ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Differentiable Inner Product Transform for 3D Reconstruction

Bachelor Thesis

Tobias Robert Eeksman

May 15, 2026

Advisors: Prof. Dr. P. Biran, Prof. Dr. B. Rieck
Department of Mathematics, ETH Zürich

Abstract

Automated 3D building reconstruction from airborne LiDAR point clouds is difficult because of architectural diversity and varying point densities. Standard volumetric neural networks often use local loss functions that ignore how a building's structure is connected globally. In this thesis, we test if adding a differentiable Inner Product Transform (IPT) loss term can work as a structural regularizer for a 3D U-Net reconstruction pipeline.

Our experimental results on the Zürich LoD2 building dataset show that the IPT loss term can lead to marginal improvements in specific configurations, such as our Res30 model. However, we find that the 64^3 grid resolution creates a limit for the accuracy. Most of the recorded error comes from turning the buildings into voxels rather than the performance of the model. We find that regularized models require significantly more iterations than a standard baseline to reach these error levels. We also identify a qualitative trade-off where the global nature of the IPT constraints, when applied to discrete 64^3 grids, can lead to curved architectural features. This work provides an analysis of these limitations and characterizes the optimization challenges involved in balancing differentiable topological transforms with volumetric occupancy.

Contents

Chapter 1

Introduction

Turning raw urban sensor data into structured 3D models is a key task in fields like remote sensing and urban planning. Airborne Light Detection and Ranging (LiDAR) is the most common tool for this because it can capture the geometry of entire cities from above. However, turning these unstructured point clouds into clean, consistent 3D building models remains a difficult task.

1.1 Current Challenges and Motivation

Building reconstruction from LiDAR is difficult because of the high diversity in architectural designs and the fact that sensor data is often incomplete due to occlusions from trees or buildings [1].

Digital models of cities are needed for real-time applications like autonomous vehicle navigation. In these cases, 3D polygonal meshes are better than raw point clouds because they are compact and easier for computers to use. However, there is currently a trade-off between accuracy and speed. Modern generative models like Point2Building [1] are accurate but slow because they generate geometry one piece at a time. On the other hand, single-pass networks like the 3D U-Net are very fast but use local loss functions like Binary Cross-Entropy (BCE). Theoretically, BCE only looks at individual pixels and does not understand the overall shape of a building [2].

The motivation for this thesis is to see if we can get the best of both worlds. We want to see if a fast U-Net can be improved by adding a global structural guide. We investigate the Inner Product Transform (IPT), a method that uses inner products to encode the overall structure and geometry of point clouds [3]. Our goal is to see if adding an IPT-based loss term can give a simple U-Net the structural intelligence of more complex models without losing its inference speed.

1.2 Thesis Objective

The main goal of this thesis is to implement and test a differentiable Inner Product Transform (IPT) loss term. Unlike other topological methods that are very slow to compute, the IPT works directly on point coordinates. This makes it efficient enough to use during neural network training [4].

Our work focuses on four specific goals:

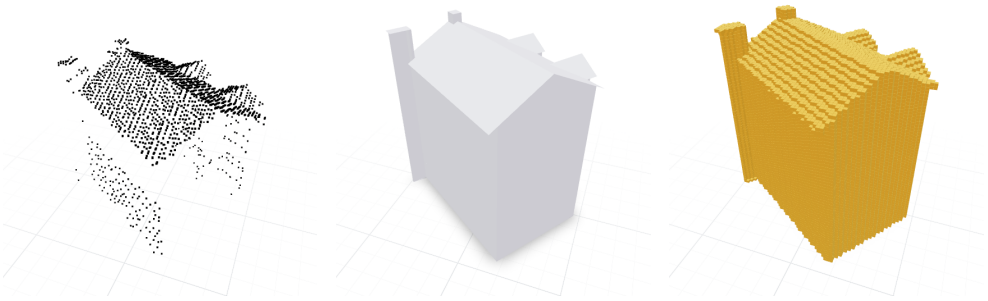
- **Implementation:** Adding a differentiable version of the IPT using a sigmoid-based approximation [5] into a 3D U-Net to allow the model to learn topological features through backpropagation.
- **Calibration:** Finding the right balance (λ_{topo}) between the standard voxel loss and the new topological loss to keep the training process stable.
- **Comparative Evaluation:** Testing the model on the Zürich LoD2 building dataset. We measure performance using geometric metrics like Mean Distance Error (MDE), Hausdorff distance, and Chamfer distance, and compare against other methods like City3D [6] and 2.5D Dual Contouring [7].
- **Analysis:** Examining the costs of this approach. Specifically, we analyze the regularization overhead, which is the fact that adding this loss makes training take much longer, and look at why global constraints sometimes cause oversmoothing on small voxel grids.

By the end of this study, we aim to show how the IPT affects a neural network's ability to reconstruct buildings and provide an honest look at the trade-offs between training time and structural accuracy.

Chapter 2

Theoretical Background

2.1 3D Representations



(a) Point Cloud **(b)** Polygon Mesh **(c)** Voxel Grid

Figure 2.1: Common 3D representations of a building: (a) discrete points sampled from surfaces, (b) continuous polygonal surfaces, and (c) a discretized volumetric occupancy grid.

2.1.1 Point Clouds

Point clouds are among the most fundamental and widely used data structures for representing three-dimensional geometry. Formally, a point cloud $\mathcal{P}$ is defined as a set of n discrete points in a d-dimensional Euclidean space. Since we are working in $\mathbb{R}^3$, we can write:

$$\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, \ldots, n\}$$

Each element $\mathbf{p}_i$ is a vector of Cartesian coordinates (x, y, z). In many modern applications, such as autonomous driving and robotics, point clouds are generated via LiDAR (Light Detection and Ranging) sensors. These sensors utilize Time-of-Flight (ToF) principles, where laser pulses are emitted and the distance to an object is calculated based on the time it takes for the reflected signal to come back.

Point clouds are considered an unstructured representation. Unlike grids or meshes, they do not explicitly encode topological connectivity or surface information. The data contains only spatial positions, meaning there is no inherent knowledge of which points constitute a single surface or the boundary between the interior and exterior of an object. This lack of structure necessitates the use of specialized algorithms to infer geometry and topology from the raw coordinates.

2.1.2 Voxels

A voxel, a word combination of "volumetric" and "pixel," represents a value on a regular grid in three-dimensional space. To discretize a continuous 3D volume into voxels, the space is subdivided into a series of uniform cubic cells. Mathematically, a voxel grid is represented as a 3D tensor $\mathbf{V} \in \{0,1\}^{L \times W \times H}$. The state of each voxel is determined by an occupancy function $o(\mathbf{x})$:

$$o(\mathbf{x}) = \begin{cases} 1 & \text{if the spatial coordinate } \mathbf{x} \text{ is occupied} \\ 0 & \text{otherwise} \end{cases}$$

In computational frameworks, voxels are typically stored as dense or sparse arrays. Their primary advantage lies in their structured nature, which allows for the direct application of 3D Convolutional Neural Networks (CNNs). However, voxels suffer from high memory consumption that scales cubically with resolution ($O(N^3)$).

2.1.3 Meshes and Abstract Simplicial Complexes

Polygon meshes are the standard representation for geometric modeling in computer graphics. A mesh $\mathcal{M}$ is defined by its geometry and its topology, often represented as $\mathcal{M} = (V, F)$, where V is a set of vertices in $\mathbb{R}^3$ and F is a set of faces that define the connectivity.

In a formal mathematical context, these are generalized as abstract simplicial complexes. A k-simplex σ is defined as the convex hull of $k+1$ affinely independent points $\{v_0, \ldots, v_k\}$. An abstract simplicial complex $\mathcal{K}$ is a finite collection of such simplices that satisfies two conditions:

1. $\forall \sigma \in \mathcal{K}$, if $\tau \subseteq \sigma$, then $\tau \in \mathcal{K}$.
2. $\forall \sigma_1, \sigma_2 \in \mathcal{K}$, if $\sigma_1 \cap \sigma_2 \neq \emptyset$, then $\sigma_1 \cap \sigma_2$ is a face of both σ_1 and σ_2.

By representing a building mesh as a simplicial complex, we can utilize algebraic topology to capture its global structure through discrete, coordinate-invariant components, such as identifying features such as holes or connected components.

2.1.4 Sublevel / Superlevel Set Filtrations

To extract topological information from a simplicial complex, we study its evolution across a sequence of nested subcomplexes, a process known as a filtration. A filtration of $\mathcal{K}$ is a sequence of subcomplexes

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = \mathcal{K}.$$

One way to obtain such a sequence is to construct the sublevel set filtrations. Given a continuous function $f : \mathcal{K} \to \mathbb{R}$ (often referred to as a filter or height function), the sublevel set for a height $h \in \mathbb{R}$ is defined as:

$$K_h = \{\sigma \in \mathcal{K} \mid f(\sigma) \leq h\} \tag{2.1}$$

As h increases from $-\infty$ to $+\infty$, we obtain a nested sequence of subcomplexes that capture the birth and merging of topological features. If we instead looked at $f(\sigma) \geq h$, we would get the superlevel set filtration.

2.2 Euler Characteristic Transform (ECT)

The Euler characteristic, denoted by χ, is a fundamental topological invariant that summarizes the shape of a simplicial complex independently of its geometric deformation. For a finite simplicial complex $\mathcal{K}$, $\chi(\mathcal{K})$ is defined as the alternating sum of the number of simplices in each dimension:

$$\chi(\mathcal{K}) = \sum_{n=0}^{d} (-1)^n |K^n| \tag{2.2}$$

where $|K^n|$ represents the cardinality of the set of n-dimensional simplices. In the context of 3D surfaces, this simplifies to the well-known formula $\chi = V - E + F$, where $V, E,$ and F denote the number of vertices, edges, and faces, respectively.

The Euler Characteristic Transform (ECT) extends this concept to capture geometric information by examining a shape from multiple perspectives. Let S^{n-1} denote the unit sphere in $\mathbb{R}^n$. Given a direction $\xi \in S^{n-1}$ and a height $h \in \mathbb{R}$, we define a height function f_ξ that maps each simplex $\sigma \in \mathcal{K}$ to its maximum projection along ξ: $f_\xi(\sigma) = \max_{v \in \sigma} \langle v, \xi \rangle$, where $\langle v, \xi \rangle$ denotes the Euclidean dot product. This allows for the construction of a sublevel set filtration:

$$\mathcal{K}_{\xi,h} = \{\sigma \in \mathcal{K} \mid f_\xi(\sigma) \leq h\} \tag{2.3}$$

Evaluating the Euler characteristic of each subcomplex $\mathcal{K}_{\xi,h}$ yields the Euler Characteristic Curve (ECC) associated with the direction ξ (see Figure 2.2). The ECC thus formally maps a simplicial complex $\mathcal{K}$, together with a

direction ξ and a threshold h, to the Euler characteristic of its corresponding subcomplex:

$$ECT : \mathcal{K} \times S^{n-1} \times \mathbb{R} \to \mathbb{Z} \tag{2.4}$$

$$(\mathcal{K}, \xi, h) \mapsto \chi(\mathcal{K}_{\xi,h}) \tag{2.5}$$

The ECT is the collection of these curves across all directions $\xi \in S^{n-1}$ (see Figure 2.3). A significant theoretical property of the ECT is its injectivity: it was originally proven that given an infinite number of directions, the transform uniquely characterizes the original shape [8]. While this result established the ECT as a powerful descriptor, practical applications rely on the more recent discovery that a finite number of directions is actually sufficient to ensure this uniqueness for many classes of shapes [9].

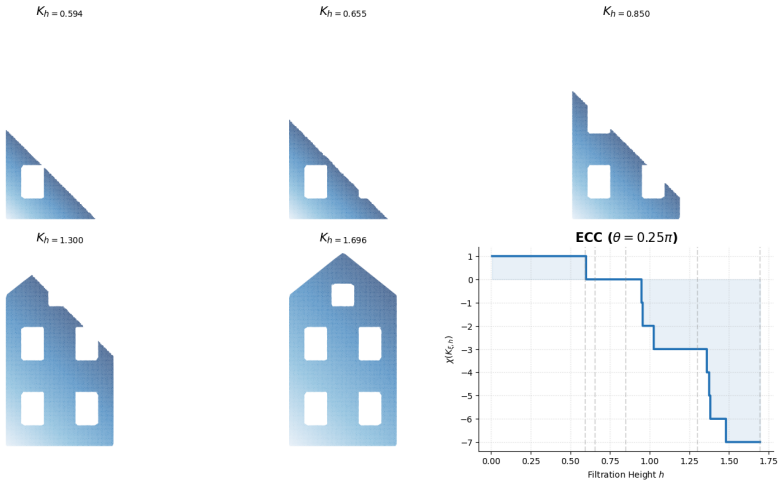


Figure 2.2: Diagonal filtration direction $(\theta = 0.25\pi)$ on a 2D building silhouette. The snapshots K_h illustrate the evolving subcomplexes as the filtration height h increases. Discrete jumps in the resulting ECC (bottom-right) occur at heights where a window opening becomes fully enclosed, creating a new hole that reduces the Euler characteristic χ.

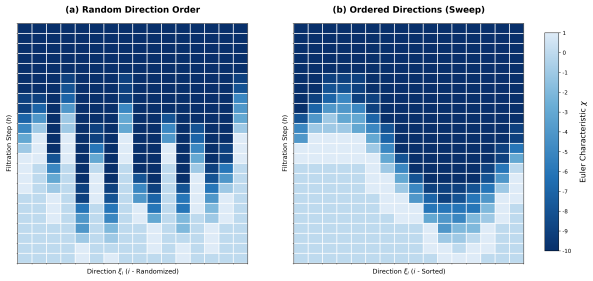


Figure 2.3: Discrete ECT matrix representations of the building silhouette (16×24).
(a) Illustrates a random ordering of direction indices, reflecting the unstructured nature of 3D direction sampling. (b) Shows the same data with ordered directions.

2.3 Inner Product Transform (IPT)

The Inner Product Transform (IPT) is a topological descriptor specifically optimized for point cloud data. Unlike the ECT, which requires the construction of a full simplicial complex $\mathcal{K}$, the IPT operates directly on the coordinates of a point set $X \subset \mathbb{R}^n$. This avoids the high computational cost associated with building higher-order simplices.

Given a point cloud $X \subset \mathbb{R}^n$, a fixed direction vector $\xi \in S^{n-1}$, and a height $h \in \mathbb{R}$, we define the set $X_{\xi,h} := \{x \in X \mid \langle x, \xi \rangle \leq h\}$. The set $X_{\xi,h}$ contains all points (0-dim simplices) below the hyperplane spanned by $\langle x, \xi \rangle = h$. The IPT is then defined as the mapping:

$$\begin{aligned} IPT(X) : S^{n-1} \times \mathbb{R} &\to \mathbb{N} \\ (\xi, h) &\mapsto \sum_{x \in X} \mathbb{1}(\langle x, \xi \rangle \leq h) \end{aligned} \tag{2.6}$$

This effectively counts how many points in the cloud are below a certain slice in a given direction. Despite its simplicity, the IPT retains strong theoretical guarantees. For a point cloud in $\mathbb{R}^n$, it has been proven that $n+1$ affinely independent directions are sufficient to ensure that the point cloud can be perfectly reconstructed [3].

2.4 Neural Networks

While the IPT provides a mathematical way to describe shapes, we need a framework that can use this information to actually reconstruct buildings.

Neural networks are a class of models, designed to act as universal function approximators. Formally, a neural network defines a mapping $\mathbf{y} = f(\mathbf{x}; \mathrm{w})$, where an input $\mathbf{x}$ is transformed into a prediction $\mathbf{y}$ through a series of computational layers parameterized by a set of learnable weights and biases, collectively denoted as w.

2.4.1 Fundamental Components

The most basic unit of a neural network is the **neuron**. For an input vector $\mathbf{x}$, a single neuron performs a weighted sum and adds a bias term b:

$$z = \sum_i w_i x_i + b = \mathbf{w}^\top \mathbf{x} + b$$

To enable the network to learn complex, non-linear relationships, this linear result is passed through an **activation function** $\sigma(z)$. Without non-linear activation, a network with any number of layers would still behave like

a simple linear model. In modern architectures, the Rectified Linear Unit (ReLU) is commonly used as the activation function:

$$\sigma(z) = \text{ReLU}(z) = \max(0, z).$$

By stacking these neurons into layers, the network can extract increasingly abstract features from the input data (see Figure 2.4).

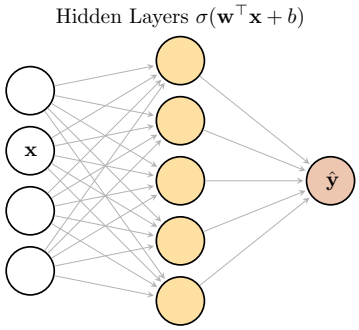


Figure 2.4: Multi-Layer Perceptron (MLP) architecture. Input features $\mathbf{x}$ are transformed by hidden layers using weights $\mathbf{w}$ and non-linear activations to produce prediction $\hat{\mathbf{y}}$.

2.4.2 The Learning Process

Learning in a neural network is the process of finding the parameters w that minimize the error of the predictions. This is achieved through three interconnected concepts: the loss function, gradient descent, and backpropagation.

1. The Loss Function: The loss function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ quantifies the discrepancy between the ground truth $\mathbf{y}$ and the network's prediction $\hat{\mathbf{y}}$.

2. Gradient Descent: To minimize the loss, we use an optimization algorithm called gradient descent. The gradient $\nabla_{\text{w}}\mathcal{L}$ is a vector that points in the direction of the steepest increase in the loss function. By moving the weights in the opposite direction of the gradient, we nudge the model toward a state of lower error:

$$\text{w} \leftarrow \text{w} - \eta \nabla_{\text{w}} \mathcal{L},$$

where η is the learning rate, a small positive scalar that controls the size of the update step.

3. Backpropagation: Calculating the gradient for millions of weights is made possible by backpropagation. This method uses the **chain rule** to propagate the error from the output layer back through the network. By calculating how much each weight contributed to the final loss, the algorithm determines exactly how to adjust every parameter in the system (See Figure 2.5).

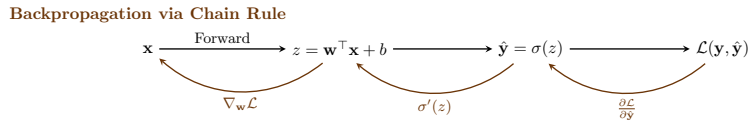


Figure 2.5: The learning process through backpropagation. The top row illustrates the forward pass where input $\mathbf{x}$ is transformed into prediction $\hat{\mathbf{y}}$ and evaluated by the loss $\mathcal{L}$. The curved arrows represent the backward pass, where gradients are propagated using the chain rule to determine parameter updates.

2.5 Convolutional Neural Networks (CNNs)

While standard neural networks (Multi-Layer Perceptrons) treat every input as independent, Convolutional Neural Networks (CNNs) are designed to exploit the spatial structure of grid-like data, such as images or 3D voxel grids.

2.5.1 The Convolution Operation

The core of a CNN is the convolutional layer. Instead of every neuron connecting to every input, a small window of weights called a **kernel** slides across the input (see Figure 2.6). For a 3D input I and a 3D kernel K of size $k \times k \times k$, the output at position (i, j, l) is calculated as:

$$S(i,j,l) = (I * K)(i,j,l) = \sum_m \sum_n \sum_p I(i+m, j+n, l+p) K(m,n,p)$$

This operation allows the network to detect local features, such as edges in 2D or surfaces and corners in 3D.

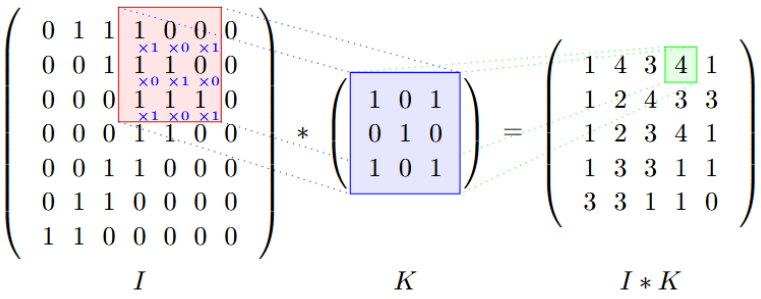


Figure 2.6: Illustration of a discrete 2D convolution operation, serving as a simplified representation of the 3D operation defined above. A 3×3 kernel K slides across the input matrix I, where each value in the resulting feature map $I * K$ (green) is the sum of element-wise products between the kernel and the local receptive field (red).

2.5.2 Key Advantages

CNNs offer three primary benefits that make them useful for 3D reconstruction tasks:

- **Sparse Connectivity:** Each neuron only looks at a small local neighborhood (the receptive field) rather than the entire volume, which drastically reduces the number of parameters.
- **Weight Sharing:** The same kernel is used across the entire input grid. This means if the network learns to recognize a "roof corner" in one part of the voxel grid, it can recognize that same feature anywhere else.
- **Translation Invariance:** Because the kernels slide across the entire grid, the network's ability to detect a feature does not depend on its specific position in space.

2.5.3 Pooling and Striding

To capture larger structural features, CNNs use downsampling techniques. **Striding** involves skipping voxels during the convolution to produce a smaller output grid. **Pooling** (specifically Max-Pooling) takes the maximum value within a small window:

$$y = \max\{x_1, x_2, \ldots, x_n\}$$

This reduces the spatial resolution of the data while keeping the most important signals, allowing the deeper layers of the network to see a much larger portion of the building at once (see Figure 2.7).

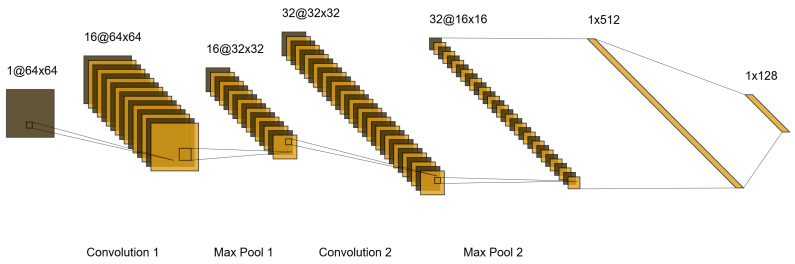


Figure 2.7: Hierarchical feature extraction in a Convolutional Neural Network. The architecture follows a 2D encoder pattern, reducing the spatial resolution ($64^2 \rightarrow 16^2$) through striding and pooling (darker blocks) while increasing feature depth to capture complex architectural patterns.

2.6 Differentiable TDA

In modern machine learning, models are trained using gradient-based optimization, like we've seen described in section 2.4.2. However, traditional topological tools like the Euler characteristic are discrete functions; they consist of integer steps that result in a derivative of zero almost everywhere. This lack of differentiability prevents them from being used directly into the backpropagation process of a neural network.

To bridge this gap, we utilize Differentiable Topological Data Analysis (TDA). The core idea is to replace the discrete indicator function with a smooth, differentiable approximation, such as the sigmoid function $S(x) = \frac{1}{1+e^{-x}}$. This method, originally proposed for the Differentiable Euler Characteristic Transform (DECT), allows topological signatures to be integrated into neural network training [5]. When applied to the IPT, the approximation becomes:

$$\hat{\chi} \approx \sum_{x \in X} (-1)^n S(\lambda(h - \langle x, \xi \rangle))$$

In this formulation, λ is a hyperparameter that controls the "sharpness" of the approximation. As λ increases, the sigmoid more closely resembles the original step function. Because this formulation is differentiable, we can calculate gradients with respect to the point coordinates x and the directions ξ. This allows the use of a topological loss term that encourages the network to optimize the shape to match a target topological signature.

2.7 Marching Cubes Algorithm

The Marching Cubes algorithm is a standard procedure, used to convert voxels into a polygonal mesh [10]. This is essential for converting the discrete outputs of a neural network back into a mesh representation suitable for standard 3D rendering engines.

The algorithm functions by iterating through every cube (formed by eight adjacent voxels) in the grid. It determines how the surface passes through the cube based on which of the eight vertices are above or below a specific threshold. There are $2^8 = 256$ possible occupancy configurations, which are mapped to a specific set of triangular faces using a pre-defined lookup table. By "marching" through the entire volume, the algorithm generates a continuous triangular mesh that approximates the boundary of the volumetric object (see Figure 2.8).

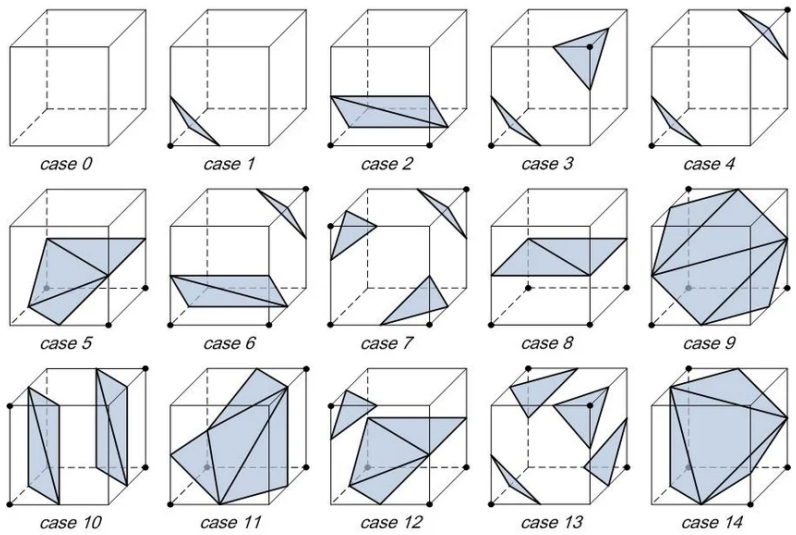


Figure 2.8: The 15 fundamental topological cases used in the Marching Cubes lookup table. While 256 occupancy configurations exist, they are reduced to these unique patterns through rotational and reflective symmetry. Black dots indicate vertices above the intensity threshold, which determine the orientation and connectivity of the resulting triangular facets within each voxel cell.

Chapter 3

Related Works

This chapter provides a technical overview of the research that informs this thesis. We focus on the intersection of topological data analysis (TDA) and machine learning, and how 3D building reconstruction has evolved from rigid geometric templates to flexible generative models.

3.1 Topological Machine Learning

Topological Data Analysis (TDA) captures global structure and connectivity that standard geometric methods often overlook [11]. Unlike traditional descriptors that rely on exact coordinates, TDA extracts features that are stable under noise and stay the same even if the data is rotated or shifted [4].

3.1.1 Persistent Homology and Topological Loss Functions

Standard loss functions like Binary Cross-Entropy (BCE) or Dice loss treat each voxel as an independent target. As a result, they lack "connectivity awareness" and cannot easily correct global errors like disconnected building components [2]. To solve this, researchers have created differentiable loss terms based on cubical complexes. By using a process called superlevel set filtration, one can track when topological features like components or holes appear and disappear. This information is stored in a persistence diagram, and the distance between the predicted and target diagrams can be used to guide the training process [12].

3.1.2 The Euler Characteristic Transform (ECT)

As a faster alternative to persistent homology, the Euler Characteristic Transform (ECT) was introduced to describe shapes by looking at the sum of their parts from many directions [8]. This transform is injective, meaning it can uniquely identify a shape using only a finite set of directions [13, 9].

This thesis is primarily inspired by the Differentiable Euler Characteristic Transform (DECT) [5], which makes the ECT end-to-end trainable by using smooth sigmoid functions. This led to the Inner Product Transform (IPT), a simplified version that works directly on point coordinates for high-efficiency reconstruction [3].

3.2 3D Building Reconstruction

Reconstructing urban geometry from LiDAR requires handling architectural variety and missing data caused by occlusions [1].

3.2.1 Geometric and Template-Based Methods

Early research focused on model-driven reconstruction using libraries of roof templates [14]. While these produce clean models, they cannot handle non-standard architecture. Later geometric methods like Polyfit [15] used plane detection to build models from segments. However, these are often sensitive to noise. If a plane is not detected correctly, the final building will have topological errors like missing walls.

3.2.2 Learning-Based and Generative Reconstruction

Recently, the field has moved toward models that learn geometry directly from data. Implicit field models like City3D [6] represent buildings as continuous functions, allowing for smooth surfaces. Most recently, generative models like Point2Building [1] have used autoregressive transformers to predict mesh vertices and faces as a sequence. While these handle complex shapes well, they require sequential prediction and rejection sampling to stay architecturally plausible. Our work investigates if we can move these structural constraints into the loss function itself using the IPT.

3.3 Comparison Partners

To evaluate the performance of our proposed IPT regularizer, we compare our results against established geometric algorithms and modern learning-based baselines. These methods represent the current paradigms in automated urban modeling.

3.3.1 2.5D Dual Contouring

The 2.5D Dual Contouring method is a robust, geometry-driven algorithm specifically designed for urban modeling from aerial LiDAR [7]. It functions by projecting the 3D point cloud onto a 2D grid and identifying sets of points that share horizontal coordinates but have different vertical elevations.

These points are connected to form a graph that is optimized to create a watertight mesh. While effective for buildings that follow a $2.5D$ logic (no vertical overhangs), it can struggle with complex 3D facades and intricate roof details.

3.3.2 City3D

City3D represents the state of the art in learning-based building reconstruction via implicit fields [6]. Instead of predicting discrete voxels or vertices directly, it learns a continuous function that is converted into a polygonal mesh via post-processing. While highly precise, City3D often relies on initial primitive segmentation, which can introduce errors if the initial detection is slightly inaccurate.

3.3.3 Point2Building

Point2Building represents the shift toward generative, autoregressive models for building reconstruction [1]. Inspired by the PolyGen architecture [16], it generates meshes by iteratively predicting coordinate sequences and connectivity. By learning architectural features directly from raw data, it avoids the fragility of plane detection. However, to maintain structural integrity, the model employs a rejection sampling strategy that validates hypotheses against architectural rules, leading to higher computational latency than the single-pass U-Net architecture proposed in this work.

Chapter 4

Methodology and Implementation

The source code for the 3D U-Net model and the training pipeline is publicly available on GitHub at https://github.com/TobiasRobertEeksman/IPT_LiDAR.

4.1 Zürich Building Dataset

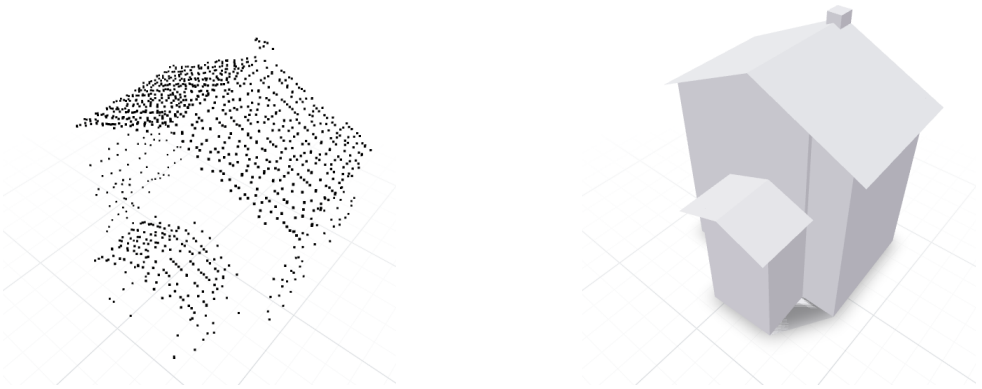
For the evaluation of our model, we utilize the Zürich Building Dataset introduced by Liu et al [1]. This dataset is a large-scale collection of urban geospatial data that combines raw sensor measurements with high-quality geometric ground truths.

4.1.1 Data Sources and Composition

The dataset is constructed from two primary sources:

- **Airborne LiDAR:** The input point clouds were obtained from the Swiss Federal Office of Topography (swisstopo). These scans are semantically classified into six categories, including ground, vegetation, and buildings. For our reconstruction task, we exclusively use the points labeled as ”buildings”.
- **Polygonal Meshes:** The ground truth models consist of Level of Detail 2 (LoD2) polygonal meshes provided by the City of Zürich. These models provide a refined representation of building structures, including precise roof geometries, but exclude minor facade details.

The study area covers a 9 km $\times$ 8 km region of Zürich, which is partitioned into spatially disjoint training and test sets to ensure the model generalizes well to unseen urban structures. Each building in the dataset is represented as a discrete point cloud subset, isolated using reference footprints from the mesh models (see Figure 4.1).



(a) Point Cloud **(b)** Polygon Mesh

Figure 4.1: Sample from the Zürich Building Dataset: (a) raw airborne LiDAR point clouds; (b) corresponding LoD2 polygonal mesh ground truths.

4.2 Voxelization Pipeline

The preprocessing set is designed to transform unstructured 3D data into a structured volumetric format suitable for a Neural Network. This process involves spatial normalization, grid discretization, and the generation of binary occupancy tensors. By converting raw point clouds and meshes into voxels, we provide the network with a fixed-size input where spatial relationships are explicitly encoded in a 3D grid (see Figure 4.2).

4.2.1 Spatial Normalization and Alignment

Before discretization, each building model is transformed into a canonical coordinate system. This step ensures that the model is invariant to its original world coordinates and scale. For each input, we calculate the axis-aligned bounding box to determine its center and maximum extent. The points or mesh vertices are shifted to the origin and scaled to fit within a unit cube defined by the bounds $[-0.5, 0.5]$ using the following transformation:

$$\mathbf{p}_{norm} = \frac{\mathbf{p} - \mathbf{c}}{s + \varepsilon}$$

where $\mathbf{c}$ is the center of the bounding box, s is the maximum dimension of the box, and ε is a small constant used to prevent numerical instability.

4.2.2 Discretization and Occupancy Mapping

The normalized coordinate space is subdivided into a regular grid of resolution $R = 64$. For LiDAR point clouds, we map each point to a specific voxel index (i, j, k) by discretizing its position:

$$idx_{i,j,k} = \lfloor (\mathbf{p}_{norm} + 0.5) \times (R - \varepsilon) \rfloor$$

A voxel is marked as occupied (1) if it contains at least one point, and empty (0) otherwise.

For the ground truth meshes, we perform a ”solid” voxelization. This involves first identifying the voxels that intersect with the mesh faces to create a shell. We then utilize a filling algorithm to mark all voxels contained within the interior of the closed surface as occupied. This solid representation is critical for the network to learn the underlying volumetric density of the buildings rather than just their surface shells. To ensure a consistent input size of $64 \times 64 \times 64$, we apply center-padding or cropping where necessary.

In essence, this pipeline standardizes the 3D data formats into binary occupancy tensors for both input and output. By establishing this uniform volumetric interface, we enable the CNN to effectively leverage spatial localizations through convolutional kernels while maintaining the computational efficiency required for high-speed inference.

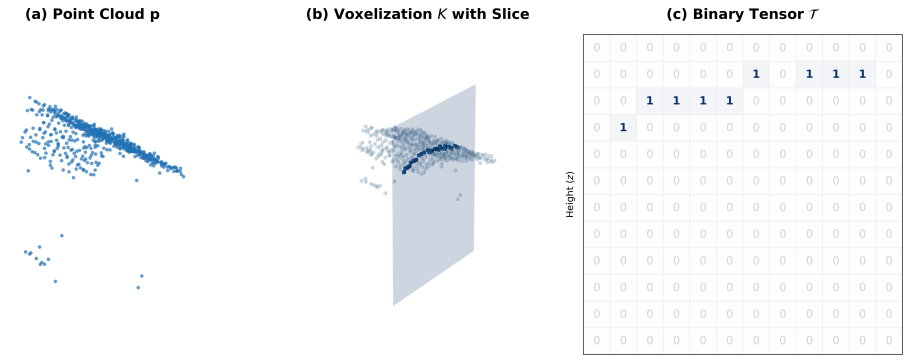


Figure 4.2: Volumetric preprocessing stages: (a) input LiDAR point cloud; (b) solid voxelization with a highlighted cross-sectional slice; and (c) a localized 12×12 binary tensor representation of the slice, oriented such that the vertical axis represents the height z.

4.3 3D U-Net CNN Architecture

The 3D U-Net is a symmetric convolutional neural network designed to capture both global semantic context and local spatial detail. Its characteristic ”U” shape consists of a contracting path (encoder) that extracts features and an expansive path (decoder) that reconstructs the spatial resolution. A defining mechanism of this architecture is the skip connection, which passes high-resolution feature maps from the encoder directly to the decoder, ensuring that geometric details lost during downsampling are preserved in the final output.

4.3.1 Core Building Blocks

The implementation utilizes three primary modules to handle the volumetric data:

- **Double Convolution:** The basic processing unit, consisting of two layers of $3 \times 3 \times 3$ convolutions. Each convolution is followed by Batch Normalization and a Rectified Linear Unit (ReLU) activation function.
- **Downscaling:** This module performs spatial reduction using a $2 \times 2 \times 2$ MaxPool3D operation, followed by the double convolution block to extract deeper features.
- **Upscaling:** This module utilizes Transposed 3D Convolutions with a stride of 2 to double the spatial resolution. After upsampling, the feature map is concatenated with the corresponding map from the encoder (skip connection) before being processed by a double convolution.

4.3.2 Implementation Details

The network is configured to process binary voxel grids with a resolution of $64 \times 64 \times 64$. The architecture is summarized by the following pipeline:

1. **Contracting Path (Encoder):** The input is passed through an initial convolution and four downscaling blocks. The spatial resolution is progressively halved ($64^3 \rightarrow 32^3 \rightarrow 16^3 \rightarrow 8^3 \rightarrow 4^3$) while the filter count doubles at each step ($32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512$).
2. **Bottleneck:** The deepest part of the network operates on a $4 \times 4 \times 4$ grid with 512 feature channels, representing the most compressed semantic information.
3. **Expansive Path (Decoder):** Four upscaling blocks reconstruct the spatial resolution back to 64^3. The concatenation of skip connections allows the decoder to combine global semantic features with precise local geometry.
4. **Output:** A final $1 \times 1 \times 1$ convolution maps the last 32 feature channels to a single output value, preserving the original 64^3 spatial dimensions. In a last step, we use the Sigmoid function in order to create a voxel-wise probability map (see Figure 4.3).

4.4 Initial Approach: DECT and Marching Cubes

Our initial architecture utilized the 3D U-Net to generate a probability (”soft”) voxel grid. To refine this output, we intended to integrate a topological loss term derived from the Differentiable Euler Characteristic Transform (DECT).

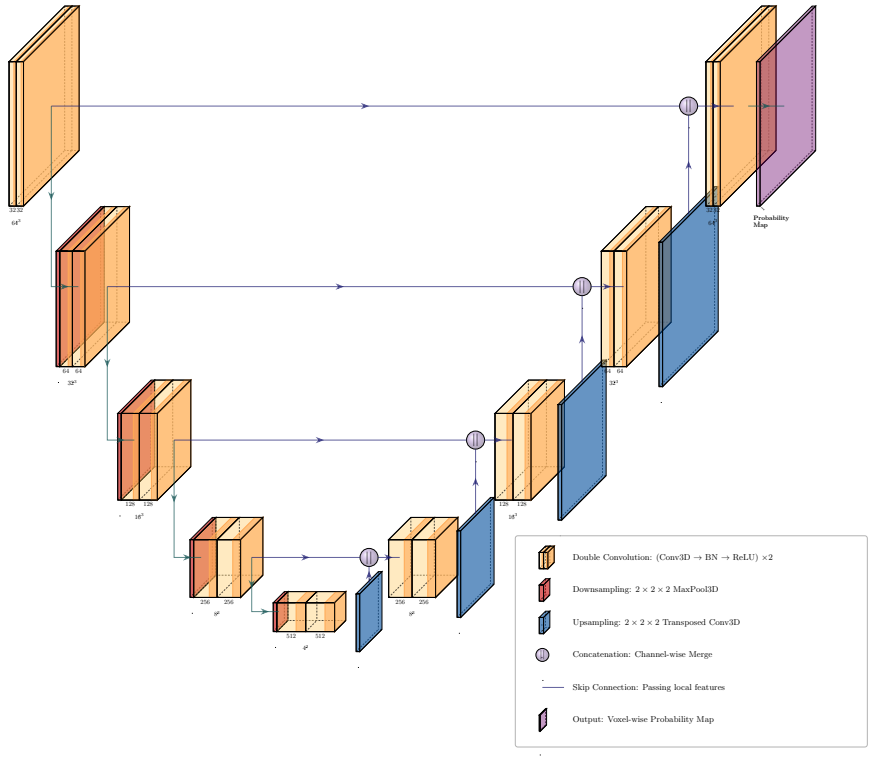


Figure 4.3: Diagram of the 3D U-Net architecture. The contracting path reduces spatial resolution from 64^3 to 4^3 while doubling feature channels. The expansive path reconstructs the volume, using skip connections to preserve local geometric detail.

4.4.1 The Mesh-Based Topological Pipeline

The core of our initial attempt was the addition of a topological regularizer. Since the Euler Characteristic Transform (ECT) is conventionally defined on simplicial complexes (such as meshes) rather than raw voxel grids, we designed a multi-step pipeline to bridge this gap:

1. **Thresholding:** The soft voxel grid was passed through a cutoff function, typically at a threshold of 0.5, to generate a discrete binary occupancy grid.
2. **Voxelized Object Extraction:** The binary grid was converted into a structured voxelized object representing the building's solid volume.
3. **Surface Reconstruction:** We applied the **Marching Cubes** algorithm to extract a triangular mesh from the binary voxel grid.
4. **Topological Evaluation:** The resulting mesh served as the input for the DECT, allowing us to calculate topological features across multiple directions and compare them to the ground truth mesh (see Figure 4.4).

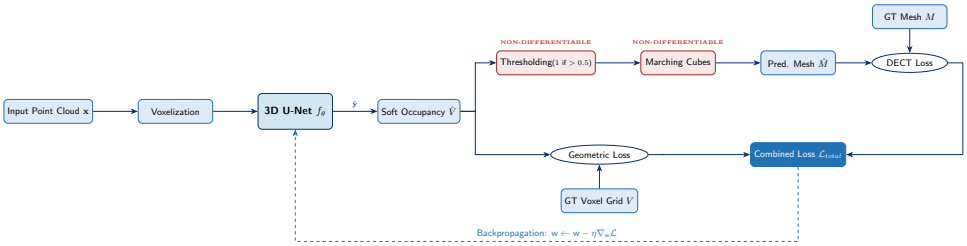


Figure 4.4: Architecture of the initial reconstruction pipeline. The system branches into two parallel paths: a standard geometric path comparing voxel occupancies, and a topological path that extracts a triangular mesh via Marching Cubes to compute the Differentiable Euler Characteristic Transform (DECT) loss. The red highlighting indicates the non-differentiable bottleneck caused by discrete thresholding and surface reconstruction, which prevents the direct propagation of topological gradients back to the 3D U-Net parameters w.

4.4.2 The Differentiability Obstacle

Despite the mathematical soundness of the ECT as a shape descriptor, this initial approach encountered critical failures during implementation due to a lack of differentiability. For a neural network to learn via gradient descent, every operation in the pipeline must provide a meaningful gradient to update the network's weights. We identified two primary bottlenecks:

- **The Cutoff Function:** The thresholding operation is essentially a step function. Its derivative is zero almost everywhere and undefined at the threshold, which prevents the flow of gradients back to the U-Net.
- **Marching Cubes:** The algorithm relies on discrete lookup tables to generate mesh faces from voxel configurations. This process is inherently non-differentiable and cannot propagate topological errors.

To resolve these issues, we initially investigated probabilistic models that could assign probabilities to various simplicial complex configurations. However, the complexity of maintaining differentiability through a mesh-generation stage ultimately led us to reconsider the transform itself. This shift motivated the adoption of the **Inner Product Transform (IPT)**, which allows for topological analysis directly on voxel coordinates.

4.5 The IPT Loss Layer

To overcome the non-differentiability of mesh-based transforms, we implement a custom Inner Product Transform (IPT) Loss Layer (see Figure 4.5). This layer allows the 3D U-Net to be optimized directly against topological targets without the need for intermediate surface reconstruction via Marching Cubes. The key innovation is treating the voxel grid as a weighted point set, where the coordinates are fixed and the presence of each point is defined by the network's output probability.

4.5.1 Differentiable Voxel-to-IPT Mapping

The 3D U-Net produces a soft probability grid $\hat{\mathbf{V}} \in [0,1]^{64\times64\times64}$. To calculate the IPT differentiably, we define a set of canonical voxel centers $\mathcal{C}$ corresponding to the grid coordinates. Rather than thresholding the grid, we treat each center $\mathbf{c}_i \in \mathcal{C}$ as a point with an associated weight p_{c_i} provided by the voxel's probability value.

The transform is then calculated by projecting these weighted centers onto a set of N direction vectors ξ. To ensure the counting process is differentiable, we replace the discrete indicator function $\mathbb{I}$ with a sigmoid function S:

$$\widehat{IPT}(\xi, h) = \sum_{i=1}^{N_{vox}} p_{c_i} \cdot S(\lambda(h - \langle \mathbf{c}_i, \xi \rangle))$$

where λ is a scaling factor that controls the steepness of the approximation. This formulation allows the gradients to propagate from the topological signature back through the weights w_i to the convolutional kernels of the U-Net.

4.5.2 Soft Boundary Extraction

A significant challenge in building reconstruction is ensuring that the topological signature captures structural features (such as roof types and wall alignments) rather than simple volumetric mass. If the IPT is calculated on a solid voxelized volume, the signal is dominated by interior points, which provides little information about surface geometry.

To resolve this, we implement a **Soft Boundary Extraction** step within the loss layer. We isolate the building's "shell" by calculating the difference between the predicted volume and a version of that volume shrunken by a morphometric erosion operation. In a differentiable context, erosion is achieved using a 3D minimum pooling operation (implemented as $-\text{MaxPool3D}(-V)$). The boundary $\mathbf{B}$ is defined as:

$$\mathbf{B} = \hat{\mathbf{V}} - \text{Erode}(\hat{\mathbf{V}})$$

The IPT is then computed exclusively on this boundary grid. This forces the model to focus its "topological attention" on the exterior surfaces of the building.

4.5.3 Direction Sampling

The expressivity of the Inner Product Transform (IPT) is fundamentally tied to the selection of the projection directions $\xi \in S^2$. To capture a comprehensive topological signature of a 3D building, the shape must be "viewed" from a sufficiently diverse set of perspectives. While theoretical results suggest

that $n+1$ affinely independent directions are sufficient for reconstruction in $\mathbb{R}^n$, practical applications involving complex architectural geometries benefit from a larger, more redundant set of directions.

In our implementation, we utilize a stochastic approach to sample directions uniformly across the unit sphere. For a specified number of directions D, we generate vectors in $\mathbb{R}^3$ where each component is drawn from a standard normal distribution $\mathcal{N}(0,1)$. To ensure these vectors represent valid directions on the unit sphere S^2, we normalize each vector by its Euclidean norm:

$$\xi_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}, \quad \text{where } \mathbf{v}_i \sim \mathcal{N}(0, \mathbf{I})$$

This method exploits the rotational symmetry of the Gaussian distribution to achieve a statistically uniform distribution of directions, preventing the topological loss from being biased toward specific coordinate axes.

To ensure scientific reproducibility and consistency during training, these directions are sampled once using a fixed random seed and stored as a persistent parameter set. This guarantees that the "topological feature space" remains identical for both the ground truth IPT generation and the differentiable IPT calculated during the U-Net's forward pass. By maintaining a static set of directions, the network can reliably learn to map geometric deviations in the voxel grid to specific discrepancies in the topological signature.

4.6 The Combined Loss Function

The optimization of our 3D U-Net is driven by a multi-component loss function. Because building reconstruction requires both pixel-level accuracy and global structural integrity, we combine standard geometric losses with our custom topological regularizer. The total loss $\mathcal{L}_{total}$ is defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} + \mathcal{L}_{Dice} + \lambda_{topo}\mathcal{L}_{topo}$$

where λ_{topo} is a weighting parameter that controls the influence of the topological term.

4.6.1 Geometric Components: BCE and Dice

The first two components focus on the volumetric geometry of the building.

Binary Cross Entropy (BCE)

BCE is a standard loss for voxel-wise classification. It measures the discrepancy between the predicted probability p_i and the ground truth label

$y_i \in \{0, 1\}$ for every voxel i in the grid. The loss is defined as:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

BCE is highly effective for pixel-level accuracy, but it can struggle with class imbalance. In urban LiDAR data, building voxels are often much rarer than empty space voxels, which can lead the model to favor predicting empty space.

Dice Loss

To address the sparsity of building data, we incorporate the Dice Loss. Instead of looking at voxels individually, Dice Loss evaluates the global overlap between the predicted volume P and the ground truth volume Y:

$$\mathcal{L}_{Dice} = 1 - \frac{2\sum_{i=1}^{N} p_i y_i + \varepsilon}{\sum_{i=1}^{N} p_i + \sum y_i + \varepsilon}$$

where ε is a smoothing term to avoid division by zero. By focusing on the intersection of the two volumes, Dice Loss ensures that the model prioritizes the "shape" of the building rather than just the total number of correct voxels. In our implementation, we use a combination of both BCE and Dice to balance local precision with global volumetric overlap.

4.6.2 Topological Component: IPT Loss

The third component is the Inner Product Transform (IPT) loss. This term compares the topological features of the predicted building to that of the target.

For the ground truth, the IPT is pre-calculated during the preprocessing stage. We take the solid ground-truth mesh, convert it to a voxelized shell, and compute the discrete IPT across our fixed set of directions. During training, the predicted "soft" voxel grid is passed through the differentiable IPT layer (as described in 4.5.1) to produce a predicted signature $\widehat{IPT}$.

To implement the IPT loss computationally, we discretize the transform into a fixed-dimensional tensor. Let D denote the number of projection directions ξ_i sampled from S^2, and let R denote the resolution of the filtration, representing the number of discrete height bins h_j along each direction. The resulting topological signature is a matrix $\mathcal{T} \in \mathbb{R}^{D \times R}$, where each entry $\mathcal{T}_{i,j}$ represents the (soft) point count for direction ξ_i at height h_j.

To calculate the loss, we utilize the Mean Squared Error (MSE) between the predicted and target matrices. However, because the IPT values represent

a summation of sigmoid "soft counts" across the voxel grid, raw values can reach several thousands, leading to gradients that would overwhelm the geometric learning process. We therefore normalize the signatures by a constant factor N_{norm}:

$$\mathcal{L}_{topo} = \text{MSE}\left(\frac{\widehat{IPT}}{N_{norm}}, \frac{IPT_{gt}}{N_{norm}}\right) = \frac{1}{D \cdot R} \sum_{d=1}^{D} \sum_{r=1}^{R} \left(\frac{\widehat{IPT}_{d,r} - IPT_{gt,d,r}}{N_{norm}}\right)^2$$

The choice of N_{norm} is intrinsically coupled to the dimensionality of the IPT matrix. For our configuration of $D = 10$ and $R = 20$, a value of $N_{norm} = 1'500$ was empirically determined to map the topological error into a range comparable with the volumetric losses ($\mathcal{L}_{BCE} \approx 0.7$). This normalization ensures that the MSE remains stable and provides a meaningful gradient signal throughout the optimization.

To determine an effective value for λ_{topo}, we analyze the relative contribution of the geometric and topological terms. We initialize the model with random weights and perform inital passes to calculate the raw magnitudes of $\mathcal{L}_{BCE}$ and $\mathcal{L}_{topo}$. We then set λ_{topo} based on a target contribution ratio α:

$$\lambda_{topo} = \frac{\alpha \cdot \mathcal{L}_{BCE}}{\mathcal{L}_{topo,raw}}$$

In our primary experiments, we utilize $\alpha = 1.0$ to provide an "equal weighting" scenario.

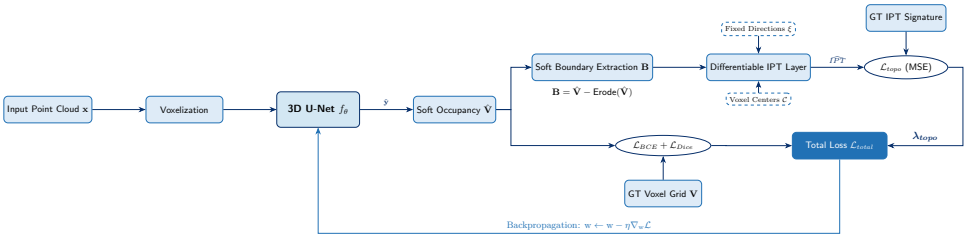


Figure 4.5: The proposed differentiable reconstruction pipeline. The 3D U-Net output is processed through a soft boundary extraction layer to isolate structural features. This boundary grid is mapped to a topological signature via a differentiable Inner Product Transform (IPT) layer using fixed voxel centers $\mathcal{C}$ and stochastic directions ξ. The total loss $\mathcal{L}_{total}$ enables end-to-end gradient propagation, allowing the model to optimize simultaneously for volumetric accuracy and topological consistency.

Chapter 5

Experiments and Evaluation

In this chapter, we detail the experimental setup used to validate the Inner Product Transform (IPT) as a structural regularizer for 3D building reconstruction. We describe the custom visualization tools developed for qualitative analysis, the monitoring framework for training stability, and the design of the parameter study used to evaluate the impact of topological constraints.

5.1 Visualizing Results: The 3D Evaluation Dashboard

We built a custom 3D Evaluation Dashboard to look at our results more closely. The tool is web-based and uses React with the Three.js library. It shows multiple synchronized views at the same time. We also included a Compare Mode. This lets us compare different mesh predictions side by side. We get these meshes by applying the Marching Cubes algorithm to the predicted voxels. This allows us to compare model predictions in a way that standard metrics cannot. The source code for the dashboard is available on GitHub at https://github.com/TobiasRobertEeksman/IPT_LiDAR_Webapp (see Figure 5.1).

The dashboard features a four-way comparison layout:

- **Source LiDAR:** The raw, noisy point cloud input.
- **Ground Truth Mesh:** The original LoD2 polygonal mesh from the Zürich dataset.
- **Voxelized Reference (Upper Bound):** The ground truth mesh converted into the 64^3 voxel space.
- **Model Prediction:** The voxel output of the 3D U-Net.

The tool also includes a dedicated Compare Mode for side-by-side analysis (see Figure 5.2).

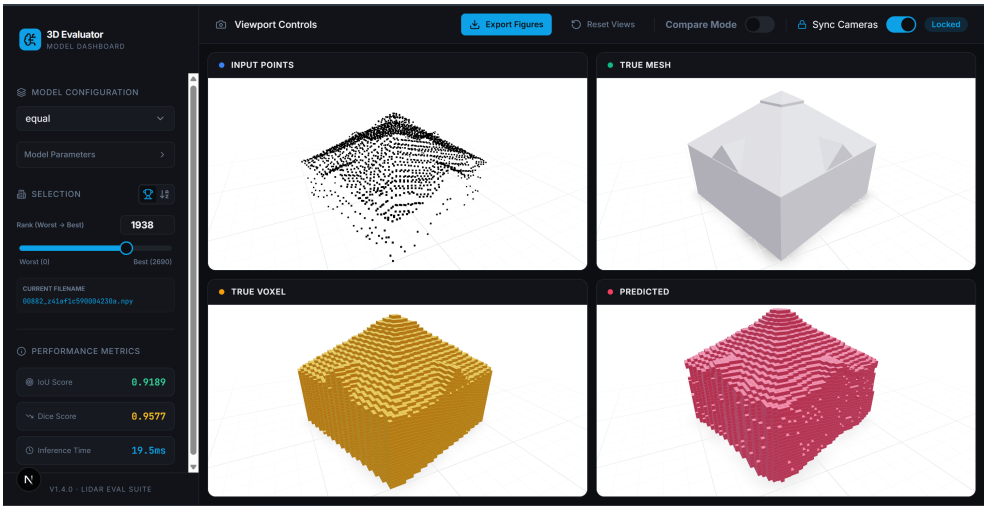


Figure 5.1: Interface of the 3D Evaluation Dashboard developed for qualitative model assessment. The four-viewport environment enables synchronized camera control for a direct comparison between the raw source LiDAR (top-left), the original LoD2 ground truth mesh (top-right), the voxelized reference (bottom-left), and the resulting 3D U-Net prediction (bottom-right). Quantitative metrics such as IoU and Dice scores are integrated into the primary sidebar to facilitate real-time correlation between visual quality and mathematical performance.

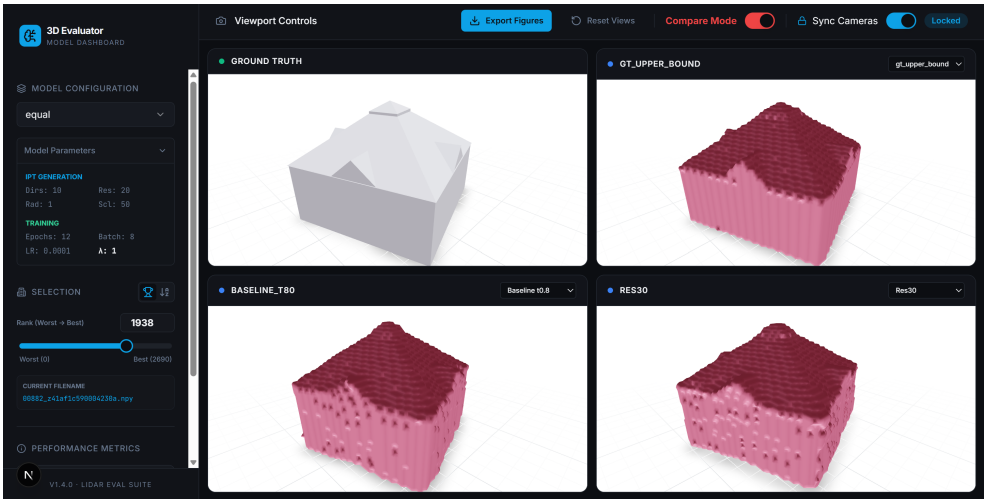


Figure 5.2: The 3D Evaluation Dashboard in Compare Mode. This layout allows us to compare mesh results from four different model configurations at the same time. These meshes are created by applying the Marching Cubes algorithm to the predicted voxels. The synchronized view helps us see how changes in the topological weight and grid settings affect the shape of the building.

5.2 Experimental Configurations

The performance of the topological regularizer is governed by the parameters used to compute the Inner Product Transform (IPT) matrix. To evaluate the impact of these hyperparameters on reconstruction accuracy, we conducted a parameter study across six primary configurations:

1. **Baseline:** The model is trained exclusively with standard voxel-wise losses ($\mathcal{L}_{BCE}$ and $\mathcal{L}_{Dice}$). This serves as a reference for the inherent capabilities of the 3D U-Net architecture without topological guidance.
2. **Equal (Random-10):** A balanced configuration utilizing 10 randomly generated projection directions with a regularization weight of $\lambda_{topo} = 1.0$. This puts a rather high weight on the IPT term, in order to check its interaction with the geometrical loss terms.
3. **Minimal Grid-Aligned (XYZ+1):** This configuration employs the three primary orthogonal axes (X, Y, Z) supplemented by a single random direction, with a reduced weight of $\lambda_{topo} = 0.5$. It investigates the impact of orthogonal vectors, as well as the claim, that 4 directions would be enough for reconstruction.
4. **Standard Grid-Aligned (XYZ+7):** An expansion of the minimal grid approach using 10 directions (3 orthogonal, 7 random) and $\lambda_{topo} = 0.5$. This assesses whether increasing directional density improves the results.
5. **Stress Test:** An extreme configuration using a high regularization weight of $\lambda_{topo} = 5.0$ and a sparse set of 4 directions. This setting is designed to prioritize global topology over local geometric precision, highlighting the behavior of the IPT under high enforcement.
6. **Res30:** The IPT resolution is increased to $R = 30$ bins (from the default 20) with $\lambda_{topo} = 0.5$. This configuration evaluates whether a more refined topological descriptor allows the network to better distinguish between architectural features.

By varying the density D, resolution R, and the weighting factor λ_{topo}, we analyze the trade-offs between computational complexity and structural accuracy.

5.3 Training with TensorBoard

For real-time monitoring of the learning process, we utilize TensorBoard. This framework allows us to track the convergence of the individual loss components (Binary Cross Entropy (BCE), Dice, and the Topological (IPT) loss) across thousands of training steps.

As shown in Figure 5.3, we observe a significant regularization overhead when the topological loss is introduced. While the baseline model (no topological loss) converges rapidly, reaching a BCE threshold of 0.1 within approximately 5'000 steps, the models utilizing the IPT regularizer require substantially more iterations. Specifically, the equal weighting scenario ($\lambda_{topo} = 1.0$) takes roughly three times as long to reach the same error level, while the stress test ($\lambda_{topo} = 5.0$) would require over 35'000 steps.

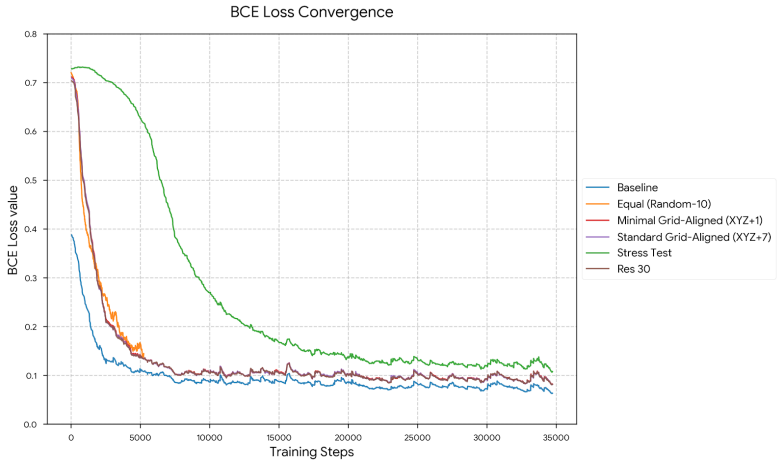


Figure 5.3: BCE loss convergence across different topological weighting configurations. Higher values of λ_{topo} lead to a significant increase in the number of iterations required to achieve comparable geometric accuracy.

The convergence behavior of the Dice loss, which evaluates global volumetric overlap, further reinforces these observations. As illustrated in Figure 5.4, the baseline model begins with a significantly lower initial error, indicating that without structural constraints, the U-Net can rapidly identify the primary building mass. In contrast, models with high topological weighting ($\lambda_{topo} \geq 1.0$) exhibit some difficulty at the start.

Around 5'000 steps, we fall into a plateau. This suggests that the topological regularizer acts as a strict geometric constraint during the early stages of training. The network is essentially prevented from optimizing for volumetric overlap until it finds a shape that aligns with the global structural prior encoded in the IPT matrix. Once this is satisfied, the Dice loss follows a similar descent pattern to the baseline, though it consistently maintains a higher final error floor (approximately 0.09 vs. 0.05). This confirms that while the IPT ensures structural consistency, it introduces a trade-off in raw volumetric precision that is most visible during the "warm-up" phase of optimization.

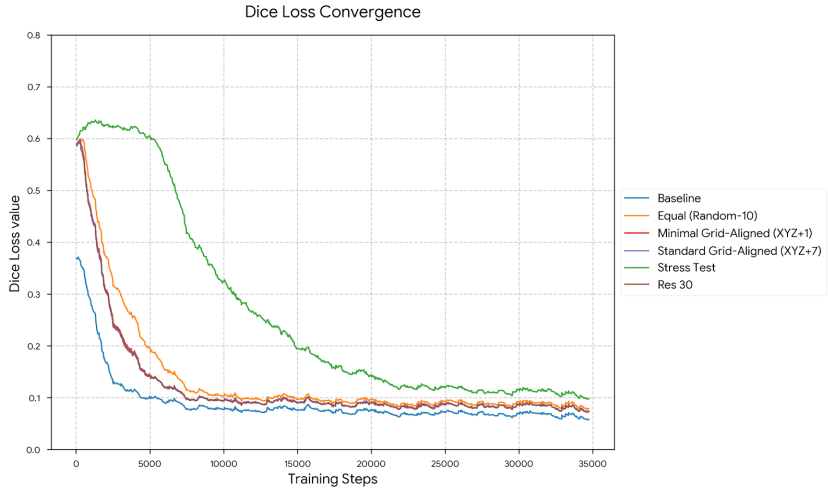


Figure 5.4: Dice loss convergence across experimental configurations. The plateauing behavior in the $\lambda_{topo} = 5.0$ configuration highlights the conflict between global structural alignment and volumetric overlap optimization.

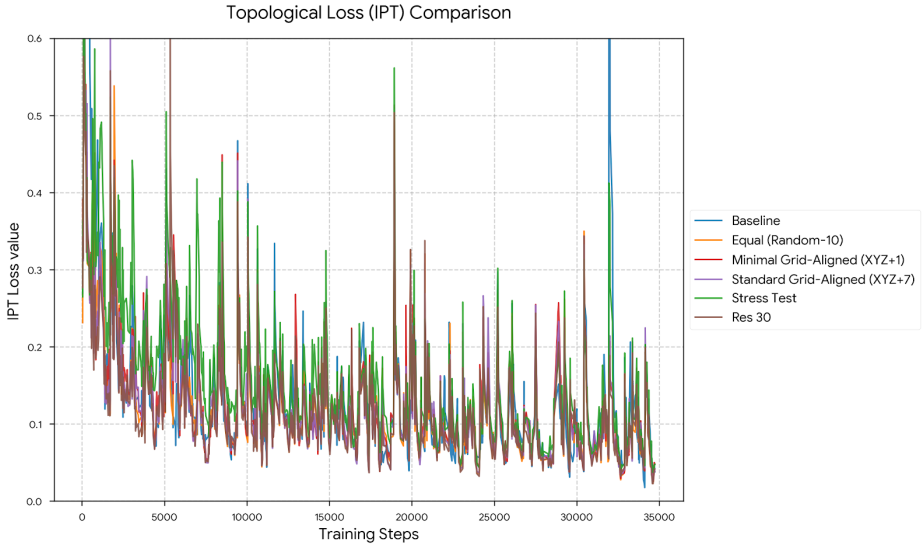


Figure 5.5: Topological (IPT) loss convergence across configurations. The frequent spikes (even in the late stages of training) highlight the difficulty of optimizing for global topological features in a discrete voxel space compared to local occupancy losses.

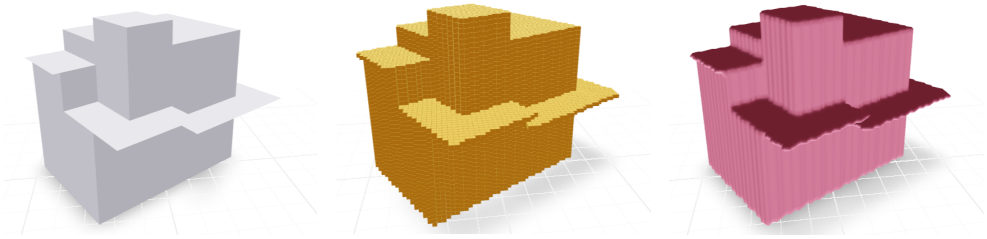
The convergence behavior of the topological loss is notably different from the smooth descent observed in BCE and Dice losses. While all regularized models eventually reach a lower error range than the baseline, the signal remains massively volatile throughout the entire training period (see Figure 5.5). This behavior indicates that the IPT matrix is a highly sensitive objective,

where small voxel adjustments can trigger large non-local changes in the projection matrix. This instability justifies our use of a smaller weighting factor (λ_{topo}), as the topological loss serves more effectively as a structural guide than a primary driver for geometric optimization.

5.4 Evaluation Metrics

To evaluate the geometric accuracy of our reconstructions, we utilize four primary distance metrics based on surface point sampling. As our network outputs a volumetric occupancy grid (64^3 voxels), we first transform the discrete voxel representation into a surface mesh using a **threshold probability** (usually $t = 0.5$) and the **Marching Cubes algorithm** [10] (see Figure 5.6). This conversion is critical for a fair comparison with mesh-based architectures, as it allows us to sample points directly from the predicted surface.

For each test building, we uniformly sample sets of 10,000 points from the ground truth mesh surface (S_{gt}) and the predicted mesh surface (S_p), denoted as point sets A and B, respectively. These metrics were specifically selected to maintain consistency with the benchmarks established in the Point2Building study [1].



(a) Target LoD2 Mesh **(b)** Discrete Voxel Grid (64^3) **(c)** GT Upper Bound (MC)

Figure 5.6: Visual analysis of discretization-induced artifacts. A sharp architectural mesh (a) is voxelized into a discrete volumetric grid (b). The subsequent surface reconstruction via Marching Cubes (c).

- **Mean Distance Error (MDE):** This metric quantifies the average deviation of the reconstructed surface from the ground truth. It is calculated as the mean Euclidean distance from each point in the ground truth set A to its nearest neighbor in the predicted set B:

$$MDE(A, B) = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} \|a - b\|_2 \tag{5.1}$$

  In our implementation, this is approximated using a KDTree to efficiently find the closest point correspondences between the sampled

clouds.

- **Hausdorff Distance:** This metric identifies the "worst-case" structural discrepancy by measuring the maximum extent of deviation between the two surfaces. It is defined as:

$$H(A,B) = \max\left\{ \sup_{a\in A} \inf_{b\in B} d(a,b), \sup_{b\in B} \inf_{a\in A} d(a,b) \right\} \tag{5.2}$$

  where $d(\cdot,\cdot)$ denotes the Euclidean distance.

- **Chamfer Distance:** The Chamfer distance evaluates the overall shape similarity by calculating the average closest point distance bidirectionally between the two sets:

$$C(A,B) = \frac{1}{|A|}\sum_{a\in A} \min_{b\in B} d(a,b) + \frac{1}{|B|}\sum_{b\in B} \min_{a\in A} d(b,a) \tag{5.3}$$

  This metric provides a global sense of the deformation required to align the predicted model with the target.

- **Precision, Recall, and F1-Score:** To measure reconstruction accuracy within specific tolerances, we calculate these metrics using a distance threshold $d = 0.01$.

  Precision measures the accuracy of the predicted surface. For a set of points A sampled from the prediction, it is the fraction of points that lie within distance d of the ground truth surface S_{gt}:

$$\text{Precision}(d) = \frac{1}{|A|}\sum_{a\in A} [\min_{b\in S_{gt}} \|a-b\| < d] \tag{5.4}$$

  Recall measures the completeness of the reconstruction. For a set of points B sampled from the ground truth, it is the fraction of points that are "covered" by the predicted surface S_p within distance d:

$$\text{Recall}(d) = \frac{1}{|B|}\sum_{b\in B} [\min_{a\in S_p} \|b-a\| < d] \tag{5.5}$$

  The F1-score is the harmonic mean of the two, providing a single value that represents the overall balance between accuracy and completeness:

$$F_1(d) = 2 \cdot \frac{\text{Precision}(d)\cdot \text{Recall}(d)}{\text{Precision}(d) + \text{Recall}(d)} \tag{5.6}$$

5.4.1 Accuracy Comparison

Table 5.1 provides a comprehensive comparison of geometric accuracy. To provide a benchmark for maximum attainable precision, we include the **GT Upper Bound**, representing the best mesh we can get by voxelization of the ground truth (see Figure 5.7 (c)). We further include two high-threshold variants ($t = 0.8$) to assess the impact of probability filtering on reconstruction precision.

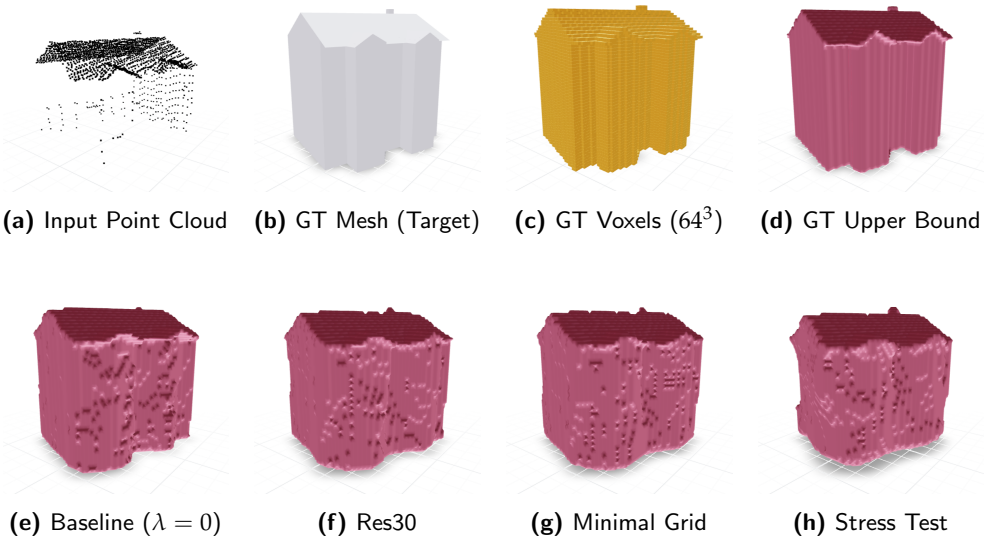
Configuration	MDE ↓	Hausdorff ↓	Chamfer ↓	F1@0.01 ↑
GT Upper Bound (Reference)	0.0096	0.0329	0.0193	0.6020
Standard Threshold (t = 0.5)				
Baseline ($\lambda_{topo} = 0$)	0.0154	**0.0555**	0.0313	0.3592
Equal (Random-10)	0.0157	0.0589	0.0317	0.3568
Stress Test ($\lambda_{topo} = 5.0$)	0.0164	0.0660	0.0329	**0.3679**
Minimal Grid (XYZ+1)	0.0154	0.0572	0.0312	0.3665
Standard Grid (XYZ+7)	0.0158	0.0575	0.0321	0.3548
Res30	**0.0153**	0.0568	**0.0311**	0.3670
High Confidence Threshold (t = 0.8)				
Baseline ($t = 0.8$)	0.0164	0.0606	0.0330	0.3572
Minimal Grid ($t = 0.8$)	0.0158	0.0593	0.0317	**0.3751**

Table 5.1: Quantitative comparison of reconstruction accuracy. Metrics are reported in Normalized Units (NU). The F1 score assesses sub-voxel precision at a strict 1% tolerance.

The results show that all models operate near the discretization-induced limit of the 64^3 grid. As the *GT Upper Bound* yields an F1@0.01 of 0.6020, it is clear that the volumetric format itself accounts for the majority of the recorded error.

Among the standard configurations, the **Res30** model achieves the highest surface precision, suggesting that finer topological descriptors assist the network in accurate surface placement. In contrast, the *Stress Test* highlights a trade-off: while it exhibits the highest distance errors, it maintains a superior F1-score (0.3679) compared to the baseline. This indicates that high topological regularization ensures a more complete architectural shell, even if it induces a slight volumetric dilation.

A notable finding occurs when increasing the probability threshold to $t = 0.8$. While the *Baseline* experiences a decrease in F1-score, the IPT-guided *Minimal Grid* model reaches its peak structural recall (0.3751). This suggests that the topological loss helps concentrate the model's probability mass along the ground truth surface, whereas the baseline's probability distribution is more diffused at the boundaries.



(a) Input Point Cloud **(b)** GT Mesh (Target) **(c)** GT Voxels (64^3) **(d)** GT Upper Bound

(e) Baseline ($\lambda = 0$) **(f)** Res30 **(g)** Minimal Grid **(h)** Stress Test

Figure 5.7: Visual comparison of reconstruction results for a building. The top row shows the input data and discretization targets. The bottom row presents the results from different model configurations. While the *Stress Test* (h) exhibits slightly curved edges compared to the *Baseline* (e), it reconstructs the chimney structure more accurately. This highlights the trade-off between maintaining straight walls and capturing fine architectural details.

Comparison to External Baselines

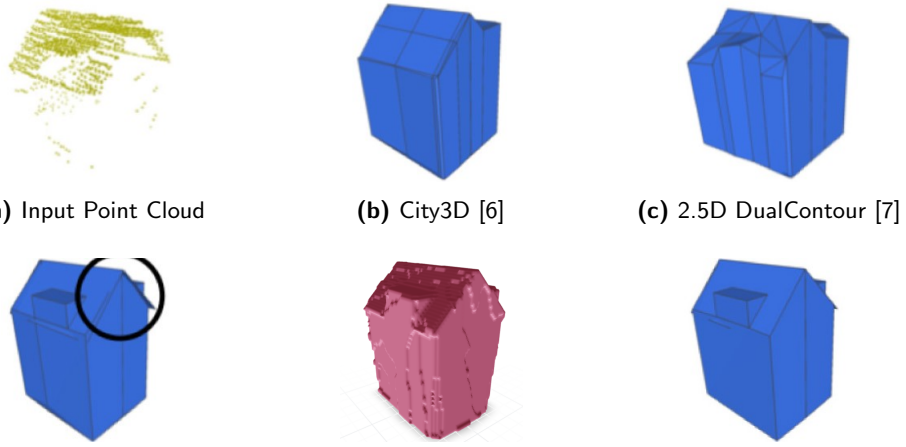
To situate our work within the broader field of 3D building reconstruction, we refer to the performance metrics of established models as reported in the literature [1]. Table 5.2 summarizes the results for **City3D** [6], **2.5D DualContour** [7], and the state-of-the-art **Point2Building** [1] generator.

It is important to note that these reference models utilize datasets where absolute metric scale (meters) were reinforced. In contrast, our study utilizes a normalized unit space ($[0, 1]^3$) due to the lack of available real-world metadata for the specific test set. Consequently, a direct numerical comparison between our results and these external baselines is mathematically unfeasible, as any assumed scale factor would lack a rigorous empirical foundation.

Method	Output Type	MDE (m) ↓	Hausdorff (m) ↓	Chamfer (m) ↓
City3D	Voxel (128^3)	0.3046	1.4723	0.3708
2.5D DualContour	Mesh	0.3646	1.9157	0.4368
Point2Building	Mesh	**0.2542**	**1.1200**	**0.3060**

Table 5.2: Geometric accuracy metrics for established baselines as reported in their respective studies. These values provide context for the current state-of-the-art in architectural reconstruction using absolute metric scales.

Instead of a numerical comparison, we look at the visual differences between our voxel approach and existing mesh methods. Figure 5.8 shows these different styles. Mesh-based models like Point2Building are very good at representing sharp edges and flat surfaces with no thickness. This makes them ideal for high-quality architectural images. Our volumetric U-Net, however, prioritizes structural completeness and topological consistency. While the 64^3 discretization results in an observable volumetric error, it successfully captures the global structure. This qualitative difference suggests that the choice of representation (voxel versus mesh) should be guided by the intended application: the former being more suitable for structural and volumetric analysis, and the latter for precise architectural rendering.



(a) Input Point Cloud **(b)** City3D [6] **(c)** 2.5D DualContour [7]

(d) Point2Building [1] **(e) Ours (Res30)** **(f)** Ground Truth (LoD2)

Figure 5.8: Qualitative comparison of LoD2 reconstruction from sparse LiDAR. Mesh-based methods (d) provide sharper architectural edges. Our volumetric approach (e) demonstrates topological consistency and global alignment compared to previous voxel-based (b) and geometric (c) methods. Note the reconstruction of the roof in our model despite the sparse input cloud.

Chapter 6

Discussion

6.1 Dataset and Resolution Limits

Our resolution of 64^3 also creates a hard floor on accuracy. In our normalized unit space, a single voxel has a width of approximately 0.0156 units. Our recorded Mean Distance Error (MDE) of roughly 0.0153 to 0.0154 is essentially equal to this discretization limit. This suggests that the bottleneck is not our loss function, but the grid size itself. As shown in Figure A.1 and Figure A.2 in the appendix, low resolution can lead to low confidence regions where the model is not sure where to place the surface. Because the error is already at the single-voxel level, even a minor change, like placing one extra layer of voxels on a wall, makes the model numerically less accurate than high-resolution or mesh-based methods. Future work using higher resolutions would likely let the IPT regularizer create much sharper edges.

The lack of a big improvement over the baseline is also because the Zürich LoD2 dataset is very clean. The ground truth models consist of perfect and noise-free planes. Because these building shapes are so simple, the basic Dice loss is already very effective at recovering the structure. The benefits of a topological regularizer would likely be more obvious on a dataset with more complex features. For example, buildings with windows, archways, or interior holes present much harder topological challenges. In those cases, the IPT loss could prevent the model from filling in openings or blurring architectural details. The regularizer would also be more useful for messier datasets where the building shapes are harder to distinguish from noise.

6.2 The Accuracy-Regularization Trade-off

A primary finding of this study is that while the IPT loss layer provides a differentiable way to add structural constraints, it does not lead to a large numerical reduction in Mean Distance Error (MDE). We were able to slightly

beat the baseline in some configurations, but the difference is small (see Figure A.4 for a visual comparison). This suggests that the small gains could be the result of statistical randomness or luck during the training process rather than a major change in model capability.

This result highlights a fundamental trade-off between geometric and topological accuracy. Standard distance metrics like MDE and Chamfer distance are local, point-wise operators that prioritize the exact placement of individual voxels. Conversely, the IPT loss is a global structural regularizer. At a 64^3 resolution, the staircase effect or small voxel noise dominates the error. In this coarse regime, the IPT loss tries to enforce global rules like wall alignments that the limited voxel resolution cannot easily follow. This results in the observed surface jitter in voxel placements when the topological weight is too high.

6.3 Structural Sensitivity and Parameter Effects

The weight of the topological loss (λ_{topo}) has a strong effect on the final shape of the building. When we used a very high lambda in the stress test, the walls of the buildings became bendy (see Figure 5.7 and the comprehensive failure analysis in Figure A.5). This is likely a side effect of how the weighted point cloud is calculated in the IPT. Because the math calculates a center of mass for projections, a high weight can pull the geometry in a way that satisfies the projection but distorts the actual 3D volume. For the IPT to work best as a loss layer, the lambda should stay small so it guides the building without changing the walls too much.

We also found that there is no huge difference between using 4 directions and 10 directions. Using 10 directions might even be slightly noisier (see Figure A.3). This suggests that for a simple 64^3 grid, a few directions are enough to define the building. Adding more views might just add more conflicting information that the low resolution cannot handle.

6.4 Training Dynamics and Computational Bottlenecks

Using the IPT layer introduces a significant overhead during training. We noticed that models using the topological loss converge much slower than the baseline. This happens because the network has to balance two different goals. The local losses like BCE focus on individual voxels, while the IPT loss looks at the building as a whole. The topo loss curves show that the topological matrix is quite hard for the network to predict. However, this is acceptable for a regulator. The goal of the IPT layer is not to reach a perfect zero loss, but to guide the model so it does not create impossible shapes.

There is also a significant computational bottleneck as the resolution and scale of the transform increase. In our experiments, a standard configuration with a resolution of 20 and a scale of 50 required roughly 30 minutes per batch. However, increasing these parameters to a resolution of 60 and a scale of 100 caused the training time to jump to approximately 5 hours per batch. This 10x slowdown occurs because the IPT calculation involves projecting every voxel in the 64^3 grid onto a set of directional bins. When the resolution (R) increases, the number of bins in the resulting matrix grows, requiring more precise coordinate mapping for every voxel.

6.5 Computational Efficiency vs State-of-the-Art

The main advantage of our U-Net approach is how fast it is. The Point2Building model is an **autoregressive transformer** [1]. This means it builds the house one corner and one wall at a time. This is a slow process, and the model often has to try several times before it gets a result that looks right.

In contrast, our model is a **single-pass 3D U-Net**. Once we turn the points into voxels, the network predicts the voxelized building at once in a single step. Our model takes only **23.7 ms** to reconstruct one building. While preprocessing the points and generating the final mesh with Marching Cubes adds a small amount of overhead, the total pipeline remains highly efficient. This speed makes it possible to process thousands of buildings in a city very quickly, which would be much harder to do with slower, iterative models.

Bibliography

[1] Y. Liu, A. Obukhov, J. D. Wegner, and K. Schindler, "Point2building: Reconstructing buildings from airborne lidar point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 215, pp. 351–368, 2024.

[2] D. J. Waibel, S. Atwell, M. Meier, C. Marr, and B. Rieck, "Capturing shape information with multi-scale topological loss terms for 3d reconstruction," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2022, pp. 150–159.

[3] E. Röell and B. Rieck, "Point cloud synthesis using inner product transforms," *arXiv preprint arXiv:2410.18987*, 2024.

[4] B. Rieck, "Topology meets machine learning: An introduction using the euler characteristic transform," *Not. Am. Math. Soc*, vol. 72, no. 7, pp. 719–727, 2025.

[5] E. Roell and B. Rieck, "Differentiable euler characteristic transforms for shape classification," *arXiv preprint arXiv:2310.07630*, 2023.

[6] J. Huang, J. Stoter, R. Peters, and L. Nan, "City3d: Large-scale building reconstruction from airborne lidar point clouds," *Remote Sensing*, vol. 14, no. 9, p. 2254, 2022.

[7] Q.-Y. Zhou and U. Neumann, "5 d dual contouring: A robust approach to creating building models from aerial lidar point clouds," in *European conference on computer vision*, Springer, 2010, pp. 115–128.

[8] K. Turner, S. Mukherjee, and D. M. Boyer, "Persistent homology transform for modeling shapes and surfaces," *Information and Inference: A Journal of the IMA*, vol. 3, no. 4, pp. 310–344, 2014.

[9] J. Curry, S. Mukherjee, and K. Turner, "How many directions determine a shape and other sufficiency results for two topological transforms," *Transactions of the American Mathematical Society, Series B*, vol. 9, no. 32, pp. 1006–1043, 2022.

[10] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

[11] H. Edelsbrunner and J. Harer, *Computational topology: an introduction*. American Mathematical Soc., 2010.

[12] M. Carriere, F. Chazal, M. Glisse, Y. Ike, H. Kannan, and Y. Umeda, "Optimizing persistent homology based functions," in *International conference on machine learning*, PMLR, 2021, pp. 1294–1303.

[13] R. Ghrist, R. Levanger, and H. Mai, "Persistent homology and euler integral transforms," *Journal of Applied and Computational Topology*, vol. 2, no. 1, pp. 55–60, 2018.

[14] H.-G. Maas and G. Vosselman, "Two algorithms for extracting building information from raw laser altimetry data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2-3, pp. 153–163, 1999.

[15] L. Nan and P. Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2353–2361.

[16] C. Nash, Y. Ganin, S. A. Eslami, and P. Battaglia, "Polygen: An autoregressive generative model of 3d meshes," in *International conference on machine learning*, PMLR, 2020, pp. 7220–7229.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

DIFFERENTIABLE INNER PRODUCT TRANSFORM FOR 3D RECONSTRUCTION

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
ROBERT EEKSMAN	TOBIAS

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZÜRICH, 15.05.2026	T. Robert

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Appendix A

Appendix

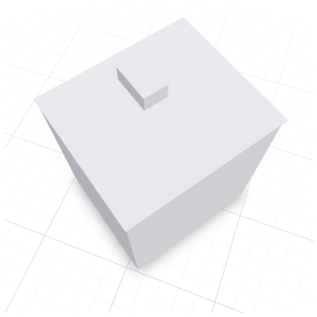
This appendix provides supplementary visual results and case studies to support the observations made in the Discussion. These examples illustrate how different hyperparameters and model configurations affect the final building reconstruction across various architectural types.

A.1 Visual Case Studies

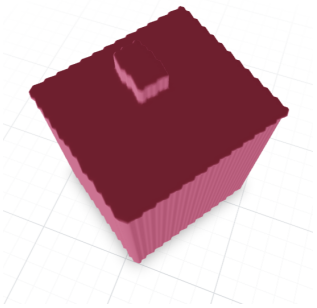
The figures in this section highlight two main behaviors observed during our experiments:

- **Model Stability:** Comparison between the baseline and regularized models (such as Res30) across different roof geometries. These cases show how the IPT loss helps maintain surface continuity where unregularized models struggled.
- **Probability Diffusion and Failure Modes:** Analysis of cases where the model exhibits high spatial uncertainty. As shown in Figure A.2, a ”diffused” probability distribution leads to a recognizable shape at $t = 0.5$, but causes a total structural collapse when the confidence threshold is increased to $t = 0.8$.

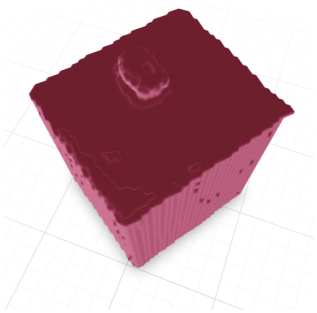
These examples serve to validate that the IPT loss acts as a ”sharpening” mechanism. By encouraging the network to concentrate probability mass along the ground truth surface, the regularized models produce architectural shells that are more robust to strict probability filtering than the baseline.



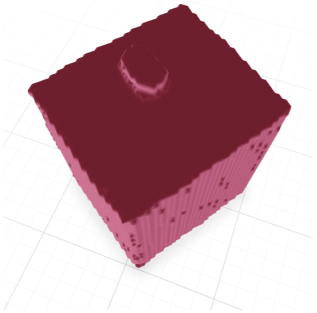
(a) Target LoD2 Mesh



(b) GT Upper Bound (MC)

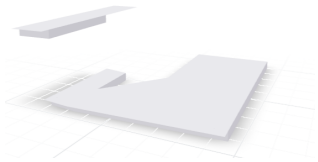


(c) Baseline ($t = 0.5$)

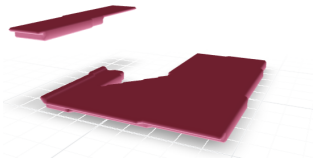


(d) Baseline ($t = 0.8$)

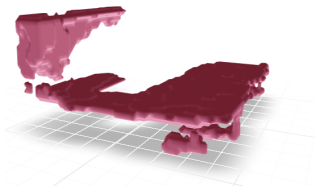
Figure A.1: Visual impact of isosurface thresholding on building 01255. A comparison between (c) and (d) reveals the sensitivity of the baseline model to probability filtering; at a higher confidence threshold ($t = 0.8$), the structure begins to erode, losing the sharp definition seen in the target LoD2 geometry (a).



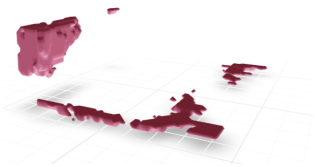
(a) Target LoD2 Mesh



(b) GT Upper Bound (MC)

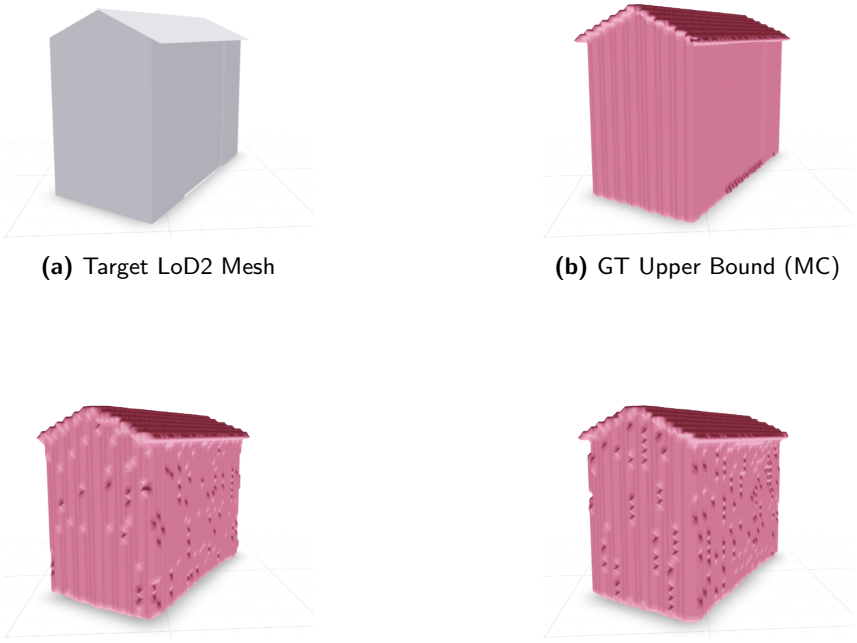


(c) Baseline ($t = 0.5$)



(d) Baseline ($t = 0.8$)

Figure A.2: Analysis of low-confidence reconstruction in building 02463. The transition from $t = 0.5$ (c) to $t = 0.8$ (d) demonstrates a complete structural breakdown, where the model's confidence in the surface placement is too low to survive strict filtering. This visualization serves as a representative example of the probability diffusion limits of the current architecture.



(a) Target LoD2 Mesh

(b) GT Upper Bound (MC)

(c) 4-Direction Projection

(d) 10-Direction Projection

Figure A.3: Comparison of projection direction density for building 02107. While the 4-direction model (c) yields a cleaner reconstruction, the 10-direction model (d) exhibits increased surface artifacts and localized noise. This supports the hypothesis that for 64^3 voxel resolutions, excessive projection views can introduce conflicting geometric constraints that the model cannot resolve effectively.

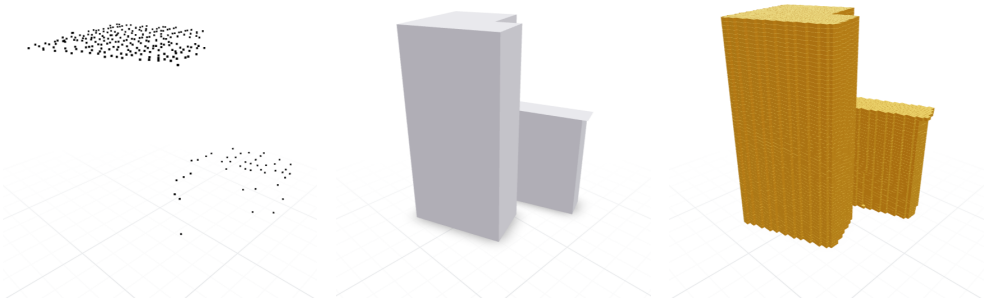
(a) Target LoD2 Mesh

(b) GT Upper Bound (MC)

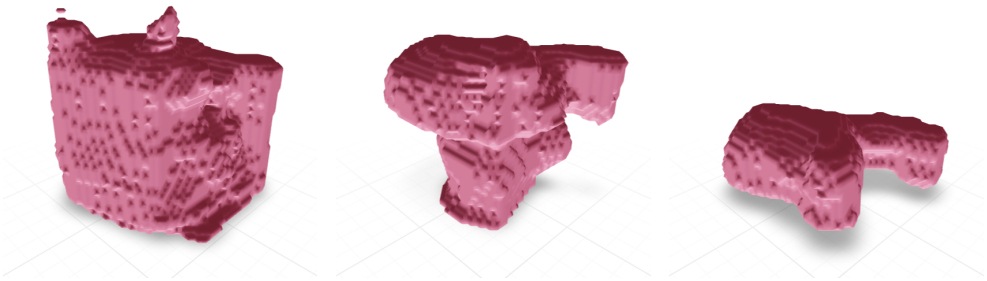
(c) Baseline Prediction

(d) Res30 Prediction ($\lambda_{topo} = 0.5$)

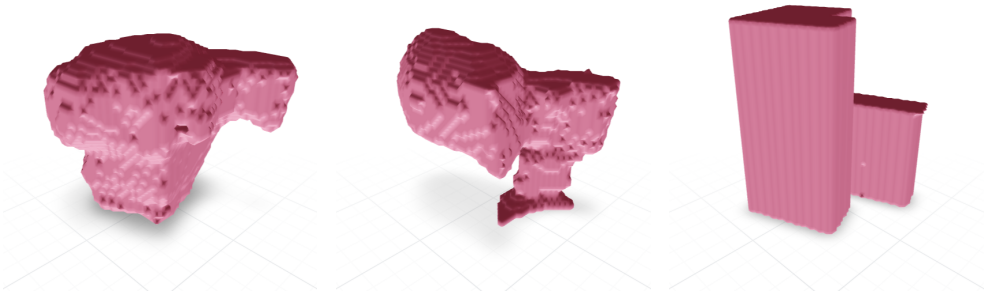
Figure A.4: Comparative analysis of structural integrity for building 02925. The Res30 model (d) demonstrates superior surface reconstruction with a straighter edge (c). This suggests that the differentiable IPT loss successfully regularizes the model to produce more coherent building envelopes, especially in regions where the LiDAR density is sparse or noisy.



(a) Source LiDAR **(b)** Target LoD2 Mesh **(c)** GT Upper Bound



(d) Baseline **(e)** Res30 ($\lambda = 0.5$) **(f)** Equal Weight ($\lambda = 1.0$)



(g) 4-Directions **(h)** Stress Test ($\lambda = 5.0$) **(i)** GT Voxelization

Figure A.5: Comprehensive analysis of reconstruction failure for building 03086. Despite varying topological weights and projection configurations, all models struggle to resolve the roof geometry. The Stress Test (h) demonstrates how an excessive topological penalty can lead to over-regularization and structural collapse, while the 4-direction model (g) fails to capture even the coarse outline, highlighting the challenges posed by high-curvature features in a low-resolution 64^3 space.