The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-4/W10-2024
8th International Conference on Smart Data and Smart Cities (SDSC), 4–7 June 2024, Athens, Greece

Integrating SAM and LoRA for DSM-Based Planar Region Extraction in Building Footprints

Oktay Eker[1,*], Murat Avcı[1], Selen Çiğdem[2], Oğuzhan Özdemir[1], Fatih Nar[1,4], Dmitry Kudinov[3]

[1] Esri, Ankara R&D Center, ODTU Teknokent Galyum Blok 06800 Çankaya Ankara, Türkiye - oeker@esri.com
[2] Esri, nFrames, Stuttgart, Germany [3] Esri, Redlands, California, U.S.A.
[4] Department of Computer Engineering, Ankara Yıldırım Beyazıt University, Ankara, Türkiye

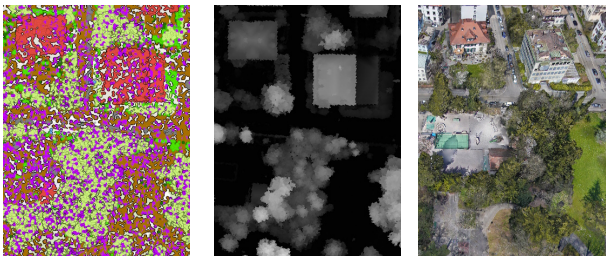
Keywords: Plane Detection, Segment Anything Model (SAM), Low-Rank Adaptation (LoRA), Elevation, 3D Building Model.

Abstract

In this paper, we present a novel approach for segmenting planar regions in Digital Surface Models (DSMs) by adapting the Segment Anything Model (SAM), an open-source framework. Our approach specifically tailors SAM to recognize planar regions within given building footprints, employing the Low-Rank Adaptation (LoRA) technique. This adaptation benefits from a detailed and realistic synthetic dataset, coupled with a novel labeling strategy for planar regions in our ground truth, enhancing the model's effectiveness and reproducibility. Unlike traditional plane detection techniques, our method consistently and accurately identifies equivalent planar regions across identical DSM inputs. Following the segmentation phase, we introduced a novel plane fitting algorithm to determine the parameters for each planar region. This enables us to refine the edges of these areas and utilize the resulting plane equations to construct precise, watertight 3D models of buildings. Despite its training on synthetic data, our model exhibits remarkable performance on both synthetic and real-world datasets, exemplified by its application to the Zurich dataset.

1 Introduction

Light Detection and Ranging (LiDAR) systems generate point cloud data, consisting of scattered 3D points, by emitting laser pulses to capture Earth's surface elevations (Wang et al., 2018). On the contrary, a Digital Surface Model (DSM) is formed by photogrammetry from overlapping aerial images, featuring elevation values arranged within a structured 2D grid (Zhou, 2017). Note that point clouds can also undergo transformation into a DSM if needed (Çağdaş Bak et al., 2016). Given that both approach provide representations of the Earth's surface, different processing methods are required to transform these data types for subsequent tasks. Within these techniques, plane detection and estimation of plane parameters is crucial in transforming this complex data into planar segments. Detecting planar regions in facades and roofs from point clouds (Liu et al., 2024) (see Figure 1a) or DSMs (see Figure 1b) enables the generation of 3D building models. Such precise yet simple 3D building models reduce storage and processing demands, while also enabling applications like flood simulation, noise or air pollution simulations, and assessment of solar potential (Rottensteiner and Briese, 2002, Li et al., 2005).



(a) Point Cloud (b) DSM (c) 3D Mesh

Figure 1. (a) Scattered 3D Point Cloud *versus* (b) Elevations on a Structured DSM Grid.

* Corresponding author

Plane detection is crucial for generating 3D building models, particularly from point cloud and DSM data. Up to now, the literature has emphasized plane detection from point cloud (Liu et al., 2024), with less focus on DSM. However, recently, there has been an increase in algorithms leveraging DSM data for various tasks (Nar et al., 2018, Aktaş et al., 2018). This trend aligns with the enhanced availability of high-resolution DSM data, thanks to the reduced costs of aerial vehicles and superior-quality affordable cameras.

Point cloud-based methods, have been extensively studied and applied in plane detection tasks. Among these, the Random Sample Consensus (RANSAC) (Fischler and Bolles, 1981) stands out as one of the earliest and most widely adopted method. It iteratively fits plane parameters to point subsets, providing a robust estimation of these parameters even in noisy data with outliers, enhancing method's popularity. Nevertheless, nondeterministic nature of RANSAC poses a significant drawback by leading to different outcomes with each run, thus hindering the reproducibility of the results. Another widely known geometric approach is the Hough Transform (Hough, 1962, Duda and Hart, 1972). Hough Transform employs a robust voting mechanism to map points onto potential planes within the parameter space, subsequently detecting clusters indicative of a common plane through the identification of peaks in the accumulator array. However, its efficient handling of noise and outliers is overshadowed by limitations in practical application due to its computational load. To mitigate this limitation, Deschaud and Goulette introduce an efficient plane detection method (Deschaud and Goulette, 2010), which markedly improves both speed and accuracy compared to the traditional approaches. Their proposed algorithm first estimates filtered normals to reduce noise impact. Then, it computes a local planarity score for each point and selects the best seed plane for voxel growing. This method speeds up plane detection by employing a voxel grid structure. However, there's a trade-off between speed and accuracy; increasing voxel size can sacrifice accuracy for speed. Additionally, the algorithm's performance is sensitive to hyper-parameters.

Statistical methods like Principle Component Analysis (PCA) (Jolliffe and Cadima, 2016) represent one of the earliest techniques for plane detection, albeit limited to a single plane. PCA estimates plane normals by identifying principal components that capture the most data variance. However, it is susceptible to noise and can be computationally inefficient due to singular value decomposition (SVD). Nurunnabi et al. proposed a robust PCA method (Nurunnabi et al., 2014), which utilizes the Minimum Covariance Determinant (MCD) to iteratively estimate a covariance matrix in a manner less influenced by outliers, albeit at a much slower pace. A significant limitation of PCA (or SVD)-based plane detection is its confinement to a single plane.

Plane detection, as a first step in the 3D model reconstruction pipeline, is a foundational element for numerous 3D model reconstruction studies. In their study, Wonka and Nan presented an approach for reconstructing piecewise planar objects (Nan and Wonka, 2017). The approach adopted in this study shows the importance of having accurate planes for generating 3D models. This method efficiently crafts a lightweight polygonal surface model by carefully selecting candidate faces, to create detailed 3D meshes through binary linear programming. Thereby, it ensures compliance with manifold and watertight constraints while preserving the sharp features of the objects. The method's efficacy relies on the precise detection and accurate parameter estimation of plane candidates, underscoring the pivotal role that robust plane detection techniques play in the reconstruction process. Similarly, the approach by Ochmann et al. for automating the reconstruction of building layouts from point cloud data emphasizes the importance of accurate wall candidate generation (Ochmann et al., 2016). This generation process, crucial for encapsulating building layouts, entails fitting parallel lines to support point sets derived from overlapping line segments. Facilitated by precise plane detection, the optimization of layouts through these candidates is key to enabling automated reconstruction.

While traditional methods often rely on handcrafted features, predefined rules, heuristics, or optimization techniques for plane detection, our proposed approach embraces the profound capabilities of deep learning (DL). Thus, we propose leveraging the Segment Anything Model (SAM) (Kirillov et al., 2023), as it excels in capturing the deep data semantics while effectively encompassing context and spatial dependencies. Subsequently, we fine-tuned SAM with Low-Rank Adaptation (LoRA) (Hu et al., 2021) using a novel planar region labeling scheme on top of our realistically looking synthetic dataset. Following, we used a custom developed robust fitting algorithm to estimate plane parameters for the regions generated by SAM. Hence, our method efficiently generates multiple planes from a given DSM and building footprint in a reproducible manner, setting it apart from many widely used methods. Lastly, segmented planar regions are refined using the estimated plane equations, leading to the construction of a simplified yet accurate LOD2.2 building model. Thus, our research explores utilizing DSM for plane detection, a departure from the predominant focus on point cloud utilization in prior studies. Utilizing DSM enables us to harness inherent spatial structure, facilitating the development of an accurate and efficient plane detection approach. Note that, high-resolution DSMs are increasingly accessible since drones and high-quality cameras become more affordable. On the other hand, point cloud can be easily converted into a DSM using tools available in commercial software like *Esri ArcGIS Pro* or using ready algorithms (Çağdaş Bak et al., 2016).

2 Proposed Method

Our approach consists of two phases: training and generation.

In the training phase, we fine-tuned SAM (Kirillov et al., 2023) architecture with Low-Rank Adaptation (LoRA) (Hu et al., 2021) using an extensive and realistic synthetic training set (refer to Section3.1). Here, SAM is fine-tuned to identify planar regions for given DSM data for specified building footprints. Details of the ground truth labeling schema and fine-tuning with LoRA are provided in Section 2.3.

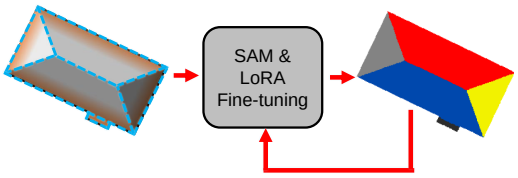


Figure 2. Training Phase.

In the generation phase, we first perform inference using the fine-tuned SAM on a given DSM and building footprint. Then, we apply robust plane fitting to estimate the parameters of each planar region identified by the SAM segmentation results within the building footprint (refer to Section 2.4). Finally, the planar regions obtained through SAM are refined, and accurate and simplified 3D building model is generated using the DSM data, building footprint, refined planar regions, and estimated plane equations (refer to Section 2.5).

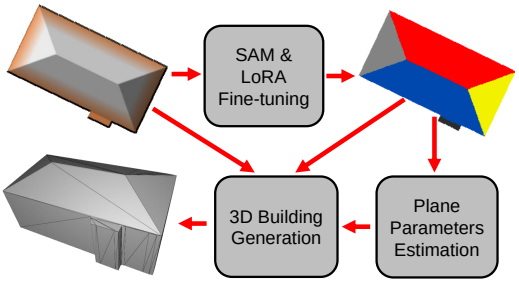


Figure 3. Generation Phase.

2.1 Segment Anything Model (SAM)

Segment Anything Model (SAM), is a foundational model in computer vision, aimed at enhancing image segmentation through promptable instructions (Kirillov et al., 2023). The Vision Transformer (ViT), a core component, revolutionizes the way images are processed. SAM is available in three versions: ViT-B (base), ViT-L (large), and ViT-H (huge), each differing in scale and capacity to handle complex segmentation tasks.

Basically, SAM integrates three core components:

• Image encoder: Extracts and transforms images into a feature space for segmentation by encoding key visual details.

• Prompt encoder: Processes given textual prompts to define segmentation tasks in natural language. This component links textual commands with visual information, ensuring alignment between user instructions and image content.

• Mask decoder: Combines both encoder outputs in order to generate accurate segmentation masks. Then it integrates visual and textual information to accurately delineate target areas in images following user prompts.

2.2 Low-Rank Adaptation (LoRA)

SAM shows promise in general segmentation tasks but it still requires fine-tuning for targeted applications, which can be data-intensive and costly. This requirement has rendered such processes impractical for individuals and small organizations, leading to the rise of parameter-efficient fine-tuning (PEFT) techniques. PEFT approaches, such as Low Rank Adaptation (LoRA), focuses on optimizing a minimal set of parameters in large models, reducing data needs and costs significantly. This approach has gained interest, evidenced by numerous studies on LoRA-based fine-tuning.

Low-Rank Adaptation (LoRA) enables the efficient training of certain dense layers in a neural network by focusing on the optimization of rank decomposition matrices. It targets the modifications within the dense layers during adaptation, while maintaining the pre-trained weights in a frozen state (Hu et al., 2021). Diagram given in Figure 4 shows how LoRA works for fine tuning a large-scale pre-trained model.

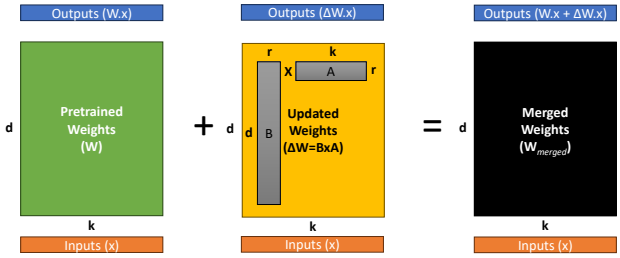


Figure 4. LoRA Diagram.

In above Figure 4, the pre-trained weight matrix is denoted as $\mathbf{W} \in \mathbb{R}^{d\times k}$. Consider $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$ matrices to be the low-rank decomposition matrices. Note that, the rank $r \ll \min(d, k)$ signifies that r is much less than the minimum of d and k. During training, $\mathbf{W}$ remains frozen, while $\mathbf{A}$ and $\mathbf{B}$ are updated. The composition of $\mathbf{A}$ and $\mathbf{B}$ defines the resultant weight matrix, as specified in Equation 1.

$$\begin{aligned} \Delta\mathbf{W} &= \mathbf{BA} \\ \mathbf{W_{merged}} &= \mathbf{W} + \Delta\mathbf{W} \end{aligned} \tag{1}$$

where $\mathbf{W}$ is the frozen weight matrix of the pretrained model, $\mathbf{A}$ and $\mathbf{B}$ denote the low-rank matrices, $\Delta\mathbf{W}$ corresponds to the weights adjusted during fine-tuning, and $\mathbf{W_{merged}}$ is the resultant weight matrix after merging.

We can apply multiplication to both $\mathbf{W}$ and $\Delta\mathbf{W}$ with the input vector x. The aggregate of these products yields the resultant output vector h, as delineated by Equation 2.

$$h = \mathbf{W}x + \Delta\mathbf{W}x \tag{2}$$

where x is input vector, and h is output vector.

As suggested by Hu et al. (Hu et al., 2021), initializing $\mathbf{A}$ with random Gaussian values and $\mathbf{B}$ with zeros is a viable strategy. Consequently, $\Delta\mathbf{W}$ starts as zero at the onset of training. This change in weight, $\Delta\mathbf{W}$, can be adjusted by scaling it with $\frac{\alpha}{r}$, where α represents a constant and r denotes rank. By setting α to match the initial rank r value and maintaining it constant throughout training, it becomes possible to minimize the loss for adjusting hyperparameters.

2.3 SAM with LoRA for Roof Segmentation

Given its efficacy in adapting Large Language Models (LLMs) through PEFT, the LoRA can also be used for fine-tuning SAM. We inspired from (Zhang and Liu, 2023) for this study. LoRA layers are added to the ViT blocks in the SAM image encoder. Although the image encoder itself is frozen these additional LoRA layers are trainable and can be used for fine-tuning SAM to our customized segmentation. Default embeddings are used in prompt encoder, so no prompt engineering is needed for the inferencing. The simplified diagram of the implementation of LoRA layers into the SAM is shown in Figure 5.

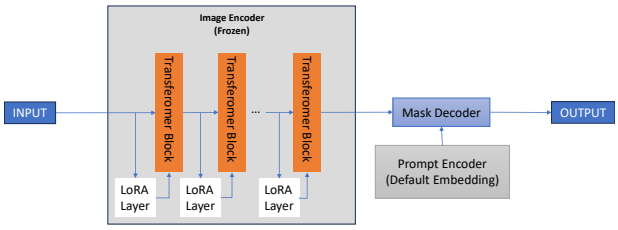


Figure 5. SAM+LoRA Diagram.

Incorporating LoRA into the projection layers of query (q), key (k), or value (v) uses the multi-head self-attention mechanism's use of cosine similarity to refine attention scores. In our case, LoRA is specifically utilized within the projection layers for q and v. Finally, applying Equation 3 to these layers results in the following equations:

$$\begin{aligned} \mathbf{Q} &= \mathbf{W_q}x + \mathbf{B_q}\mathbf{A_q}x \\ \mathbf{K} &= \mathbf{W_k}x \\ \mathbf{V} &= \mathbf{V_v}x = \mathbf{W_v}x + \mathbf{B_v}\mathbf{A_v}x \end{aligned} \tag{3}$$

where $\mathbf{W_q}$, $\mathbf{W_k}$, and $\mathbf{W_v}$ represent the frozen projection matrices in SAM, whereas $\mathbf{A_q}$, $\mathbf{B_q}$, $\mathbf{A_v}$, and $\mathbf{B_v}$ are the matrices trained as LoRA parameters. The variable x denotes the input vector, and q, k, v correspond to the output vectors.

We classified roof planes into semantic classes by aspect angles, focusing on flat, gable, and hip roofs (Figure 6). Class 1 to 4 are assigned for sloped roofs. Class 1 encompasses roof planes with aspect angles less than $45°$ or equal to or greater than $315°$, indicating a Northwest-to-Northeast (NW-NE) orientation. Class 2 includes planes with aspect angles ranging from $45°$ (inclusive) to less than $135°$, corresponding to a Northeast-to-Southeast (NE-SE) orientation. Planes within Class 3 have aspect angles from $135°$ (inclusive) to less than $225°$, representing a Southeast-to-Southwest (SE-SW) orientation. Finally, Class 4 consists of planes with aspect angles from $225°$ (inclusive) to less than $315°$, denoting a Southwest-to-Northwest (SW-NW) orientation. Flat roofs are identified with consistent aspect angles across the plane, classified under Class 5.

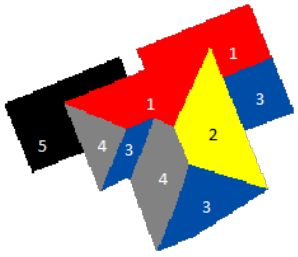


Figure 6. Semantic classification.

Similar to SAM, our method also predicts multiple segmentation masks (instances) each corresponding to a unique roof plane region class. We denote C as the set of roof plane region instances, for which we have established 50 distinct instances. Our objective is to generate a segmentation map where each pixel belongs to a class in the set $C = c_0, c_1, c_2, \ldots, c_k$. Here, c_0 is designated for the background, while c_i, for $i \in 1, \ldots, k$, represents the 50 predefined instances. It is assumed that every roof plane region assigned to the same semantic class may have 10 different instances on the same building, both in training and in inference. For a class z where $1 \leq z \leq 5$, plane regions are labeled from $100z+1$ to $100z+10$. For instance, Class 1 plane regions are labeled from 101 to 110 (Table 1). At training time, the planes are sorted by area, with instance labels assigned from largest to smallest within each semantic class, ranging from 1 to 50 and then remapped to the original values for inference.

Class	Class Definition	Labels
1	$315 \leq \gamma \leq 315+90$	$101, 102, \ldots, 110$
2	$45 \leq \gamma < 45+90$	$201, 202, \ldots, 210$
3	$135 \leq \gamma < 135+90$	$301, 302, \ldots, 310$
4	$225 \leq \gamma < 225+90$	$401, 402, \ldots, 410$
5	Flats	$501, 502, \ldots, 510$

Table 1. Semantic classes and labels (γ = aspect angle).

In our study, the DSM raster in the training dataset undergoes per-building per-tile normalization, scaling values to the range between 0 and 1 via the min-max method. Figure 7 shows sample tiles of the image and label data used in training. Figure 7(a) represents the normalized DSM tile, while Figure 7(b) depicts its corresponding label tile. The numbers on the label tile shows the labels of the roof plane regions. The numbers in parenthesis shows the mapped labels used during the training.

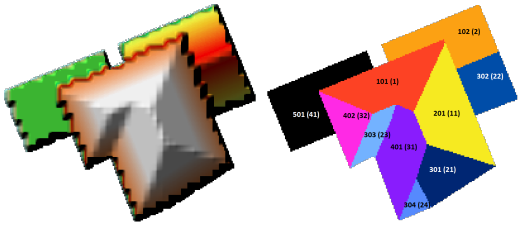


Figure 7. A training sample: normalized DSM tile (one with shaded relief symbology), and the matching ground-truth labels.

We used 7520 buildings in TrainVille and 108 buildings in TestVille, all synthetic (Section 3.1.1), for training and testing, respectively. Roof plane vector data served as labels, with DSM images as inputs, assuming building footprints are given for DSM masking. Resolutions of DSM and label rasters were 10 cm, with tiles sized 1024×1024 pixels. As in (Razuvayevskaya et al., 2023), we utilize a dynamic batch processing strategy to enhance GPU efficiency and accelerate training. We set the Cross-Entropy weight to 0.2 and the Dice loss weight to 0.8. The initial learning rate is 10^{-4}, with a *warmup* period of 250 epochs, and a maximum of 500 epochs with an early stop at 160 epochs. Parameters β_1, β_2, and weight decay for the *AdamW* optimizer are 0.9, 0.999, and 0.1, following (Zhang and Liu, 2023). We don't use rotation and flip data augmentations, given the rotation-sensitive nature of our labeling and the ample availability of synthetically generated data. We adopt LoRA to fine-tune the frozen q and v projection layers of the transformer blocks. Starting with a rank value of 4 for the ViT-B backbone, we sequentially increased it through 5, 6, 7, to 8, where a significant improvement was noted, establishing 8 as the base rank value. Training experiments and comparisons were then extended to rank values of 16, 32, 64, 128, 256, and 512.

2.4 Robust Single Plane Fitting

The SAM, fine-tuned using LoRA, outputs planar regions as its segmentation result. However, these results are provided as mask images, which do not include the parameters of the planes. Consequently, it becomes necessary to estimate the plane parameters for each segmented region. Given that these regions may contain noise and outliers, the plane fitting method employed must be robust against such imperfections. Thus, we devised a plane fitting methodology that leverages the L_1-norm, known for its robustness to outliers. Also, we developed an efficient numerical minimization strategy for the cost function specifically tailored to robust single plane fitting.

Let us define robust plane fitting as minimization of the cost function $J(.)$ with respect to the plane parameter $\mathbf{p} = [d, a, b]^\top$, where $\mathbf{r}_i = \hat{z}_i - z_i = \mathbf{g}_i^\top \mathbf{p} - z_i$ is the residual for the input data $\mathbf{g}_i = [1, x_i, y_i]^\top$ and elevation z_i, and v_i is the weight factor. Weight factor of i^{th} residual is computed as $v_i = e^{-\frac{r_i}{\text{median}(r)}+\epsilon}$ where v_i gets closer to zero for large residuals, such as outliers. At last, the cost function is given by:

$$J(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n} v_i |r_i| \tag{4}$$

with respect to the plane parameter $\mathbf{p} = [d, a, b]^\top$.

First, we need to approximate the non-differentiable absolute function (Ozcan et al., 2016) as below:

$$v_i |r_i| \approx v_i \frac{(\mathbf{g}_i^\top \mathbf{p} - z_i)^2}{|\mathbf{g}_i^\top \tilde{\mathbf{p}} - z_i| + \epsilon} = h_i (\mathbf{g}_i^\top \mathbf{p} - z_i)^2 \tag{5}$$

where

$$h_i = \frac{v_i}{|\mathbf{g}_i^\top \tilde{\mathbf{p}} - z_i| + \epsilon} = \frac{e^{-\frac{\tilde{r}_i}{\text{median}(\tilde{r})}+\epsilon}}{|\mathbf{g}_i^\top \tilde{\mathbf{p}} - z_i| + \epsilon} \tag{6}$$

Note that, weight v_i for i^{th} data is computed using the median of the residuals where $v_i = e^{-\frac{\tilde{r}_i}{\text{median}(\tilde{r})}+\epsilon}$ and $\tilde{r}$ is proxy constant for the residuals.

Finally, cost function becomes:

$$J^{(k)}(\mathbf{p}) \approx \frac{1}{n}\sum_{i=1}^{n} h_i (\mathbf{g}_i^\top \mathbf{p} - z_i)^2 \tag{7}$$

$$= \frac{1}{n}(\mathbf{G}\mathbf{p} - \mathbf{z})^\top \mathbf{H} (\mathbf{G}\mathbf{p} - \mathbf{z}) \tag{8}$$

where $\mathbf{G}$ is the data matrix composed of $\mathbf{g}_i$ values and $\mathbf{z}$ is the vector form of z_i values.

Taking the derivative of the cost function $J^{(k)}(\mathbf{p})$ with respect to $\mathbf{p}$ and equalizing it to zero leads to an iterative solution where k is the iteration index:

$$\frac{\partial J^{(k)}(\mathbf{p})}{\partial \mathbf{p}} = \frac{1}{n} 2\mathbf{G}^\top \mathbf{H} (\mathbf{G}\mathbf{p} - \mathbf{z}) = 0 \tag{9}$$

$$\mathbf{G}^\top \mathbf{H} \mathbf{G}\mathbf{p} - \mathbf{G}^\top \mathbf{H}\mathbf{z} = 0 \tag{10}$$

$$\mathbf{G}^\top \mathbf{H} \mathbf{G}\mathbf{p} = \mathbf{G}^\top \mathbf{H}\mathbf{z} \tag{11}$$

$$\mathbf{A}\mathbf{p} = \mathbf{b} \tag{12}$$

where $\mathbf{A} = \mathbf{G}^\top \mathbf{H} \mathbf{G}$ and $\mathbf{b} = \mathbf{G}^\top \mathbf{H}\mathbf{z}$.

2.5 3D Building Model Generation

Our objective is to produceLOD2.2 building models (Biljecki et al., 2016). Prior to estimating plane equations and 3D extrusion, it is essential to undertake several preprocessing steps to clean the data. These steps are enumerated as follows:

• The output from the segmentation process, planar regions, is transformed into vector polygons. For this task, we utilize the *Raster to Polygon* tool available in the *ArcGIS Pro*.

• Vertices of the obtained polygons are processed through the *Simplify Shared Edges* tool in the *ArcGIS Pro* where the *Retain Effective Areas (Visvalingam-Whyatt)* algorithm is applied.

• Polygons with an area smaller than the threshold value of 0.1 square meters are removed.

• The *Fill Gaps* tool in the *ArcGIS Pro* is employed to close the spaces left by the removal of the polygons.

• Next, the vertices of the polygons are simplified again using the *Simplify Shared Edges* tool in the *ArcGIS Pro* using the *Retain Critical Points (Douglas-Peucker)* algorithm.

After obtaining the simplified polygons using refined regions, the plane parameters for each planar region are estimated. The exterior vertices of the plane polygons are projected onto the building footprint vector, and superfluous vertices are eliminated. Surface normals for the planes are computed, and adjacent polygons with similar orientations are combined. Finally, the planes undergo triangulation, and the building walls are formed by extending the roof planes downward to the ground level.

3 Experiments

For all training and testing tasks, a Linux-based machine equipped with a GV-100 graphics card with 32GB of vRAM was utilized. This system comprises an Intel® Xeon® W-2123 CPU with 4 hardware cores and 8 threads, operating at 3.60 GHz, with 64 GB of RAM.

3.1 Training and Test Data

Deep learning architectures rely heavily on extensive labeled datasets. However, collecting and annotating these datasets is resource-intensive. Utilizing synthetic data in machine learning offers several advantages such as reducing costs, improving accuracy, enabling scalability, and easing licensing restrictions.

• Cost-effectiveness: It is a cost-effective alternative to real-world data acquisition, reducing the need for time-consuming and expensive manual collection and annotation.

• Accuracy enhancement: It offers the ability to tailor and generate diverse datasets, capturing a broad spectrum of scenarios, including rare events difficult to obtain in real-world data.

• Scalability: Synthetic data generation is scalable, enabling quick and efficient production of large data volumes, ideal for areas with limited labeled data or needing frequent updates.

• Licensing constraints: Using real-world datasets can involve complex licensing, privacy laws, and data-sharing limits. Synthetic data bypasses these issues, avoiding privacy and intellectual property concerns.

3.1.1 Synthetic Data Generation Pipeline

A synthetic data production pipeline was developed, wherein the synthetically generated data is utilized as input for both training and testing phases. First, training and test datasets were procedurally created through Esri's ArcGIS CityEngine, utilizing Computer-Generated Architecture (CGA) rules. ArcGIS CityEngine is a sophisticated 3D modeling tool that employs procedural modeling to craft extensive, interactive, and immersive urban landscapes more efficiently compared to traditional modeling methods. The procedural modeling language employed in CityEngine, known as CGA, facilitates the generation of 3D urban settings and architectural structures. Buildings were produced featuring a mix of three roof types: flat, gable, and hip, as depicted in Figure 8. The procedural generation ensures that buildings exhibit features typical of four categories: high-rise, commercial, apartment, and residential.

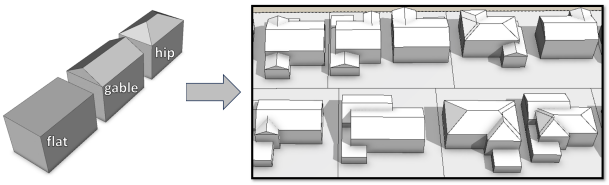


Figure 8. Procedurally generated roof types.

Two synthetic cities, TrainVille and TestVille (see Figure 9), with around 25,000 and 12,000 buildings respectively, were created using CityEngine and exported as multipatch geometry.

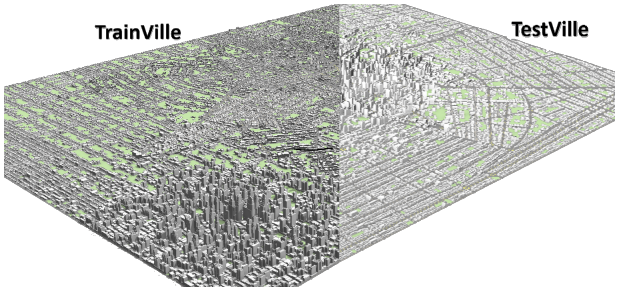


Figure 9. Synthetic cities (TrainVille & TestVille).

The *Esri ArcGIS Pro* Data Interoperability Extension, known as *FME Software*, was utilized to extract roof planes and construct as separate vector layers. Therefore, *FME Software* was utilized to extract roof planes and construct as separate vector layers (see Figure 10). At this stage, all necessary attributes (e.g., aspect angle, surface normals) were calculated, and an ID was assigned to the planes of each building.

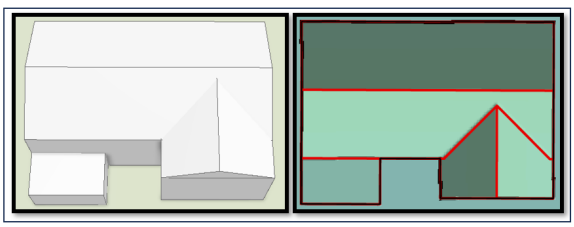


Figure 10. Extracted roof planes (red) and roof prints (black) with calculated attributes from multipatches (left image).

3.1.2 Real-world Data

Real-world test data were selected from Zurich[1], where a DSM with a 10 cm resolution for the test area was generated using photogrammetric methods from drone imagery. There are 756 buildings, each with its associated footprint vectors. A sample of this real-world test data is presented in Figure 11.

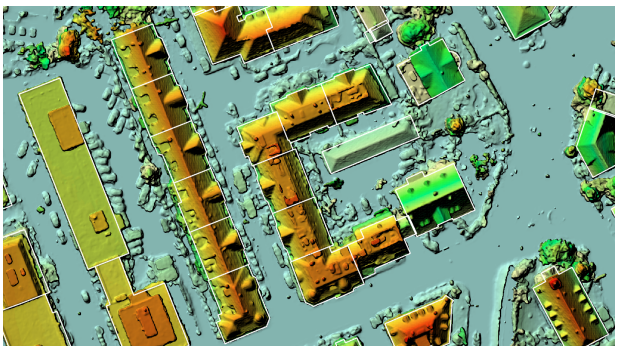


Figure 11. Real-world test data: City of Zurich.

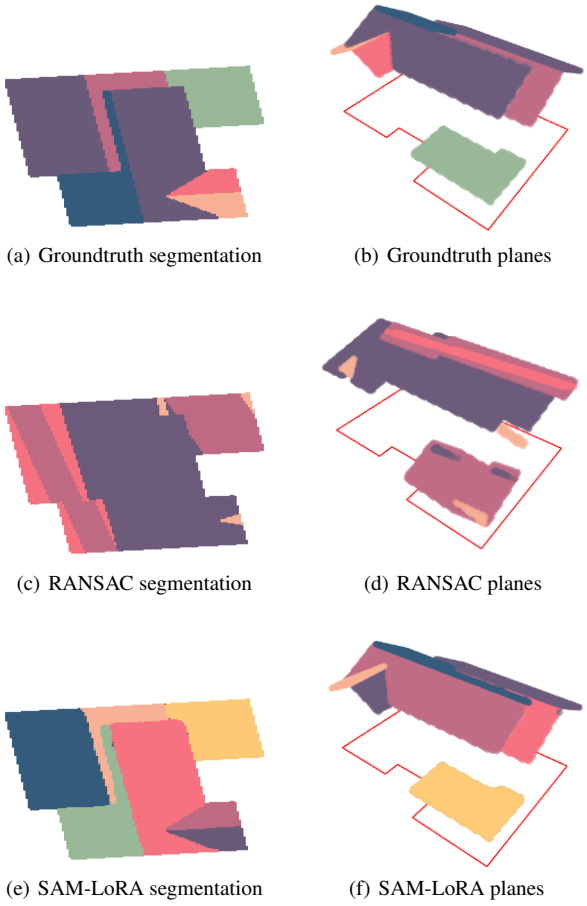
3.2 Quantitative Results

We tried different rank values to determine the best one for our study. We fine-tuned pre-trained SAM models: ViT-B (around 367 MB) and Vit-L (around 1.2 GB). The inferencing times for these 2 pre-trained backbone SAM models for one tile are respectively 0.5 seconds and 1 seconds. We don't use ViT-H (approximately 2.5 GB) for further experiments because it is too large and inferencing process time takes longer than the other two models (2 seconds).

To evaluate the performance of each trained model, we compute $F1$-score, precision, recall, and accuracy metrics for two different epochs: 160 and 500, respectively. These metrics were calculated using the synthetic TestVille data, which contained ground truth labels. We also assess the performance of planar region segmentation using the Jaccard Index, Intersection over Union (IoU), which is computed as $IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$ where A represents the ground truth labels, and B is the predicted planar segmentation masks.

We assess our method's planar region detection against the RANSAC algorithm using 108 synthetic buildings in TestVille (see example results in Figure 12). The process involves extracting unique labels from the ground truth and predicted masks, then calculating the IoU for each ground truth label against its predicted counterpart by finding the intersection and union of pixels for each label pair. The aggregated IoU scores provide a measure of the overall agreement between the ground truth and predicted masks for planar regions. RANSAC's inherent randomness produced variations between different runs as illustrated in Figure 13, which shows consecutive RANSAC trials. Its effectiveness also depends significantly on chosen hyperparameters. In our setup, we've configured RANSAC with a maximum of 750 iterations and a distance threshold of 0.6, using the L_1 distance metric to calculate distances between ground truth and predicted z values. Our SAM-LoRA based method significantly outperforms the RANSAC approach in segmenting planar regions, achieving a Jaccard Index of 0.96 compared to RANSAC's 0.66.

[1] Data courtesy of Wingtra



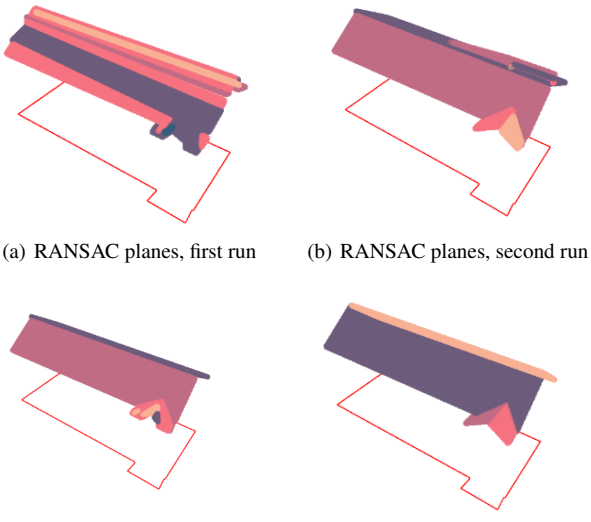
(a) Groundtruth segmentation (b) Groundtruth planes

(c) RANSAC segmentation (d) RANSAC planes

(e) SAM-LoRA segmentation (f) SAM-LoRA planes

Figure 12. Test sample results: RANSAC and SAM-LoRA

As seen in Figure 13, results of the RANSAC method exhibit variation across different executions, underscoring its intrinsic randomness. This randomness arises from the approach of RANSAC, which involves the random selection of subsets from the dataset for iterative model fitting. This random selection process, while enabling RANSAC to robustly handle outliers, introduces variability in the results.



(a) RANSAC planes, first run (b) RANSAC planes, second run

(c) RANSAC planes, third run (d) Groundtruth planes

Figure 13. Results of multiple RANSAC runs.

The performance metrics for 160 and 500 epochs, during fine-tuning with LoRA across various ranks for ViT-L and ViT-BB, are given in Table 2. Higher rank values correspond to larger LoRA models. Additionally, the predicted instance numbers per class detail some missing instances, primarily due to their scarce or small area presence in the ground truth. LoRA model sizes substantially increases at higher ranks (256, 512) when using the ViT-L pretrained model. The analysis suggests that the rank 128 LoRA model trained for 500 epochs with the Vit-L pretrained SAM model achieves the best performance.

Table 2. Comparison of fine-tuning SAM with different ranks.

Rank (r) / File size (ViT-L)	Epochs = 160						Epochs = 500					
	Classes / Instances	Predicted Instances	f1	Precision	Recall	Accuracy	Classes / Instances	Predicted Instances	f1	Precision	Recall	Accuracy
r = 8	1/4	3	0.9661	0.9429	0.9905	0.9345	1/4	4	0.9703	0.9501	0.9914	0.9424
	2/6	3	0.9804	0.9889	0.9721	0.9616	2/6	3	0.9837	0.9879	0.9795	0.9679
	3/4	3	0.9661	0.9827	0.9500	0.9344	3/4	3	0.9704	0.9879	0.9535	0.9425
44885 KB	4/4	3	0.9826	0.9825	0.9827	0.9658	4/4	3	0.9856	0.9860	0.9851	0.9716
	5/6	5	0.9961	0.9982	0.9940	0.9923	5/6	5	0.9964	0.9983	0.9945	0.9928
	Total = 24	**All = 17**	**0.9783**	**0.9790**	**0.9779**	**0.9577**	**Total = 24**	**All = 18**	**0.9813**	**0.9820**	**0.9808**	**0.9634**
r = 16	1/4	3	0.9691	0.9491	0.9900	0.9401	1/4	3	0.9728	0.9550	0.9911	0.9469
	2/6	3	0.9827	0.9889	0.9766	0.9660	2/6	3	0.9860	0.9891	0.9829	0.9723
	3/4	3	0.9696	0.9840	0.9555	0.9409	3/4	3	0.9724	0.9876	0.9577	0.9464
	4/4	3	0.9864	0.9868	0.9860	0.9732	4/4	3	0.9882	0.9898	0.9866	0.9766
47957 KB	5/6	5	0.9958	0.9985	0.9930	0.9916	5/6	5	0.9948	0.9984	0.9911	0.9896
	Total = 24	**All = 17**	**0.9807**	**0.9815**	**0.9802**	**0.9624**	**Total = 24**	**All = 17**	**0.9828**	**0.9840**	**0.9819**	**0.9664**
r = 32	1/4	3	0.9728	0.9584	0.9877	0.9470	1/4	4	0.9751	0.9590	0.9918	0.9515
	2/6	3	0.9847	0.9888	0.9806	0.9699	2/6	3	0.9859	0.9863	0.9855	0.9722
	3/4	3	0.9719	0.9867	0.9576	0.9454	3/4	3	0.9739	0.9893	0.9589	0.9491
	4/4	3	0.9874	0.9867	0.9880	0.9750	4/4	3	0.9878	0.9910	0.9845	0.9758
54101 KB	5/6	5	0.9959	0.9984	0.9934	0.9919	5/6	5	0.9963	0.9981	0.9944	0.9925
	Total = 24	**All = 17**	**0.9825**	**0.9838**	**0.9814**	**0.9658**	**Total = 24**	**All = 18**	**0.9838**	**0.9848**	**0.9830**	**0.9682**
r = 64	1/4	3	0.9667	0.9449	0.9895	0.9355	1/4	4	0.9723	0.9546	0.9908	0.9462
	2/6	3	0.9720	0.9857	0.9586	0.9454	2/6	3	0.9778	0.9883	0.9674	0.9565
	3/4	3	0.9679	0.9857	0.9508	0.9378	3/4	3	0.9722	0.9841	0.9606	0.9459
	4/4	3	0.9866	0.9891	0.9841	0.9736	4/4	3	0.9887	0.9905	0.9870	0.9777
66390 KB	5/6	5	0.9948	0.9959	0.9938	0.9897	5/6	5	0.9953	0.9964	0.9941	0.9906
	Total = 24	**All = 17**	**0.9776**	**0.9802**	**0.9754**	**0.9564**	**Total = 24**	**All = 18**	**0.9813**	**0.9828**	**0.9800**	**0.9634**
128	1/4	4	0.9671	0.9439	0.9915	0.9363	1/4	4	0.9751	0.9578	0.9930	0.9514
	2/6	3	0.9816	0.9840	0.9792	0.9639	2/6	3	0.9877	0.9892	0.9861	0.9757
	3/4	3	0.9684	0.9887	0.9490	0.9388	3/4	3	0.9759	0.9908	0.9614	0.9529
	4/4	3	0.9863	0.9914	0.9813	0.9729	4/4	3	0.9890	0.9915	0.9864	0.9782
90966 KB	5/6	5	0.9955	0.9975	0.9934	0.9910	5/6	5	0.9966	0.9981	0.9950	0.9931
	Total = 24	**All = 18**	**0.9798**	**0.9811**	**0.9789**	**0.9606**	**Total = 24**	**All = 18**	**0.9848**	**0.9855**	**0.9844**	**0.9703**
r = 256	1/4	3	0.9702	0.9535	0.9876	0.9422	1/4	4	0.9724	0.9533	0.9924	0.9464
	2/6	3	0.9843	0.9816	0.9870	0.9691	2/6	3	0.9867	0.9871	0.9863	0.9738
	3/4	3	0.9693	0.9908	0.9486	0.9403	3/4	3	0.9726	0.9908	0.9551	0.9467
	4/4	3	0.9864	0.9891	0.9837	0.9732	4/4	3	0.9878	0.9901	0.9854	0.9758
140118 KB	5/6	5	0.9961	0.9981	0.9942	0.9923	5/6	5	0.9960	0.9984	0.9937	0.9920
	Total = 24	**All = 17**	**0.9813**	**0.9826**	**0.9802**	**0.9634**	**Total = 24**	**All = 18**	**0.9831**	**0.9839**	**0.9826**	**0.9669**
r = 512	1/4	3	0.9714	0.9525	0.9911	0.9445	1/4	4	0.9728	0.9535	0.9929	0.9471
	2/6	3	0.9854	0.9890	0.9817	0.9711	2/6	3	0.9860	0.9856	0.9863	0.9723
	3/4	3	0.9694	0.9903	0.9494	0.9406	3/4	3	0.9729	0.9905	0.9560	0.9473
	4/4	3	0.9870	0.9857	0.9883	0.9743	4/4	3	0.9871	0.9905	0.9836	0.9745
238422 KB	5/6	5	0.9935	0.9984	0.9887	0.9871	5/6	5	0.9948	0.9982	0.9914	0.9896
	Total = 24	**All = 17**	**0.9813**	**0.9832**	**0.9798**	**0.9635**	**Total = 24**	**All = 18**	**0.9827**	**0.9837**	**0.9821**	**0.9662**

Vit-L SAM model fine-tuned with different ranks.

Rank (r) / File size (ViT-B)	Epochs = 160						Epochs = 500					
	Classes / Instances	Predicted Instances	f1	Precision	Recall	Accuracy	Classes / Instances	Predicted Instances	f1	Precision	Recall	Accuracy
r = 8	1/4	3	0.9487	0.9242	0.9745	0.9024	1/4	3	0.9641	0.9388	0.9908	0.9307
	2/6	3	0.9713	0.9851	0.9580	0.9442	2/6	3	0.9820	0.9857	0.9783	0.9646
	3/4	3	0.9536	0.9878	0.9217	0.9113	3/4	3	0.9634	0.9888	0.9393	0.9294
	4/4	3	0.9814	0.9815	0.9813	0.9635	4/4	3	0.9839	0.9842	0.9836	0.9683
42950 KB	5/6	5	0.9924	0.9953	0.9895	0.9848	5/6	5	0.9952	0.9984	0.9920	0.9904
	Total = 24	**All = 17**	**0.9695**	**0.9748**	**0.9650**	**0.9412**	**Total = 24**	**All = 17**	**0.9777**	**0.9792**	**0.9768**	**0.9567**
r = 16	1/4	3	0.9578	0.9294	0.9880	0.9190	1/4	3	0.9654	0.9421	0.9899	0.9331
	2/6	3	0.9782	0.9779	0.9786	0.9574	2/6	3	0.9821	0.9816	0.9826	0.9648
	3/4	3	0.9502	0.9895	0.9139	0.9051	3/4	3	0.9638	0.9890	0.9398	0.9301
	4/4	3	0.9816	0.9830	0.9802	0.9638	4/4	3	0.9847	0.9873	0.9821	0.9699
44102 KB	5/6	5	0.9949	0.9985	0.9913	0.9898	5/6	5	0.9959	0.9982	0.9936	0.9918
	Total = 24	**All = 17**	**0.9725**	**0.9757**	**0.9704**	**0.9470**	**Total = 24**	**All = 17**	**0.9784**	**0.9796**	**0.9776**	**0.9579**
r = 32	1/4	3	0.9657	0.9406	0.9920	0.9336	1/4	3	0.9634	0.9375	0.9907	0.9293
	2/6	2	0.9750	0.9877	0.9626	0.9512	2/6	3	0.9775	0.9854	0.9697	0.9560
	3/4	3	0.9657	0.9873	0.9451	0.9337	3/4	3	0.9662	0.9856	0.9476	0.9346
	4/4	3	0.9849	0.9881	0.9818	0.9703	4/4	3	0.9874	0.9899	0.9850	0.9752
46406 KB	5/6	5	0.9913	0.9962	0.9864	0.9827	5/6	5	0.9954	0.9976	0.9932	0.9908
	Total = 24	**All = 16**	**0.9765**	**0.9800**	**0.9736**	**0.9543**	**Total = 24**	**All = 17**	**0.9780**	**0.9792**	**0.9772**	**0.9572**
r = 64	1/4	3	0.9716	0.9570	0.9866	0.9448	1/4	4	0.9621	0.9343	0.9916	0.9269
	2/6	3	0.9839	0.9843	0.9835	0.9683	2/6	3	0.9805	0.9849	0.9761	0.9617
	3/4	3	0.9668	0.9868	0.9477	0.9358	3/4	3	0.9684	0.9889	0.9487	0.9387
	4/4	3	0.9851	0.9862	0.9840	0.9707	4/4	3	0.9859	0.9892	0.9827	0.9723
51014 KB	5/6	5	0.9951	0.9982	0.9920	0.9902	5/6	5	0.9955	0.9982	0.9927	0.9910
	Total = 24	**All = 17**	**0.9805**	**0.9825**	**0.9788**	**0.9620**	**Total = 24**	**All = 18**	**0.9785**	**0.9791**	**0.9784**	**0.9581**
128	1/4	3	0.9700	0.9496	0.9914	0.9418	1/4	3	0.9716	0.9517	0.9924	0.9448
	2/6	3	0.9841	0.9832	0.9849	0.9686	2/6	3	0.9858	0.9862	0.9855	0.9721
	3/4	3	0.9670	0.9890	0.9460	0.9362	3/4	3	0.9717	0.9921	0.9520	0.9449
	4/4	2	0.9854	0.9898	0.9810	0.9712	4/4	3	0.9867	0.9889	0.9845	0.9738
60230 KB	5/6	5	0.9962	0.9982	0.9941	0.9923	5/6	5	0.9964	0.9982	0.9946	0.9928
	Total = 24	**All = 16**	**0.9805**	**0.9795**	**0.9795**	**0.9620**	**Total = 24**	**All = 17**	**0.9824**	**0.9834**	**0.9818**	**0.9657**
r = 256	1/4	3	0.9721	0.9614	0.9830	0.9457	1/4	3	0.9703	0.9489	0.9926	0.9423
	2/6	3	0.9828	0.9759	0.9898	0.9661	2/6	3	0.9865	0.9872	0.9857	0.9733
	3/4	3	0.9699	0.9912	0.9496	0.9416	3/4	3	0.9720	0.9909	0.9538	0.9455
	4/4	3	0.9853	0.9884	0.9823	0.9711	4/4	3	0.9879	0.9914	0.9845	0.9761
78662 KB	5/6	5	0.9962	0.9981	0.9942	0.9924	5/6	5	0.9946	0.9984	0.9907	0.9892
	Total = 24	**All = 17**	**0.9813**	**0.9830**	**0.9798**	**0.9634**	**Total = 24**	**All = 17**	**0.9822**	**0.9834**	**0.9815**	**0.9653**
r = 512	1/4	3	0.9681	0.9459	0.9913	0.9381	1/4	3	0.9720	0.9522	0.9927	0.9456
	2/6	3	0.9842	0.9816	0.9868	0.9689	2/6	3	0.9857	0.9848	0.9866	0.9719
	3/4	3	0.9679	0.9912	0.9457	0.9378	3/4	3	0.9723	0.9917	0.9537	0.9461
	4/4	3	0.9857	0.9920	0.9795	0.9718	4/4	3	0.9869	0.9913	0.9826	0.9742
115526 KB	5/6	5	0.9948	0.9985	0.9911	0.9896	5/6	5	0.9919	0.9982	0.9856	0.9839
	Total = 24	**All = 17**	**0.9801**	**0.9819**	**0.9789**	**0.9612**	**Total = 24**	**All = 17**	**0.9818**	**0.9836**	**0.9802**	**0.9643**

Vit-B SAM model fine-tuned with different ranks.

We compare constructed 3D buildings (for ViT-L, Rank 128, 500 epochs) on synthetic TestVille and real Zurich test sites. Random checkpoints ranging from 30 to 100 are generated on these 3D buildings. An initial blunder test eliminates outlier checkpoints before calculating RMSE, mean error, and median value for height discrepancies. These calculated values are presented in Table 3. Despite the model's training on synthetic data without real-world noise, height errors are minimal for both the synthetic and the real-world Zurich datasets.

Test Data	RMSE (m)	Mean (m)	Median (m)
TestVille	0.103	0.0006	2.2×10^{-12}
Zurich	0.67	0.0002	0.002

Table 3. Generated 3D building heights versus DSM heights.

3.3 Qualitative Results

Our approach generates LOD2 building models through planar region detection and robust plane fitting, followed by constructing 3D models from these roof planes (Figure 14).

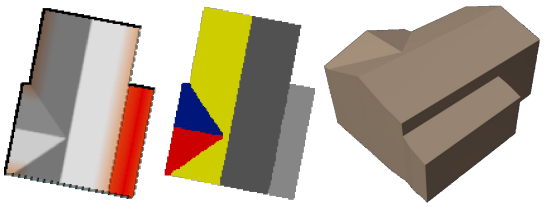


Figure 14. A constructed 3D model from synthetic TestVille.

Testing on noisy real-world Zurich dataset also validates our algorithm's proficiency in addressing real-world complexities effectively (Figure 15 and Figure 16). Although the 3D models could be further improved by post-processing the segmentation mask of planar regions, our approach already reliably yields accurate LOD2.2 representations of buildings.

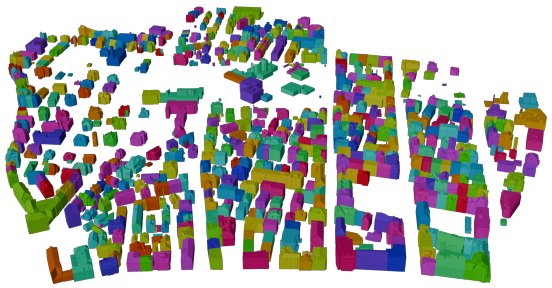


Figure 15. 3D building reconstructions of a piece of Zurich city.

Performance of the proposed method for a small piece of Zurich city is illustrated in Figure 15. Also, a more detailed view of a building within this area is shown in Figure 16.

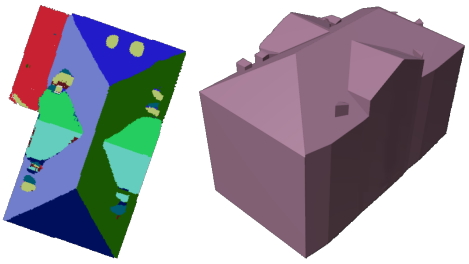


Figure 16. A constructed 3D model from the city of Zurich.

4 Conclusion

In this paper, we introduce a novel method for mapping planar regions within given DSM and building footprint, utilizing a fine-tuned SAM. We apply the LoRA for fine-tuning the SAM, which enables precise detection of planar areas within given building boundaries. Furthermore, we have crafted a robust plane fitting method that employs the L_1-norm to estimate the plane parameters of the planar regions produced by the fine-tuned SAM. Our experiments show that our approach surpasses conventional RANSAC-based approaches in segmenting planar regions on building footprints from DSMs. This superiority is reflected through a higher Jaccard index along with providing consistent and replicable outcomes. Our approach mitigates the inherent randomness of RANSAC by leveraging a SAM variant enhanced with LoRA and a unique labeling technique, ensuring precise and uniform results for the same DSM inputs. Even though trained on synthetic data, our approach's effectiveness has also been confirmed with real-world Zurich dataset, showcasing significant results.

In future research, we aim to enhance our model's training by integrating both noisy synthetic and diverse real-world datasets. We will also explore reorienting the buildings to align with the standard canonical axis, aiming to enhance the effectiveness of our labeling schema. We also plan to increase segmentation precision by merging adjacent planar regions with similar plane equations and dividing regions that require differentiation into multiple planes. This approach aims to better capture the complexities of real-world structures. Additionally, we will focus on refining the delineation of planar regions through an optimization-based method that considers plane intersections, aiming for a higher fidelity to the original data and more accurate physical world representations. Furthermore, we plan to employ computational geometry libraries to support the generation of simpler, yet more precise, 3D building models.

References

Aktaş, G., Nar, F., Vural, F. T. Y., 2018. Correlation-based variational change detection for elevation models. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 1930–1933.

Biljecki, F., Ledoux, H., Stoter, J., 2016. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59, 25-37.

Deschaud, J.-E., Goulette, F., 2010. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing.

Duda, R. O., Hart, P. E., 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1), 11-15.

Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395.

Hough, P. V. C., 1962. Method and means for recognizing complex patterns. U.S. Patent 3,069,654.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., 2021. Lora: Low-rank adaptation of large language models.

Jolliffe, I. T., Cadima, J., 2016. Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng Sci.*, 374(2065), 303-309.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., Girshick, R., 2023. Segment anything.

Li, Y., Gong, P., Babu, M., 2005. Automated 3D building geometrical moeling from DSM.

Liu, Y., Obukhov, A., Wegner, J. D., Schindler, K., 2024. Point2Building: Reconstructing buildings from airborne lidar point clouds.

Nan, L., Wonka, P., 2017. Polyfit: Polygonal surface reconstruction from point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2353–2361.

Nar, F., Yilmaz, E., Camps-Valls, G., 2018. Sparsity-driven digital terrain model extraction. *2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 1316–1319.

Nurunnabi, A., Belton, D., West, G., 2014. Robust statistical approaches for local planar surface fitting in 3D laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 96, 106-122.

Ochmann, S., Vock, R., Wessel, R., Klein, R., 2016. Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54, 94-103.

Ozcan, C., Sen, B., Nar, F., 2016. Sparsity-Driven Despeckling for SAR Images. *IEEE Geoscience and Remote Sensing Letters*, 13(1), 115-119.

Razuvayevskaya, O., Wu, B., Leite, J., Heppell, F., Srba, I., Scarton, C., Bontcheva, K., Song, X., 2023. Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification.

Rottensteiner, F., Briese, C., 2002. A new method for building extraction in urban areas from high-resolution LiDAR data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34.

Wang, R., Peethambaran, J., Chen, D., 2018. LiDAR Point Clouds to 3-D Urban Models: A Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(2), 606-627.

Zhang, K., Liu, D., 2023. Customized segment anything model for medical image segmentation.

Zhou, Q., 2017. *Digital Elevation Model and Digital Surface Model*. John Wiley Sons, Ltd, 1–17.

Çağdaş Bak, Ergül, M., Aktaş, G., Nar, F., 2016. DSM extraction using variational methods. *Signal Processing and Communication Application Conference (SIU)*, 265–268.