understanding of which points correspond to structural vertices. Our method addresses this by combining learned 2D detection with a 3D voxel representation that aggregates multi-view ray information through a transformer architecture.

Our pipeline operates in three stages. First, a lightweight CNN predicts per-view vertex heatmaps, from which candidate rays are extracted via non-maximum suppression and back-projected into 3D. Second, these rays are accumulated into a voxel grid along with structure-from-motion occupancy, forming a 6-channel volume that is processed by our RayVoxelTransformer—a dual-pathway architecture combining dense image feature unprojection with sparse ray-guided attention. Third, a hybrid NMS scheme extracts vertex predictions which are fused with the organizers' learned baseline via geometric snapping.

The core idea of our approach is to produce highly accurate 3D vertex predictions and use these to guide the learned baseline toward a more accurate and complete wireframe. Rather than attempting to predict both vertices and edges from scratch, we focus our architectural innovations on precise vertex localization and leverage the baseline's edge connectivity, snapping its vertices toward our more accurate predictions.

2. Motivation and Observations

The organizers' learned baseline predicts wireframe edges as directed segments from a midpoint, parameterized by a direction vector and length. While this produces reasonably realistic wireframe topologies, we observed that vertex accuracy degrades significantly for longer edges: a

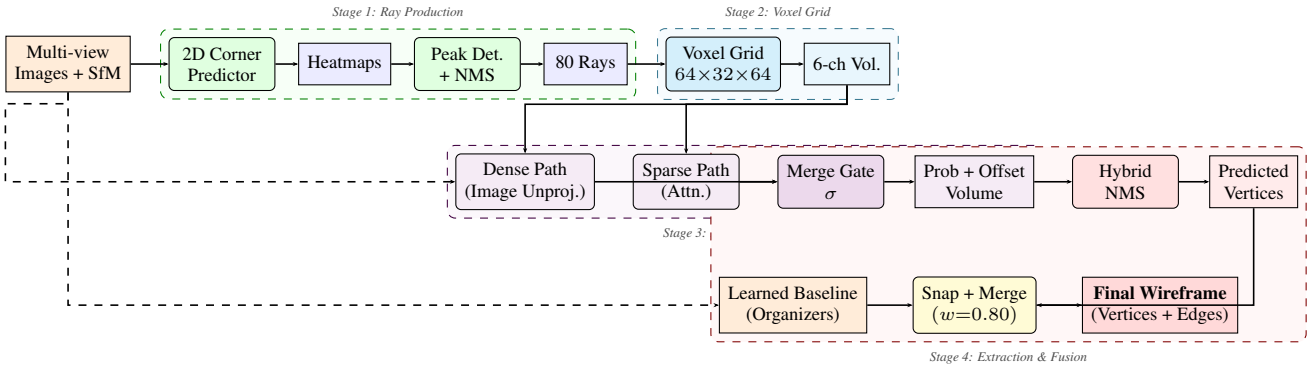

Figure 1. Overview of our method. Multi-view inputs are processed by a 2D corner predictor to produce rays (Stage 1), which are accumulated into a 6-channel voxel grid (Stage 2). The RayVoxelTransformer combines dense and sparse pathways via a learned merge gate (Stage 3). Hybrid NMS extracts vertices that are snapped onto the learned baseline to produce the final wireframe (Stage 4). Dashed arrows indicate auxiliary data flow.

small angular error in the predicted direction is amplified by the edge length, causing endpoint positions to drift far from their true locations. For an edge of length l with angular error θ, the endpoint displacement grows as $l \sin \theta$, meaning that even modest directional inaccuracies of a few degrees can produce vertex errors exceeding 1m on long roof ridges or eaves.

A second limitation is recall. The baseline operates on a fused point cloud derived from COLMAP and depth maps, which may lack coverage in textureless or occluded regions. Vertices in such areas have weak point cloud support and are frequently missed entirely.

These observations motivated our approach: rather than improving the baseline's segment prediction directly, we focus on independently estimating accurate vertex positions with high recall. By detecting structural corners in 2D views—where they appear as clear intensity junctions regardless of 3D occlusion—and triangulating them through a learned voxel representation, we can localize vertices that are poorly supported by the point cloud. The snapping fusion then corrects the baseline's endpoint drift by pulling vertices toward our more accurate predictions, while appending unmatched vertices to increase coverage. This division of labor allows each component to focus on its strength: the baseline provides plausible edge connectivity, while our method provides precise and complete vertex localization.

3. Method

3.1. Overview

Our method predicts the 3D wireframe vertices of a building from multi-view imagery through a pipeline of four sequential stages, illustrated conceptually below:

1. **Ray Production**: A 2D corner predictor CNN predicts vertex likelihood heatmaps in each view. Peaks are detected, refined to sub-pixel precision, and back-projected as 3D rays using calibrated camera parameters.

2. **Voxel Grid Construction**: A $64 \times 32 \times 64$ voxel grid is placed around the scene using SfM points. Rays are marched through the grid, accumulating per-voxel statistics (ray count, mean direction, mean confidence). SfM occupancy is splatted separately.

3. **RayVoxelTransformer**: The 6-channel voxel volume is processed by a dual-pathway architecture: a dense path unprojecting image features into voxels, and a sparse path applying transformer attention to active voxel tokens and SfM point tokens. A learned gate merges both pathways to predict per-voxel vertex probability and sub-voxel offset.

4. **Vertex Extraction and Fusion**: Hybrid NMS combines ray-guided and dense peak extraction to produce vertex candidates. These are used to guide the organizers' learned baseline wireframe via geometric snapping, producing an accurate and complete final wireframe.

3.2. Ray Production

The first stage identifies candidate vertex locations in 2D and lifts them to 3D rays that guide downstream processing.

2D Corner Predictor. For each view, we construct a 7-channel input by concatenating the gestalt segmentation (3 channels), metric depth (1 channel), and ADE20k semantic confidence (3 channels). A lightweight CNN with dilated convolutions processes this input at resolution 576×768 and produces a single-channel heatmap $H \in [0, 1]^{576 \times 768}$ indicating the likelihood of a structural vertex at each pixel.

Peak Detection. Local maxima in H are identified using a threshold $\tau_{\text{peak}} = 0.30$. Each detected peak is refined to sub-pixel precision by computing the center of mass within a 7×7 window around the maximum:

$$\hat{\mathbf{p}} = \frac{\sum_{(i,j)\in\mathcal{N}} (i,j) \cdot H(i,j)}{\sum_{(i,j)\in\mathcal{N}} H(i,j)} \tag{1}$$

where $\mathcal{N}$ denotes the 7×7 neighborhood.

Spatial NMS. Cross-view spatial NMS suppresses redundant detections: peaks within 20 pixels of a higher-confidence peak in the same view are removed. The top $N_{\text{rays}} = 80$ peaks are retained globally across all views.

Ray Construction. Each retained peak at pixel coordinates $(\hat{u}, \hat{v})$ with confidence c is unprojected to a 3D ray. Given camera intrinsics K and extrinsics $(R, \mathbf{t})$ mapping world to camera coordinates:

$$\mathbf{d} = R^\top K^{-1} \begin{bmatrix} \hat{u} \\ \hat{v} \\ 1 \end{bmatrix}, \quad \mathbf{o} = -R^\top \mathbf{t} \tag{2}$$

The ray is parameterized as $\mathbf{r}(t) = \mathbf{o} + t\hat{\mathbf{d}}$ where $\hat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|$. Each ray carries its associated confidence score c from the heatmap.

The 2D corner predictor is trained with a per-pixel binary cross-entropy loss against Gaussian-blurred ground truth vertex projections:

$$\mathcal{L}_{\text{corner}} = -\frac{1}{|\Omega|} \sum_{\mathbf{p}\in\Omega} \big[y_{\mathbf{p}} \log H(\mathbf{p}) + (1 - y_{\mathbf{p}}) \log(1 - H(\mathbf{p})) \big] \tag{3}$$

where $y_{\mathbf{p}}$ is the Gaussian-smoothed ground truth at pixel $\mathbf{p}$.

3.3. Voxel Grid Construction

The second stage constructs a structured 3D representation that encodes the spatial distribution of rays and SfM evidence.

Grid Placement. An axis-aligned voxel grid of resolution $R_x \times R_y \times R_z = 64 \times 32 \times 64$ is placed around the scene. The anisotropic resolution (shorter in Y) reflects the typical aspect ratio of building rooftops. The grid extent is determined from the SfM point cloud: points are trimmed to the 5th–95th percentile per axis to remove outliers, then 40% padding is added. The Y-axis origin is centered at the 80th percentile of point heights, placing the ground-roof boundary at a predictable grid location.

Ray Marching. Each ray $\mathbf{r}(t) = \mathbf{o} + t\hat{\mathbf{d}}$ is intersected with the grid's axis-aligned bounding box (AABB) to find entry and exit parameters $t_{\min}, t_{\max}$. We sample $S = 128$ points uniformly along the intersection:

$$\mathbf{x}_s = \mathbf{o} + \Big(t_{\min} + \tfrac{s}{S-1} (t_{\max} - t_{\min}) \Big) \hat{\mathbf{d}} \tag{4}$$

for $s = 0, \ldots, S-1$. Each sample point maps to a voxel index $(i, j, k) = \lfloor (\mathbf{x}_s - \mathbf{g}_{\text{origin}})/v \rfloor$ where v is the voxel size. After deduplication (each ray contributes at most once per voxel), we accumulate:

$$n_{ijk} \mathrel{+}= 1 \qquad \text{(ray count)} \tag{5}$$

$$\mathbf{D}_{ijk} \mathrel{+}= \hat{\mathbf{d}} \qquad \text{(direction sum)} \tag{6}$$

$$C_{ijk} \mathrel{+}= c \qquad \text{(confidence sum)} \tag{7}$$

SfM Splatting. Each SfM point $\mathbf{q}_m$ is mapped to its nearest voxel, incrementing an occupancy counter O_{ijk}.

Volume Assembly. The final 6-channel volume $V \in \mathbb{R}^{6 \times R_x \times R_y \times R_z}$ is assembled as:

$$V_{ijk} = \Big[\log(1 + n_{ijk}),\ \frac{\mathbf{D}_{ijk}}{\max(n_{ijk}, 1)},\ \frac{C_{ijk}}{\max(n_{ijk}, 1)},\ \log(1 + O_{ijk}) \Big] \tag{8}$$

where the direction and confidence channels are normalized by ray count to produce mean values.

3.4. RayVoxelTransformer

The core of our method is a dual-pathway transformer architecture that combines dense image-based features with sparse ray-guided attention.

3.4.1 Dense Pathway

Image Encoder. A stride-4 CNN encoder processes each 7-channel input image (rescaled to 288×384) into a 64-dimensional feature map. The encoder consists of two downsampling stages with GELU activations.

Dense Unprojection. For each voxel center $\mathbf{g}_{ijk}$ in world coordinates, we project into each camera view:

$$\mathbf{p}_{\text{cam}} = R\mathbf{g}_{ijk} + \mathbf{t}, \quad [u, v]^\top = \frac{1}{z_{\text{cam}}} K \mathbf{p}_{\text{cam}} \tag{9}$$

Image features at (u, v) are sampled via bilinear interpolation and averaged across visible views (where $z_{\text{cam}} > 0$), producing a feature volume $F_{\text{dense}} \in \mathbb{R}^{64 \times R_x \times R_y \times R_z}$ and a visibility count volume.

3D Convolutions. The concatenated features $[F_{\text{dense}}; \text{visibility}] \in \mathbb{R}^{65\times R_x\times R_y\times R_z}$ are processed by two 3D convolutional blocks (each containing two Conv3d layers with GELU activation) to produce dense probability and offset predictions.

3.4.2 Sparse Pathway

Voxel Tokenization. Active voxels (where $n_{ijk} > 0$) are extracted and tokenized. Each token combines ray features (6 channels) with dense image features (64 channels), projected via a two-layer MLP to dimension D. Sinusoidal positional encodings based on voxel coordinates are added. If more than 2048 active voxels exist, we retain those with the highest ray confidence scores.

SfM Point Tokens. Up to 1024 cleaned SfM points are encoded by a PointTransformer [8] encoder with $k = 16$ nearest-neighbor self-attention (2 blocks), producing D-dimensional tokens that capture local geometric context.

Joint Transformer. Voxel tokens (N_v tokens) and SfM tokens (N_s tokens) are concatenated and processed by 4 self-attention blocks [7], each consisting of pre-norm LayerNorm, multi-head attention, and a feed-forward network with hidden dimension $4D$. The first N_v output tokens yield sparse probability and offset predictions at active voxel locations.

3.4.3 Merge Gate

A learned sigmoid gate adaptively combines the dense and sparse pathway outputs:

$$\hat{p}_{ijk} = \sigma(g_{ijk}) \cdot p_{ijk}^{\text{sparse}} + (1 - \sigma(g_{ijk})) \cdot p_{ijk}^{\text{dense}} \tag{10}$$

$$\hat{\boldsymbol{\delta}}_{ijk} = \sigma(g_{ijk}) \cdot \boldsymbol{\delta}_{ijk}^{\text{sparse}} + (1 - \sigma(g_{ijk})) \cdot \boldsymbol{\delta}_{ijk}^{\text{dense}} \tag{11}$$

where g_{ijk} is computed from the concatenation of ray features, image features, and visibility count via a linear layer. This allows the network to rely on the image-based dense path in regions with good multi-view coverage, while trusting the ray-guided sparse path where strong heatmap detections provide precise localization.

Training Loss. The RayVoxelTransformer is trained with a combination of binary cross-entropy on the per-voxel probability and L_1 loss on the sub-voxel offset:

$$\mathcal{L}_{\text{V9}} = \text{BCE}(\hat{p}_{ijk}, y_{ijk}) + \lambda \cdot \mathbf{1}[y_{ijk} = 1] \cdot \|\hat{\boldsymbol{\delta}}_{ijk} - \boldsymbol{\delta}_{ijk}^{*}\|_1 \tag{12}$$

where $y_{ijk} \in \{0, 1\}$ indicates whether a ground truth vertex falls within voxel (i, j, k), and $\boldsymbol{\delta}_{ijk}^{*}$ is the fractional offset of the vertex within that voxel.

3.5. Hybrid NMS and Vertex Extraction

The predicted probability and offset volumes are decoded into a set of 3D vertex candidates through a two-stage hybrid NMS procedure.

Ray-Based Peaks. For each input ray, we sample 64 points along its AABB intersection and identify the voxel with maximum predicted probability. If this probability exceeds $\tau_{\text{ray}} = 0.35$, a vertex candidate is placed at:

$$\hat{\mathbf{v}} = (\mathbf{i} + 0.5 + \hat{\boldsymbol{\delta}}_{\mathbf{i}}) \cdot v + \mathbf{g}_{\text{origin}} \tag{13}$$

Candidates within radius $r_{\text{ray}} = 0.3$m of a higher-confidence candidate are suppressed. Up to 48 ray-based vertices are retained.

Dense Peaks. All voxels with probability exceeding $\tau_{\text{dense}} = 0.45$ are considered. The top 128 by probability are converted to world coordinates using the same offset formula. Dense peaks within radius $r_{\text{add}} = 0.5$m of any existing (ray-based) prediction are suppressed; remaining peaks are appended up to a total of 48 vertices.

3.6. Fusion with Learned Baseline

We use the organizers' learned baseline [4] as a source of edge connectivity and global wireframe structure. The baseline takes fused COLMAP and depth-derived point clouds as input and predicts wireframe segments (vertex pairs with edges) using a Perceiver-based architecture. Our contribution is not in the baseline itself, but in how we use our accurate vertex predictions to improve its output through geometric snapping.

Snapping Strategy. For each baseline vertex $\mathbf{v}_{\text{baseline}}$, we find the nearest vertex $\mathbf{v}_{\text{ours}}$ from our RayVoxelTransformer predictions. If the distance falls within a snapping radius $r_{\text{snap}} = 2.0$m, the baseline vertex is displaced toward our prediction:

$$\hat{\mathbf{v}}_{\text{fused}} = (1 - w) \cdot \mathbf{v}_{\text{baseline}} + w \cdot \mathbf{v}_{\text{ours}} \tag{14}$$

with snapping weight $w = 0.80$, reflecting high confidence in our vertex localization. Baseline vertices with no nearby match are left unchanged, preserving the baseline's coverage of vertices our model may have missed.

Appending Unmatched Vertices. Our predicted vertices that are not matched to any baseline vertex (i.e., farther than r_{snap} from all baseline vertices) are appended to the final wireframe as additional vertices. This ensures that accurate detections from our model are not lost when the baseline fails to predict a corresponding vertex.

The resulting fused wireframe inherits edge connectivity from the baseline while benefiting from the superior vertex localization of our ray-voxel approach.

3.7. Loss Functions

Beyond the per-component losses described above, we evaluate wireframe quality using geometric distance metrics.

Varifold Loss. Following [1], we define a multi-scale varifold distance between predicted and ground truth wireframes. For two sets of oriented line segments $\mathcal{P}$ and $\mathcal{Q}$, the varifold kernel evaluates geometric proximity and directional alignment:

$$k_\sigma(\mathbf{e}_i, \mathbf{e}_j) = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{2\sigma^2}\right) \cdot |\langle \hat{\mathbf{t}}_i, \hat{\mathbf{t}}_j \rangle| \qquad (15)$$

where $\mathbf{c}_i$ is the segment midpoint and $\hat{\mathbf{t}}_i$ is the unit tangent. The multi-scale loss combines evaluations at $\sigma \in \{0.5, 1.0, 2.0\}$m with weights $\alpha \in \{0.2, 0.6, 0.2\}$:

$$\mathcal{L}_{\text{varifold}} = \sum_s \alpha_s \left[\mu_\sigma(\mathcal{P}, \mathcal{P}) - 2\mu_\sigma(\mathcal{P}, \mathcal{Q}) + \mu_\sigma(\mathcal{Q}, \mathcal{Q})\right] \qquad (16)$$

where $\mu_\sigma(\mathcal{A}, \mathcal{B}) = \sum_{i,j} \ell_i \ell_j \, k_\sigma(\mathbf{e}_i, \mathbf{e}_j)$ and ℓ_i denotes segment length (weighted with exponent 1.0).

Sinkhorn Loss. As a regularizer for segment matching, we employ an optimal transport distance [2] between predicted and ground truth edge midpoints with entropic regularization $\varepsilon = 0.05$, computed via 10 Sinkhorn iterations. A dustbin cost of 0.1 handles unmatched segments.

4. Results

We evaluate our method on the hoho22k_2026_trainval validation set using the challenge metrics: mean and median HSS (Hausdorff-based Structural Similarity), and Vertex F1 at 0.5m and 1.0m thresholds.

4.1. Main Results

Table 1 compares the organizers' learned baseline alone against our full pipeline with snapping fusion on 100 validation scenes.

	HSS		Vertex F1	
Method	Mean	Med.	@0.5m	@1.0m
Baseline	0.352	0.369	—	—
+ Snap	**0.411**	**0.453**	0.494	0.685

Table 1. Validation results (100 scenes). Snapping improves the baseline by +0.059 mean HSS, winning on 85/100 scenes.

The snapping fusion yields an absolute improvement of +0.059 in mean HSS and +0.084 in median HSS, demonstrating that our RayVoxelTransformer produces vertex predictions that are substantially more accurate than those of the baseline. The fused method wins on 85 out of 100 validation scenes, indicating consistent improvement across diverse building geometries.

4.2. Snap Contribution Analysis

To understand how snapping contributes across scenes of varying difficulty, we evaluate on an extended set of 321 scenes using weight $w = 0.80$, radius $r = 2.0$m, and vertex-F1 threshold 0.5m. Scenes are bucketed by baseline IoU. Table 2 reports the change in IoU, HSS-F1, and vertex-F1 due to snapping.

Across all 321 scenes, snapping improves IoU in 236 cases (73.5%) with a mean gain of $+0.0379$, and improves HSS-F1 in 190 cases (59.2%) with a mean gain of $+0.0678$. The improvement is strongest in the mid-difficulty bucket ($[0.25, 0.50)$), where the baseline produces reasonable but imprecise wireframes that benefit most from vertex correction: mean ΔIoU of $+0.0498$ with a 79/21 win/loss ratio and mean ΔHSS-F1 of $+0.0991$ with a 72/15 win/loss ratio.

For scenes where the baseline already performs well ($[0.75, 1.00)$), snapping still yields positive IoU gains ($+0.0275$, 16 wins vs. 5 losses) but slightly degrades HSS-F1 (-0.0212, 3 wins vs. 6 losses). This indicates that displacing already-accurate vertices can occasionally break topological consistency evaluated by the HSS metric.

The vertex-F1 contribution from appending unmatched vertices is more modest: an overall mean gain of $+0.0112$ with 147 wins vs. 115 losses. This metric measures the recall benefit of vertices our model predicts that the baseline missed. The near-even win/loss ratio in the $[0.50, 0.75)$ bucket suggests that for moderately well-performing baselines, our appended vertices sometimes introduce false positives that offset the recall improvement.

4.3. Snapping Hyperparameter Sweep

We swept the snapping weight w over the range $[0.6, 0.9]$ on an extended validation set of 170 scenes. Table 3 reports the effect of the snapping weight on mean HSS.

The optimal weight of $w = 0.80$ indicates that the baseline benefits from strongly displacing its vertices toward our predictions. This confirms that the RayVoxelTransformer provides more geometrically accurate vertex localization, while the baseline contributes valuable edge connectivity that should not be discarded entirely.

4.4. NMS Parameters

The hybrid NMS parameters were selected via grid search: ray probability threshold $\tau_{\text{ray}} = 0.35$, ray sup-

Table 2. Snap contribution analysis across baseline IoU buckets (321 scenes, $w = 0.80$, $r = 2.0$m). Δ denotes the metric change from snapping. Win/Loss counts the number of scenes improved vs. degraded.

		Δ IoU					
Baseline IoU	N	Mean	Max	Min	Std	Wins	Losses
$[0.00, 0.25)$	100	$+0.0304$	$+0.2153$	-0.0833	0.0501	69	26
$[0.25, 0.50)$	100	$+0.0498$	$+0.2088$	-0.1859	0.0723	79	21
$[0.50, 0.75)$	100	$+0.0357$	$+0.1526$	-0.1412	0.0617	72	28
$[0.75, 1.00)$	21	$+0.0275$	$+0.1448$	-0.0439	0.0447	16	5
Overall	**321**	$\mathbf{+0.0379}$	$+0.2153$	-0.1859	0.0616	**236**	**80**

		Δ HSS-F1					
Baseline IoU	N	Mean	Max	Min	Std	Wins	Losses
$[0.00, 0.25)$	100	$+0.0601$	$+0.3684$	-0.3158	0.1169	54	17
$[0.25, 0.50)$	100	$+0.0991$	$+0.4400$	-0.1667	0.1154	72	15
$[0.50, 0.75)$	100	$+0.0629$	$+0.3871$	-0.2500	0.1077	61	18
$[0.75, 1.00)$	21	-0.0212	$+0.0909$	-0.1667	0.0642	3	6
Overall	**321**	$\mathbf{+0.0678}$	$+0.4400$	-0.3158	0.1146	**190**	**56**

		Δ Vertex-F1 (unmatched vertices)					
Baseline IoU	N	Mean	Max	Min	Std	Wins	Losses
$[0.00, 0.25)$	100	$+0.0193$	$+0.1421$	-0.0606	0.0395	44	30
$[0.25, 0.50)$	100	$+0.0185$	$+0.2126$	-0.0959	0.0566	55	34
$[0.50, 0.75)$	100	-0.0006	$+0.1342$	-0.1833	0.0573	43	43
$[0.75, 1.00)$	21	-0.0059	$+0.1000$	-0.1019	0.0439	5	8
Overall	**321**	$\mathbf{+0.0112}$	$+0.2126$	-0.1833	0.0523	**147**	**115**

Snap Weight w	Mean HSS
0.60	0.406
0.70	0.410
0.80	**0.415**
0.90	0.413

Table 3. Snap weight sweep (170 val scenes, radius $r = 2.0$m). Higher weight (stronger pull toward our vertices) improves HSS up to $w = 0.80$.

pression radius $r_{\text{ray}} = 0.3$m, dense probability threshold $\tau_{\text{dense}} = 0.45$, dense addition radius $r_{\text{add}} = 0.5$m, and maximum vertex count of 48. The two-stage design (ray-guided followed by dense fill-in) ensures that high-confidence detections along rays are prioritized while lower-confidence but spatially distinct vertices are still recovered.

5. Conclusion and Future Work

We presented a ray-voxel transformer approach for structured 3D reconstruction that focuses on accurate vertex prediction to guide the learned baseline wireframe. Through a simple observation—that independently localized vertices can correct the baseline's endpoint drift and increase recall—our method achieves a significant improvement of +0.059 mean HSS over the baseline alone.

However, the current vertex prediction pipeline is limited by several factors. First, the 2D corner predictor lacks occlusion awareness: it detects peaks independently per view without reasoning about which vertices are actually visible, leading to spurious rays from occluded junctions. Second, the fixed $64\times32\times64$ voxel grid imposes a resolution ceiling on localization accuracy and struggles with buildings that deviate from typical aspect ratios. Third, the handcrafted hybrid NMS with manually tuned thresholds is brittle and cannot adapt to varying scene densities. An end-to-end approach for vertex set prediction—directly regressing an unordered set of 3D vertices from multi-view features without intermediate discretization or heuristic post-processing—would overcome these limitations.

Additionally, the fusion with the baseline is currently limited to geometric snapping, which is suboptimal: it can only displace existing vertices and append new ones, but cannot modify edge connectivity or remove spurious edges. A learned refinement stage that takes the baseline wireframe and our predicted vertices as joint input—for example, a

graph neural network that re-predicts vertex positions and edge existence conditioned on both sources—should be able to further lift performance significantly by enabling topological corrections alongside geometric ones.

Finally, we note that the training data contains several scenes with misaligned ground truth annotations, which inject noise into the supervision signal. Adopting outlier-robust loss functions—such as truncated or Huber-style losses that downweight high-residual samples—would reduce the influence of these corrupted scenes and allow training to focus on improving accuracy where the annotations are reliable.

References

[1] N. Charon and A. Trouvé. The varifold representation of nonoriented shapes for diffeomorphic registration. *SIAM Journal on Imaging Sciences*, 6(4):2547–2580, 2013.

[2] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. a. Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, 2021.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[6] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[8] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *International Conference on Computer Vision (ICCV)*, 2021.

[9] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *International Journal of Computer Vision*, 127:302–321, 2019.

[10] S. Zhou et al. Holistic 3d reconstruction challenge. In *ECCV Workshop on Structured 3D Reconstruction*, 2022.