Edge Prediction for Roof Wireframe Reconstruction with Transformers

Gustav Hanning*, Ludvig Dillén*, Jonathan Astermark*, Johanna Lidholm*, Viktor Larsson
Centre for Mathematical Sciences, Lund University
{gustav.hanning,ludvig.dillen,jonathan.astermark,johanna.lidholm,viktor.larsson}@math.lth.se

Abstract

This paper presents a competitive solution to the S23DR Challenge 2026, which aims to reconstruct 3D house roof wireframe models from sparse SfM point clouds and ground-level semantic segmentations and depth maps. Our proposed method utilizes an end-to-end Transformer encoder-decoder architecture inspired by DETR. To effectively process the geometric and semantic data, the sparse SfM point cloud input is dynamically subsampled based on semantic priority and augmented with Gestalt and ADE20k class features. To further increase segmentation context, we fuse the point features with additional Gestalt feature encodings which are obtained by projecting the points into latent feature maps produced by a frozen autoencoder. Learned query embeddings are then decoded directly into 3D wireframe edges via cross-attention mechanisms. Evaluated on the "HoHo 22k" dataset, our approach significantly outperforms both handcrafted and learned baselines, achieving a Hybrid Structure Score (HSS) of 0.6476 and securing the second-highest position on the challenge's private leaderboard.

1. Introduction

We present our solution to the Structured Semantic 3D Reconstruction (S23DR) Challenge 2026 [1]. The goal of the challenge is to reconstruct house roof wireframe models given sparse Structure-from-Motion (SfM) point clouds and ground-level Gestalt and ADE20k [8, 22] semantic segmentation images and depth maps.

Our method is based on a simple Transformer [19] architecture inspired by DETR [3]. We train the network end-to-end to directly predict wireframe edges from the input SfM point cloud, which is augmented with features from the segmentation images and subsampled to include the most relevant points. The point cloud is passed through our encoder-decoder model, where a set of learned queries are decoded into edges. We additionally fuse the latent point features with multi-view informed Gestalt feature vectors, encoding local segmentation details, which are obtained by projecting the points into feature maps generated by a frozen autoencoder.

The solution achieves the second-highest score on the private leaderboard. An overview of our method is shown in Fig. 3.

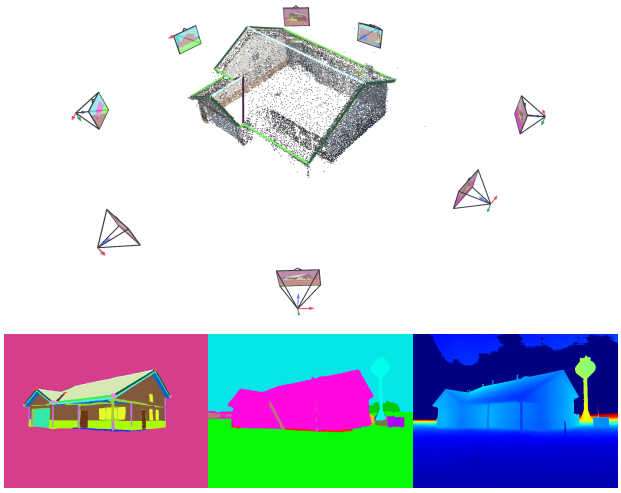

Figure 1. Example scene from the HoHo 22k dataset. **Top**: Sparse SfM point cloud and cameras. **Bottom**: Gestalt/ADE20k segmentations and depth map for one of the views.

2. Dataset

In this section, we describe the "HoHo 22k" dataset and the evaluation protocol used in the challenge. We also summarize the main differences compared to last year's challenge and discuss some data-related issues encountered.

2.1. Modalities

Each scene in the provided dataset consists of the following:

- A 3D point cloud generated by a VGGT-based [20] SfM pipeline, with tracks, intrinsics and poses.
- Gestalt and ADE20k segmentations for the images.
- MoGe-2 [21] depth maps.

* Equal contribution

- Ground truth wireframes (vertices and edges) and semantic edge classes.

The original images are not included in the dataset. An example scene is shown in Fig. 1.

2.2. Splits and Submissions

The dataset is split into a training set with 19677 scenes, a validation set with 170 scenes, and a test set only accessible through the Hugging Face platform. The test set consists of a public split on which participants are allowed 5 submission attempts per day (each with a 2-hour runtime limit) and a hidden split announced at the end of the challenge. Each team is allowed to submit two of their solutions for evaluation on the hidden test split. The performance on the hidden split determines the winner.

2.3. Metrics

The evaluation metric used is called Hybrid Structure Score (HSS) [12], which is the harmonic mean of the F1 score of the vertices and the intersection-over-union (IoU) of the edges. A vertex is predicted positive if it is within 0.5 m from a ground-truth vertex. The edges are modeled with a 0.5 m radius cylinder for deciding the IoU score.

2.4. Updates from S23DR 2025

The dataset went through a major cleanup for this year's challenge. The original dataset was reviewed and inconsistent scenes were removed, resulting in a total of 22k scenes compared to last year's 25k. The pose ambiguity reported by participants from last year [14] seems to have been resolved, as only one unique set of camera parameters is provided per camera. The reconstruction quality was improved and more images per scene included. Last year's depth maps were generated from Metric3D [5] but are now produced by MoGe-2 [21]. The images were reported to be downscaled to 768 px for compute and storage reasons.

2.5. Data-related Challenges

Working with the challenge, we have encountered a few dataset issues. This includes problems with downloading the dataset as well as the quality of the data.

Problem downloading the dataset: Due to a few corrupt images in the training set, we were first only able to download around 13k scenes with the Hugging Face dataset library. After implementing a workaround in our code, we could eventually train our model on the full set, minus the corrupt samples.

Ground-truth wireframe misalignment: We observed that some scenes in both the train and validation splits had misalignment issues between the point cloud and the ground-truth wireframe; see Fig. 2 for an example. For this reason, we manually went through the validation dataset and removed 32 misaligned scenes. For the training data, a manual approach was not feasible so we took two different approaches. First we excluded the samples identified by the organizer's learned baseline [11]. Then we ran our current best model at the time on the training set and filtered away all scenes with HSS < 0.01. That gave a reduction of training scenes by 781. The filtering was conservative as we did not want to remove hard training scenes.

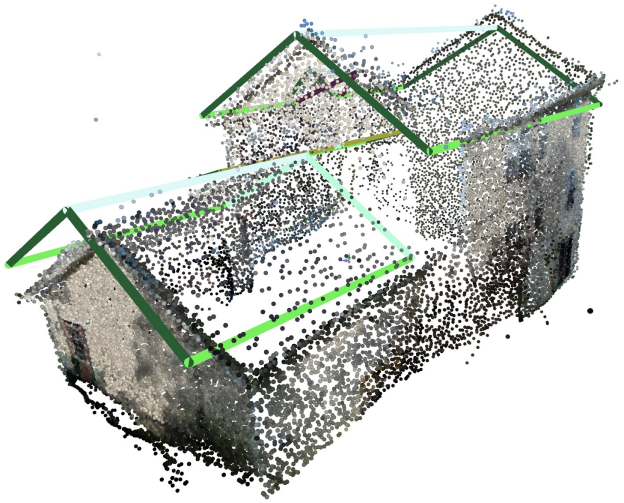

Figure 2. The ground truth wireframe is misaligned with the SfM point cloud. This example is from the validation set.

3. Method

Our method takes as input a point cloud augmented with Gestalt and ADE20k class features, subsampled to a fixed number of points (Sec. 3.1). The network architecture follows a standard encoder-decoder design with a set of learned queries (Sec. 3.2). To increase the receptive field into the segmentation images, we fuse the point features with multi-view informed Gestalt feature vectors, encoding a richer segmentation context (Sec. 3.3). We train the network (Sec. 3.5) with cross entropy and L_1 loss (Sec. 3.4). The predicted edges are post-processed to form the final wireframe model (Sec. 3.6).

3.1. Pre-processing

First, each point in the sparse SfM point cloud is projected into every view in its track. The Gestalt and ADE20k classes are determined by majority vote and are one-hot encoded. We then subsample the point cloud to a fixed number of points, N_p. Points belonging to the Gestalt edge and vertex classes, which are the most informative for the task, are sampled with probabilities 10 and 100 times higher, respectively, than those of other points. Conversely, points with Gestalt class "unclassified" or "unknown" are assigned zero probability. The result is a subsampled point cloud $P \in \mathbb{R}^{N_p \times 3}$ with features $F \in \mathbb{R}^{N_p \times D}$, where D is the

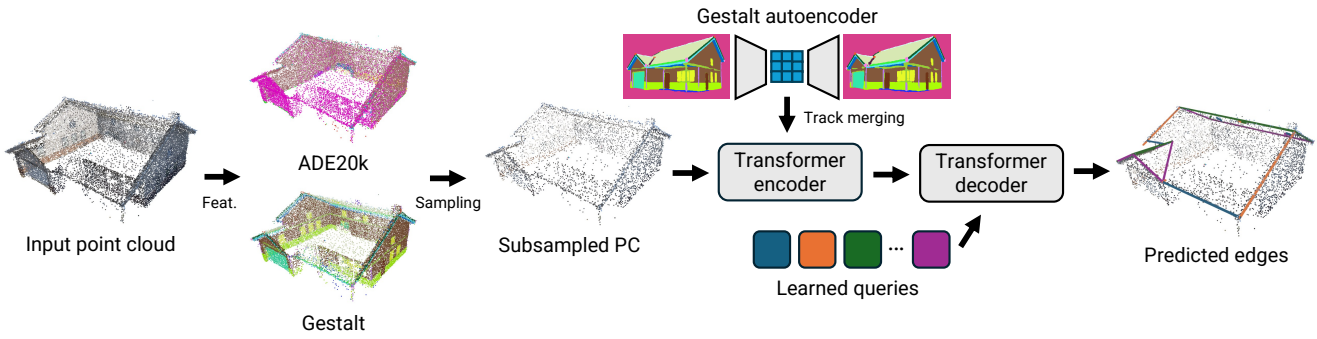

Figure 3. Our method predicts rooftop wireframe edges from a point cloud using a transformer encoder-decoder network.

point feature dimension. The one-hot encoded Gestalt and ADE20k features are of dimension 28 and 149, each, and we also include the RGB color (normalized to the $[0, 1]$ range), so $D = 28 + 149 + 3 = 180$. Finally, the point cloud is resized to fit in the unit cube.

3.2. Network Architecture

Our network has a standard encoder-decoder structure similar to DETR [3] but adapted to the problem of roof wireframe reconstruction. We project the input features F into an embedding space of dimension D_e, add fixed positional encodings [17] computed from to the point cloud P and pass the projected features to a Transformer encoder.

The encoded point cloud is, together with a set of learned anchor edges and corresponding query embeddings, the input to a Transformer decoder. Self-attention among edges and cross-attention between edges and points are used alternately in the decoder layers. Positional encoding is applied for both points and edges at each step.

There are two prediction heads in the network: a classification head and a coordinate head. The classification head is a simple linear layer that outputs logits for the edge classes. As the metrics used in the challenge are purely geometrical, we do not try to predict the actual edge class but instead utilize a single class plus a "background" class. The coordinate head is a small MLP followed by sigmoid activation, predicting the normalized coordinates of the two endpoints.

3.3. Local Gestalt Feature Fusion

While each point is assigned a Gestalt and ADE20k class by majority voting in the pre-processing step, this will inevitably not capture all of the information contained in the segmentation masks due to the sparsity of the point cloud. This should be especially true for the Gestalt maps, which contain explicit segmentations of edges and vertices. In order to increase the receptive field of the points, and recover more of the Gestalt segmentation context, we fuse each projected point feature with an additional learned Gestalt feature that is both multi-view informed and encodes a richer local context.

First, each Gestalt image is encoded to a coarse feature map using a frozen autoencoder with a 32×32×32 bottleneck (*i.e.*, a spatial 32×32 grid of patches, each with feature dimension 32). After subsampling the point cloud, we retrieve a set of corresponding 32-dimensional features based on the projections into each view. Each feature is then projected to a D_g-dimensional vector using a shared linear projector. A positional encoding of the relative view vector, defined as the normalized direction from the origin (in resized point space) to the camera center, is added to this D_g-dim vector. The feature vector is averaged over all views in the track, and then passed through a final linear layer to get the final D_e-dim Gestalt feature. This Gestalt feature is fused with the projected point feature before the encoder by addition, and weighted with a learned factor which is initialized to 0. For simplicity, we used $D_g = D_e$ and the same positional encoder as for point positions.

3.4. Loss Function

We supervise directly on the predicted wireframe edges. They are matched to the ground-truth edges using the Hungarian algorithm [10] with the matching cost between a ground truth edge e and prediction $\hat{e}$ given by

$$\mathcal{C}(e, \hat{e}) = -\hat{p}_c + \mu \min\left(\|e - \hat{e}\|_1, \|\tilde{e} - \hat{e}\|_1\right), \quad (1)$$

where $\hat{p}_c$ is the predicted probability that $\hat{e}$ belongs to the class c of e, and μ is a scale factor. We compute the L_1 distance over the six coordinates, corresponding to the two endpoints of the edge. Since edges are undirected, we take the minimum distance over the original ground-truth edge e and its flipped version $\tilde{e}$, *i.e.* the edge e with its endpoints reversed.

Assuming that the number of queries N_q is larger than the number of ground truth edges N_g the result of the Hungarian matching is a set of edge correspondences $\{(e_i, \hat{e}_i)\}_{i=1}^{N_g}$ and unmatched predictions

$\{\hat{e}_i\}_{i=N_g+1}^{N_q}$. From these we calculate a loss $\mathcal{L} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{L_1}$, where

$$\mathcal{L}_{L_1} = \frac{1}{N_g}\sum_{i=1}^{N_g} \min\left(\|e_i - \hat{e}_i\|_1, \|\tilde{e}_i - \hat{e}_i\|_1\right) \quad (2)$$

is the average L_1 distance for matched edges and

$$\mathcal{L}_{CE} = -\frac{1}{\sum_{i=1}^{N_q} w_{c_i}}\sum_{i=1}^{N_q} w_{c_i} \log \hat{p}_{c_i} \quad (3)$$

a weighted cross-entropy loss. Here, w_{c_i} is the weight for the class c_i of the matched ground truth edge e_i or the background class if there was no match.

3.5. Implementation and Training Details

We train our network on our cleaned HoHo 22k training set. During training, two types of data augmentations are applied to prevent overfitting: random rotation around the y-axis (up direction) and a 50% probability of mirroring the point cloud in the yz-plane. The weighted sampling described in Sec. 3.1 also acts as a regularizer and $N_p = 7168$ points are sampled for each scene to create training examples. We use $N_q = 100$ queries and set $\mu = \lambda = 5$ for matching and computing the loss $\mathcal{L}$, which is applied to the output of each decoder layer and summed. The class weights are $w_e = 1$ and $w_b = 0.1$ for the edge and background class, respectively.

The network has 5 encoder and 5 decoder layers, with embedding dimension $D_e = 360$. It is trained for 75 epochs with the AdamW [9, 13] optimizer using a weight decay of 10^{-4} and a one-cycle learning rate schedule [15]. The learning rate is initially set to 2×10^{-5}, anneals to a maximum of 2×10^{-4} and then gradually decreases to the final value 2×10^{-7}. The batch size is 11. Dropout with probability 0.1 is applied in both encoder and decoder layers, along with layer normalization [2]. Every 500 iterations we compute the mean HSS on our cleaned validation set and save the weights that maximize this metric to use for inference. The model takes 12 hours to train on an RTX 4090 GPU and has 23M parameters.

Gestalt autoencoder: The encoder part of the autoencoder consists of four layers, each applying a 4×4 convolution with stride 2, followed by batch norm [6] and a ReLU activation. The number of output channels are $\{32, 64, 128, 256\}$, respectively. A final 1×1 convolution projects to the 32-channel bottleneck. The decoder architecture is the transpose of the encoder, but outputs class logits which are then are mapped back to the RGB values of the winning class.

For training the autoencoder, 1000 scenes are randomly selected from the train set, giving a total of 9422 training images. Each image is reshaped to 512×512, giving the bottleneck size of 32×32 with a receptive field of 46×46 pixels. Training is done using weighted cross-entropy loss, where edge and vertex classes are assigned a weight of 10 and 100, respectively, while the rest of the weights are set to 1. We use the same optimizer and learning rate schedule as for the Transformer training, and train for 50 epochs with batch size 24.

3.6. Post-processing

At inference time, edges predicted to be of the background class and those with probability $\hat{p}_c$ less than 0.95 are first removed. The result is a set of disconnected edges from which we construct the house roof wireframe.

Since each predicted edge is represented by two independent endpoints, multiple endpoints may be predicted near the same physical roof corner. To avoid duplicate vertices, we merge nearby predicted vertices after edge prediction using an iterative centroid-based strategy: (1) calculate the distances between all vertices; (2) update the two closest vertices to their average vertex position, given that they are closer than 0.5 meters, keeping only one of the vertices. The algorithm repeats until no pair of vertices are closer than the distance threshold. Note that we do not allow the two endpoints of the same predicted edge to merge, since that would collapse an edge into a point. Duplicate edges may appear following this procedure, and all but one copy are removed for each edge.

Our model uses absolute positional encoding, making it sensitive to rotations and flips, even though the underlying reconstruction task should be invariant to such transformations. Since inference is relatively fast, taking only about 25 minutes of the allowed 2 hours on the evaluation server, we employ test-time augmentation (TTA) to improve robustness. We also ensemble two independently trained checkpoints of the same model. In total, we evaluate both checkpoints over four yaw rotations, $0°, 120°, 240°$, and $300°$, with and without horizontal flipping, yielding 14 augmented predictions (only two variants for $300°$ due to resource constraints).

4. Results

In this section, we present quantitative and qualitative results of our wireframe prediction model. We present the baselines provided, results on the public and private test sets and some ablation studies.

4.1. Baselines

Two baselines are given by the workshop organizers: one handcrafted [18] and one learned [11]. In short, the handcrafted baseline detects vertices and edges in the Gestalt segmentations and lifts these 2D detections to 3D using depth maps that were refined to fit the SfM point cloud.

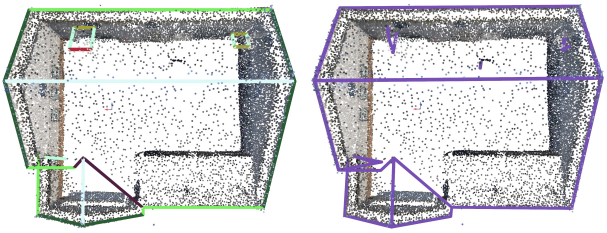

Figure 4. Ground truth (**left**) and predicted (**right**) roof wireframes for one scene in the HoHo 22k validation set. Our method can accurately predict longer edges but struggle with smaller details like chimneys.

As the same vertex/edge can typically be seen in multiple views, the lifting is followed by a de-duplication step.

The learned baseline utilizes a Perceiver-based [7] transformer architecture trained on fused 3D point clouds, created by combining the SfM point cloud with unprojected depth maps. Similar to our approach, it uses Gestalt and ADE20k class features, but not RGB color. The network is trained in a manner comparable to ours, matching predicted and ground-truth edges and using coordinate and cross-entropy loss, but in contrast employs a more complex three-stage training procedure.

4.2. Challenge Results

In Tab. 1 we report results on the public and private test sets. Our method significantly surpasses the two baselines, both in terms of vertex F1 score and edge IoU. It is outperformed by the challenge winner ("VRG_jskvrna") by 0.0066 HSS on the private test set, but is substantially better than the third-place contestant ("StarAtNyte1"). Fig. 4 shows example predictions for one of the scenes in the validation set. While our model accurately reconstructs the most significant edges of the roof, it fails to predict some of the shorter edges, for example those around the chimney in the top right part of the two images.

4.3. Ablation Experiments

We validate some of our design decision by conducting ablation experiments on the full validation set (Tab. 2), where individual components are removed or modified relative to our full model (top row).

First, we run our method with a single trained checkpoint (second row) and compare with the ensemble of two. Using the ensemble is marginally better across all metrics. Next, the test-time augmentations are also disabled (third row), resulting in a larger performance hit. A single network that does not use the Gestalt autoencoder is evaluated (fourth row). It is slightly worse than one that includes the Gestalt feature fusion (second row). We next perform the point sampling with equal probability for all points (except "unclassified" and "unknown", which are still excluded). As we sample a relatively large number of points ($N_p = 7168$), the resulting point clouds still contain many points with Gestalt vertex and edge classification, and the performance decrease is minor (fifth row compared to first). Finally, we try running with only a quarter of the points ($N_p = 1792$, now with the higher weights for vertex and edge points), and note that our method is fairly robust to the input point cloud density (last row).

5. Unsuccessful Approaches

We describe in this section some of the ideas that we tried, which did not improve the performance of our method.

5.1. Plücker Lines

One geometric primitive we tried as network input beyond points was Plücker lines [4]. The main reason for using Plücker lines was to be able to provide 3D information to the network in regions where the point cloud was sparse, often due to low co-visibility. To obtain Plücker lines, we sampled them from the Gestalt segmentation mask only at predicted edge and vertex classes, with a 10 times higher weight for sampling vertices, as there are significantly fewer vertex pixels than edge pixels. Explicitly, at a pixel sample x in an image, we form the ray from the camera center, $C = -R^T t$, to x in world coordinates as

$$d = R^T K^{-1}[x; 1] \tag{4}$$

and normalize it as $\hat{d} = \frac{d}{\|d\|}$ resulting in two degrees of freedom. Then, we define the moment $m = C \times \hat{d}$ encoding the line's offset relative to the origin, giving an additional two degrees of freedom as m has three parameters with the constraint $\hat{d} \cdot m = 0$. We then let $(\hat{d}, m) \in \mathbb{R}^6$ be the input to our network together with a one-hot encoded vector of potential classes (only edges and vertex classes here). We tried various alternatives for using Plücker lines in our network. For example, using a separate encoder for the lines and adding it to the decoder in a cross-attention layer between lines and queries, after the cross-attention between points and queries. Another thing we tried was using Rotary Pose Embedding (RoPE) [16] in the cross-attention layers between the lines and the queries. The above methods worked as we could predict edges and vertices with acceptable accuracy when we only used lines. However, performance did not improve when lines were combined with points. We are unsure why adding lines did not increase the model accuracy, but one guess is that specifying infinite lines gave too weak a signal. A solution to that problem could have been to use monocular depth to create a finite line segment between the camera center and a 3D point.

Table 1. Results on the public and private test sets.

Method	Public			Private		
	HSS	F1	IoU	HSS	F1	IoU
VRG_JSKVRNA	**0.6068**	**0.7351**	0.5252	**0.6542**	**0.7907**	0.5665
LUNDUNI (Ours)	0.6067	0.7113	**0.5375**	0.6476	0.7506	**0.5779**
STARATNYTE1	0.5746	0.6639	0.5156	0.6060	0.6955	0.5451
Learned baseline	0.4470	0.4910	0.4229	0.4739	0.5122	0.4536
Handcrafted baseline	0.3559	0.4717	0.2971	0.3907	0.5067	0.3281

Table 2. Ablation experiments on the full validation set.

Method	HSS	F1	IoU
Full model	**0.4845**	**0.5655**	**0.4377**
– Ensemble	0.4823	0.5615	0.4356
– Test-time aug.	0.4690	0.5502	0.4233
– Gestalt feat. fusion	0.4764	0.5535	0.4335
– Weighted sampling	0.4802	0.5630	0.4330
– Num. points	0.4791	0.5602	0.4322

5.2. Fewer One-hot Classes

Another approach we tested was to limit the size of the input vector to the network. The reasoning behind this was that most of the 177 one-hot classes were rarely used. We tried to assign classes that did not appear in enough pixels or scenes to a dustbin class. However, this did not improve performance. Furthermore, because it was hard to decide on a suitable threshold for when to include a class and when to assign it to the dustbin, we decided not to decrease the number of classes. We believe that the network was likely able to learn to downweight uncommon classes, and thus it did not matter much whether we included them or not. In terms of computational complexity, keeping all classes only increased the computational workload marginally.

5.3. Vertex Consistency Loss

Our model predicts edges independently of each other, despite the fact that the wireframe is connected in a graph-like structure. Consequently, predictions corresponding to the same ground truth vertex may still be spatially separated. To address this, we introduced an additional vertex consistency loss.

The Hungarian matching induces correspondences between predicted edge endpoints and ground truth vertices. For each set of endpoint coordinates $\hat{v}_i$ assigned to a specific ground truth vertex k, we form the group $G_k = \{\hat{v}_i\}$. We compute the mean position $\hat{\mu}_k = \frac{1}{|G_k|} \sum_{i \in G_k} \hat{v}_i$ and define a coordinate consistency loss $\mathcal{L}_{vc}^k = \frac{1}{|G_k|} \sum_{i \in G_k} ||\hat{v}_i - \hat{\mu}_k||_1$. The total loss is the mean over all sets containing at least two edge endpoints. Adding this loss to the model gave roughly a 1% unit increase on the validation set but did not show any improvement on the public test set, so this was omitted in the final model.

5.4. Prediction Heads for Post-processing

In addition to the edge prediction model, two prediction heads were investigated with the goal of simplifying the post-processing stage. The motivation was to let the network directly predict properties of the wireframe instead of relying on heuristics in the post-processing step. In the final model, none of the method provided a significant improvement on the public test set.

Vertex feature head. Our model relies on distance-based merging between vertices that are close enough in space. This can be problematic as wireframes can contain more than one vertex within our distance threshold. To address this, a vertex feature head was added to learn feature embeddings for each predicted vertex. The idea was that vertices belonging to the same ground-truth vertex would get similar feature representations that could make the vertex merging in the post-processing more robust.

The training was performed using a contrastive loss consisting of both positive and negative terms. The positive term encouraged features belonging to the same ground truth vertex to be similar, while the negative term penalized high similarity between features belonging to different vertices. During inference, the vertex features were incorporated into the vertex merging post-processing step by considering both the spatial distance and feature similarity. The rest of the network was frozen during training of this prediction head. Including the vertex features in the post-processing step gave only marginal improvements.

Edge class prediction head. The vertex merging procedure modifies vertex locations to satisfy connectivity constraints between edges. As a side effect, this can change the orientation of the edges and cause initially horizontal edges to lose their horizontal alignment. To mitigate this, the edge classification head was introduced.

The training data contains semantic edge classes, some which correspond to horizontal structures. Based on an

analysis of the validation set, the original edge classes were divided into horizontal and non-horizontal categories. The prediction head was trained to predict this label using cross-entropy loss. The edge classification achieved high accuracy, indicating that the distinction between the categories could be learned.

The predicted class was then used during post-processing. Edges classified as horizontal were constrained to remain horizontal after vertex merging. Despite the high classification accuracy, incorporating this information did not provide any significant improvement on the HSS score.

6. Conclusion

In this paper, we present our contribution to the S23DR Challenge 2026, which aims to reconstruct the roof wireframes from a sparse SfM point cloud and semantic image cues. Our approach is based on a DETR [3] inspired transformer architecture that combines geometric information from the point cloud with semantic features from Gestalt and ADE20k segmentations, fused with a multi-view aggregated local-context Gestalt feature.

To improve the model's performance, we introduce a post-processing stage where we iteratively merge nearby vertices. We further employ test-time augmentation and model ensembling, which provides additional performance gains.

Our method achieves an HSS score of 0.6476 on the private test set, securing second place in the challenge leaderboard and outperforming all but the winning submission.

Acknowledgments The work was supported by ELLIIT, the Swedish Research Council (Grant No. 2023-05424), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Compute was provided by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

References

[1] S23DR Competition at 3rd Workshop on Urban Scene Modeling @ CVPR 2026, 2026. 1

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3, 7

[4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003. 5

[5] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3Dv2: A Versatile Monocular Geometric Foundation Model for Zero-shot Metric Depth and Surface Normal Estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2024. 2

[6] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015. 4

[7] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *International Conference on Machine Learning (ICML)*, 2021. 5

[8] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 1

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

[10] Harold W Kuhn. The Hungarian Method for the Assignment Problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3

[11] Jack Langerman. S23DR 2026 - Learned Submission Baseline. https://huggingface.co/jacklangerman/s23dr-2026-submission, 2026. 2, 4

[12] Jack Langerman, Denys Rozumnyi, Yuzhong Huang, and Dmytro Mishkin. Explaining Human Preferences via Metrics for Structured 3D Reconstruction. In *International Conference on Computer Vision (ICCV)*, 2025. 2

[13] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *International Conference on Learning Representations (ICLR)*, 2019. 4

[14] Jan Skvrna and Lukas Neumann. Structured Semantic 3D Reconstruction (S23DR) Challenge 2025 - Winning solution. *arXiv preprint arXiv:2506.16421*, 2025. 2

[15] Leslie N Smith and Nicholay Topin. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 4

[16] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. 5

[17] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Neural Information Processing Systems (NeurIPS)*, 2020. 3

[18] usm3d Team. S23DR 2026 - Handcrafted Submission Baseline. https://huggingface.co/usm3d/handcrafted_submission_2026, 2026. 4

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Neural Information Processing Systems (NeurIPS)*, 2017. 1

[20] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer. In *Computer Vision and Pattern Recognition (CVPR)*, 2025. 1

[21] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate Monocular Geometry with Metric Scale and Sharp Details. *Neural Information Processing Systems (NeurIPS)*, 2026. 1, 2

[22] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic Understanding of Scenes Through the ADE20K Dataset. *International Journal of Computer Vision (IJCV)*, 127(3):302–321, 2019. 1