

COMBINING MONOCULAR GEOMETRY AND SEMANTIC SEGMENTATION FOR STRUCTURAL PLANE EXTRACTION IN INDOOR SCENES

ERIK STRAND

Master's thesis
2026:E33



LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Combining Monocular Geometry and Semantic Segmentation for Structural Plane Extraction in Indoor Scenes

Erik Strand



LUND
UNIVERSITY

Centre for Mathematical Sciences

Abstract

Network cameras observe physical spaces but lack an inherent understanding of the scenes they see. This thesis presents a pipeline for floor and wall instance segmentation from a single uncalibrated RGB image, requiring no scene-specific prior information such as camera intrinsics, depth sensors, floor plans, or labelled floor and wall data for training. The detected instances are intended to support downstream applications such as room-boundary estimation, object detector validation, and camera placement verification. The method combines monocular geometry estimation from MoGe with semantic segmentation from Mask2Former, clustering surface normals by orientation, splitting co-oriented regions by projected depth, and classifying each candidate region via a semantic majority vote.

The pipeline is evaluated on 34 annotated images from 21 unique indoor scenes using Intersection over Union (IoU), which measures overlap between predicted and ground-truth regions. The strongest backbone combination tested achieves a floor instance IoU of 0.945 and a wall instance IoU of 0.854 across all 34 images. Real scenes captured with Axis network cameras scored higher (0.975 floor, 0.900 wall) than the synthetic Hypersim subset (0.938 floor, 0.844 wall). Wall detection was more sensitive to backbone choice than floor detection.

Acknowledgements

I would like to thank my supervisor Robert Stahr for being a great person to work with. You made the project enjoyable throughout, helped me when things were hard, and were good company during the good times too.

I would also like to thank the entire Sophia Antipolis team for the warm welcome and for the many good conversations, both technical and otherwise. Getting a glimpse of French work culture and daily life was really fascinating for me.

Thank you to Axis Communications for giving me the opportunity to write a thesis on a topic I found really interesting, and for letting me do it from the south of France. Getting to live abroad for a few months, get to know a new place, and experience something beyond the usual student routine was something I will not forget.

Finally, thank you to Viktor Larsson and Gustav Hanning for being excellent supervisors from the university side. Your feedback was always helpful and to the point, and you were easy to reach whenever I needed input.

Contents

1. Introduction	1
1.1 Background	1
1.2 Problem	1
1.3 Aim of Research	2
1.4 Delimitations	2
1.5 Report Outline	2
2. Background	3
2.1 Monocular Depth and Geometry Estimation	3
2.2 Surface Normal Estimation	7
2.3 Semantic Segmentation	7
2.4 Transformer-Based Vision Encoders	8
2.5 Structural Plane Detection	8
2.6 Related Work	9
3. Methodology	11
3.1 Dataset and Annotations	11
3.2 System Overview	14
3.3 Model Selection	15
3.4 Model Inference	16
3.5 Runtime Measurement	16
3.6 Plane Segmentation Algorithm	17
3.7 Evaluation Protocol	27
3.8 Parameter Optimisation	27
3.9 Comparison Variants	28
4. Results	29
4.1 Plane Segmentation Results	29
4.2 Effect of Model Combination	34
4.3 Model Inference Time	35
4.4 Algorithm Runtime	35
4.5 Letterbox Effect	36
4.6 Hard Images	37
4.7 Method Comparison	42
5. Discussion and Conclusions	45
5.1 Discussion	45
5.2 Conclusions	54
5.3 Future Work	55
Bibliography	57

1

Introduction

1.1 Background

Network cameras are IP-connected devices that capture and transmit video from a fixed vantage point. Like those developed by Axis Communications, they observe physical spaces but have no inherent understanding of the scenes they see. Gaining that knowledge usually requires external inputs such as camera intrinsics and extrinsics, a floor plan, or supplementary depth sensors. Such inputs require manual work to obtain, are not always available, and do not scale to large deployments.

Monocular geometry estimation has improved substantially in recent years [1], with models now able to recover dense depth maps and per-pixel surface normals from a single RGB image without any calibration data or prior scene knowledge [2]. Extracting structural scene information from a single uncalibrated frame is now feasible, opening a wide range of possibilities for camera-based systems.

This thesis explores floor and wall instance segmentation as one such possibility. The task is to assign each pixel a label of floor, wall, or neither, and groups pixels belonging to the same surface into a single instance. Monocular geometry produces per-pixel estimates of the spatial layout of the scene, and semantic segmentation produces per-pixel class predictions. Both are estimations rather than measurements, since a geometrically flat wall can appear curved in the estimated point cloud and semantic labels can bleed across true surface boundaries, so the challenge is to derive reliable pixel-level labels from imperfect inputs. Together, geometry and semantics form a basis for identifying structural surfaces without any scene-specific prior information such as camera calibration, a floor plan, or physical measurements of the space. Labeled floor and wall regions have concrete uses for camera analytics, such as flagging geometrically inconsistent detections, identifying potential occlusions at furniture boundaries, and estimating room boundaries from wall-floor intersections. Floor and wall instance segmentation is one of many potential applications that monocular geometry enables for surveillance systems, and one relevant to Axis.

1.2 Problem

The purpose of this thesis is to investigate how well monocular geometry in combination with semantic segmentation can produce labeled floor and wall instances from a single uncalibrated indoor image.

1.3 Aim of Research

The research questions of this thesis can be stated as follows:

- RQ1: Algorithm performance.** Can monocular geometry in combination with semantic segmentation reliably detect and separate floor and wall instances at the pixel level across a set of indoor scenes, and how does detection quality vary across different backbone model sizes?
- RQ2: Practical limits.** What are the failure modes of such a method and which scene types are most challenging?

1.4 Delimitations

- **Indoor scenes only.** The method is designed and evaluated exclusively on indoor environments. Outdoor scenes have different surface geometry, lighting conditions, and semantic content, and no claim is made about performance in those types of scenes.
- **On-device deployment.** All experiments are run on a desktop GPU. Profiling inference latency on Axis embedded hardware (ARTPEC), and system-level questions such as when to trigger a run and how to detect that conditions are suitable, are left as future work.
- **Temporal integration.** Each frame is processed independently. Using estimates from previous frames to improve or stabilise the current result is not explored.
- **Model selection.** This thesis does not attempt an exhaustive comparison of monocular geometry estimators or semantic segmentation models. The goal is to demonstrate what the combination of depth estimation and semantic segmentation can achieve for floor and wall labelling, not to determine which individual models are optimal.

1.5 Report Outline

Chapter 2 covers the models, methods, and related work that form the basis of this thesis. **Chapter 3** describes the pipeline, dataset, and parameter optimisation. **Chapter 4** presents the quantitative evaluation across all 34 images, including a comparison against RANSAC-based depth splitting and a no-segmentation ablation. **Chapter 5** discusses the results, states conclusions against each research question, and outlines future work.

2

Background

This chapter provides the technical background for the methods used in this work. It covers monocular depth and geometry estimation, surface normal prediction, semantic segmentation, and the transformer-based encoder architectures the models rely on. It also covers approaches to structural plane detection and related work on room layout estimation and plane instance segmentation.

2.1 Monocular Depth and Geometry Estimation

Monocular depth estimation infers a per-pixel depth map from a single RGB image. The same image projection can arise from objects at different scales and distances, making the problem inherently ambiguous [3]. All methods must therefore impose some prior, either explicit geometric constraints or learned statistical regularities from training data.

Relative and Metric Depth

Depth models are commonly divided into two categories: *relative*, or affine-invariant, depth models and *metric* depth models [1, 4, 5].

A relative depth model predicts depth only up to an unknown global scale and shift. Its output satisfies $\hat{d} \approx sd + t$ for unknown constants s and t , capturing which surfaces are closer or farther away and preserving the ordinal depth structure of the scene, but not providing absolute distances in metres. This affine ambiguity makes relative depth models well suited for training on heterogeneous datasets collected with different sensors and annotation conventions. By factoring out the per-image scale and shift in the loss function, the model can learn from datasets that provide consistent relative or ordinal depth information even when their absolute depth scales are incompatible [1].

A metric depth model predicts depth in real-world units. This requires calibrated sensor data such as RGBD cameras, LiDAR, or stereo with a known baseline. Metric models typically train on domain-specific data and often generalise poorly across domain shifts involving different camera configurations or scene scales [5]. Metric depth is needed by any application that requires real-world coordinates, for example computing the physical distance to an object or projecting a bounding box onto a floor plane.

Depth Anything V2

Depth Anything V2 (DA2) [4] is a relative depth model with a DINOv2-pre-trained [6] ViT [7] encoder and a DPT (Dense Prediction Transformer) [8] decoder. The main contribution over V1 is replacing noisy real-image depth labels with synthetic ground truth for teacher training, followed by large-scale knowledge distillation from a ViT-Giant teacher to a smaller student. DA2 also provides metric fine-tuned variants, but these are domain-specific. The indoor metric variant is fine-tuned on Hypersim [9] synthetic indoor data, so metric accuracy in scenes with substantially different camera configurations or room scales cannot be assumed.

DA2 was evaluated in this work but rejected because recovering metrically consistent 3D geometry from a relative depth map requires camera intrinsics, which are unavailable in an uncalibrated deployment, whereas MoGe predicts an affine-invariant 3D point map directly, from which focal length can be recovered post-hoc without requiring known intrinsics as input [2].

MoGe

MoGe [2] (Monocular Geometry Estimation) predicts a per-pixel 3D point map from a single image. Rather than predicting depth directly, MoGe outputs a per-pixel *affine-invariant point map* $\mathbf{P} \in \mathbb{R}^{H \times W \times 3}$ of 3D coordinates. Affine-invariant means that the predicted geometry is defined only up to an unknown global affine transform, here a scale and translation, so the relative arrangement of points is preserved while the absolute coordinate system is not. From this representation, focal length can be recovered post-hoc without requiring known intrinsics as input. The network uses a ViT-Large [7] encoder pre-trained with DINOv2 [6] and a lightweight CNN-based upsampler decoder.

MoGe is the geometry back-end used throughout this work. The pipeline in this thesis uses the surface normals output by MoGe directly rather than computing normals in a separate step. In the public MoGe repository, a later MoGe-2 normal-estimation feature adds a lightweight convolutional head trained with a squared angular loss, using normals derived from depth and camera intrinsics for supervision. According to the repository note, this feature was added after the MoGe-2 submission and is not part of the original MoGe-2 paper [10].

MoGe Output Visualisation

Figure 2.1 shows the reference image for the Lunch Room Tables image, which is used as the input to MoGe throughout the following figures. Figure 2.2 shows the depth map MoGe produces for the same image. Closer surfaces appear brighter and the colour bar gives the depth values in metres. Figure 2.3 shows the corresponding normal map. Each pixel’s normal direction is encoded as an RGB colour, where the red channel encodes the x component (left/right), green encodes y (up/down in world space), and blue encodes z (toward/away from camera). MoGe uses the OpenCV camera convention where the y axis points downward, so a positive y normal component means the surface faces downward [11]. The perceived colour of any surface depends on its orientation relative to the camera as well as any tilt, so no single colour corresponds to a fixed surface type across all images. To aid interpretation, the figure overlays arrows that represent the projected normal direction at a regular grid of pixels, each coloured by its own normal value. Figure 2.4 shows the same data lifted into 3D in two views. The per-pixel depth values and estimated camera intrinsics back-project the image into metric 3D coordinates, coloured by the original image colours. The perspective view shows how the floor, walls, and furniture are arranged in space, while the top-down view reveals the room footprint.

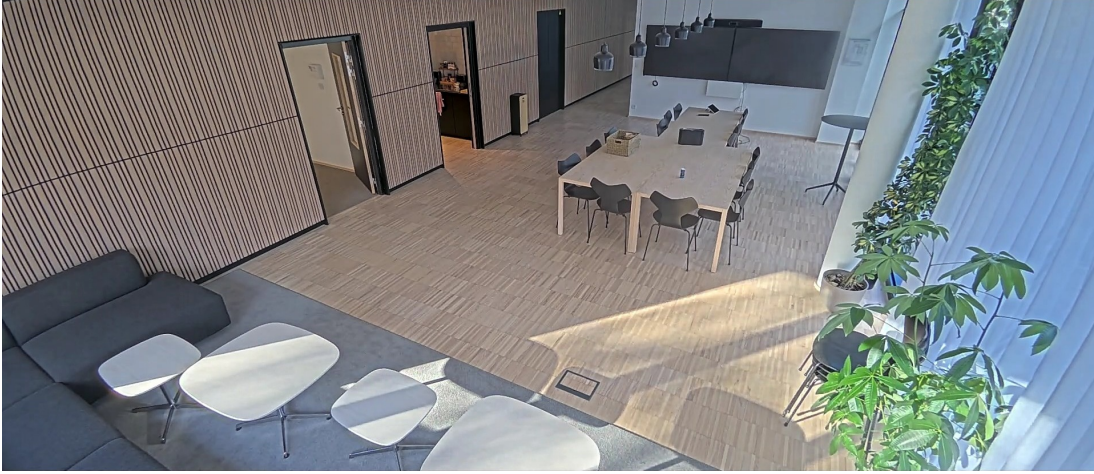


Figure 2.1 Reference image for the Lunch Room Tables image. This image is the input to MoGe in the following figures.

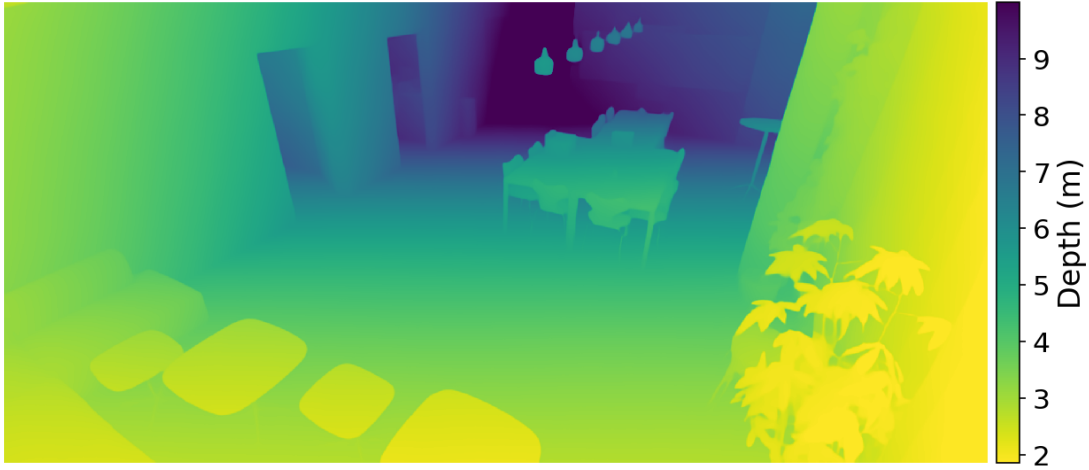


Figure 2.2 MoGe depth map for the Lunch Room Tables image. The colour bar gives the depth values in metres. Closer surfaces appear brighter.

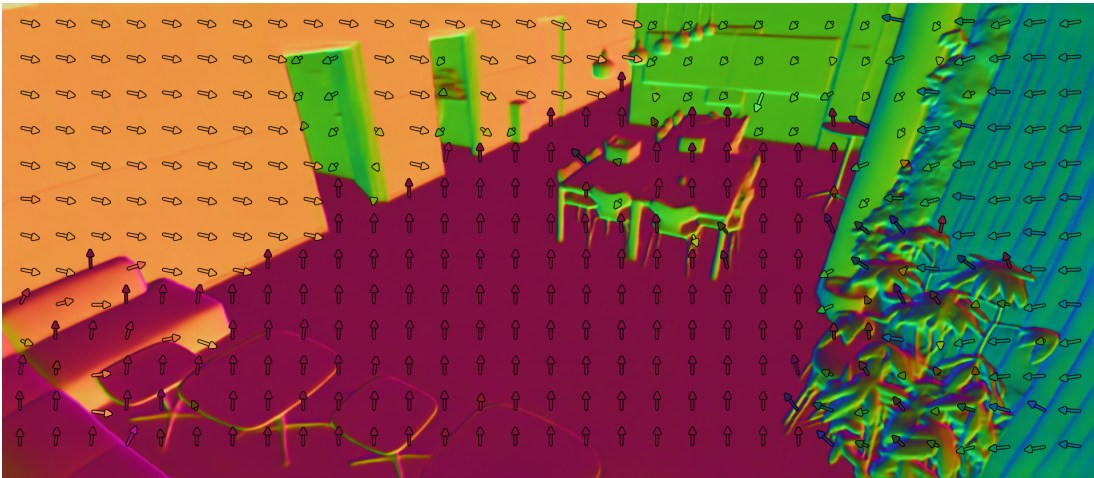
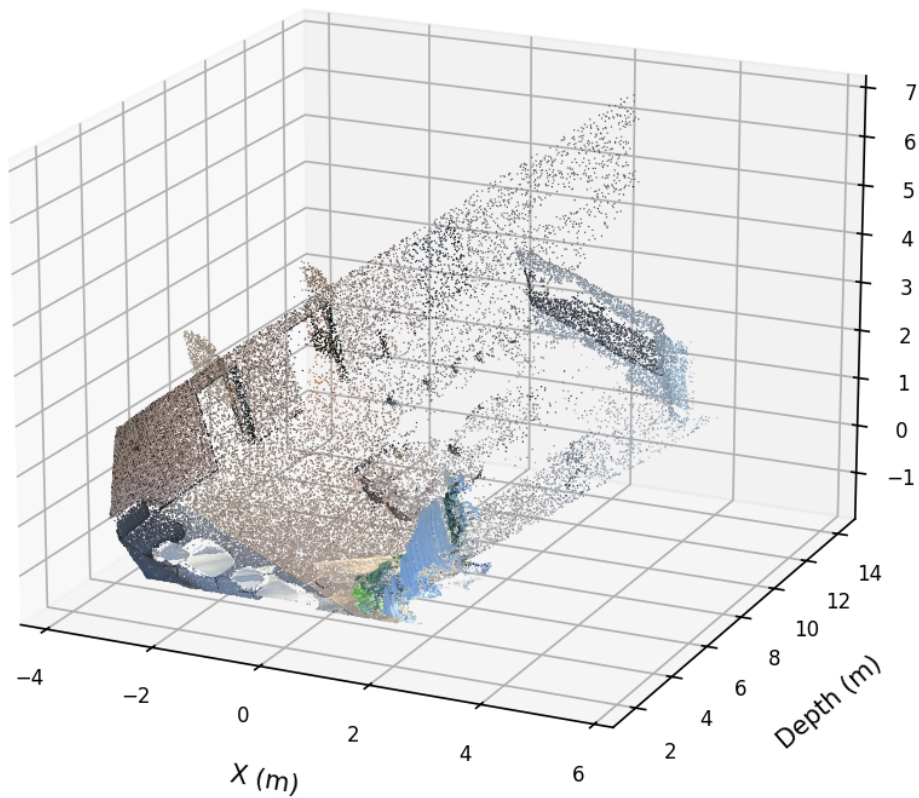
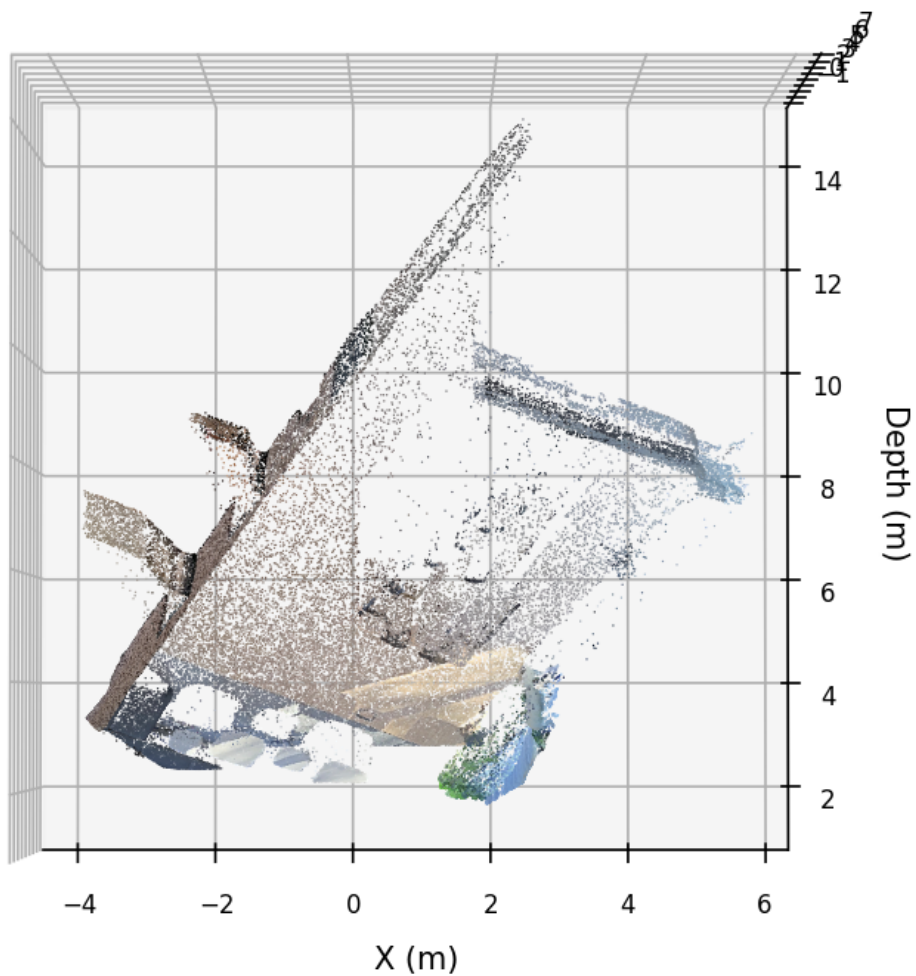


Figure 2.3 MoGe normal map for the Lunch Room Tables image. Each pixel is coloured by its surface normal direction ($R=x$, $G=y$, $B=z$). Arrows show the projected normal direction at a regular grid of pixels, each coloured by its own normal value.



(a) Perspective view of the reconstructed point cloud.



(b) Top-down view of the reconstructed point cloud.

Figure 2.4 MoGe 3D point cloud for the Lunch Room Tables image, coloured by the reference image.

2.2 Surface Normal Estimation

A surface normal at a point $\mathbf{p}$ on a surface is the unit vector $\hat{n}$ perpendicular to the tangent plane at $\mathbf{p}$. All points on a flat surface share the same normal direction.

Normals can be computed from a depth map by first reconstructing neighbouring 3D points and then estimating the local tangent plane, for example using horizontal and vertical 3D displacement vectors or a least-squares fit. Because this operation depends on local depth differences, it is sensitive to depth noise. Small errors in adjacent depth values can lead to large angular errors in the resulting normal. Direct normal prediction reduces this dependence on local depth differentiation by learning normal directions from image and geometric context, and can produce smoother normal fields in noisy or weakly textured regions [2].

2.3 Semantic Segmentation

Semantic segmentation assigns a class label to every pixel. Instance segmentation additionally separates individual object instances within each class.

Mask2Former

Mask2Former [12] is a semantic segmentation model built on a Swin Transformer [13] backbone, a multi-scale pixel decoder, and a Transformer decoder that produces per-query binary masks and class probabilities. The key contribution is masked attention, where each query attends only within its predicted foreground region rather than the full feature map, which sharpens segmentation boundaries.

The checkpoint used in this work is trained on ADE20K [14], a scene-parsing benchmark with 150 semantic categories including wall (class 1), floor (class 4), and ceiling (class 6). Mask2Former provides the per-pixel semantic labels used to classify geometric plane candidates in the pipeline, a candidate region is accepted as a wall or floor only if the majority of its pixels carry a semantically consistent label.

Mask2Former Output Visualisation

Figure 2.5 shows the Mask2Former output for the Lunch Room Tables reference image shown in Figure 2.1. Each pixel is assigned the ADE20K class with the highest predicted probability. The legend lists every predicted class that covers at least 0.1% of total pixels, ordered from largest to smallest region.

SegFormer

SegFormer [15] pairs a hierarchical Mix Transformer (MiT) encoder with a lightweight all-MLP decoder and is trained on ADE20K, producing per-pixel predictions over the same 150-category vocabulary as Mask2Former. Six backbone sizes are available, from B0 (3.7 M parameters) to B5 (85 M parameters). SegFormer was evaluated as an alternative to Mask2Former but was outperformed across all backbone combinations in an untuned comparison on the real-image subset and was not used in subsequent experiments.



Figure 2.5 Mask2Former semantic segmentation output for the Lunch Room Tables image (Swin-Large). Each colour corresponds to a predicted ADE20K class. The legend gives the class name and percentage of pixels assigned to it. Classes covering less than 0.1% of pixels are omitted.

2.4 Transformer-Based Vision Encoders

MoGe and DA2 both use a Vision Transformer (ViT) [7] encoder pre-trained with DINOv2 [6]. Mask2Former uses a Swin Transformer [13] encoder. Both architectures are available in multiple sizes that trade off accuracy against inference cost. The ViT sizes evaluated in this work are ViT-S (22 M parameters), ViT-B (86 M parameters), and ViT-L (307 M parameters). The Swin sizes evaluated are Swin-T (28 M), Swin-S (50 M), Swin-B (88 M), and Swin-L (197 M parameters).

2.5 Structural Plane Detection

Depth Histogram Splitting

When multiple physically distinct surfaces share the same normal direction, they occupy different positions along that normal axis. A table top above a floor or a raised platform in front of the main floor level are examples. These co-oriented layers can be separated by projecting each point $\mathbf{p}_i$ onto the shared normal $\hat{\mathbf{n}}$ to obtain a signed depth offset $d_i = \mathbf{p}_i \cdot \hat{\mathbf{n}}$, then finding modes in the resulting one-dimensional histogram. Points on the same physical surface concentrate around the same offset and produce a peak, while a valley between two peaks indicates a physical gap between layers. The same approach has been applied to LiDAR point clouds for automated floor-plan reconstruction, where point-density histograms along the vertical axis separate floor and ceiling layers [16].

RANSAC

RANSAC (Random Sample Consensus) [17] is a robust model-fitting algorithm that tolerates a large fraction of outliers. In each iteration, a minimal random subset of points defines a candidate model, all remaining points within a distance threshold ϵ are counted as inliers, and the candidate with the most inliers is retained. The accepted model is then refit by least squares on all inliers.

Applied to 3D plane fitting, three points define a candidate plane $\mathbf{n}^\top \mathbf{x} = d$ and inliers are measured by perpendicular distance. The same principle applies to lower-dimensional models, where fitting a constant depth layer to projected depth values requires only a single sample point and measures inliers by scalar depth difference.

RANSAC finds one dominant model per run. Extracting all walls and the floor requires running it sequentially, removing inliers after each round. Adjacent walls with similar orientations are captured together in a single round unless inlier removal is precise, and later rounds operate on increasingly sparse point clouds.

2.6 Related Work

Sensor-Based Structural Surface Detection

Consumer RGB-D cameras made structured indoor understanding from depth data broadly accessible from around 2010. Silberman et al. [18] introduced the NYU Depth v2 dataset and demonstrated indoor scene interpretation from RGB-D images, combining colour and Kinect depth cues to label regions such as floor, structural elements, furniture, and props. Sensor-based methods benefit from dense metric point clouds with a well-characterised noise model, where RANSAC and region-growing reliably fit explicit planes. The limitation is hardware dependency. A depth sensor must be co-located with the camera and integrated into the system from deployment, which is impractical for existing network camera installations.

LiDAR-equipped mobile robots commonly extract floor and wall planes from 3D point clouds using iterative RANSAC-style plane fitting [19], where each detected plane’s inliers are removed before fitting the next plane, building on the robust model-fitting principle introduced by Fischler and Bolles [17]. This works well with the dense, low-noise measurements of laser scanners but transfers poorly to monocular depth estimates, where residual surface curvature means no single inlier threshold handles all surfaces consistently across diverse scenes.

Room Layout Estimation

Hedau et al. [20] introduced a box-layout formulation in which a room is approximated as a cuboid observed from the inside. It provides a compact estimate of the dominant room envelope, including floor, wall, and ceiling regions, but it still represents the scene as one global layout instead of separately segmented structural surfaces.

Newer methods are more general. LayoutNet predicts a full 3D room layout from a single RGB image with a learned model [21]. Another line of work uses panoramic input to capture the full room context. LED2-Net formulates 360-degree room layout estimation as depth prediction on the horizon line of a panorama and uses differentiable depth rendering for training [22]. In practice, LED2-Net is designed for panoramic images rather than standard perspective images. These newer approaches reduce several limits of early box-layout methods, but they still target global room-envelope estimation rather than separate floor and wall instance segmentation.

Learning-Based Plane Instance Segmentation

Liu et al. [23] proposed PlaneNet, the first end-to-end neural architecture for piece-wise planar reconstruction from a single RGB image. The network predicts plane parameters and per-plane segmentation masks directly from RGB appearance, training on piece-wise planar depth maps extracted from ScanNet. PlaneRCNN [24] extends the Mask R-CNN [25] instance-detection framework with plane-specific prediction heads for segmentation masks and 3D plane parameters. It further uses a multi-view consistency loss to refine masks by enforcing consistency with nearby views during training. The associative embedding approach of Yu et al. [26] maps each pixel to a learned embedding space where coplanar pixels cluster together. Mean-shift clustering [27]

groups pixels into plane instances, after which a second stage estimates 3D plane parameters per instance. These methods can detect arbitrary planar surfaces, but they do not by themselves assign application-specific semantic surface classes such as floor, wall, or platform without additional interpretation or post-processing. They also rely on plane-instance supervision, or training data derived from processed RGB-D datasets.

The approach in this thesis uses off-the-shelf geometry and segmentation models with no plane-specific training data, and assigns semantic labels directly from surface orientation using domain-specific geometric constraints. The trade-off is that the method is limited to the surface types for which those constraints hold, whereas general plane detectors can in principle detect any planar surface given sufficient training data.

3

Methodology

This chapter describes the dataset, pipeline, and evaluation framework used to investigate floor and wall instance segmentation from single indoor images. The pipeline combines monocular geometry estimation with semantic segmentation and is evaluated on 34 annotated images drawn from real Axis camera footage and the Hypersim synthetic dataset [9]. The chapter covers model selection, the plane segmentation algorithm, runtime measurement, the evaluation and parameter optimisation procedures, and two comparison variants used to isolate the contributions of individual pipeline stages.

3.1 Dataset and Annotations

Scene Collection

The evaluation dataset contains 34 images from 21 unique scenes, drawn from two sources.

Real scenes. Footage was captured by Axis network cameras installed in the building where this project was conducted. Five of the six images are from the same office floor and include a lunch room covered by two cameras at opposite corners, a middle office area framed through a doorway, a kitchen with a partial corridor along one side, and a lounge area in which a corridor recedes into the background. The sixth image is the garage on the floor below, which is visually distinct from the office footage. The five office images share floor and wall finishes but vary in furniture density and visible corridor depth.

Synthetic scenes. These come from Hypersim [9], a photorealistic indoor dataset rendered from artist-modelled environments. The 28 synthetic images come from 19 unique synthetic scenes. Several images are different views of the same underlying scene, captured from different camera angles and positions. Because of viewpoint changes, visible layout and furniture arrangement can differ strongly between images from the same scene. Texture and lighting style still remain similar within each source scene. Synthetic images were chosen specifically to cover cases that do not appear in the office footage, such as heavy occlusion, non-orthogonal wall layouts, multiple wall planes at different depths, and raised floor levels. Because of this selection criterion the synthetic subset is intentionally harder than the real one, and performance numbers across the two subsets should be interpreted with that asymmetry in mind.

All 34 images show the scene from a downward-tilted angle, consistent with how a typical surveillance camera would view the scene.

Annotation Procedure

Ground-truth annotations were produced in LabelMe [28] and stored as polygon-based JSON files. LabelMe represents each shape as a single, non-self-intersecting polygon and has no native support for polygons with holes. Because of this the schema uses four labels and a group id to encode each instance. Multiple polygons sharing the same group id are unioned to form the instance mask, and subtraction polygons are used to cut out occluded regions. In practice, whichever approach is more convenient is used so a complex wall visible as several disconnected patches may be drawn as multiple Wall polygons with the same group id, while a mostly-visible floor with furniture in the way may be drawn as one large Floor polygon plus several Floor Sub cutouts. Both strategies produce the same final per-pixel mask and can even be combined within a single instance.

- **Wall / Wall Sub:** filled polygons for a visible wall instance, all tagged with the same group id. Any pixel covered by a Wall Sub polygon with that group id is excluded from the final mask.
- **Floor / Floor Sub:** the same scheme for the floor surface.

Figure 3.1 shows a ground-truth annotation overlay for the coffee lounge image, illustrating how floor and wall instances are labelled together with the effect of subtraction polygons.

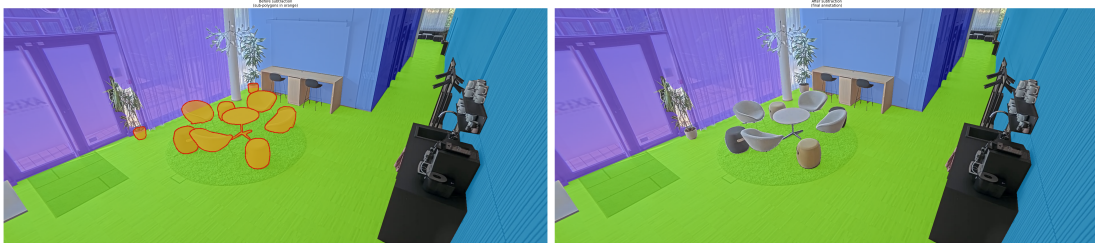


Figure 3.1 Ground-truth annotation for the coffee lounge image. The left panel shows the full enclosing polygons with subtraction polygons highlighted in orange. The right panel shows the final annotation after subtraction is applied, removing occluded regions such as furniture from the floor. The floor instance is shown in green and individual wall instances are shown in blue shades.

Defining surface boundaries precisely is harder than it appears. Some boundaries require a judgement call that has no uniquely correct answer. Figure 3.2 shows an example from a bathroom image where both visible walls contain a recessed indentation that interrupts the flat surface. The right wall also has a window set into its indentation. The decision in this dataset was to include both the indentations and the window within the respective wall annotations, but any of them could equally be excluded. Similar questions arise for doorway reveals, alcoves, and partially visible wall sections throughout the dataset.



Figure 3.2 A bathroom image illustrating an annotation ambiguity. Both walls contain a recessed indentation and the right wall also has a window set into its indentation. All were included in the wall annotations, though a reasonable annotator could also have excluded any of them. Consistent application of a fixed rule across all such cases matters more than the rule itself.

Beyond defining a visually plausible geometric ground truth, the annotations define the target against which the algorithm is evaluated and optimised. A labelling decision is therefore also a specification of what correct behaviour looks like for the intended application. For ambiguous regions such as window indentations, or alcoves, the application may be indifferent to whether those pixels belong to a wall instance or not. Treating one arbitrary interpretation as the only correct one creates a scoring artefact. The algorithm is penalised for errors that carry no functional consequence, and parameter optimisation may be steered toward recovering performance on borderline regions at the cost of pixels that matter more. An evaluation protocol designed to handle this explicitly would annotate uncertain regions with a separate label and exclude or weight those pixels differently during scoring, requiring two additional annotation classes covering ambiguous floor and ambiguous wall regions respectively. This approach was not pursued here because it increases annotation effort, requires a label type that is unlikely to be present in any external dataset, and has no established convention for how it should be applied. The annotations in this study follow a fixed rule applied consistently.

Each image took approximately 45 minutes to annotate. All annotations were made by the same person in an attempt to keep the labelling criteria consistent across all images. Complete consistency is difficult to achieve in practice since walls interact with each other and with scene objects in many ways. The annotation should therefore be seen as a reasonable and internally consistent ground truth rather than a uniquely correct one.

Hypersim Ground-Truth Preparation

Hypersim distributes per-pixel semantic labels and instance IDs in HDF5 format [29] using the NYU-40 taxonomy [18], but not polygon annotations compatible with the schema in section 3.1. To bridge this gap, floor (NYU-40 class 2) and wall (class 1) instance masks were extracted from the semantic instance data, the contour of each mask was traced and simplified with the Douglas–Peucker algorithm [30], and any pixels belonging to overlapping objects were converted into the corresponding Wall Sub or Floor Sub subtraction polygons. The result was saved as a LabelMe JSON file so that Hypersim scenes are processed identically to hand-annotated ones by all downstream evaluation code.

Instance-level wall labelling still required manual intervention. Hypersim assigns instance IDs to movable objects but treats structural surfaces such as walls and floors as a single undifferentiated semantic region. The automated conversion therefore produced a single wall/floor polygon per image rather than one per visible instance. Each Hypersim image therefore required a manual pass to split the wall region into separate instances, which meant that preparing the synthetic ground truth was still time-consuming even when the data was pre-annotated.

3.2 System Overview

Processing is split into two stages that run independently. The first stage runs MoGe [2] and Mask2Former [12] on each image and writes their outputs to disk. The second stage reads those cached outputs, runs the plane-segmentation algorithm, and computes evaluation metrics against ground-truth annotations.

The two stages are independent because the segmentation algorithm and its parameters have no effect on the model outputs, since MoGe and Mask2Former take only the input image. Caching their outputs once allows the segmentation algorithm to be developed, adjusted, and re-evaluated without re-running inference.

Figure 3.3 shows the full data flow.

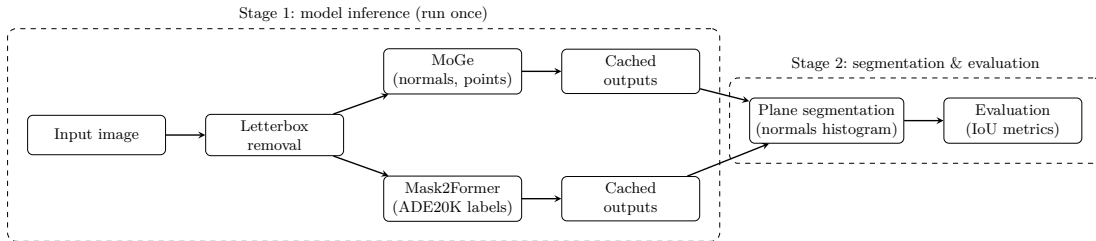


Figure 3.3 System pipeline. Stage 1 runs MoGe and Mask2Former on each image once and caches the outputs. Stage 2 reads those cached outputs and can be re-run independently for any algorithm variant or parameter configuration.

3.3 Model Selection

Four models were evaluated before settling on MoGe and Mask2Former. These were a relative-depth estimator, a monocular geometry estimator, and two semantic segmentation networks.

Monocular Geometry Estimation

MoGe was chosen as the geometry back-end. Two properties make it suitable for this. First, it estimates the camera field of view directly from image content without requiring known intrinsics, which matters because Axis cameras are commonly deployed without precise calibration. Second, it predicts surface normals as a direct network output rather than deriving them by finite differences from a depth map.

Depth Anything v2 [4] was evaluated first but produces relative depth. Converting relative depth to metrically consistent surface normals requires the camera field of view [2], which is unavailable in the uncalibrated deployment scenario targeted here. DA2 was therefore abandoned early.

Semantic Segmentation: Mask2Former vs. SegFormer

Development began with SegFormer [15]. On visual inspection of early results, SegFormer misclassified a large table as floor in the Kitchen image even with the largest backbone, which prompted a switch to Mask2Former. Because both models output per-pixel ADE20K [14] class indices, they are completely interchangeable downstream, so no other component needed modification to switch between them.

To decide which model to keep, both were evaluated on the six real images using identical algorithm parameters that had not yet been tuned to either model. Table 3.1 reports both parameter count and combined IoU score for each combination. The IoU metrics used to compute this score and how they should be interpreted are defined in section 3.7. All four Mask2Former variants outperform all six SegFormer variants in mean score. The Mask2Former variants are also substantially larger models. The smallest Mask2Former variant (Swin-Tiny, 47 M parameters) is comparable in size to SegFormer B3 (47 M parameters), placing it in the middle of the SegFormer backbone size range, and the largest Mask2Former variant (Swin-Large, 216 M parameters) is considerably larger than all SegFormer variants. The performance advantage should therefore be understood partly as a consequence of model capacity rather than architectural superiority alone. The Kitchen image accounts for most of the gap, where Mask2Former achieves 0.72–0.73 against SegFormer’s 0.50–0.65, and the table misclassification described above was present even with the largest SegFormer backbone. Among Mask2Former sizes, Swin-Large achieves the highest mean (0.82) but by a negligible margin over Swin-Base (0.82) and Swin-Tiny (0.82), and Swin-Small is the only Mask2Former variant that shows an inconsistent result, dropping to 0.657 on Kitchen.

Because the subsequent parameter optimisation was conducted using Mask2Former outputs, repeating this comparison with tuned parameters would favour Mask2Former by construction. The untuned comparison above is therefore the appropriate basis for the model-selection decision. SegFormer was excluded from all subsequent parameter tuning and final evaluation.

Table 3.1 Parameter count and combined IoU score for each segmentation model on the six real images, evaluated with untuned algorithm parameters. Best score per scene in bold. Cof. = Coffee lounge, Gar. = Garage, Kit. = Kitchen, LR-C = Lunch Room Corridor, LR-T = Lunch Room Tables, Mid. = Middle office.

Model	Params (M)	Cof.	Gar.	Kit.	LR-C	LR-T	Mid.	Avg
Mask2Former Swin-Large	215.5	0.908	0.727	0.727	0.853	0.752	0.948	0.819
Mask2Former Swin-Base	106.9	0.906	0.727	0.724	0.854	0.752	0.948	0.819
Mask2Former Swin-Tiny	47.4	0.908	0.727	0.732	0.855	0.723	0.947	0.815
Mask2Former Swin-Small	68.8	0.908	0.727	0.657	0.855	0.752	0.948	0.808
SegFormer B2	27.5	0.881	0.728	0.645	0.853	0.721	0.948	0.796
SegFormer B3	47.3	0.905	0.725	0.634	0.853	0.728	0.930	0.796
SegFormer B4	64.1	0.876	0.727	0.566	0.855	0.752	0.948	0.787
SegFormer B1	13.7	0.886	0.728	0.551	0.855	0.722	0.939	0.780
SegFormer B5	84.7	0.904	0.728	0.495	0.850	0.731	0.948	0.776
SegFormer B0	3.8	0.905	0.728	0.554	0.848	0.115	0.939	0.681

3.4 Model Inference

Letterbox Removal

Axis cameras apply barrel-distortion correction internally, which produces black borders around the output image. Left in place, these borders cause depth models to produce erroneous geometry near the image edges. Before running any model, the letterbox is detected and removed automatically by scanning each row and column for the first and last positions whose maximum pixel intensity exceeds a threshold of 10, then cropping to that bounding box. The cropped image is saved alongside the original and re-used on subsequent runs. The quantitative effect of retaining the letterbox is reported in Table 4.5.

MoGe: Depth and Surface Normal Estimation

Three MoGe backbone sizes are evaluated: ViT-S, ViT-B, and ViT-L. Running all three allows the results chapter to show how much geometry quality affects segmentation performance, since ViT-L produces noticeably cleaner normal fields than ViT-S. For each image and backbone, the outputs written to disk are the depth map, the 3D point cloud ($H \times W \times 3$), the per-pixel surface normals ($H \times W \times 3$), the estimated camera intrinsics, and the validity mask.

Mask2Former: Semantic Segmentation

Four Mask2Former backbone sizes are evaluated: Swin-Tiny, Swin-Small, Swin-Base, and Swin-Large. As with MoGe, covering the full size range lets the results show how much the choice of segmentation backbone matters in practice. Model logits are interpolated to the input resolution, and the per-pixel class with the highest score produces a per-pixel integer label map.

3.5 Runtime Measurement

Model inference was timed on an NVIDIA GeForce RTX 5090. Algorithm timing ran on an AMD Ryzen 9 9900X with the precomputed model outputs loaded from disk, since the segmentation algorithm involves no GPU operations. Timing was split into two separate studies, one for model inference and one for the segmentation algorithm.

Model inference times were measured by loading each model, running three warmup passes to prime the GPU, and then recording ten timed passes with CUDA synchronisation before and after the timing window. The mean of the ten runs was reported as the inference time per image. A single real-scene image was used as the test input for all measurements.

Algorithm times were measured separately from model inference. The timing loop loaded precomputed MoGe and Mask2Former outputs from disk and ran the full segmentation algorithm without any GPU involvement. One warmup run per image was discarded, and three timed repeats were averaged to give a per-image estimate. The mean and standard deviation across all 34 images were then computed per model combination. This design isolates the algorithm from model-loading overhead and from GPU scheduling variability.

3.6 Plane Segmentation Algorithm

The algorithm reads the cached normal map, 3D point cloud, validity mask, and semantic label map, and returns a set of labelled plane instances for each image. It has six steps. Unless otherwise noted, all numerical parameter values cited in this section are the outputs of the Optuna [31] optimisation study described in section 3.8. Figure 3.4–Figure 3.6 show the algorithm inputs for the Lunch Room Tables image, which achieves a combined IoU of 0.945 with ViT-L and Swin-Large. Figure 3.7 through Figure 3.15 then illustrate each processing step.

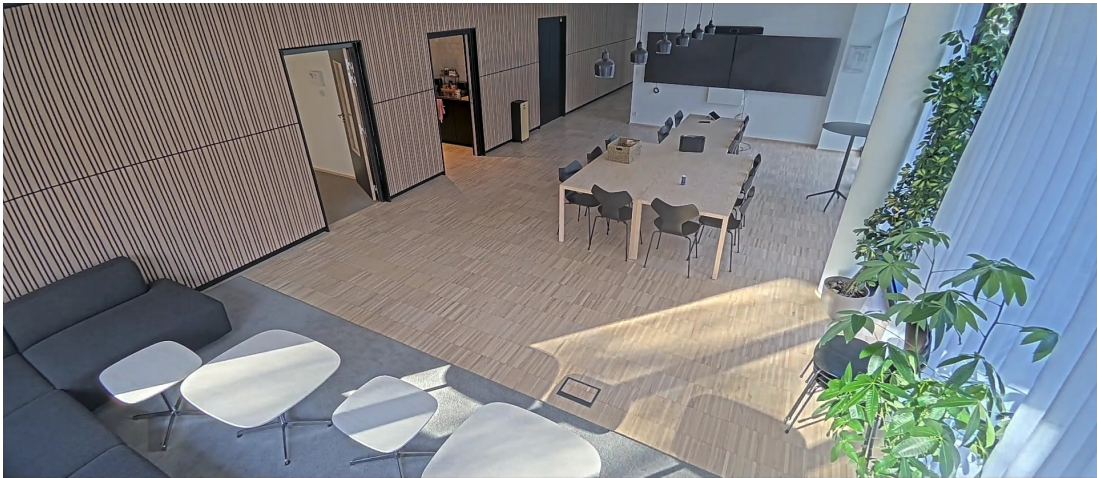
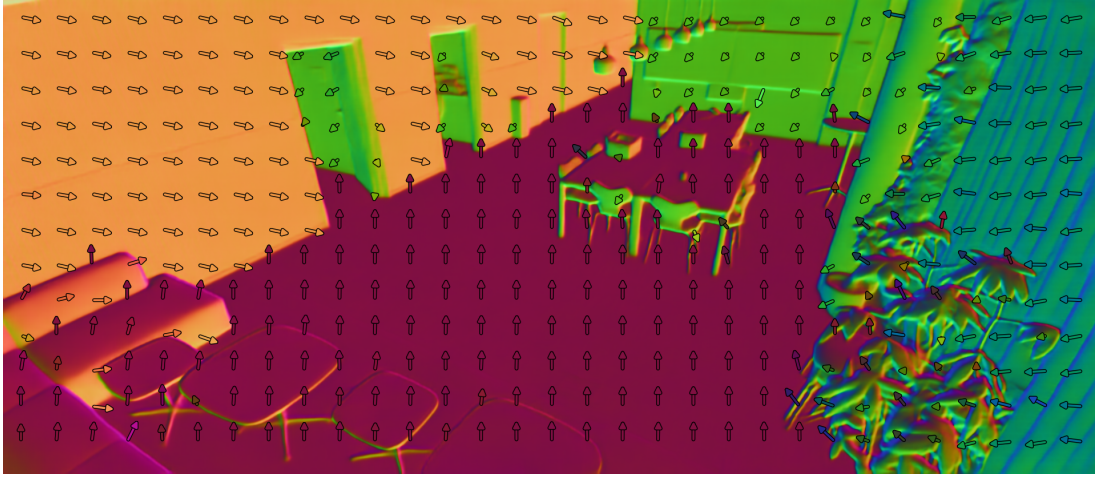
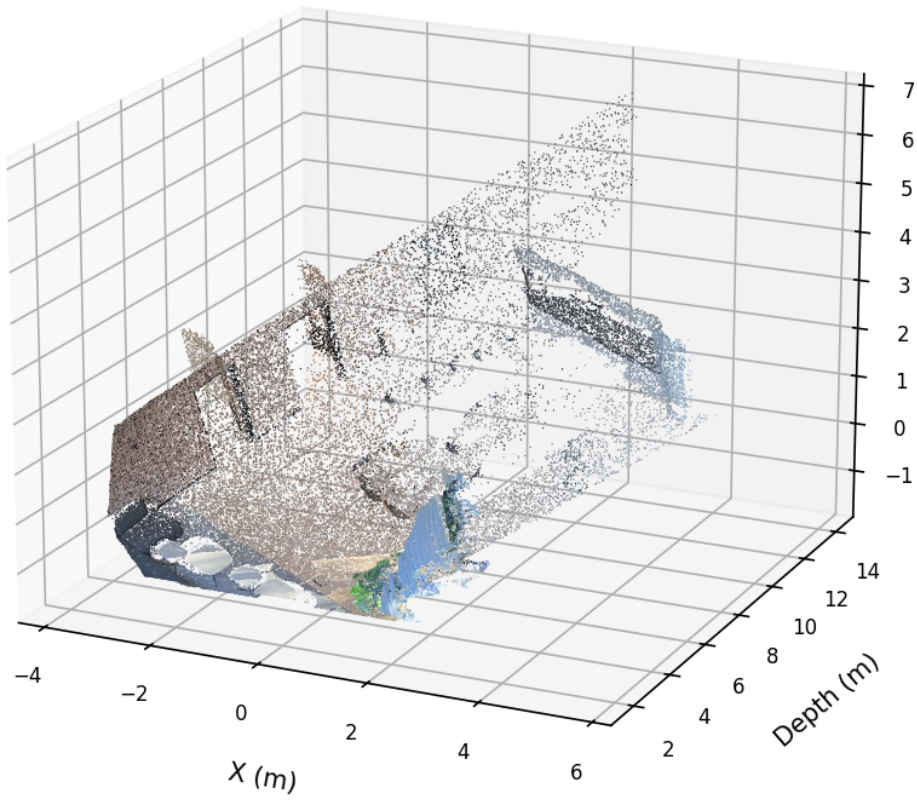


Figure 3.4 Original image: Lunch Room Tables (input to both MoGe and Mask2Former).



(a) MoGe surface normals ($R=n_x$, $G=n_y$, $B=n_z$). Arrows show the projected normal direction at a regular grid of pixels.



(b) Perspective view of the MoGe point cloud, coloured by the reference image.

Figure 3.5 MoGe surface normals and 3D point cloud for the Lunch Room Tables image, used as algorithm inputs.

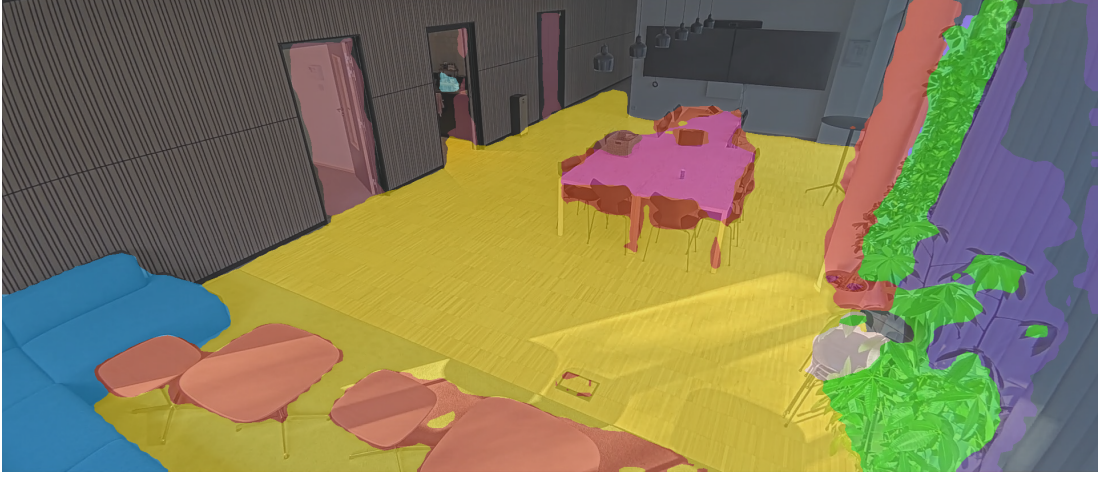


Figure 3.6 Mask2Former semantic label map (ADE20K, 150 classes) used as algorithm input. Each colour represents a distinct class.

Floor and wall instances are treated separately from step 3 (depth histogram splitting) onward. After greedy normal clustering, each cluster is labelled floor or wall based on the y-component of its mean unit normal. MoGe outputs normals in the OpenCV camera coordinate system, where the Y axis points downward, so a surface normal pointing upward in the scene has a negative y-component. Clusters for which $n_y < -0.25$ are therefore handled as floor clusters and all others are handled as wall clusters. From that point, every subsequent step applies independent logic for the two types. The histogram splitting uses separate prominence and gap thresholds, and connectivity splitting is applied only to wall clusters. Floor instances proceed directly to classification after histogram splitting. This separation is motivated by two observations. First, they present fundamentally different segmentation challenges. A floor must be flat, because any surface with strongly varying topology is not walkable and therefore not a floor in the functional sense relevant to this application. This physical constraint means that a real floor should ideally produce a tight cluster of upward-facing normals, and any surface whose normals deviate substantially from that direction can safely be excluded without meaningful loss. Walls carry no equivalent constraint. They can face any lateral direction, bear highly varied surface finishes, textures, and decorative elements, and appear in many disconnected fragments. All of these factors introduce normal variation that makes wall segmentation inherently harder than floor segmentation. Secondly, decoupling the parameters removes a trade-off, since settings that maximise floor segmentation quality tend to over-split walls, and vice versa.

Normal Denoising

The normal fields produced by MoGe contains per-pixel variation even on physically flat surfaces due to model prediction imprecision. A per-channel Gaussian blur with kernel size 13 ($\sigma = 13/3$) is applied to the normal field, followed by per-pixel re-normalisation to unit length. The idea is to make the subsequent clustering step more robust to individual noisy normals without blurring across surface boundaries where normal discontinuities are large. In practice the effect was more pronounced for the smaller ViT-S and ViT-B models, which produce noisier normal fields than ViT-L, and the denoising step therefore matters more when using a smaller backbone. Figure 3.7 shows the result after denoising.

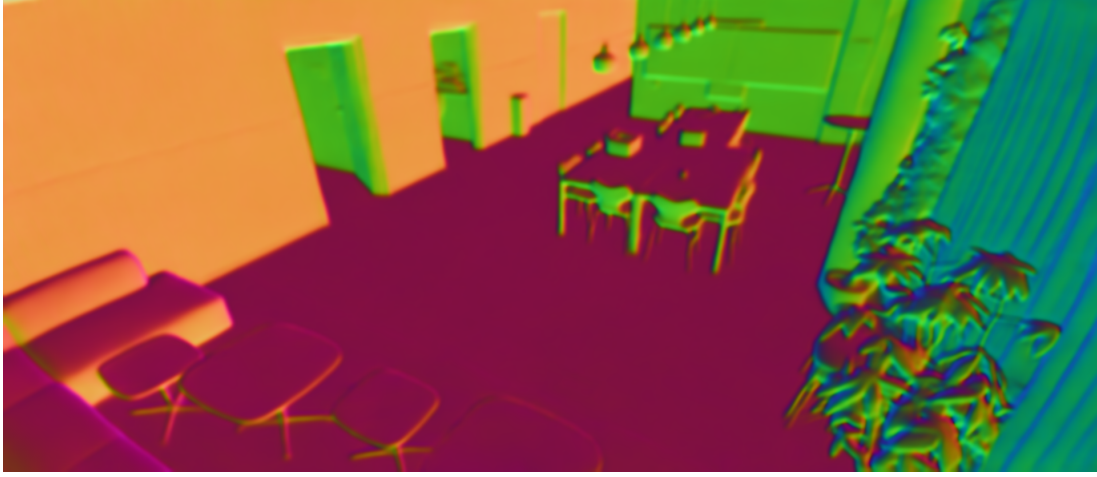


Figure 3.7 Surface normals after per-channel Gaussian denoising (kernel 13, $\sigma = 13/3$). The change is subtle but reduces per-pixel variance on flat surfaces.

Greedy Normal Clustering

Pixels are partitioned into groups of approximately co-planar orientation by the following greedy procedure.

1. Find the mode direction among all unassigned valid normals by coarsely quantising each normal to a grid with step 0.1 (approximately 6° angular resolution), identifying the most-populated bin, and computing the mean of all normals in that bin to obtain the seed normal $\hat{\mathbf{n}}_s$.
2. Compute the dot product between the seed normal $\hat{\mathbf{n}}_s$ and every remaining unassigned valid normal $\hat{\mathbf{n}}_i$. Assign every pixel where $\hat{\mathbf{n}}_s \cdot \hat{\mathbf{n}}_i \geq \tau_n$ to the current cluster, with $\tau_n = 0.85$ (corresponding to an angular tolerance of approximately 31.8°).
3. Compute the mean of the assigned normals and re-normalise it to unit length to obtain the cluster representative normal $\hat{\mathbf{n}}_c$. Remove all assigned pixels from the candidate set.
4. Repeat from step 1 until the candidate set is empty or 30 clusters have been formed.
5. Discard clusters covering fewer than 1.4% of image pixels. This threshold prevents very small clusters from being treated as meaningful plane instances.
6. Label each remaining cluster as a floor cluster if the y -component of its representative normal satisfies $(\hat{\mathbf{n}}_c)_y < -0.25$ (pointing sufficiently upward), and as a wall cluster otherwise.

Figure 3.8 shows the clustering result for the Lunch Room Tables scene.

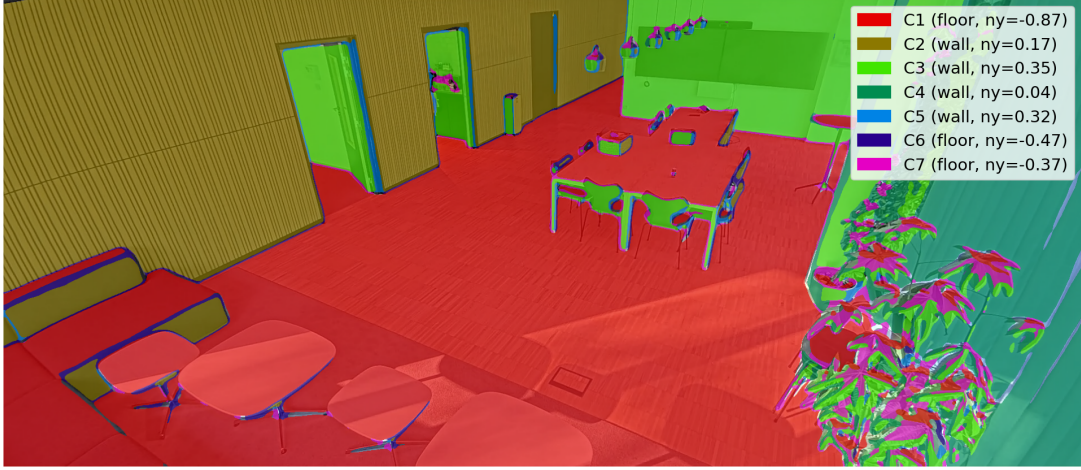


Figure 3.8 Greedy normal clustering result. Each cluster is shown in a distinct colour. The legend states whether the cluster is labelled floor ($n_y < -0.25$) or wall and its mean normal y -component. This scene produces seven clusters, three floor and four wall clusters.

Depth Histogram Splitting

A normal cluster produced in the previous step may contain multiple physically distinct surfaces that share approximately the same orientation but sit at different depths. A floor and a tabletop, for example, both produce upward-facing normals but are separate surfaces. The key geometric observation is that all points on a plane with normal $\hat{\mathbf{n}}_c$ yield the same value when projected onto that normal. For any 3D point $\mathbf{p}_i \in \mathbb{R}^3$, where $\mathbf{p}_i$ is the position vector from the camera origin to the surface location as estimated by MoGe’s point cloud, the scalar projection

$$d_i = \mathbf{p}_i \cdot \hat{\mathbf{n}}_c$$

is constant for all points on the same plane. This constant equals the perpendicular distance from the camera origin to the plane along $\hat{\mathbf{n}}_c$. Two co-normal surfaces at different depths therefore produce two distinct peaks in the distribution of $\{d_i\}$, and splitting at the valleys between those peaks separates the cluster into depth-coherent sub-clusters.

The projected depths $\{d_i\}$ are binned into a histogram (288 bins for floor clusters, 100 bins for wall clusters) and smoothed with a 5-bin moving average to suppress spurious local minima. Representative histograms for a floor cluster and a wall cluster from the example scene are shown in Figure 3.9 and Figure 3.11 respectively. The smooth peaked structure in those figures illustrates why valley-based splitting is well-suited to this distribution. A local minimum at bin v between a left peak at bin l and a right peak at bin r is accepted as a split boundary when both of the following conditions hold. The valley must be sufficiently deep relative to the flanking peaks, satisfying

$$h_v \leq (1 - \rho) \min(h_l, h_r),$$

where h denotes smoothed bin height and ρ is the valley-prominence threshold ($\rho = 0.652$ for floor, $\rho = 0.785$ for wall). The projected-depth distance between the two flanking peaks must also exceed a minimum gap (0.004 m for floor, 0.175 m for wall). After all candidate valleys have been identified, a sequential suppression pass discards any valley whose depth position lies within the minimum gap of the previously accepted valley, preventing a cluster of near-identical splits when a broad shallow region produces multiple adjacent candidate bins. The wall thresholds are set conservatively to bias toward under-segmentation, as it is preferable to keep one wall

as a single instance than to split it incorrectly, and any over-conservative decision is corrected by the coplanar merging step described below. Each accepted valley bin is converted back to a depth boundary value on the d_i axis. Every pixel in the cluster is then assigned to a sub-cluster according to which interval between consecutive boundary values its projected depth d_i falls into. The bin indices are not used for assignment, only the continuous depth values matter.

In the histogram figures, the filled curve is the smoothed histogram on which the split detection operates and the thin line behind it is the raw unsmoothed histogram shown for reference. A red dashed line marks each accepted split boundary. The many fluctuations visible in the raw histogram are largely absent in the smoothed curve, and those that survive smoothing are further filtered by the strict prominence threshold. Figure 3.10 and Figure 3.12 show the corresponding depth-split pixel overlays. The remaining five clusters are not shown but follow the same pattern, and Figure 3.13 shows all instances after splitting has been applied across all seven clusters.

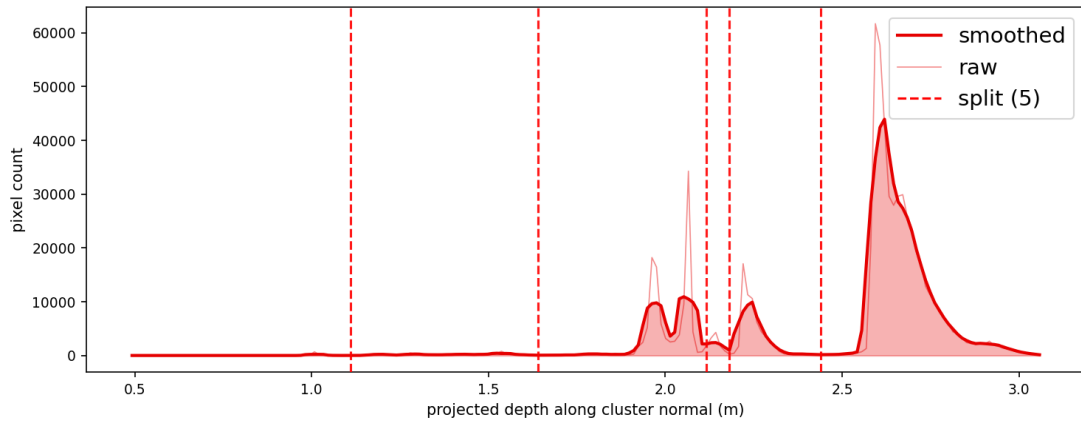


Figure 3.9 Depth histogram: cluster 1 (floor). Red dashed lines mark accepted valley splits where a local minimum in the smoothed histogram passes the prominence threshold ($\rho = 0.652$, valley below 34.8% of the smaller flanking peak) and the minimum depth gap condition (0.004 m).

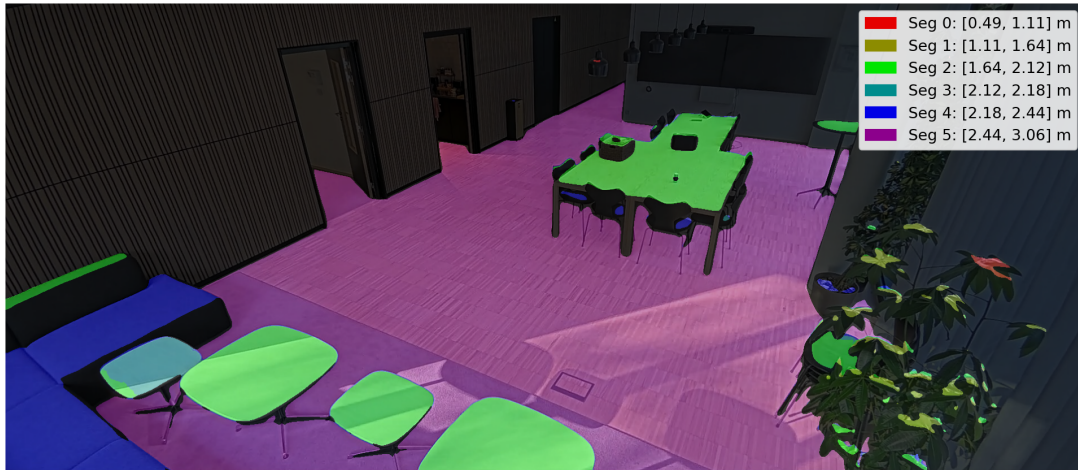


Figure 3.10 Depth-split overlay: cluster 1 (floor). Only this cluster's pixels are shown at full brightness. All other pixels are dimmed. Each distinct colour corresponds to one depth segment. The legend gives the projected-depth range of each segment.

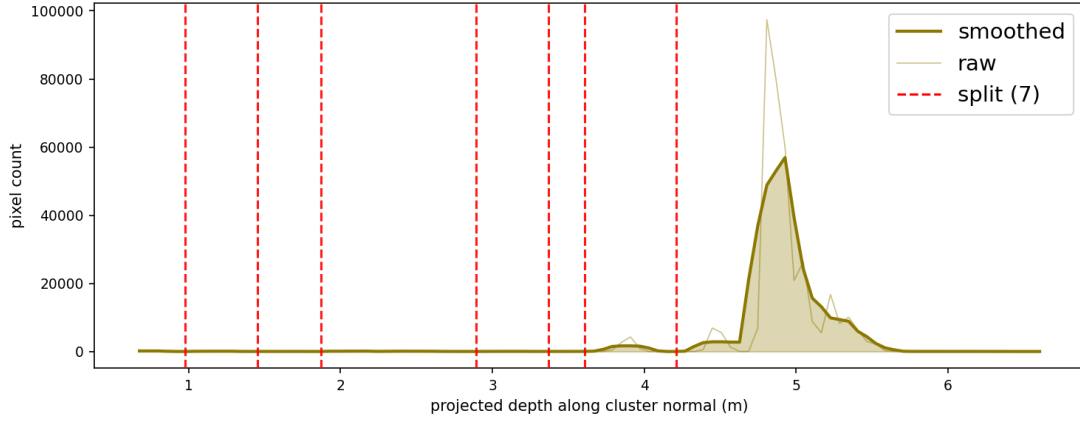


Figure 3.11 Depth histogram: cluster 2 (wall). Red dashed lines mark accepted valley splits where a local minimum in the smoothed histogram passes the prominence threshold ($\rho = 0.785$, valley below 21.5% of the smaller flanking peak) and the minimum depth gap condition (0.175 m).



Figure 3.12 Depth-split overlay: cluster 2 (wall). Only this cluster's pixels are shown at full brightness. All other pixels are dimmed. Each distinct colour corresponds to one depth segment. The legend gives the projected-depth range of each segment.

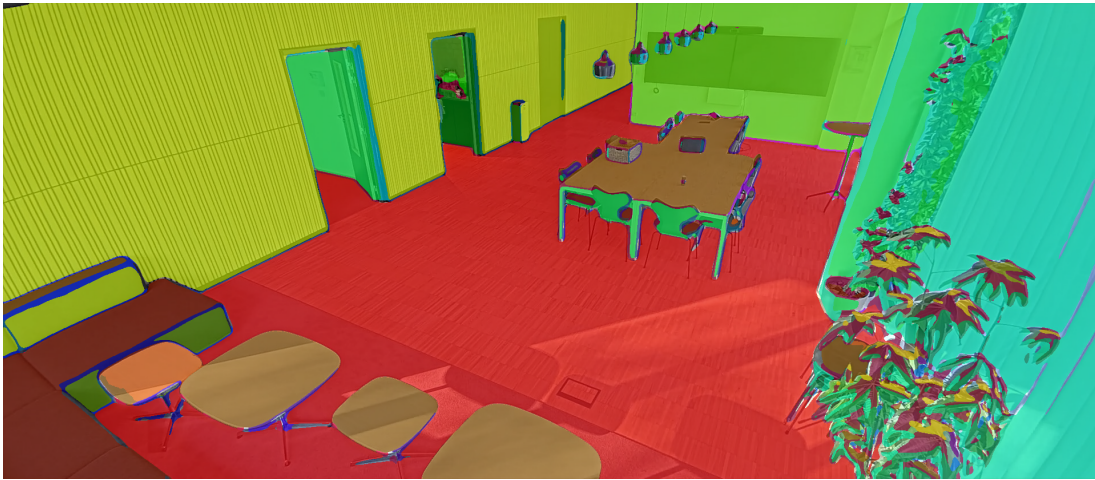


Figure 3.13 Instances after depth histogram splitting. Sub-clusters separated along the projected depth axis each receive a new distinct colour.

Spatial Connectivity Splitting

A depth-split sub-cluster may still cover two spatially disjoint regions, such as the same wall segment appearing on either side of a doorway. Each wall sub-cluster is split into its connected components by running connected-component labelling directly on the original cluster mask, with no morphological pre-processing. Any fragments that belong to the same physical surface but are separated by thin gaps in the mask are recovered in the subsequent coplanar merging step rather than by pre-emptively bridging the gaps here. Connected components covering fewer than 0.13% of image pixels are discarded. Floor sub-clusters are not connectivity-split because the floor tends to appear as one large connected region. Figure 3.14 shows the result after connectivity splitting. In this image, the sofa back on the left shares a similar normal direction with the adjacent wall and was grouped into the same normal cluster by the previous step, so connectivity splitting separates it from the wall as a distinct instance. The lower part of the sofa, which formed a single segment after histogram splitting, is further divided into two fragments here because a small table interrupts the mask and breaks the spatial continuity between the left and right sections.

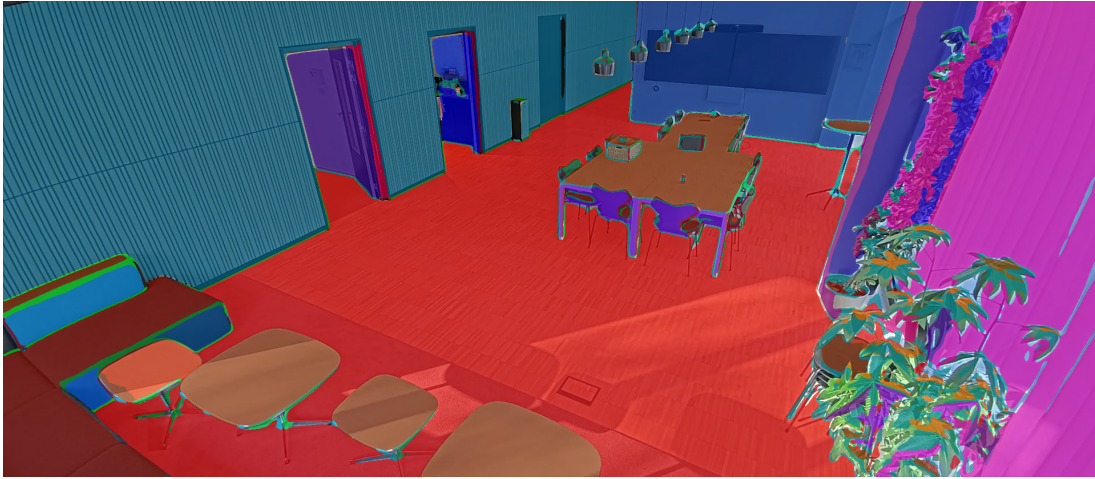


Figure 3.14 Instances after spatial connectivity splitting (wall clusters only). Disjoint regions that survived histogram splitting as one instance are now separated.

Coplanar Wall Fragment Merging

Connectivity splitting can over-segment a wall whenever the normal cluster mask contains thin gaps caused by depth-estimation noise or occluders, producing two fragments that should be part of the same surface. These fragments are re-merged by comparing every pair of wall instances that share the same normal cluster. Two instances are merged when all three conditions hold. Their median projected depths, computed fresh from the 3D point cloud as $\tilde{d} = \text{median}_i(\mathbf{p}_i \cdot \hat{\mathbf{n}}_c)$, differ by at most 0.323 m. The vertical gap between their axis-aligned bounding boxes does not exceed 180 px. The horizontal gap does not exceed 24 px.

The depth test uses a per-instance median of the raw 3D projections rather than the histogram-derived depth centre because it is independent of binning artefacts and reflects the actual geometry of each fragment. The asymmetric gap thresholds reflect the asymmetry of the problem, a wall is more likely to be interrupted vertically by a horizontal occluder like a shelf or a window sill than horizontally by a narrow vertical element such as a doorframe. A generous vertical threshold handles the common case without also merging genuinely distinct walls at

similar depths. The small horizontal threshold prevents two walls meeting at a corner from being incorrectly merged. When multiple fragments satisfy the conditions, they are grouped with union-find and the largest fragment in each group absorbs the others. In the Lunch Room Tables scene used throughout this section the step produces no visible change, because no wall was split into disconnected fragments by the previous step. The step is important in scenes where a horizontal occluder such as a staircase interrupts a wall, splitting it into a top and bottom fragment. Image ai_054_007/c1/f53 in Figure 4.4c is one such case, where the right-hand wall is partially occluded by some stairs and would remain as two separate instances without this merging step.

Instance Classification

The normal-direction criterion applied in the clustering step was used to route instances through separate processing paths, not to assign final labels. Classification assigns those labels and proceeds in two phases. In the first phase, a reference floor normal $\hat{\mathbf{n}}_f$ is estimated by taking a weighted mean of the normals of all instances for which more than 60% of pixels carry a floor-type ADE20K label, weighted by pixel count. Each instance is then bucketed by the cosine similarity of its mean unit normal to $\hat{\mathbf{n}}_f$. An instance is a floor candidate if $|\hat{\mathbf{n}}_c \cdot \hat{\mathbf{n}}_f| \geq 0.82$ (within roughly 35° of the floor plane), and a wall candidate otherwise.

In the second phase, each bucket is filtered by a combination of semantic voting and geometric checks. A floor candidate is accepted if its plurality ADE20K label belongs to the floor class (floor, ground, earth, rug, stairs, sidewalk), at least 50% of its pixels carry a floor-class label, and it covers at least 0.3% of image pixels. The two semantic conditions are not equivalent. The plurality condition can pass with less than 50% floor coverage when non-floor labels are split across many classes, while the 50% condition can fail even when a floor-type label is the most common single label. Together they require that a floor-type label both wins the modal vote and forms an outright majority of pixels. A wall candidate must cover at least 0.15% of image pixels, have a perimeter-to- $\sqrt{\text{area}}$ ratio of at most 25 (to reject fragmented or patchy instances), and a plurality ADE20K label that belongs to the set of wall-like classes (wall, window, door, curtain, mirror, painting, and related surfaces). No equivalent pixel-fraction threshold is imposed because the wall-like set is broad enough that a genuine wall instance is very likely to carry a wall-like plurality label even under label bleed, making the additional condition redundant. Finally, any wall instance smaller than 7.2% of the largest accepted wall in the image is dropped. All other components are labelled as other and excluded from the output. Figure 3.15 shows the classification result for the example image.

Table 3.2 lists all numerical parameters of the algorithm together with their values. The values were arrived at through a combination of manual experimentation and Optuna [31] optimisation, as described in section 3.8.

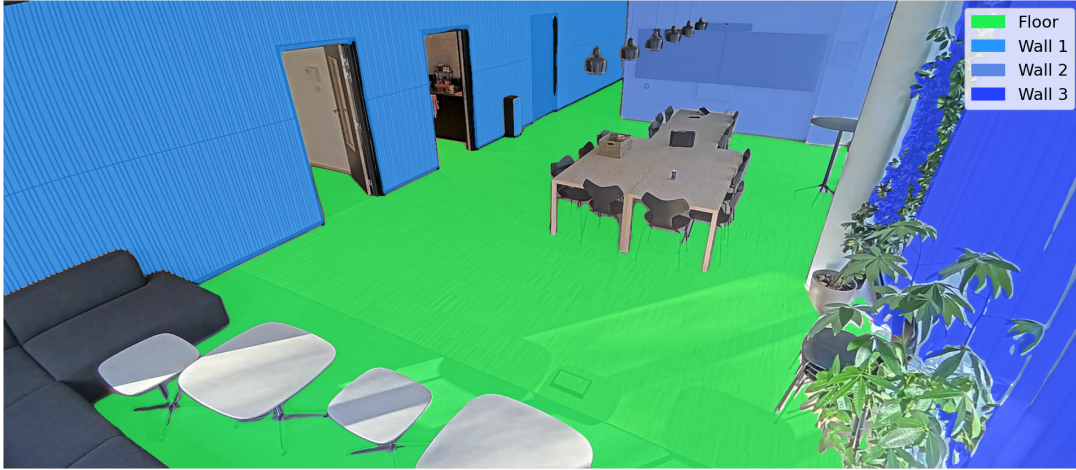


Figure 3.15 Instances after classification. Green = floor, blue shades = individual wall instances, grey = rejected as other.

Table 3.2 All numerical parameters of the plane segmentation algorithm. Values were determined through manual experimentation and Optuna optimisation.

Step	Parameter	Value
Normal denoising	Gaussian kernel size (px)	13
Normal clustering	Normal similarity threshold τ_n	0.85
	Minimum cluster size	1.4% of pixels
	Maximum number of clusters	30
	Floor/wall normal split threshold (n_y)	-0.25
Depth histogram splitting	Number of bins (floor)	288
	Valley prominence ρ (floor)	0.652
	Minimum peak separation (floor)	0.004 m
	Number of bins (wall)	100
	Valley prominence ρ (wall)	0.785
	Minimum peak separation (wall)	0.175 m
Connectivity splitting	Smoothing window	5 bins
	Minimum component size	0.13% of pixels
Coplanar wall merging	Maximum depth difference	0.323 m
	Maximum vertical bounding-box gap	180 px
	Maximum horizontal bounding-box gap	24 px
Instance classification	Floor seed: minimum floor label fraction	60%
	Floor candidate: cosine similarity threshold	0.82
	Floor acceptance: minimum pixel fraction	0.3%
	Floor acceptance: minimum floor-label majority	50%
	Wall candidate: minimum pixel fraction	0.15%
	Wall candidate: maximum perimeter ratio	25
	Wall acceptance: minimum relative size	7.2%

3.7 Evaluation Protocol

For each image, two metrics are computed separately for floor and wall, giving four scores per image in total.

Per-instance best-match IoU (floor / wall). For every ground-truth instance of the class, the algorithm finds the predicted instance with the highest pixel-level intersection-over-union (IoU). If no predicted instance overlaps the ground-truth region, the score for that instance is zero. Instance scores are then aggregated into a class mean weighted by each instance’s pixel count as a fraction of the total pixels belonging to that class in the scene. This pixel-weighted mean prevents a single very small instance from disproportionately penalising the score. A missed instance that covers only 2% of the wall pixels contributes far less than a missed instance covering 40%.

Per-class IoU (floor / wall). All ground-truth instances of the class are merged into a single binary mask, and all predicted instances are merged likewise. The IoU of these two merged masks gives a single class-level score that measures overall pixel coverage without regard to instance boundaries.

The two metrics capture different aspects of performance and can differ. A system that correctly covers all wall pixels but merges two distinct walls into a single prediction scores well on per-class IoU but poorly on per-instance IoU, because each ground-truth instance is matched against a prediction that also covers the other wall, reducing the overlap ratio. Conversely, a system that predicts each ground-truth wall instance accurately but also classifies the ceiling as a wall scores well on per-instance IoU but poorly on per-class IoU. Because no ceiling region is annotated as a wall instance, the ceiling prediction is never matched against any ground-truth instance and therefore cannot harm the per-instance score. However, the per-class score is hurt because the merged prediction mask now includes the ceiling pixels which inflates the union without adding to the intersection. Neither metric is sufficient on its own, so they should be read together.

3.8 Parameter Optimisation

The algorithm has several numerical parameters. Optuna was used throughout development to guide their values. Parameters related to normal denoising and normal clustering were set early in development, primarily through experimentation on the six real images before the synthetic images were added to the dataset, and were not revisited in the final Optuna studies. The remaining parameters were tuned more systematically with Optuna on all 34 images, though the degree of tuning effort varied across parameters and across different stages of the project. A useful property of this workflow is that as tuning progresses it becomes clear which parameters consistently settle in a narrow region regardless of how others vary. Those can be left unchanged in subsequent studies without meaningfully limiting performance, which reduces the search space and makes later studies faster.

Because floor and wall have separate scores and largely separate parameter sets, they can be tuned in independent studies each using only the score for its own class as the objective. This avoids the problem that arises from a single combined objective, where a trial that improves wall performance may incidentally shift floor parameters that were already working well.

Each study used a Tree-structured Parzen Estimator (TPE) sampler run against all 34 annotated images with MoGe ViT-L and Mask2Former Swin-Large. The wall study maximised

the mean of the wall class IoU and the pixel-weighted wall instance IoU across all 34 images, with floor parameters held fixed. The floor study maximised the corresponding floor scores with wall parameters held fixed. Trial results were cached by parameter hash in a SQLite database so that an interrupted study could resume without repeating completed evaluations.

3.9 Comparison Variants

Two alternative configurations were evaluated alongside the main algorithm to isolate the contributions of specific stages. Both keep the complete geometric pipeline (normal denoising, greedy clustering, connectivity splitting, and coplanar wall merging) unchanged. One replaces the depth-splitting step and the other replaces the classification step.

RANSAC Depth Splitting

The depth histogram valley detection step was replaced with iterative RANSAC depth fitting applied to the one-dimensional projected depth values $\{d_i\} = \{\mathbf{p}_i \cdot \hat{\mathbf{n}}_c\}$. For each cluster, RANSAC samples one projected depth value at random, identifies inlier points within a distance tolerance, refits the plane depth as the inlier mean, and accepts the plane if the inlier count reaches a minimum fraction of the cluster. Accepted inliers are removed and the procedure repeats until no plane passes the acceptance criterion or a maximum plane count is reached. Pixels not assigned to any accepted plane are allocated to the nearest one. If no plane was accepted the entire cluster forms one instance.

Floor and wall clusters each have separate inlier threshold and minimum inlier fraction parameters. These were tuned with a balanced objective in an 80-trial Optuna study, averaging the floor class IoU, wall class IoU, floor instance IoU, and wall instance IoU across all 34 images. The trial count was considered sufficient because the best objective value had not improved for the final third of trials, indicating that the search had converged.

No-Segmentation Ablation

The no-segmentation variant replaces the semantic voting step in instance classification with geometry-only rules. Mask2Former output is not used. An instance is routed to the floor candidate set if its cluster was identified as floor-oriented during splitting or if its mean unit normal satisfies $n_y < -0.25$. Instances with $n_y > 0.95$ are treated as ceiling-like and discarded. All remaining instances are accepted as wall, subject to the same minimum size and perimeter ratio thresholds as in the main method.

The geometric pipeline is identical to the main method, so this ablation measures the contribution of semantic labels to correct class assignment independently of the instance boundaries. No separate parameter tuning was performed for this variant and it uses the same geometric parameter values as the main method.

4

Results

This chapter reports how the pipeline performed across all 34 images and all twelve backbone combinations. It covers per-image segmentation accuracy, the effect of backbone size on floor and wall detection, inference and algorithm runtimes, the impact of letterbox removal, a detailed analysis of the images where performance was lowest, and a comparison against two alternative variants, RANSAC-based depth splitting and a no-segmentation ablation that removes semantic labelling entirely.

4.1 Plane Segmentation Results

All results in this section use the ViT-L + Swin-Large combination, which achieves the highest wall class IoU across the evaluation grid (see section 4.2). Floor class IoU averages 0.945 and wall class IoU averages 0.854 across all 34 images. The corresponding instance metrics are 0.945 (floor) and 0.853 (wall). Table 4.1 gives the full per-image breakdown.

Real images outperform synthetic ones on both metrics. The 6 real images average 0.975 floor class IoU and 0.900 wall class IoU. Floor detection is consistent across all six, with the lowest value being 0.961 (Lunch Room Tables). Wall performance varies more, from 0.810 (Garage) to 0.974 (Kitchen).

The 28 synthetic images average 0.938 floor class IoU and 0.844 wall class IoU. Most images achieve wall class IoU above 0.80, with 20 of the 28 falling above this threshold. The remaining 8 images spread from 0.48 to 0.76, with two below 0.65 (see section 4.6). Floor detection on the synthetic images is generally stable. Of the 27 images with a visible floor, 25 achieve floor class IoU above 0.85. The two exceptions are ai_054_007/c1/f53 (0.785) and ai_054_010 (0.838).

Figure 4.1 through Figure 4.3 illustrate example outputs for three images that achieved good results. These are Kitchen, which is a real image with floor-cls 0.971 and wall-cls 0.974, ai_001_005/c1/f0, which is a synthetic image with floor-cls 0.927 and wall-cls 0.920, and ai_005_003/c0/f37, which is a synthetic image with floor-cls 0.951 and wall-cls 0.845.

For each image, the six subfigures are ordered left to right and top to bottom as semantic segmentation, ground truth, prediction, floor coverage, wall coverage, and false predictions. In the semantic, ground-truth, and prediction panels, floor instances are shown in green hues and wall instances in blue hues. Shade variation is used only to separate neighbouring instances of the same class.

In floor coverage, green marks GT floor pixels that are recovered by the prediction, while red marks GT floor pixels that are missed. In wall coverage, sky blue marks GT wall pixels that are recovered, while magenta marks GT wall pixels that are missed. In false predictions, magenta denotes predicted floor on GT wall, yellow denotes predicted floor on unlabeled pixels,

red denotes predicted wall on GT floor, and cyan denotes predicted wall on unlabeled pixels. Non-highlighted image regions are darkened.

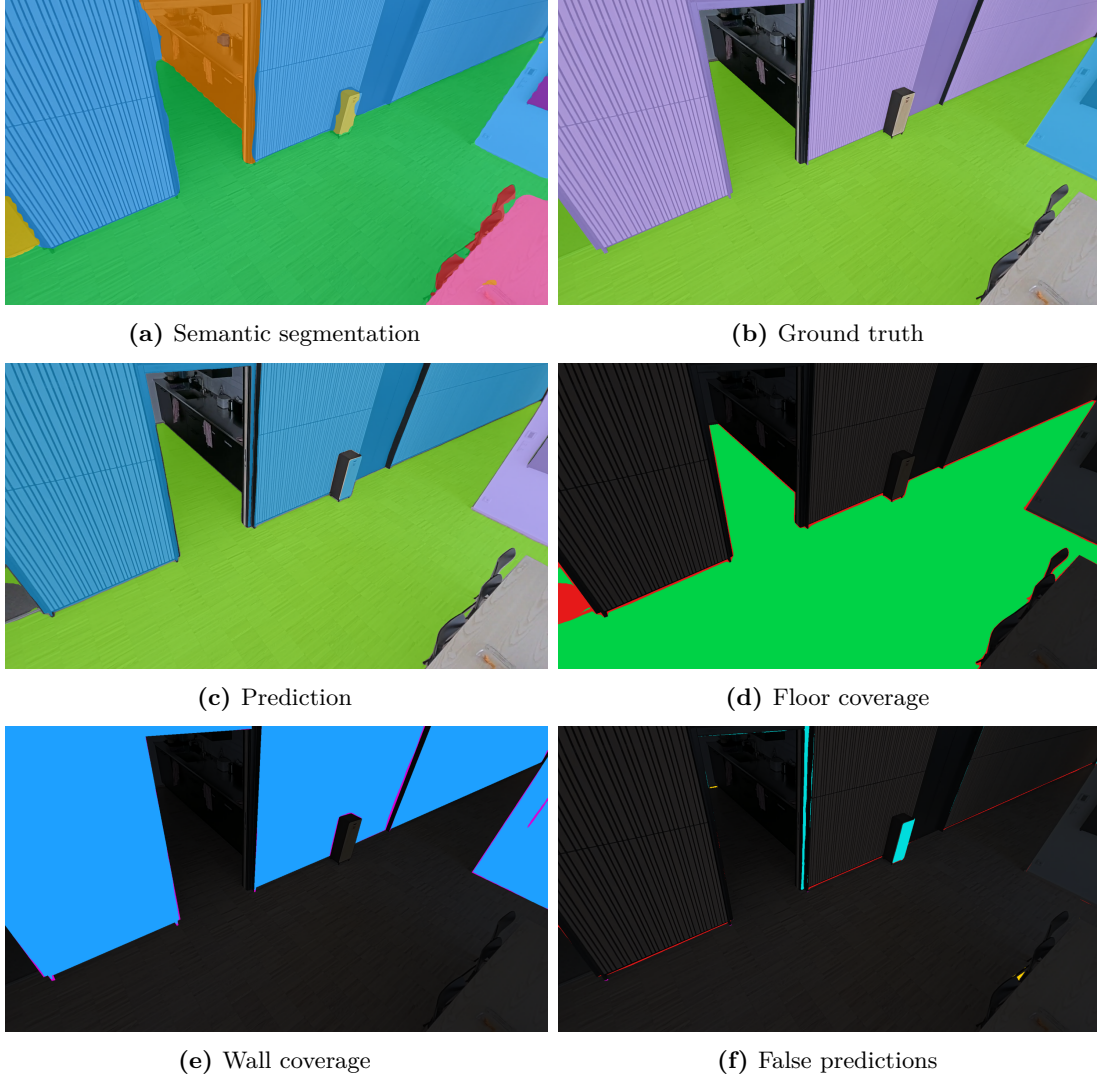


Figure 4.1 Kitchen qualitative panels (floor-cls 0.971, wall-cls 0.974).



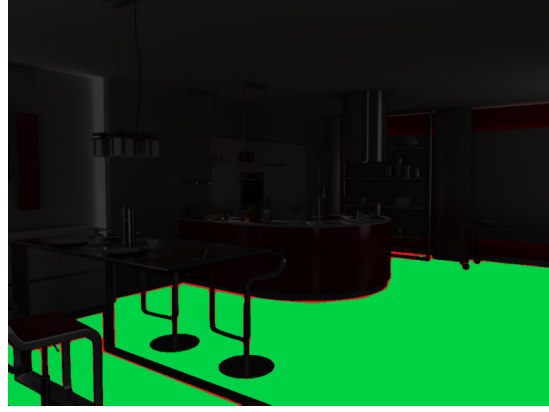
(a) Semantic segmentation



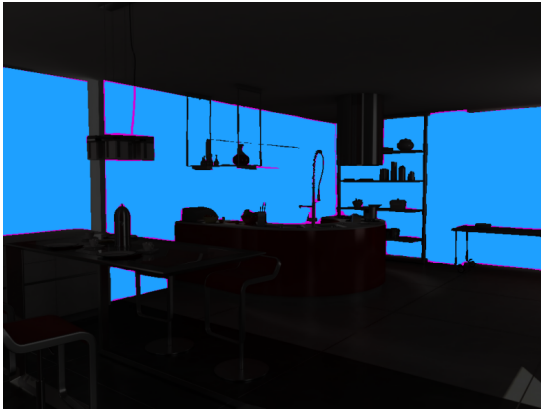
(b) Ground truth



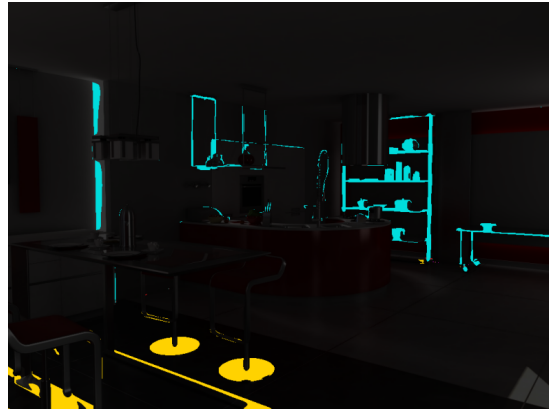
(c) Prediction



(d) Floor coverage



(e) Wall coverage



(f) False predictions

Figure 4.2 ai_001_005/c1/f0 qualitative panels (floor-cls 0.927, wall-cls 0.920).

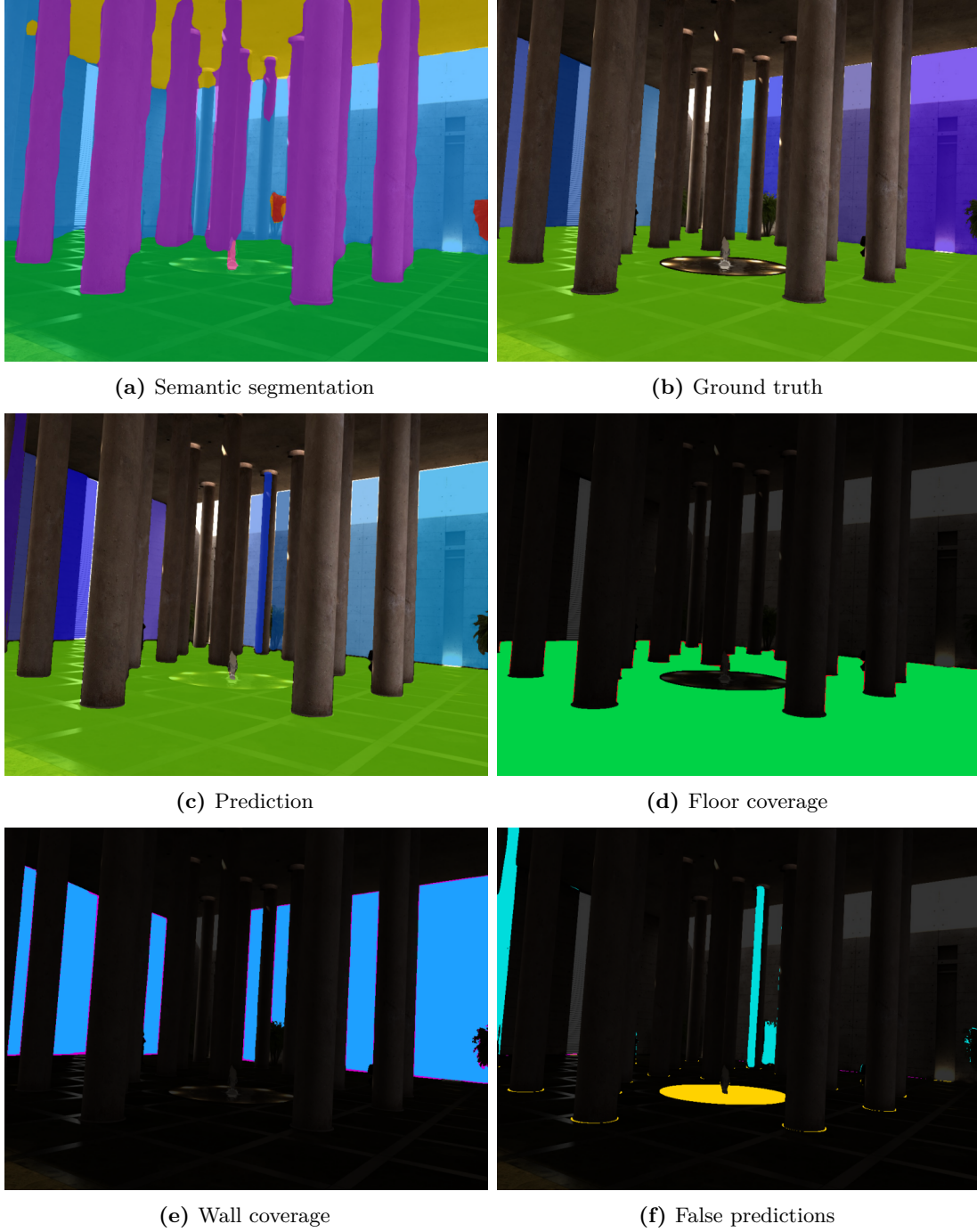


Figure 4.3 ai_005_003/c0/f37 qualitative panels (floor-cls 0.951, wall-cls 0.845).

Table 4.1 Per-image results for all 34 images (ViT-L + Swin-Large). N/A indicates no visible floor.

Image	Floor-cls	Wall-cls	Floor-inst	Wall-inst
<i>Real images</i>				
Coffee lounge	0.972	0.904	0.972	0.905
Garage	0.990	0.810	0.990	0.655
Kitchen	0.971	0.974	0.971	0.974
LR Corridor	0.971	0.880	0.971	0.884
LR Tables	0.961	0.903	0.961	0.907
Middle office	0.983	0.931	0.983	0.932
<i>Real mean</i>	0.975	0.900	0.975	0.876
<i>Synthetic images</i>				
ai_001_001/c0/f12	0.961	0.878	0.961	0.879
ai_001_001/c0/f19	0.992	0.954	0.992	0.953
ai_001_001/c0/f29	0.911	0.899	0.911	0.928
ai_001_001/c0/f34	0.995	0.944	0.995	0.945
ai_001_003/c0/f0	0.960	0.891	0.960	0.732
ai_001_003/c0/f69	0.956	0.978	0.956	0.978
ai_001_005/c0/f0	0.964	0.764	0.964	0.643
ai_001_005/c1/f0	0.927	0.920	0.927	0.919
ai_001_005/c2/f19	0.885	0.950	0.885	0.950
ai_001_005/c2/f69	0.957	0.853	0.957	0.924
ai_001_006/c1/f33	0.994	0.950	0.994	0.950
ai_001_007/c0/f0	0.944	0.975	0.944	0.975
ai_001_008/c0/f27	0.955	0.954	0.955	0.954
ai_001_010/c0/f12	0.952	0.889	0.952	0.889
ai_001_010/c0/f29	N/A	0.481	N/A	0.437
ai_002_001/c0/f48	0.963	0.883	0.963	0.858
ai_002_003/c0/f0	0.906	0.718	0.906	0.713
ai_003_002/c1/f27	0.980	0.939	0.980	0.941
ai_003_009/c0/f1	0.984	0.845	0.984	0.873
ai_003_010/c0/f18	0.985	0.663	0.985	0.858
ai_005_003/c0/f37	0.951	0.845	0.951	0.946
ai_005_004/c0/f5	0.952	0.749	0.952	0.813
ai_008_010/c0/f7	0.958	0.563	0.958	0.660
ai_009_003/c0/f0	0.870	0.755	0.870	0.742
ai_053_002/c0/f33	0.933	0.927	0.933	0.927
ai_054_007/c0/f7	0.876	0.718	0.888	0.775
ai_054_007/c1/f53	0.785	0.870	0.785	0.868
ai_054_010/c0/f39	0.838	0.885	0.838	0.705
<i>Synthetic mean</i>	0.938	0.844	0.939	0.848
Overall mean	0.945	0.854	0.945	0.853

4.2 Effect of Model Combination

Table 4.2 reports floor and wall metrics averaged over all 34 images for all twelve backbone combinations. The parameters were tuned on ViT-L + Swin-Large as described in section 3.8, so results for smaller combinations are conservative lower bounds.

Floor class IoU is stable across the full grid, ranging from 0.910 to 0.945, and is largely unaffected by backbone choice on either axis. Wall class IoU is more sensitive, spanning 0.786 to 0.854.

Fixing Swin-Large and stepping down MoGe gives wall class IoU of 0.854 (ViT-L), 0.844 (ViT-B), and 0.827 (ViT-S), a total drop of 0.027. Fixing ViT-L and stepping down Mask2Former gives 0.854 (Swin-L), 0.825 (Swin-T), 0.822 (Swin-S), and 0.810 (Swin-B), a total drop of 0.044. The Mask2Former axis is front-loaded, with the initial Swin-Large to Swin-Tiny step costing 0.029 wall IoU points while the subsequent Swin-Tiny to Swin-Small transition adds only 0.003. The MoGe axis degrades more gradually, at 0.010 from ViT-L to ViT-B and 0.017 from ViT-B to ViT-S.

Table 4.2 Mean floor and wall metrics over all 34 images for each backbone combination. Floor-cls and Wall-cls are binary class IoU. Floor-inst and Wall-inst are pixel-weighted best-match instance IoU. Best value in each column is in bold.

MoGe	M2F	Floor-cls	Wall-cls	Floor-inst	Wall-inst
ViT-L	Swin-L	0.945	0.854	0.945	0.853
ViT-L	Swin-T	0.943	0.825	0.943	0.845
ViT-L	Swin-S	0.945	0.822	0.945	0.854
ViT-L	Swin-B	0.914	0.810	0.920	0.839
ViT-B	Swin-L	0.941	0.844	0.942	0.829
ViT-B	Swin-T	0.941	0.818	0.940	0.824
ViT-B	Swin-S	0.942	0.818	0.942	0.831
ViT-B	Swin-B	0.938	0.806	0.942	0.817
ViT-S	Swin-L	0.934	0.827	0.934	0.821
ViT-S	Swin-T	0.932	0.790	0.931	0.804
ViT-S	Swin-S	0.922	0.789	0.920	0.815
ViT-S	Swin-B	0.910	0.786	0.912	0.809

4.3 Model Inference Time

Table 4.3 reports Stage 1 inference time per image for each backbone combination, measured on a representative 1920×827 image.

Table 4.3 Stage 1 inference time per image on an NVIDIA GeForce RTX 5090 (1920×827 px, 10-run mean). Stage 1 total = MoGe time + Mask2Former time for that combination.

MoGe	M2F	MoGe (ms)	M2F (ms)	Total (ms)
ViT-L	Swin-L	42.6	21.0	63.6
ViT-L	Swin-T	42.6	12.2	54.8
ViT-L	Swin-S	42.6	15.5	58.1
ViT-L	Swin-B	42.6	16.8	59.4
ViT-B	Swin-L	25.8	21.0	46.8
ViT-B	Swin-T	25.8	12.2	38.0
ViT-B	Swin-S	25.8	15.5	41.3
ViT-B	Swin-B	25.8	16.8	42.6
ViT-S	Swin-L	21.1	21.0	42.1
ViT-S	Swin-T	21.1	12.2	33.3
ViT-S	Swin-S	21.1	15.5	36.6
ViT-S	Swin-B	21.1	16.8	37.9

The largest combination (ViT-L + Swin-L) processes one image in 63.6 ms. The smallest (ViT-S + Swin-T) reduces this to 33.3 ms, a factor of roughly two. MoGe ViT-L accounts for 67% of the largest combination’s total time. Reducing to ViT-S saves 21 ms while ViT-B saves 17 ms. Replacing Swin-Large with Swin-Tiny saves 8.8 ms. These timings cover Stage 1 only.

4.4 Algorithm Runtime

Table 4.4 reports the Stage 2 segmentation algorithm runtime per image for each backbone combination, averaged across all 34 images on an AMD Ryzen 9 9900X.

Runtime is largely independent of backbone choice, ranging from 374 ms to 403 ms across all twelve combinations. The algorithm operates on the cached normal map, point cloud, and segmentation labels, all of which have the same spatial resolution regardless of which model produced them. A consistent gap of roughly 25 ms separates the ViT-L tier (396–403 ms) from the ViT-S tier (373–378 ms), despite identical input sizes. The cause and its implications are discussed in Chapter 5. The high standard deviation (133–170 ms across images) reflects differences in scene complexity and input image size.

Table 4.4 Stage 2 algorithm runtime per image on an AMD Ryzen 9 9900X (3 timed runs per image after 1 warmup, mean $\pm$ std across all 34 images).

MoGe	M2F	Mean (ms)	Std (ms)
ViT-L	Swin-L	399	162
ViT-L	Swin-T	403	168
ViT-L	Swin-S	402	166
ViT-L	Swin-B	396	164
ViT-B	Swin-L	386	165
ViT-B	Swin-T	398	170
ViT-B	Swin-S	393	168
ViT-B	Swin-B	389	167
ViT-S	Swin-L	378	138
ViT-S	Swin-T	376	133
ViT-S	Swin-S	373	136
ViT-S	Swin-B	375	134

4.5 Letterbox Effect

The Axis cameras used for the real-image captures apply internal barrel-distortion correction, which leaves black letterbox borders along the top and bottom of the image. When the letterbox is retained, MoGe interprets the black borders as scene content, producing erroneous depth and normal estimates near the image edges. To quantify the effect, the full pipeline was run on two versions of each real image, namely the cropped image (letterbox removed, used in all other evaluations) and a letterboxed version. For Coffee and Garage, the letterboxed version is the original uncropped camera output, which retains 253 and 324 pixels of black border respectively. For the remaining four images, which have no meaningful letterbox in their original captures, a synthetic letterbox of 253 pixels (126 top, 127 bottom) was added to the cropped content image to provide a controlled comparison. In all cases the model outputs were cropped back to the content rows before segmentation and evaluation, so the ground-truth annotations required no modification.

This ablation was run on an earlier version of the pipeline and annotation set, so the cropped-condition values in Table 4.5 do not exactly match the corresponding entries in the main results table. The difference is small enough that the comparison between cropped and letterboxed conditions remains valid and the conclusion unaffected.

Table 4.5 reports wall class IoU for both conditions across all three MoGe sizes with Swin-Large fixed.

Table 4.5 Wall class IoU comparing cropped (letterbox removed) and letterboxed inputs across all six real images and all three MoGe sizes (Swin-Large fixed). Coffee and Garage used their original uncropped camera output. The remaining four images had a 253-pixel synthetic letterbox added to the cropped content image.

Image	MoGe	Cropped	Letterboxed
Coffee lounge	ViT-L	0.826	0.778
	ViT-B	0.820	0.669
	ViT-S	0.822	0.653
Garage	ViT-L	0.796	0.548
	ViT-B	0.712	0.515
	ViT-S	0.691	0.488
Kitchen	ViT-L	0.974	0.969
	ViT-B	0.971	0.964
	ViT-S	0.971	0.883
Middle office	ViT-L	0.935	0.780
	ViT-B	0.920	0.893
	ViT-S	0.869	0.907
LR Corridor	ViT-L	0.856	0.857
	ViT-B	0.851	0.806
	ViT-S	0.827	0.832
LR Tables	ViT-L	0.918	0.900
	ViT-B	0.938	0.649
	ViT-S	0.643	0.586

Across the real images, the cropped condition generally performs better than the letterboxed condition for all three backbone sizes. The clearest degradations appear in Coffee and Garage, where the black borders occupy a substantial part of the image and consistently reduce wall class IoU. The four images with a synthetic 253-pixel letterbox show a smaller and less uniform effect, but the overall trend still favours cropping.

The larger backbones also appear more robust to letterbox artifacts than the smaller ones. In several images, the performance drop is larger for ViT-B and especially ViT-S than for ViT-L.

This is consistent with the expectation that larger models are better at discounting irrelevant image regions and focusing on the actual scene content.

4.6 Hard Images

Four images that produce low scores for identifiable structural reasons are examined in Table 4.6 and discussed below.

Table 4.6 Four hard images with their metrics (ViT-L + Swin-Large). N/A indicates no visible floor.

Image	Floor-cls	Wall-cls	Floor-inst	Wall-inst
ai_054_007/c1/f53	0.785	0.870	0.785	0.868
ai_001_010/c0/f29	N/A	0.481	N/A	0.437
ai_008_010/c0/f7	0.958	0.563	0.958	0.660
ai_003_010/c0/f18	0.985	0.663	0.985	0.858

ai_054_007/c1/f53 (floor-cls 0.785, floor-inst 0.785). The sofa and table tops lie close in depth to the floor and are grouped into the same instance as the floor. Because the majority of that instance is floor, the whole instance is classified as floor, so sofa and table top pixels are included in the floor prediction. This misclassification hurts both floor class IoU and floor instance IoU.

ai_001_010/c0/f29 (wall-cls 0.481, wall-inst 0.437). The image contains three annotated wall instances. Two of them are problematic. A large cabinet occupies most of the visible area, and the actual wall behind it sits at a similar projected depth with a similar normal. The algorithm therefore merges these two into a single plane instance. The merged instance contains predominantly cabinet pixels, so the semantic vote returns cabinet and the whole instance is excluded from the wall output. Both the cabinet and the wall behind it are consequently missed. The low score comes from two compounding problems. The depth-based merging fails to separate the cabinet from the wall behind it, and the resulting merged instance is then dominated by cabinet pixels and excluded by the semantic vote. This lowers both wall class IoU and wall instance IoU.

ai_008_010/c0/f7 (wall-cls 0.563, wall-inst 0.660). The image contains a slanted surface that the algorithm classifies as wall, while the ground truth does not annotate it as wall. The resulting false-positive wall pixels lower wall class IoU. The slanted surface is geometrically connected to the adjacent real wall, causing the prediction to merge them into a single instance. This merged instance extends beyond the ground-truth wall boundary, which also reduces wall instance IoU.

ai_003_010/c0/f18 (wall-cls 0.663, wall-inst 0.858). The image contains a slanted surface in the ambiguous region between a wall and a ceiling, which the algorithm classifies as wall while the ground truth does not. The false-positive wall pixels from this surface lower wall class IoU. Unlike the previous image, the slanted surface is not merged with the real upright wall in the prediction, so the predicted wall instances still match their ground-truth counterparts well and wall instance IoU is much less affected.

Figure 4.4 through Figure 4.7 use the same six-panel order and the same colour convention as Figure 4.1 through Figure 4.3, so each hard-image figure can be read directly with the same interpretation.



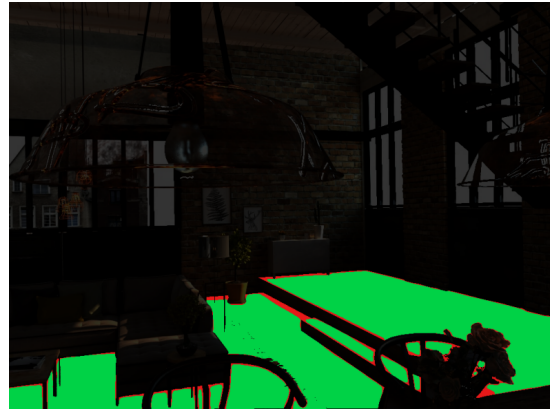
(a) Semantic segmentation



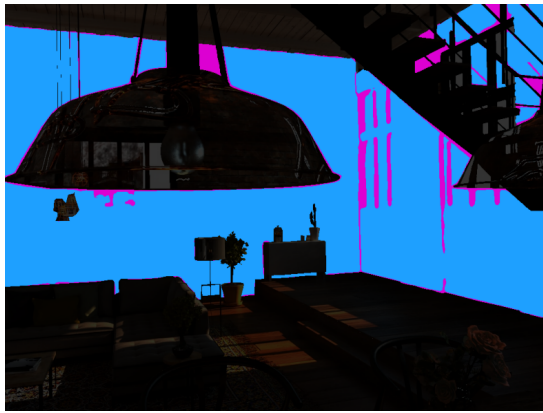
(b) Ground truth



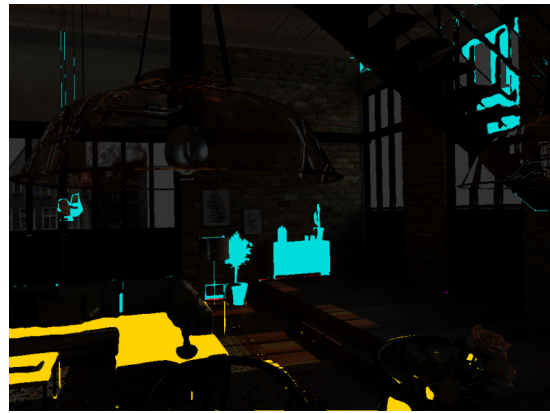
(c) Prediction



(d) Floor coverage

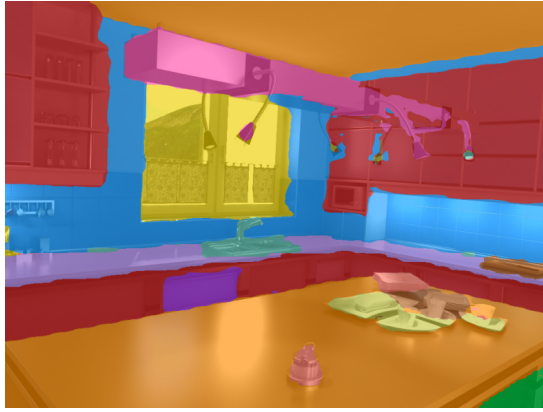


(e) Wall coverage



(f) False predictions

Figure 4.4 ai_054_007/c1/f53 qualitative panels (floor-cls 0.785, wall-cls 0.870).



(a) Semantic segmentation



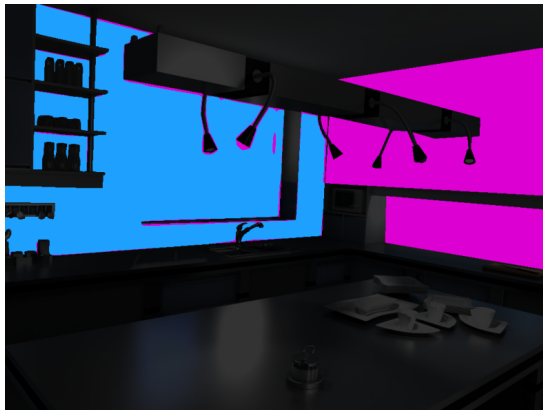
(b) Ground truth



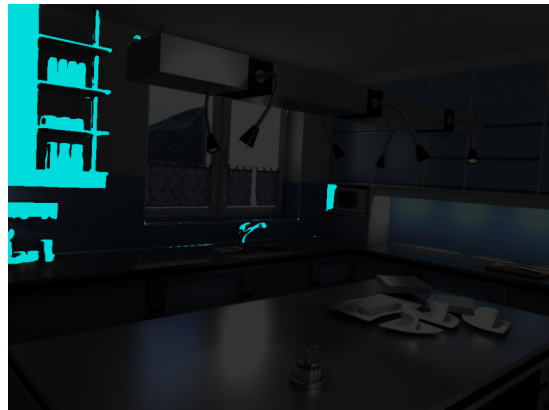
(c) Prediction



(d) Floor coverage



(e) Wall coverage



(f) False predictions

Figure 4.5 ai_001_010/c0/f29 qualitative panels (wall-cls 0.481).

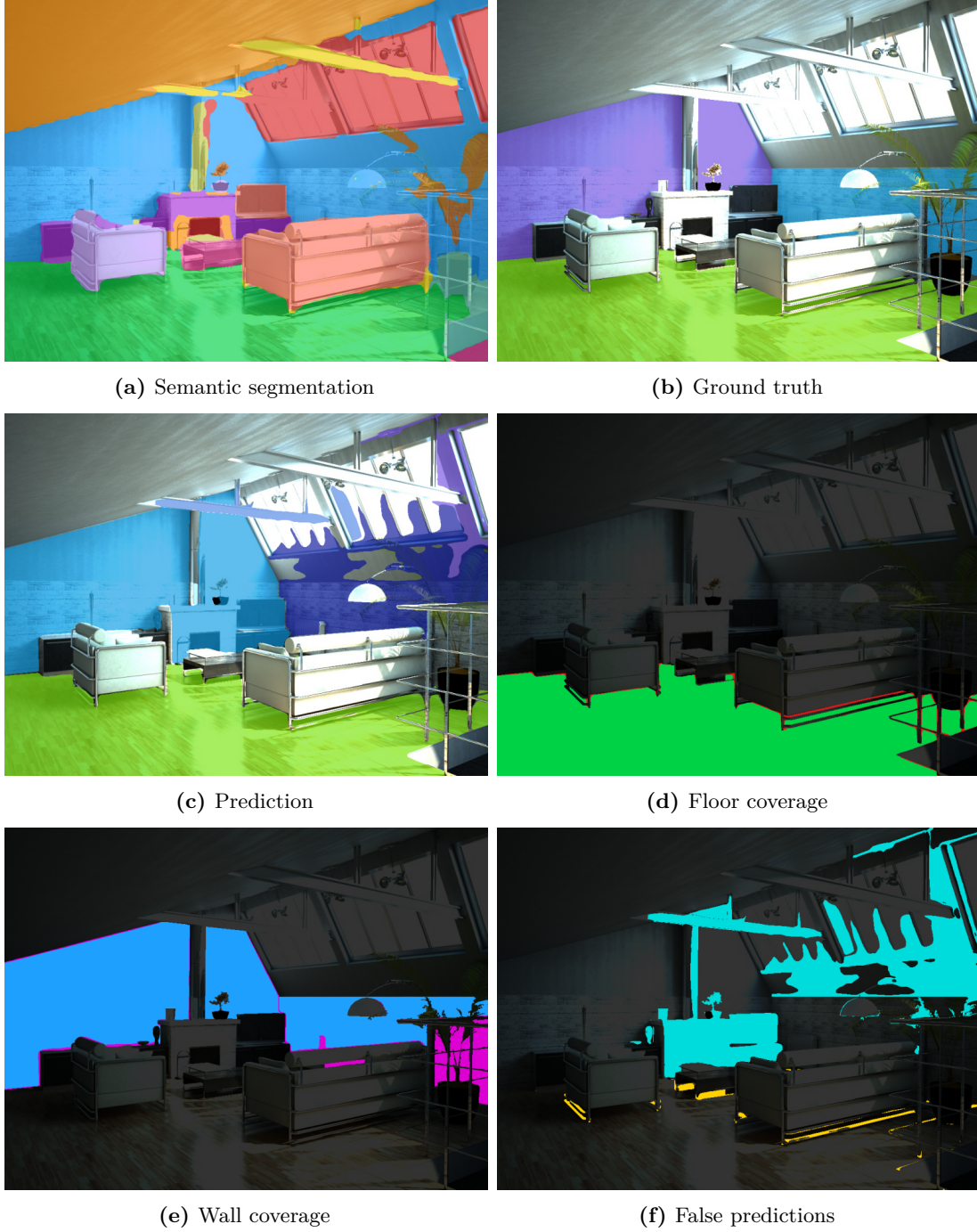
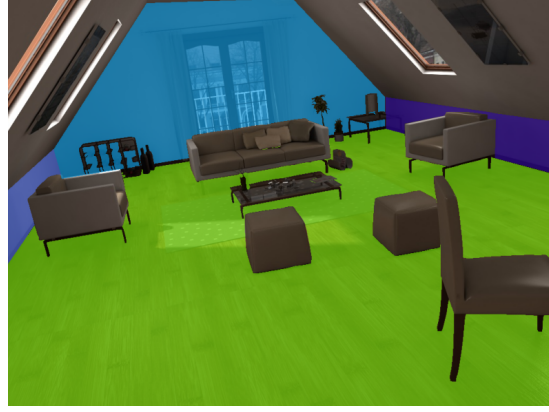


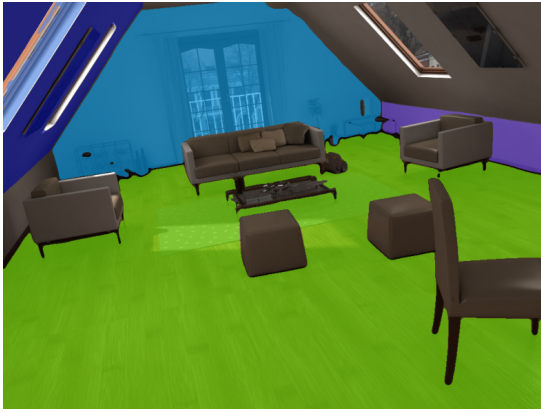
Figure 4.6 ai_008_010/c0/f7 qualitative panels (floor-cls 0.958, wall-cls 0.563).



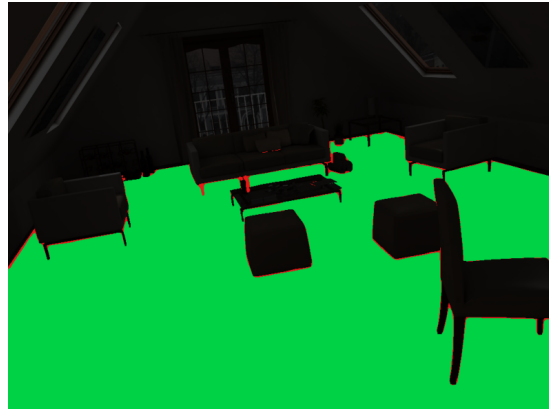
(a) Semantic segmentation



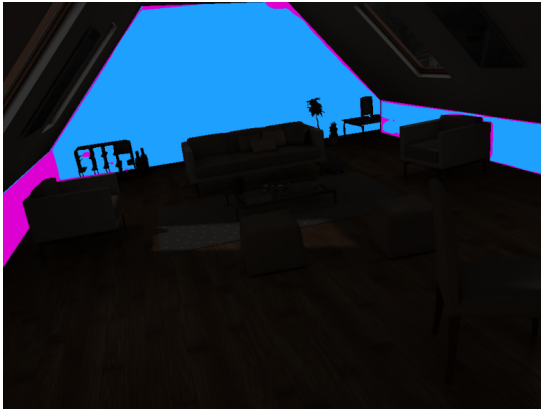
(b) Ground truth



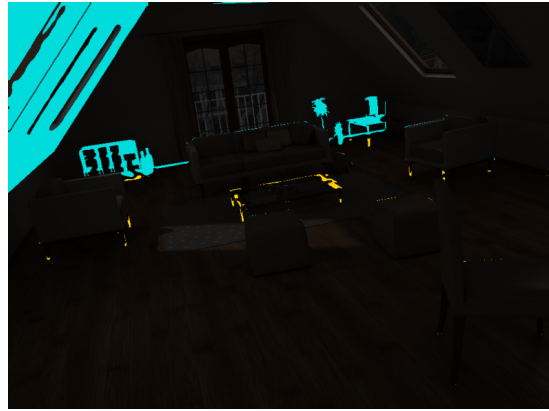
(c) Prediction



(d) Floor coverage



(e) Wall coverage



(f) False predictions

Figure 4.7 ai_003_010/c0/f18 qualitative panels (floor-cls 0.985, wall-cls 0.663).

4.7 Method Comparison

This section reports evaluation results for the RANSAC depth splitting variant and the no-segmentation ablation. Each variant is evaluated separately because the studies differ in scope. The RANSAC variant was run across all 12 model combinations, while the no-segmentation ablation bypasses Mask2Former entirely and produces results independent of the segmentation backbone.

RANSAC Depth Splitting

Table 4.7 reports results for the histogram method and the RANSAC variant averaged over all 34 images for each of the three ViT backbone sizes paired with Swin-Large. Because both methods operate on MoGe’s depth estimates rather than on the segmentation output, the geometry backbone is varied while the segmentation backbone is held fixed at Swin-Large.

Table 4.7 Histogram versus RANSAC averaged over all 34 images (Swin-Large fixed).

MoGe	Method	Floor-cls	Wall-cls	Floor-inst	Wall-inst
ViT-L	Histogram	0.945	0.854	0.945	0.853
	RANSAC	0.932	0.854	0.919	0.824
ViT-B	Histogram	0.941	0.844	0.942	0.829
	RANSAC	0.924	0.835	0.907	0.800
ViT-S	Histogram	0.934	0.827	0.934	0.821
	RANSAC	0.923	0.805	0.904	0.754

The histogram method outperforms the RANSAC variant on all four metrics across all three backbone sizes. The gap is largest on wall instance IoU with ViT-S, where RANSAC falls 0.067 points below the histogram method. Class IoUs are comparatively stable, with wall class IoU dropping by at most 0.022 (ViT-S) and floor class IoU changing by at most 0.017 across all three backbone sizes. The relatively small class IoU drop indicates that RANSAC does not introduce many spurious floor or wall detections on surfaces that belong to neither class, so the overall pixel coverage of each class remains close to the histogram result. The larger instance IoU penalty reflects that RANSAC splits or merges the correct GT segments less accurately, degrading how well each predicted instance aligns with its individual ground-truth counterpart. Both failure modes are visible in the images below.

Figures 4.8–4.10 show prediction outputs for three images. In Lunch Room Tables, the left-side long wall is split into two separate wall instances. In the Garage, the floor is split into two instances, the car roof is merged with the floor prediction, and one wall facing the camera is also split into two instances. In the Coffee Lounge, some chair seats are merged into the floor prediction and the right-hand wall is split into two instances, with the smaller of the two absent from the output.

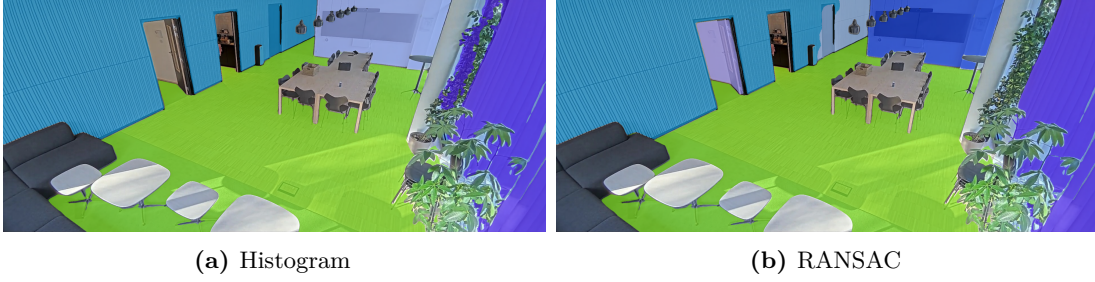


Figure 4.8 Lunch Room Tables (ViT-L + Swin-Large). In the RANSAC prediction the left-side long wall is over-segmented into two wall instances.

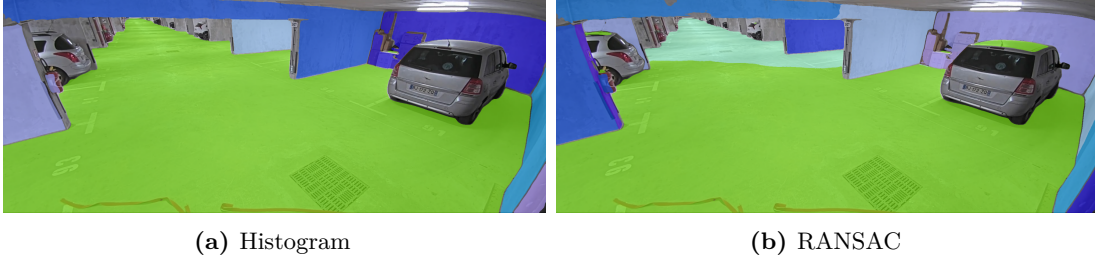


Figure 4.9 Garage (ViT-L + Swin-Large). In the RANSAC prediction the floor is over-segmented into two instances, the car roof is merged into the floor prediction (under-segmentation), and one wall facing the camera is also over-segmented.

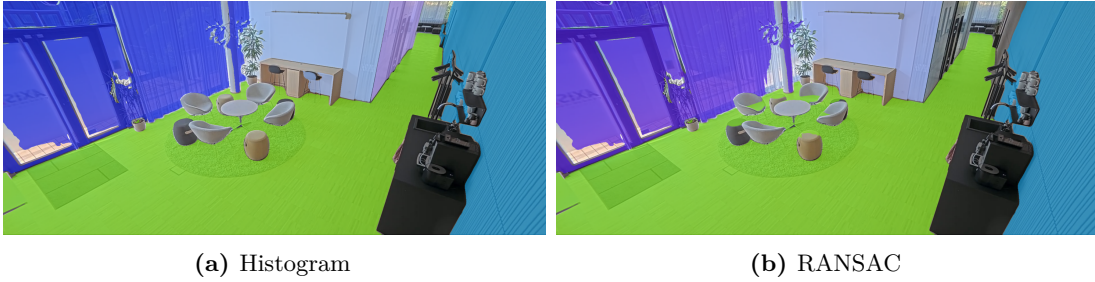


Figure 4.10 Coffee Lounge (ViT-L + Swin-Large). In the RANSAC prediction some chair seats are merged into the floor prediction (under-segmentation), the right-hand wall is over-segmented and the smaller segment is dropped and the corridor wall on the left is missing from the output.

No-Segmentation Ablation

The no-segmentation variant classifies instances by normal direction alone, without using a segmentation model. Table 4.8 reports results for each of the three MoGe backbone sizes and compares against the histogram method under the same conditions.

Table 4.8 Histogram versus no-segmentation ablation averaged over all 34 images.

MoGe	Method	Floor-cls	Wall-cls	Floor-inst	Wall-inst
ViT-L	Histogram	0.945	0.854	0.945	0.853
	No-seg	0.722	0.691	0.948	0.847
ViT-B	Histogram	0.941	0.844	0.942	0.829
	No-seg	0.717	0.687	0.943	0.822
ViT-S	Histogram	0.934	0.827	0.934	0.821
	No-seg	0.722	0.692	0.916	0.825

The histogram method outperforms the no-segmentation variant on class IoU across all three backbone sizes, while instance IoU remains close to the histogram result. The class IoU gap is largest on floor, where the no-segmentation variant falls by 0.21 to 0.22 points, and wall class IoU drops by 0.14 to 0.16 points. Floor instance IoU and wall instance IoU each change by at most 0.018 across all three backbone sizes. The class IoU drop is consistent across all three MoGe backbone sizes. The larger class IoU penalty reflects that the predicted floor and wall regions cover the ground-truth segments well but also include additional disconnected regions such as furniture surfaces, degrading class IoU. Both effects are visible in the images below.

Figures 4.11 and 4.12 show prediction outputs for two images. In the Coffee Lounge, table tops and sofa seats are predicted as floor alongside the true floor, and cabinet sides are predicted as wall. In the Kitchen, counter tops and table surfaces are predicted as floor alongside the true floor, and cabinet sides are predicted as wall.



Figure 4.11 Coffee Lounge (ViT-L + Swin-Large). In the no-seg prediction, table tops and sofa seats satisfy the upward-normal floor condition and are predicted as floor alongside the true floor. Cabinet sides are predicted as wall.

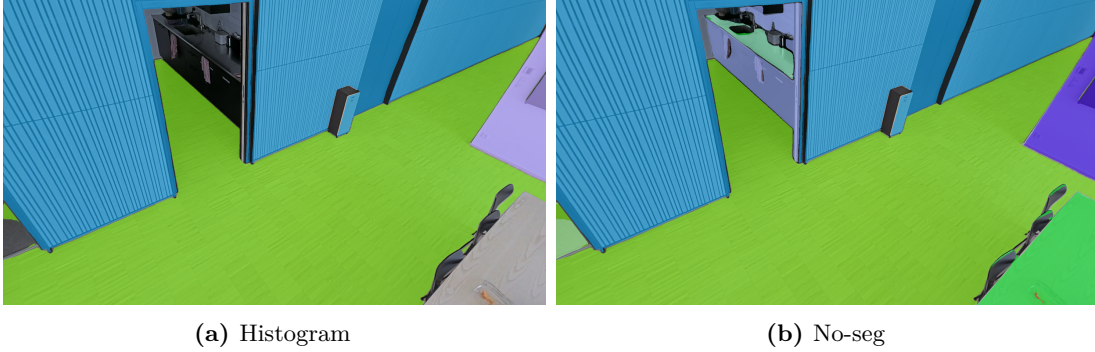


Figure 4.12 Kitchen (ViT-L + Swin-Large). In the no-seg prediction, counter tops and table surfaces are predicted as floor alongside the true floor, and cabinet sides are predicted as wall.

5

Discussion and Conclusions

This chapter interprets the results and draws conclusions against the two research questions. It covers why the pipeline performs as it does, the trade-offs in the key design choices, the failure modes seen in the hardest images, and practical considerations for deployment. The chapter closes with direct answers to both research questions and directions for future work.

5.1 Discussion

What Works and Why

Monocular depth combined with semantic segmentation detects and separates floor and wall planes from a single image across the 34 images in this evaluation. The scores reported throughout this section are for the largest backbone combination tested, ViT-L with Swin-Large. Floor detection achieves a class IoU of 0.945 averaged over all 34 images and remains consistent across both real and synthetic subsets. Across all twelve backbone combinations tested, floor class IoU ranges from 0.910 to 0.945, a spread of 0.035. Wall detection scores lower at 0.854 for the largest combination and degrades more noticeably with smaller backbones, ranging from 0.786 to 0.854 across the same twelve combinations, a spread of 0.068 and nearly twice the floor variation.

The robustness of floor detection comes from a physical constraint that does not apply to walls. A floor must be flat and level for walking and wheeled mobility, so in inhabited spaces floor normals cluster tightly around a single upward direction. This gives the normal clustering algorithm a well-conditioned, narrow target. Walls have no such constraint. They can appear at any depth, be partially occluded by furniture of arbitrary height, and their normals can coincide with those of large flat objects such as cabinets and whiteboards. This variety explains both the lower wall scores and their greater sensitivity to backbone size. Smaller depth and segmentation models produce noisier normals and less reliable semantic labels, and those errors accumulate faster on surfaces with greater natural variation.

A subtler result is the small gap between class and instance IoU for both classes. Class IoU measures whether detected pixels cover the correct region. Instance IoU measures whether those pixels are also grouped into the correct number of separate instances. The two metrics are close for both classes (floor class 0.945 and instance 0.945, wall class 0.854 and instance 0.853), indicating that the method neither systematically merges distinct planes into one nor fragments a single plane into many.

Real images score higher than synthetic (0.975 vs. 0.938 floor, 0.900 vs. 0.844 wall), but the gap does not indicate overfitting to real-world data. Five of the six real images come from the same office, a space with a relatively simple layout of perpendicular walls, low furniture density, and only modest visual clutter. The real subset is therefore straightforward by the composition of

the evaluation set. The synthetic images were deliberately chosen to cover harder configurations, including non-orthogonal rooms, heavy occlusion, and unusual depth distributions.

The intermediate outputs of the geometry estimation and semantic segmentation stages show comparable visual quality on synthetic and real images. The pipeline operates on those outputs rather than on raw images, so any quality gap in the intermediate representations would propagate to the final result. The comparable intermediate quality suggests the real-versus-synthetic performance gap is driven primarily by the real images being simpler in layout rather than by the models producing lower-quality outputs on synthetic imagery.

Scope and Generalisation

The parameter search and the final evaluation were conducted on the same 34 images, which limits what can be said about how the method will perform on environments it has not been tuned to.

The degree of overfitting is probably modest. The search optimised roughly twelve global parameter values, each required to work across all 34 images at once. No image received its own configuration, and a value that helped one image at the cost of others was penalised. Because the 34 images cover a wide range of layouts and furniture arrangements, a parameter set that works across all of them is not closely tailored to any single configuration. The backbone comparison in section 4.2 supports this. Backbone combinations other than the tuned ViT-L + Swin-Large achieve scores within a narrow band of the best result, and if the parameters were strongly overfitted to one model’s output characteristics, smaller combinations would degrade more noticeably than they do.

The results still point to a real limit on generalisation. The Garage image, which is visually quite different from the rest of the real footage, and the synthetic images with tilted walls (see section 5.1) both score below the rest of the evaluation set. Images with harder or more unusual layouts consistently produce worse results. The method will likely not generalise well to indoor environments that differ substantially from the types covered here.

Generalising to all indoor scenes is a hard problem regardless of the approach. Any method, whether tuned or trained, requires annotated data for new environment types. Large public datasets with floor and wall annotations exist, but they cover only a narrow range of environment types. Structured3D [32], for example, is limited to apartment and house-like interiors. Shopping centres, staircases, multi-level garages, and similar spaces are essentially unannotated, so the scope of what can be evaluated on existing data is narrow. This is not a limitation specific to this work.

How well the method generalises beyond this evaluation set is an open question, and the evidence above suggests the answer is not particularly well. What the results do establish is that combining monocular geometry with semantic segmentation produces meaningful floor and wall extractions on the images in this set. The small number of global parameters, the lack of per-image tuning, and the consistent scores across backbone combinations all suggest the method is picking up genuine structure in the data rather than fitting to noise. The results are best read as a proof of concept.

For practical use, the breadth of what counts as an indoor environment is enormous, and attempting to make the method work across all of it at once is unlikely to be productive. A more realistic approach is to identify a narrow target environment, collect a small annotated set drawn from that specific type, and tune the parameters towards it. The results here suggest that

a carefully selected subset can yield reliable extractions, and this kind of focused deployment is likely a more achievable starting point than aiming for general indoor coverage.

Plane Finding in Estimated Point Cloud Data

MoGe produces high-quality metric geometry from a single image, but the output is an estimation not a real measurement. In practice this means that a large flat wall or a wide floor, when represented as a point cloud from MoGe, is not a perfectly flat plane. There is residual noise and slight curvature that grows with surface size and is more pronounced closer to the camera. Points that should be coplanar in the real world are therefore not exactly coplanar in the estimated point cloud.

This imperfection creates a problem in plane segmentation. A strict plane-fitting approach that requires points to lie tightly on a single mathematical plane would correctly distinguish closely spaced surfaces like a sofa back leaning against a wall, or a small cabinet standing near a corner but would also split a single large wall or floor into multiple detections wherever the noise or curvature locally exceeds the fitting tolerance. A lenient approach that accepts more deviation from flatness would recover large surfaces as single coherent planes but would absorb nearby objects into the same detection.

The algorithm presented in this thesis takes the lenient direction, treating points within roughly one metre of each other in projected depth as the same plane instance. A painting or television mounted flush against a wall is absorbed into the wall detection rather than triggering a split. An over-merged wall would still be a valid room boundary for every downstream use considered in this work, whereas an over-split wall would introduce phantom room corners that would corrupt downstream geometry. A different use case that requires distinguishing every surface individually might prefer the opposite setting and then merge fragments in a post-processing step.

RANSAC is the natural alternative for robust plane fitting, and the comparison in section 4.7 quantifies the difference. Across all three MoGe backbone sizes the histogram method outperforms RANSAC on floor class IoU, floor instance IoU, and wall instance IoU, with the gap widening for smaller backbones. With ViT-S the floor instance IoU difference is 0.030 and the wall instance IoU difference is 0.067. The prediction figures for the three comparison images show the characteristic failure patterns. In Lunch Room Tables, a single long wall is over-segmented into two instances because depth variation along its length caused RANSAC to fit separate planes to one physical surface. In the Garage, the floor splits by the same mechanism and the car roof at a similar depth is merged into the floor instance. The Coffee Lounge shows both effects simultaneously. A wall segment is over-segmented and the smaller part falls below the minimum size threshold and is discarded, leaving a gap in the wall output, while a chair seat is absorbed into the floor by under-segmentation. These failures share a common cause. The acceptance radius wide enough to handle the distributed depth of a genuinely large surface in MoGe’s noisy point cloud also admits points from an adjacent surface, causing under-segmentation. A tighter radius fragments large surfaces, causing over-segmentation. Eighty Optuna trials identified the best available trade-off in the search space, but no single global parameter setting resolved this tension across 34 diverse images. The histogram method avoids it because it operates on the one-dimensional distribution of projected depths within a normal cluster rather than on individual point distances. A broad, slightly curved surface produces a single coherent peak in the smoothed histogram even when its constituent points are not exactly coplanar, and the

valley between two peaks is detectable regardless of how wide each individual peak is.

Limitations of Semantic Labels

The pipeline’s use of semantic segmentation as a final classification step introduces a dependency that differs from MoGe’s geometric uncertainty. Semantic labels are not crisp at the pixel level. Mask2Former tends to bleed labels across object boundaries, so pixels at the edge of a wall instance may receive furniture labels and vice versa. Beyond this boundary imprecision, the model sometimes makes outright classification errors that are difficult to anticipate. MoGe’s geometric errors are smooth and follow predictable physical patterns (curvature increasing with surface size, noise scaling with distance from the camera), whereas segmentation errors can be arbitrary and image-specific. One image may have near-perfect wall classification while a geometrically similar image produces an unexpected misclassification of the same surface type.

Large cabinets, built-in shelving, and tall display screens are perceptually similar to walls, being flat, vertical, and often spanning the full height of the room. Whether a cabinet should be treated as a wall depends on the downstream use case, and there is no single correct answer. For room-boundary estimation a cabinet flush against a wall would be harmless to absorb, whereas for camera placement verification the same cabinet might block a relevant line of sight. The semantic vote filter makes a single static choice, and this choice will be correct for many images but wrong for others. Any threshold that allows cabinets to count as walls in the images where this is desirable would also allow them to count in images where it is not.

The point cloud comes from MoGe and the semantic labels come from Mask2Former, and neither source is aware of the other. When Mask2Former misclassifies a region, the semantic vote for the plane instance containing that region shifts accordingly. A large misclassification covering a substantial fraction of a genuine wall or floor instance can flip the vote and cause the entire instance to be discarded, producing a miss that traces entirely to a segmentation error rather than a geometry failure. This effect is more pronounced for small plane instances, where a few misclassified pixels represent a larger share of the total.

Geometry-First Pipeline Ordering

The geometry-first ordering of the current algorithm is a deliberate response to this unpredictability. By extracting planar instances from the point cloud first, each candidate plane is classified independently based on the majority semantic label of its pixels. The majority vote makes the system robust to local boundary bleed, since a few misclassified pixels at the edge of a large wall plane do not flip its classification. Starting from geometry also means the algorithm operates on coherent planar regions, which are more likely to have a dominant semantic label than arbitrary image patches, further diluting the effect of individual label errors.

An alternative design could apply semantic segmentation first, selecting pixels labelled as wall or floor before performing any geometry, and then use normal clustering and depth splitting to separate instances within those labelled regions. Provided the segmentation model reliably classifies sofas, televisions, and similar objects as furniture rather than walls, this ordering would prevent them from entering the pipeline entirely and would address some of the failure cases examined later in this chapter. A segmentation model that occasionally misclassifies a sofa as a wall would introduce those very objects at the earliest stage. Beyond the furniture exclusion benefit, many of the current challenges would remain unchanged. Boundary bleed would clip wall regions at their edges, requiring some form of post-processing to recover the missing boundary pixels, which could in turn introduce its own unforeseen problems. A large

misclassification at the start of the pipeline would be harder to recover from than in the current design where geometry constrains which label errors matter, the close-wall merging problem would persist because MoGe’s depth precision is unchanged, and the ambiguous class boundary question (whether to treat a large cabinet as a wall) would remain regardless of ordering. A semantics-first approach is a viable alternative, and at current Mask2Former quality could plausibly achieve similar overall scores, but it should not be seen as a solution to the general problem. Most of the hard cases would still be hard. The ordering shifts where errors arise, not whether they arise. The geometry-first choice was made because geometric errors from MoGe are more consistent and predictable image to image than segmentation errors, making the system easier to reason about and tune.

Failure Mode Interpretation

The four images examined in section 4.6 fall into three distinct failure categories, each illustrating a different limit of the current approach.

The first is depth histogram ambiguity, illustrated by ai_054_007/c1/f53. The sofa tops and table surfaces in this image sit at approximately the same projected depth as the floor platform. Because the algorithm under-segments by design, these surfaces are merged into the same plane instance as the floor, and the majority semantic vote for that instance returns floor, causing the sofa and table to be marked as floor in the output. For this specific image a tighter depth-valley threshold would likely produce the correct separation, but applying that threshold globally would cause large floors to be over-segmented in other images where small depth variation is surface curvature rather than a real structural step. This is the kind of per-image trade-off that motivated the Optuna-based parameter search, since manually tuning on one image improves it at the cost of degrading the aggregate score. This particular failure could likely be avoided by a semantics-first pipeline ordering, since correctly labelled sofas and tables would never enter the floor candidate set, but that ordering introduces its own challenges.

The second failure category is illustrated by ai_001_010/c0/f29 and involves both a depth merging failure and a labelling problem. The image contains three annotated wall instances. Two of them cause problems. A large cabinet occupies most of the visible area, and the actual wall behind it sits at a similar projected depth with a similar normal, so the algorithm merges those two into a single plane instance. The merged instance is dominated by cabinet pixels, so the semantic vote returns cabinet and the whole instance is excluded. Both the cabinet instance and the wall behind it are consequently missed, while the third annotated wall is unaffected. Part of the failure is geometric, as the merging step groups two surfaces that should remain separate. But even if the merging were fixed, a deeper problem would remain. Adding cabinet to the set of wall-like labels would recover this image, but the same change would cause cabinets to be detected as walls in every other image where they happen to appear, which is not desirable in general. The current static label set reflects a deliberate choice that is correct for most images and incorrect for this one, and there is no way to satisfy both outcomes with a single fixed rule.

The third failure category involves surfaces in the ambiguous zone between wall and ceiling geometry, illustrated by both ai_008_010/c0/f7 and ai_003_010/c0/f18. These images contain slanted surfaces that are not annotated as walls in the ground truth. The algorithm does partially detect them as walls, producing over-segmented detections on surfaces the annotations do not count, which appear as false positives and lower the IoU scores. Whether such a surface should be labelled as a wall is genuinely unclear. A slanted surface at an intermediate angle could

reasonably be considered a wall for room-boundary estimation purposes and a ceiling for other applications. What the algorithm produces is therefore not entirely wrong, but it does not match the annotations, and that disagreement drives the low scores. The difficulty is that no single labelling convention is objectively correct for surfaces in this intermediate zone, and the score in these images reflects annotation ambiguity as much as any algorithmic failure. The algorithm was designed for perpendicular walls and makes no claim about how to handle surfaces in this region.

No-Segmentation Ablation

The no-segmentation ablation in section 4.7 removes semantic labelling entirely and classifies plane instances by normal direction alone. Floor class IoU drops by 0.21 to 0.22 points and wall class IoU drops by 0.14 to 0.16 points across all three backbone sizes. Instance IoU is much less affected, changing by at most 0.018.

The pattern makes sense. Without semantic labels, any surface with an upward-pointing normal is classified as floor and any surface with a vertical normal is classified as wall, regardless of what the surface actually is. Table tops, sofa seats, and counter surfaces all satisfy the floor normal condition and are therefore predicted as floor alongside the true floor. Cabinet sides and similar vertical furniture satisfy the wall condition and are included in the wall output. These additional regions lower class IoU. Instance IoU is largely unaffected because the geometric stage still finds and separates the plane instances correctly. The extra furniture surfaces either fall into existing instances without breaking the match or form small separate instances that do not strongly affect the score. The separation of class IoU impact from instance IoU impact indicates that semantic labelling contributes primarily to filtering out incorrectly oriented furniture, while the geometric stage handles instance delineation.

Deployment Considerations

The letterbox ablation in section 4.5 confirms that this pre-processing step cannot be treated as optional. IoU drops appear across both the real and synthetic letterbox conditions, with the largest cases large enough to change whether the output is usable, not just to degrade a metric by a small margin. The step is simple to automate, and its cost is negligible relative to inference, so there is no practical reason to skip it.

The timings in section 4.3 and section 4.4 were measured on high-end desktop hardware, but the relative profile is informative regardless of the target platform. MoGe inference ranges from 21 ms (ViT-S) to 43 ms (ViT-L) and Mask2Former from 12 ms to 21 ms across backbone sizes, giving a combined inference budget of roughly 33–64 ms. The Stage 2 algorithm runs at 375–403 ms averaged over all 34 images, six to ten times longer than inference. The algorithm therefore dominates the total runtime, and switching to a smaller backbone does not meaningfully change throughput. If the pipeline needs to run at a higher update rate, the algorithm is the right target for optimisation.

A subtler observation is that the algorithm time is not entirely independent of backbone choice even though the algorithm stage loaded precomputed MoGe and Mask2Former outputs from disk. The mean algorithm time is roughly 25 ms longer for ViT-L combinations than for ViT-S combinations. The precise reason is not certain, but the most plausible explanation is that larger backbones tend to produce outputs with finer-grained normal and depth structure, which generates more candidate plane clusters and more histogram computation per image. The effect is small relative to the overall algorithm runtime and does not change the conclusion

that inference time is a minor component of total latency. No effort was made in this work to optimise the algorithm for speed, so the current runtimes should not be taken as a lower bound.

The backbone results in section 4.2 also inform backbone selection. Floor detection is largely unaffected by backbone size, which means a floor-only deployment can trade down to the smallest combination at roughly half the inference time without significant accuracy loss. Wall detection is more sensitive to backbone size, and the score degradation with smaller models is large enough to matter in practice. An application that needs both surfaces should use the largest combination and for wall detection specifically, the larger backbone earns its additional inference time.

Practical Applications

The pipeline output is not an end in itself. It is a structured representation of the scene that is intended to enable a range of higher-level analytics.

The most direct use is the instance segmentation itself. Once the image pixels are labelled as floor, wall, or neither (the remaining region, corresponding broadly to objects and furniture), the spatial location of any person detection could be validated against these labels. A person walking on the floor should have their bounding box base land on a floor pixel. If the base instead projects onto a wall region the localisation is geometrically implausible and the detection could be flagged as a potential false positive or as a sign that the subject is not on the floor surface, for example because they are behind glass or the detector has misfired. A detection whose base falls in the remaining region may be partially or fully occluded by nearby furniture, which means that the base of the bounding box likely does not represent the point of contact of that person, so using it to estimate the person’s distance from the camera would not give correct results.

The plane instances can be combined with MoGe’s metric point cloud output for 3D geometry reasoning. This application has not been evaluated. It is included to show one concrete downstream use, not as a validated capability. The current evaluation measures floor and wall IoU in the image plane and includes no ground-truth range measurements. Visual inspection of depth and normal outputs across all 34 images showed consistently plausible metric geometry, with walls at realistic distances and floor planes at appropriate angles, and since the plane instances are a by-product of the detection step, their 3D intersections can be computed directly. Each detected wall plane is represented by a point and a normal vector in metric 3D space, and the floor plane is likewise. The intersection of two planes is a line, obtained analytically from the two normal vectors and offsets. This line is then clipped to the lateral extent of the detected wall instance and projected back to the image plane for visualisation. The resulting skirting line could locate the room boundary even where part of the floor is fully occluded by furniture, because the line is derived from the fitted plane geometry rather than from visible floor pixels directly. Wall-wall intersection lines could similarly locate room corners in 3D, and because MoGe produces metric-scale output these coordinates are in real-world units.

With the room boundary known in 3D, any detected object with an associated depth estimate could be tested against it, and if the object’s 3D position lies inside the convex hull formed by the wall planes it would be within the room, whereas a position outside could indicate an object behind glass or in an adjacent space. The 2D room footprint derived from the intersection lines could also be compared against a pre-existing floor plan, and systematic disagreements could flag a camera mounted in the wrong position or pointing in an unexpected direction. These

geometric products are a natural extension of what the pipeline already computes rather than a separate reconstruction stage.

None of these applications need the floor and wall boundaries to be perfect. Under-segmented walls that merge two adjacent panels still produce valid spatial priors and valid intersection lines. Only systematic misses, missing a wall entirely or placing it at the wrong depth, would degrade the downstream geometry, and those cases correspond exactly to the hard images already discussed.

Figures 5.1–5.4 show the intersection lines on four real images as a concept illustration. These results are not part of the formal evaluation and no accuracy claim is made about the recovered 3D coordinates. The coloured skirting lines follow the room boundary continuously even in front of furniture, since they are projected from 3D plane fits rather than traced along visible floor edges. In an image with no detectable floor or no detectable wall the lines cannot be computed, but this is rare for the deployment context.

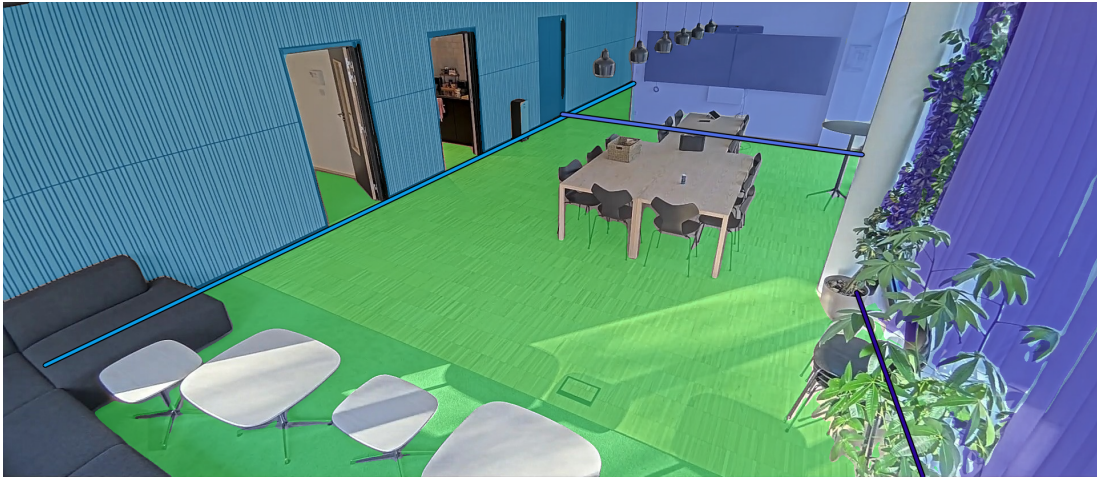


Figure 5.1 Floor-wall intersection lines, Lunch Room Tables (floor 0.961, wall 0.903, ViT-L + Swin-Large). Each coloured line segment shows the 3D intersection of a detected wall plane with the detected floor plane, projected back to the image plane. Wall and floor masks are shown as coloured overlays, and the intersection line for each wall uses the same colour as its mask.



Figure 5.2 Floor-wall intersection lines, ai_001_005/c0/f0 (floor 0.964, wall 0.764, ViT-L + Swin-Large). Each coloured line segment shows the 3D intersection of a detected wall plane with the detected floor plane, projected back to the image plane. Wall and floor masks are shown as coloured overlays, and the intersection line for each wall uses the same colour as its mask.

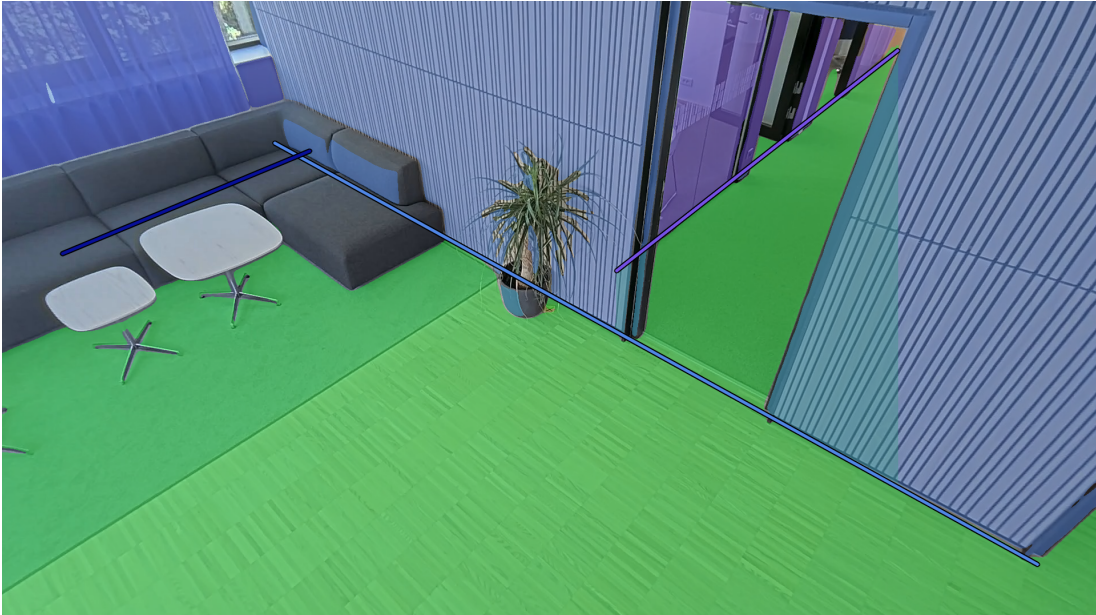


Figure 5.3 Floor-wall intersection lines, Middle (floor 0.983, wall 0.931, ViT-L + Swin-Large). Each coloured line segment shows the 3D intersection of a detected wall plane with the detected floor plane, projected back to the image plane. Wall and floor masks are shown as coloured overlays, and the intersection line for each wall uses the same colour as its mask.



Figure 5.4 Floor-wall intersection lines, Lunch Room Corridor (floor 0.971, wall 0.880, ViT-L + Swin-Large). Each coloured line segment shows the 3D intersection of a detected wall plane with the detected floor plane, projected back to the image plane. Wall and floor masks are shown as coloured overlays, and the intersection line for each wall uses the same colour as its mask.

5.2 Conclusions

RQ1. Algorithm performance. On the 34 images used in this evaluation, monocular geometry combined with semantic segmentation detects and separates floor and wall instances with high scores. Performance on general indoor environments beyond this set is expected to be limited. The evaluation images cover a narrow range of layouts, and the discussion in section 5.1 gives reasons to expect lower scores on environments that differ substantially from those covered here.

Within the evaluation set, the pipeline achieves a floor class IoU of 0.945 and a wall class IoU of 0.854 averaged over all images with the largest backbone combination (ViT-L + Swin-Large). Instance IoU closely tracks class IoU for both surfaces (0.945 floor, 0.853 wall), indicating that the method neither systematically merges distinct planes nor fragments single surfaces into many, and that false detections on surfaces that are not floor or wall are infrequent. Floor detection is robust across all twelve backbone combinations tested, ranging from 0.910 to 0.945, and is largely unaffected by backbone choice. Wall detection is more sensitive, ranging from 0.786 to 0.854, and degrades more noticeably with smaller models because walls present greater normal variation and more frequent semantic label ambiguity than floors.

RQ2. Practical limits. The practical limits of the approach are visible in the four images examined in section 4.6, where two root causes account for the underperformance. The first is the lenient depth-merging strategy, which is necessary because MoGe’s estimated point clouds are not perfectly flat for flat real-world surfaces. Surfaces close in depth are merged into a single instance, and the semantic vote on the merged instance determines the outcome. If non-floor or non-wall pixels dominate the vote, the entire instance is excluded and any real floor or wall it contains is missed. If floor or wall pixels dominate, the merged instance is accepted and the unrelated pixels are incorrectly labelled as floor or wall. The second root cause is that the algorithm assumes walls are upright. Slanted surfaces at angles between wall and ceiling pass the wall orientation filter and are predicted as wall even when not annotated as such. When such a surface is geometrically connected to a real wall, the two merge into a single over-extended instance that lowers both wall class IoU and wall instance IoU. When the slanted surface is separate, only wall class IoU falls significantly. Both failure types can occur in real deployments. The depth-merging case occurs in environments where furniture sits near walls or where large

wall-like objects such as cabinets are annotated as walls, and the slanted-surface case depends on room geometry. These are limitations of the current implementation and would need targeted solutions to address reliably. The method scores higher on the real images (0.975 floor, 0.900 wall) than on the intentionally hard synthetic subset (0.938 floor, 0.844 wall), and the gap reflects the simpler layout of the real images rather than any overfitting to real-world imagery.

A broader practical limit is that indoor environments cover an enormous range of room types, furniture densities, camera heights, and lighting conditions, and the available annotated data covers only a small fraction of that range. Attempting to make a single configuration work across all indoor scene types is therefore not a realistic goal at this stage. The more achievable path is to identify a specific target environment, collect a small annotated set from that context, and tune towards it. The scores on the office and corridor images in this evaluation suggest that a focused approach of this kind can produce reliable results. The hard synthetic images show what happens when the layout moves away from the types the parameters were tuned on.

5.3 Future Work

Broader evaluation. The 34 images in this evaluation cover a limited range of environments, and the results do not establish how the method performs on environments outside that range. The evidence from the hard images in section 4.6 suggests that performance degrades as layout complexity increases, which makes broad generalisation across all indoor scene types an unrealistic target without substantial additional work. A more tractable goal is to identify a narrow target deployment context, collect a small annotated set drawn from that specific context, and tune the parameters towards it. The questions of how many annotated images are sufficient, how sensitive the results are to which images are chosen, and how sharply performance degrades as the deployment environment drifts from the tuned set remain open and would require dedicated experiments to answer.

Temporal integration. The current pipeline processes each frame independently, but Axis cameras produce continuous video streams where structural elements such as floors and walls remain static across frames. Accumulating evidence over time would allow the method to build a more reliable scene model by reinforcing consistent detections and suppressing transient errors caused by moving objects or single-frame noise. The temporal signal could also distinguish structural surfaces, which remain stable across hundreds of frames, from temporary scene elements that change. This would reduce single-frame errors and enable detection of when the room configuration has changed.

Model exploration. This thesis evaluated three MoGe backbone sizes paired with four Mask2Former backbone sizes, but many other geometry estimators and segmentation models have not been tried. The pipeline places no hard requirements on which specific models are used, and it is possible that a different combination, whether a newer geometry model, a more specialised segmentation network, or a lighter architecture optimised for speed, could improve results or reduce inference time while achieving similar accuracy. A broader search over model combinations would establish whether the current best result is close to what the pipeline architecture can achieve, or whether substantial gains remain.

3D geometry from MoGe depth. The floor-wall intersection application in section 5.1 shows that the metric point cloud produced by MoGe can be used for downstream 3D reasoning, but this use was not evaluated. Whether MoGe’s absolute depth estimates are accurate enough to support applications such as room-corner localisation, object placement verification, or floor-plan comparison against a reference has not been established. A dedicated study with ground-truth range measurements would quantify the metric accuracy of MoGe’s output in deployment conditions and determine which 3D geometry applications are viable in practice.

On-device deployment. All experiments were run on a desktop GPU. For practical use on Axis cameras, the pipeline would need to run on embedded hardware (Axis ARTPEC), which requires profiling inference latency and memory, exploring model quantisation for MoGe and Mask2Former, and addressing system-level questions such as when to trigger a run and how to detect that scene conditions are suitable for reliable estimation.

Bibliography

- [1] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022.
- [2] R. Wang, S. Xu, C. Dai, J. Xiang, Y. Deng, X. Tong, and J. Yang, “MoGe: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [3] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [4] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, “Depth anything v2,” in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
- [5] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. V. Gool, and F. Yu, “UniDepth: Universal monocular metric depth estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [6] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, “DINOv2: Learning robust visual features without supervision,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.07193>
- [7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference on Learning Representations (ICLR)*, 2021.
- [8] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [9] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, “Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

- [10] Microsoft MoGe Contributors, “Moge-2 normal estimation,” <https://github.com/microsoft/MoGe/blob/main/docs/normal.md>, 2026, accessed: 2026-05-26.
- [11] MoGe Team at Microsoft Research, “MoGe repository README,” <https://github.com/microsoft/MoGe/blob/main/README.md>, 2026, accessed: 2026-05-28.
- [12] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, “Masked-attention mask transformer for universal image segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [13] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [14] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [15] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “SegFormer: Simple and efficient design for semantic segmentation with transformers,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [16] B. Okorn, X. Xiong, B. Akinci, and D. Huber, “Toward automated modeling of floor plans,” in *Proceedings of the Symposium on 3D Data Processing, Visualization and Transmission*, Paris, France, 2010.
- [17] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [18] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgb-d images,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760.
- [19] R. Schnabel, R. Wahl, and R. Klein, “Efficient RANSAC for point-cloud shape detection,” *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.
- [20] V. Hedau, D. Hoiem, and D. Forsyth, “Recovering the spatial layout of cluttered rooms,” in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1849–1856.
- [21] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, “LayoutNet: Reconstructing the 3D room layout from a single RGB image,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [22] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, “LED2-Net: Monocular 360 layout estimation via differentiable depth rendering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [23] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, “PlaneNet: Piece-wise planar reconstruction from a single RGB image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

- [24] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz, “PlaneRCNN: 3D plane detection and reconstruction from a single image,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [26] Z. Yu, J. Zheng, D. Lian, Z. Zhou, and S. Gao, “Single-image piece-wise planar 3D reconstruction via associative embedding,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [27] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.
- [28] K. Wada, “labelme: Image polygonal annotation with python,” <https://pypi.org/project/labelme/>, 2016.
- [29] The HDF Group, “Hierarchical Data Format, version 5,” <https://www.hdfgroup.org/solutions/hdf5/>, 1997.
- [30] D. H. Douglas and T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.
- [32] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou, “Structured3d: A large photo-realistic dataset for structured 3d modeling,” 2020. [Online]. Available: <https://arxiv.org/abs/1908.00222>

Master's Theses in Mathematical Sciences 2026:E33

ISSN 1404-6342

LUTFMA-3619-2026

Mathematics

Centre for Mathematical Sciences

Lund University

Box 118, SE-221 00 Lund, Sweden

<http://www.maths.lth.se/>