Exploring what can be done with hand-crafted 3D reconstruction pipeline

Antonio Jurić*
University of Zagreb, Faculty of Electrical Engineering and Computing
juric.antonio@hotmail.com

Abstract

This paper presents our 2nd place solution for the 2nd S23DR Challenge at CVPR 2025. The challenge focuses on reconstructing structured geometric wireframe models of buildings from posed images and Structure-from-Motion (SfM) outputs. We explored what can be done with hand-crafted pipeline, and how it can be improved. Key enhancements over the baseline include: (1) systematic hyperparameter tuning to optimize the pipeline's performance; (2) a multi-view consistency filter to prune outlier vertices based on cross-view reprojection and edge recovery of filtered vertices; (3) expanding the list of supported vertex and edge classes from the Gestalt segmentations to create more detailed models; (4) improved handling of inputs of various sizes.

Our non-learning-based optimized handcrafted approach demonstrates good performance, resulting in 0.39 score on private leaderboard, securing 2nd place in a challenge.

1. Introduction

The automatic conversion of visual data into structured 3D models is a foundational problem in computer vision. The 2nd S23DR Challenge [1] on CVPR'25 2nd Workshop on Urban Scene Modeling specifically targets this by requiring the reconstruction of semantic wireframe models for building roofs from posed imagery- in HoHo25k dataset. While contemporary approaches leverage end-to-end learning, our work investigates the capabilities of a classical geometric pipeline built upon a provided handcrafted baseline. Our core strategy is to lift rich 2D semantic cues from Gestalt segmentation maps to 3D, enforcing strong multi-view consistency to ensure geometric accuracy and produce a high-quality wireframe.

Our key enhancements over the baseline include: (1) systematic hyperparameter tuning; (2) a multi-view consistency filter to prune outliers and edge recovery of filtered vertices; (3) expanded support for more semantic feature classes; and (4) improved handling of variable input sizes. We believe this optimized pipeline serves as a good baseline and reveals the capabilities of non-learning methods. This report details our methodology and presents the experimental validation for our design choices.

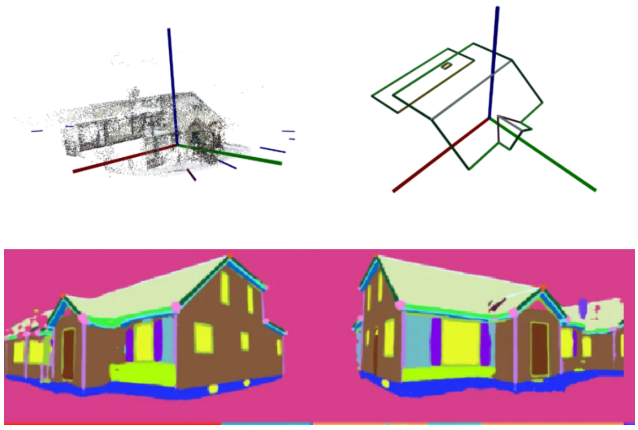

Figure 1. Example of a scene from the dataset.

2. Method

Our method improves upon a provided handcrafted baseline solution. We first briefly describe the baseline pipeline, and then detail our key enhancements.

2.1. Baseline Method

The baseline solution processes each building scene in a multi-stage, per-view manner. First, for each input image (view), it performs 2D feature extraction directly from the semantic Gestalt maps. It identifies vertices (limited to `apex` and `eave_end_point`) by finding connected components of the corresponding colors. Then, it detects line-like features (`eave`, `ridge`, `rake`, `valley`) again with connected components algorithm, and forms edges by connecting vertices that lie close to these detected lines. These 2D vertices are then lifted to 3D by using the provided

*Part-time PhD student; employed at Blackshark.ai.

dense depth maps, which are first aligned with the sparse COLMAP point cloud. Finally, after all views are processed, the 3D vertices from all views are merged based on proximity and type, and the resulting wireframe is cleaned by pruning unconnected or distant vertices.

2.2. Our Enhancements

Building upon this baseline, we introduced several modifications to improve performance, robustness, and the level of detail in the final reconstruction. These enhancements are detailed below. After every modification, we re-ran the pipeline on the validation set and reported the scores in the following tables.

2.2.1 Hyperparameter Tuning

We began by systematically tuning two critical hyperparameters: `edge_th`, the pixel distance for associating vertices to detected edge lines (default value: 10), and `search_radius`, the pixel radius for finding sparse depth priors (default value: 10). Optimal values were found to be 25 for both parameters.

Table 1. Tuning `edge_th` (with `search_radius=10`).

`edge_th`	HSS Mean	Corner F1	Edge IoU
10 (Baseline)	0.1480	0.2198	0.1222
25 (Best)	**0.1917**	**0.2515**	**0.1667**

Table 2. Tuning `search_radius` (with `edge_th=25`).

`search_radius`	HSS Mean	Corner F1	Edge IoU
10 (Baseline)	0.1917	0.2515	0.1667
25 (Best)	**0.2018**	**0.2636**	**0.1752**

Tuning `th` (default value 0.5) in `merge_vertices_3d` method (controls the 3D distance for merging vertices from different views) gave small improvement: `th` value of 0.3 prevented over-merging. Tuning `th` (default value 4.0) in `prune_too_far` method (filters final vertices based on their distance to the sparse COLMAP point cloud) also gave improvement: `th` value of 0.5 significantly cleaned up outlier points.

Table 3. Tuning merge threshold `th` (in meters).

`th`	HSS Mean	Corner F1	Edge IoU
0.5 (Default)	0.2018	0.2636	0.1752
0.3 (Best)	**0.2031**	**0.2629**	**0.1766**

Table 4. Tuning prune threshold `th` (in meters).

`th`	HSS Mean	Corner F1	Edge IoU
4.0 (Default)	0.2031	0.2629	0.1766
0.5 (Best)	**0.2157**	**0.2791**	**0.1864**

Modifying other hyperparameters did not improve the performance. Those include: depth threshold in `fit_scale_robust_median`; `keep_largest` parameter in `prune_not_connected` (setting to True degraded the performance because it only kept the largest connected component of the wireframe, removing good vertices).

So far, described experiments did not include code modifications, only hyperparameter tuning. Rest of methods include code modifications, to higher or smaller extent.

2.2.2 Alternative Vertex Detection

We experimented with an alternative vertex detection strategy controlled by the `use_colmap_for_vertices` flag. When enabled, this method projects sparse 3D points from COLMAP into the image of each view, and uses DBSCAN to cluster projections that fall within the correct semantic regions of the Gestalt map, using cluster centroids as vertices. While theoretically more robust, this approach degraded performance in practice, and thus this feature was disabled in our final pipeline.

2.2.3 Expanded Support for Semantic Feature Classes

We then noticed that the baseline pipeline only supports two vertex classes: `apex` and `eave_end_point`, but Gestalt segmentation maps contains also the `flashing_end_point` class. We added support for this class to the pipeline.

Additionally, baseline pipeline only supports four edge classes: `eave`, `ridge`, `rake`, `valley`, but Gestalt segmentation maps contains also the `flashing`, `hip`, `step_flashing`, `transition_line` classes. We added support for these classes to the pipeline.

These new vertex and edge classes contributed to the performance improvement, as shown in following table, since more Gestalt segmentation classes are now taken into consideration when predicting the wireframe.

Table 5. Effect of adding new semantic classes.

Configuration	HSS Mean	Corner F1	Edge IoU
Before	0.2157	0.2791	0.1864
With New Classes	**0.2249**	**0.2918**	**0.1941**

2.2.4 Multi-View Consistency Filtering

Our most significant enhancement was a multi-view consistency filter, implemented in the `filter_vertices_by_multi_view_consistency` function. As opposed to previous try with projecting points for each view during processing each view, this is done after all views are processed. After all views are processed, this filter prunes outlier 3D vertices by re-projecting each candidate into all overlapping views. A vertex is kept only if its projection is consistent with the semantic class in a minimum number of views. This single step proved crucial for removing noise and provided significant performance boost, as shown in following table.

Table 6. Effect of Multi-View Consistency Filtering

Configuration	HSS Mean	Corner F1	Edge IoU
Before	0.2249	0.2918	0.1941
After	**0.2437**	**0.3360**	**0.2028**

The method has couple of hyperparameters that can be tuned: `min_consistent_views` (optimal value in experiments was 1), `min_shared_points_for_overlap` (optimal value in experiments was 3), and `projection_patch_size` (optimal value in experiments was 30). Most influental parameter was `projection_patch_size`, which controls the size of the patch around the projected 3D point that is checked for consistency. Smaller patch size is more sensitive to small segmentation errors, but larger patch size is more robust to noise. In other words, increasing patch size allows the projection to check the neighborhood of the projected point, which is more robust to small segmentation errors.

Additionally, to fix the problem of disconnected components, we added a step to recover edges of multi-view filtered vertices. This is done by re-running the edge detection logic on the set of filtered, consistent vertices for each view. This step ensures that valid connections between reliable vertices are not lost. This step is implemented in `recover_edges_after_vertex_filtering` function.

2.2.5 Improved Handling of Input Sizes

Last improvement was how baseline solution works with input images of various sizes. When inspecting dataset, we noticed that not all images are horizontal (768x1024 pixels); some are vertical: 1024x768 pixels. If such case occurs, baseline solution would fail to process the scene: if only one image in scene has different orientation, code raises an exception and prints empty wireframe. Problem was simple to fix: instead of trying to create numpy array from list of images (that was cause of exception, because not all images in array have same shape, which is neccessary for numpy array), we proceed with python list of images, and cast each image into numpy array only when necessary. There is no need for resizing images, nor for skipping such images, since algorithm does not depend on fixed sized images. Following table shows the effect of this change.

Table 7. Effect of handling input sizes.

Configuration	HSS Mean	Corner F1	Edge IoU
Before	0.2437	0.3360	0.2028
After	**0.2709**	**0.3780**	**0.2251**

This gives significant improvement in HSS score on available validation set, since now the pipeline can handle all images in dataset. Validation set has 102 scenes in total, and 11 of them have images with different orientation.

2.3. Final Results

Table 8 summarizes the cumulative impact of our enhancements on the **validation set**. Each row adds one of our key contributions. The systematic tuning of hyperparameters, introduction of multi-view consistency, and robust handling of varied inputs all provided significant gains.

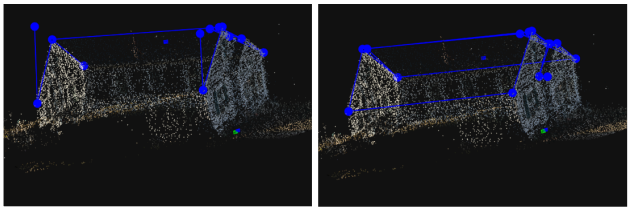

Figure 2. Performance improvement across different enhancement stages: left is original baseline, right is our improved solution. Notice how improved solution removed outliers, and found more both vertices and edges.

This solution achieved **0.39** score on the **private test set**, securing 2nd place. The CPU-only approach enables lightweight execution on edge devices.

Table 8. Ablation study showing the cumulative effect of our enhancements on the **validation set**.

Enhancement	HSS Mean	Corner F1	Edge IoU
Baseline	0.1480	0.2198	0.1222
+ Param. Tuning (2D)	0.2018	0.2636	0.1752
+ Param. Tuning (3D)	0.2157	0.2791	0.1864
+ Expanded Classes	0.2249	0.2918	0.1941
+ Multi-View Filter	0.2437	0.3360	0.2028
+ Input Size Fix (Final)	**0.2709**	**0.3780**	**0.2251**

2.4. Future Work

One of the future improvements could be to use more advanced 3D reconstruction methods, such as learning-based approaches. Learning-based approaches should accelerate in cases where these hand-crafted methods fail: e.g. when there is no visible roof in the image, or when the roof is too complex to be reconstructed by hand-crafted methods, see Figure 3.

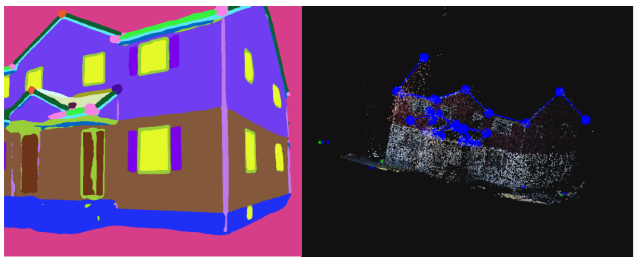

Figure 3. Example of edge failure case where hand-crafted methods struggle with complex roof structures: top roof edge is not visible in the image, and thus not detected by the baseline pipeline.

3. Conclusion

In this report, we presented our 2nd-place solution to the S23DR 2025 challenge. Our approach was to systematically enhance a baseline handcrafted pipeline. Through a series of methodical improvements — including rigorous hyperparameter tuning, the introduction of a critical multi-view consistency filter, and robust handling of varied data inputs — we significantly improved the performance of the initial solution. Our final model, which runs entirely on CPU, demonstrates that a carefully optimized classical geometry pipeline can still be highly competitive for complex structured 3D reconstruction tasks.

References

[1] J. Langerman, D. Mishkin, and Y. Huang. S23DR competition at 2nd workshop on urban scene modeling @ CVPR 2025. https://huggingface.co/spaces/usm3d/S23DR2025, 2025. 1