SUP'COM
Higher School of Communication of Tunis

المدرسة العليا للمواصلات بتونس
ÉCOLE SUPÉRIEURE DES COMMUNICATIONS DE TUNIS

Telecommunications engineering cycle

Graduation Project Report

3D Urban Areas Reconstruction from High Resolution 2D Images

Realized by:

Soulaimene Turki

Academic Supervisor:

Mr. Riadh Abdelfattah

Professional Supervisors:

Dr. Ksenia Bittner

Dr. Houda Chaabouni

Work proposed and fulfilled in collaboration with:

DLR

CRNS

Academic year: 2023-2024

Ecole Supérieure des Communications de Tunis
SUP'COM
2083 Cité Technologique des Communications - Elghazala - Ariana - Tunisie
Tél. +216 70 240 900 – site web : www.supcom.mincom.tn

Acknowledgements

To begin this report, I would like to express my deep appreciation to the German Aerospace Center (DLR) and the Digital Research Center of Sfax (CRNS) for offering me the invaluable opportunity to participate in this project as part of their collaborative partnership. The resources and support provided by these institutions were fundamental to the success of this endeavor. I would also like to sincerely acknowledge the German Academic Exchange Service (DAAD) for supporting this research.

I am also sincerely grateful to my colleagues in the Photogrammetry and Image Analysis department, especially the AI4BuildingModeling team, including PhD students and other team members. Our collaborative discussions were essential in fostering innovative ideas, and I am deeply appreciative of each team member's valuable insights and contributions.

A special thanks goes to my supervisors, Houda Chaabouni from CRNS, and Ksenia Bittner from DLR, whose guidance and support extended beyond the professional scope. Their mentorship was a cornerstone of my internship experience, shaping my growth throughout the project.

I am profoundly grateful to my academic supervisor, Riadh Abdelfattah, for his ongoing support, continuous mentorship, and invaluable insights, all of which have been instrumental in guiding my academic journey.

I would like to express my heartfelt thanks to my family and friends for their unfailing support and encouragement throughout this academic pursuit. Your confidence in me has been a continuous source of motivation.

Above all, I wish to acknowledge the divine guidance of Allah. His blessings and the paths He has opened for me have been instrumental in bringing me to this point. I am humbled and deeply thankful for the growth and opportunities this journey has offered.

As I move forward, I am eager to apply the knowledge and skills gained here to make meaningful contributions in future endeavors.

Dedication

This endeavor is dedicated to all the cherished individuals who stood by me, offering their unwavering support and unwavering belief in my journey.

To my parents, **Ali** and **Rim**, thank you for your patience, support, and prayers. I am forever indebted for all that you have done for me, and I can never adequately express the depth of my appreciation for your endless love and care.
May God grant you long life and good health,

To my beloved siblings, **Sirine**, **Sabrine** and **Safouen**, I can't imagine a world without you, who have consistently been my greatest pillars of support.
I wish you prosperity and triumph in every undertaking you pursue.

To my beloved **Chaima Kammoun** and my dear friend **Akram Dhouib**, your presence in my life has been a source of comfort, inspiration, and joy. Your encouragement and love have shaped my path in countless ways, and I can't thank you enough for being a part of my life.

- Soulaimene

List of Acronyms

AI	*Artificial Intelligence*
CD	*Chamfer Distance*
CCD-3DR	*Consistent Conditioning in Diffusion for Single-Image 3D Reconstruction*
CDPM	*Centered Denoising Probabilistic Models*
DDPM	*Denoising Diffusion Probabilistic Models*
GAN	*Generative Adversarial Networks*
GPU	*Graphics Processing Unit*
LiDAR	*Light Detection and Ranging*
MLP	*Multi-Layer Perceptron*
MSE	*Mean Squared Error*
PC²	*Projection-Conditioned Point Cloud Diffusion*
PVCNN	*Point-Voxel CNN*
SAM	*Segment Anything Model*
VAE	*Variational Autoencoders*

ViT	*Vision Transformer*
VRAM	*Video Random Access Memory*
2D	*Two-dimensional space*
3D	*Three-dimensional space*

Abstract

The reconstruction of detailed 3D point clouds of urban buildings from single-view images remains challenging due to limitations in existing methods, which often focus on rooftops while neglecting walls and the ground. The lack of comprehensive datasets containing complete 3D point clouds and the difficulty in obtaining accurate camera pose information from single-view images further complicate the process.

To address these challenges, we propose a novel approach that reconstructs detailed 3D point clouds of urban buildings, capturing the full structure, including rooftops, walls, and the ground, for a more comprehensive representation. Our method leverages a generative artificial intelligence diffusion model guided by edge-aware features, such as binary masks and Sobel edge maps, to progressively refine geometric details. These features enable the model to better capture architectural contours, improving the accuracy and precision of the 3D reconstruction.

A key contribution of our work is the creation of a custom dataset to address the scarcity of comprehensive 3D data. This dataset includes complete 3D point clouds and camera pose information, predicted directly from single-view images using our methodology. By incorporating these predicted camera parameters, the dataset ensures accurate alignment of features onto the 3D point cloud, providing a robust foundation for model training and evaluation. The results demonstrate that our method outperforms existing techniques, achieving highly accurate and detailed 3D reconstructions of urban buildings, with generalizability proven on another processed data from single images of Tallin City, Estonia.

Keywords: Point Cloud, Camera Pose, Building Reconstruction, Digital Orthophoto, Single-View Geometry, Diffusion Models.

Table of Contents

List of Figures

List of Tables

General Introduction

Accurate 3D building models are increasingly crucial for applications in fields like navigation, urban planning, and the creation of 3D city maps [1]. Traditionally, techniques such as Light Detection and Ranging (LiDAR) [2] and multi-view stereo imagery [3] have been used to develop these models. LiDAR, which employs aerial platforms equipped with laser scanners, generates highly detailed 3D point coordinates of terrain and structures. Multi-view reconstruction, on the other hand, captures multiple images from various angles to achieve a complete perspective of building structures.

While these techniques are foundational in the field, there is growing interest in developing new methods that offer improved efficiency and accessibility. This shift is evident in the literature, where researchers are increasingly focusing on innovative approaches to address the rising demand for urban data and high-quality modeling. Among these, monocular 3D building reconstruction, which creates models from single images, is rapidly gaining traction as a promising alternative [4, 5]. This novel approach, relying on single-view images, offers several compelling advantages over traditional techniques. For one, it significantly reduces the need for expensive equipment and complex processes, making it more cost-effective and widely accessible [6]. Moreover, monocular methods streamline workflows by eliminating the requirement for multiple images or specialized platforms like those used in LiDAR, making them more practical for a wider range of applications.

The motivation behind this method stems from the observation that humans can infer depth and perceive the 3D structure of objects using monocular cues [7] and prior knowledge. While machines do not naturally possess this ability, advancements

in deep learning have made it possible for algorithms to mimic human-like perception, enabling the creation of detailed urban models from a single image.

Nevertheless, even with monocular reconstruction, challenges remain, particularly in improving edge-aware reconstruction. This is about refining how accurately the model captures the edges of objects, which plays a crucial role in producing 3D models that are both more precise and lifelike.

This report dives into the challenging field of 3D building reconstruction, highlighting an innovative approach that uses aerial images taken from a single viewpoint to create more detailed and complete models.

The outline of this report is as follows:

1. **Chapter 1:** We define the problem and objectives of 3D reconstruction from single-view images, discuss the challenges involved, and present our project goals. We also review previous research, from classical methods to generative models, and explain the key concepts behind our methodology and dataset creation.

2. **Chapter 2:** We detail the dataset preparation process, the tools used, and the steps involved in creating it. We also outline our research pipeline, from feature extraction to the generative diffusion model that produces the 3D point cloud, highlighting the transition from 2D to 3D.

3. **Chapter 3:** We cover the implementation of our methodology, including hardware and software details. We define the evaluation metrics and compare our approach with other models, offering both qualitative and quantitative insights into its performance.

Chapter 1

Introduction and Foundations

Introduction

This chapter sets the stage for our project by presenting its scope and the specific problem we aim to address. We outline the project's goals and propose a solution informed by the challenges identified. To provide context, we include a literature review that begins with an overview of 3D reconstruction techniques and narrows its focus to methods specific to building reconstruction, exploring both traditional and modern advancements. Additionally, we introduce and explain key concepts and technical elements critical to our methodology, emphasizing how they align with and support the research objectives. This foundation paves the way for the detailed contributions presented later in the report.

1.1 Scope

This work entitled "3D Urban Areas Reconstruction from High Resolution 2D Images" is carried out within the framework of the Graduation Internship presented in order to obtain the National Engineering Diploma of the Higher School of Communication of Tunis for the academic year 2023/2024. The internship was hosted in collaboration between German Aerospace Center (DLR)[8] and Digital Research Center of Sfax (CRNS)[9] and supported by the German Academic Exchange Service (DAAD)[10].

1.2 Problem Definition

LiDAR uses scanners that send out pulses of light and measure how long it takes for the light to bounce back. This process helps create detailed 3D point data of the terrain or objects. However, this method is not only costly due to the expense of specialized equipment (sensors) but also because of the aerial platforms required (drones, helicopters, or airplanes), which add additional costs for fuel, maintenance, and skilled operators. Moreover, LiDAR often produces incomplete 3D models of buildings (according to [11]). A single scan typically captures only the roof and some wall structures, leaving other important features underrepresented or missing.

Similarly, multi-view reconstruction is a method of creating a 3D model by taking several pictures of an object or scene from different angles. When these pictures are combined, they provide enough information to recreate the object's shape. Satellite-based image capture, commonly used in this approach, is subject to delays due to fixed orbits of satellites [12]. As a result, obtaining updated images of a location can take days or weeks, during which significant changes (such as construction or natural disasters) could occur, leading to data gaps. Furthermore, high-resolution satellites, which are essential for accurate reconstructions, tend to have longer revisit times, further limiting the availability of up-to-date, high-quality imagery.

Even when satellite images are acquired successfully, another significant challenge arises: the management and storage of large volumes of data [13]. For an accurate reconstruction of a single building, multiple high-resolution images (typically at least two) are required, which increases storage demands and complicates data handling.

On the other hand, single-view 3D reconstruction stands out as a highly advantageous method, particularly due to its low cost and ease of use. The fact that it only requires one image instead of multiple images or complex sensors inherently makes it more affordable compared to the old methods. However, monocular 3D reconstruction still faces its own set of limitations. Many of these methods focus exclusively on generating point clouds of rooftops from a single image, resulting in models that

are incomplete and lack essential structural details [14]. These approaches often fail to capture the full complexity of the building, overlooking critical features such as walls and other architectural elements. Furthermore, we identified a significant gap in the availability of suitable datasets for our method. Specifically, most existing datasets [15] lack two critical components: point clouds of buildings and accurate camera poses. Since many machine learning-based reconstruction methods rely on point clouds for detailed 3D modeling, the absence of these data elements limits their applicability. Furthermore, accurate camera poses, which include the position and orientation of the camera in relation to the building, are crucial for creating an accurate 3D model. These poses help us figure out how to translate the pixels in a 2D image back into their correct 3D locations in space, ensuring that each pixel matches the right spot in the real world. Without this information, it would be difficult to accurately reconstruct the building, as we wouldn't know how to correctly position the parts of the image in 3D space. The lack of such data in available datasets presents a notable challenge for training robust models capable of generating accurate and complete 3D structures.

1.3 Project Goals

The issues identified in the previous section guide us in defining clear objectives for this project. Our primary goal is to address the challenges of 3D building reconstruction by generating detailed 3D models, including key structural elements such as roof outlines (the visible shape or boundary of a building's roof). These models will represent the entire building, including the roof, walls, and ground, all derived from top-down, single-view images showing only the roof, with no view of the facade, specifically digital orthophotos which are aerial images corrected to ensure accurate distances and scale for precise measurements. But we're not stopping there, another important objective is to develop a dataset of point clouds for various buildings, each paired with the camera pose (the position and orientation of the camera in space), to overcome the issues of limited datasets.

1.4 Literature Review on Single-View 3D Reconstruction

In the field of monocular 3D reconstruction, the literature reveals a variety of techniques used to address the problem, spanning from classical methods to learning-based approaches and, more recently, generative models. This progression resembles a timeline: with the introduction of deep learning, the focus shifted towards deep learning-based 3D reconstruction methods. More recently, the advent of generative AI has caused a shift in research focus once again, with many studies now exploring this new direction.

1.4.1 Classical Methods

The task of reconstructing 3D shapes from single-view images has been a primary focus in computer vision research for over two decades. Early methods relied on monocular cues, similar to how human vision interprets depth to infer 3D structure. Techniques included shading [16] (using gradients of reflected light intensity to deduce shape), texture [17] (analyzing surface patterns and variations), and silhouettes [18] (using object outlines to approximate form), each contributing to initial advancements in approximating 3D shapes from a single viewpoint. Unfortunately, these methods have very limited generalizability due to their reliance on predefined cues. Small changes in lighting, texture, or object orientation can drastically impact the performance and accuracy of the reconstruction.

When it comes to building reconstruction, early methods focused on identifying basic features like roof lines and segments. These features were then pieced together to form a complete roof structure [19, 20]. In addition to these geometric details, researchers also turned to shadow analysis to improve the accuracy of the reconstruction. By studying the shadows cast by buildings [21], they could estimate heights and better understand how different structures relate to one another. However, these methods often relied on simplified models, such as rectangular or square shapes, which limited their effectiveness for more complex buildings. Additionally, since the

accuracy of these approaches depended on precise sunlight positioning, changes in lighting could lead to errors, making the methods less reliable in real-world conditions. These limitations led researchers to explore learning-based methods, which aim to automatically learn robust features from large datasets.

1.4.2 Learning-based Methods

With the advent of machine learning and deep learning, classical methods for 3D reconstruction have been replaced by more advanced learning-based approaches, such as the widely used encoder-decoder [22, 23] frameworks. In an encoder-decoder architecture, the encoder is responsible for extracting high-level 2D features from input images, typically through convolutional layers [24] that capture spatial hierarchies of visual information. These extracted features are then passed to the decoder, which interprets and transforms them into 3D representations, such as volumetric grids or point clouds. The decoder's role is to reconstruct a 3D object by mapping the learned 2D features to a higher-dimensional 3D space, allowing for the generation of more complex and accurate 3D shapes. Recently, deep learning methods have been introduced to estimate building heights from single 2D images. These techniques also combine the building's height estimation with footprint extraction from images taken at off-nadir angles.

Recent advancements in building roof reconstruction from single-view images have leveraged the GraphX-Conv [25] architecture, as introduced in the sat2pc [14] method, to deform point clouds, followed by a refinement network for further detail enhancement. However, while these learning-based techniques are effective for reconstructing rooftops, they are often limited in their ability to capture finer details of the building's structure. This is a significant limitation, as it restricts the accuracy of the full 3D model.

1.4.3 Generative AI Methods

Generative AI techniques for 3D reconstruction generally fall into three categories: Variational Autoencoders (VAEs) [26], Generative Adversarial Networks (GANs) [27], and Diffusion Models [28]. VAEs are designed to simplify complex data by compressing it into a lower dimensional space, which helps preserve the important details while making it easier to work with. GANs, on the other hand, consist of two networks: one creates new data, while the other evaluates how realistic it is. It's like a friendly competition between the two, pushing each to improve and resulting in more convincing outcomes. Diffusion models work differently by adding noise to a dataset and then gradually removing it, refining the data bit by bit to recover its original form.

In 3D reconstruction, some researchers [29] have found that combining VAEs and GANs in a hybrid approach works well by leveraging the strengths of both methods. For instance, they use a VAE to simplify 3D point cloud data, making it easier to handle by compressing it into a more compact form. Then, this compressed data is passed on to a GAN, which does the hard work of turning it back into detailed 3D shapes. However, due to the training complexity and high computational cost of this approach, many methods are now shifting focus towards diffusion models, which are rapidly gaining popularity in the research field. As a result, our approach will primarily rely on diffusion models for more precise and efficient 3D reconstruction.

When we look at approaches for building reconstruction using diffusion models, we observe significant challenges in producing accurate 3D models. For example, in BuildDiff [30], two conditional diffusion models are used, one to generate a coarse representation (a simplified version) and the other to create a finer representation from general view images. Despite this, this method struggles with capturing the detailed complexities of buildings.

As a result, during our literature review, we focus on methods originally developed for objects and explore how these can be adapted to remote sensing datasets, refining them for more effective use in building reconstruction.

This leads us to discuss the reference methods we are focusing on, namely Projection-Conditioned Point Cloud Diffusion (PC^2)[31] and Consistent Conditioning in Diffusion for Single-Image 3D Reconstruction (CCD-3DR) [32], which were primarily designed for object data. These methods have demonstrated promising results in 3D reconstruction. However, they face some key limitations, such as their heavy dependence on accurate camera poses for reconstruction. Obtaining precise camera poses from aerial images can be challenging, and since these methods were not originally designed or trained for aerial data, they often produce incomplete models, missing important structural details like sharp edges and fine geometries.

In our approach, we aim to extend these two baseline methods to remote sensing tasks. Specifically, we will adapt the feature projection technique (discussed in section 2.4.2) used in PC^2 and integrate the diffusion model of CCD-3DR, which is based on Centred Denoising Probabilistic Models (CDPM) (detailed in section 1.5.2). While both baseline methods were trained on object datasets featuring items such as cars, planes, chairs, and teddy bears, our method will focus on aerial imagery and remote sensing applications. By developing and refining a dataset tailored to this domain, we aim to enable these models to generate more accurate and comprehensive 3D building reconstructions. We will elaborate on these adaptations and their implications in the key concepts section.

1.5 Key Concepts

As outlined in the general proposed solution, we will create a custom dataset and design a specific model. However, before diving into the development process, there are key concepts and tools we need to highlight. Therefore, we will first introduce the essential concepts required for preparing the dataset, followed by the concepts necessary for our methodology.

1.5.1 Core Data Concepts

Digital Orthophoto

A digital orthophoto [33] is an aerial or satellite image that has been adjusted to ensure the scale is consistent throughout. This process, called "orthorectification", corrects the image for distortions, making it a true, accurate representation of the Earth's surface where measurements of distances and areas can be made accurately. As shown in the fig. 1.1, the difference between an uncorrected aerial image and an orthophoto is clear. The uncorrected aerial image, which is essentially a perspective

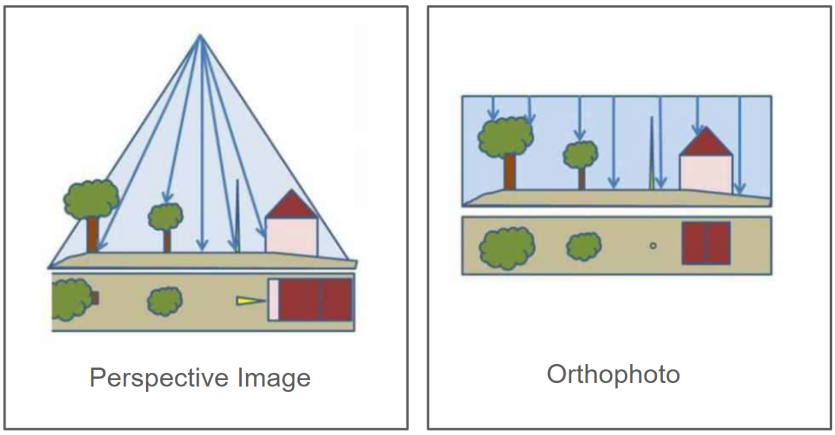


Figure 1.1: A comparison between a perspective image and a digital orthophoto [34]

image (meaning it shows objects in a way that their size and position change depending on their distance from the camera, with parallel lines converging and closer objects appearing larger), cannot accurately show the true dimensions of objects. For example, both the building facade and the tree trunk may appear to have different dimensions, which is incorrect. However, in the digital orthophoto, the image is corrected as if projected from far away, making the rays parallel, so we can see true proportions and measurements. While there are specialized tools and software for this correction process, we won't dive into the details here, as we assume all the images we're working with are already digital orthophotos.

Point-Cloud

A point cloud is a collection of data points in 3D space, where each point is defined by (x, y, z, c)). The (x, y, z) coordinates specify the point's position, while c includes extra details like color, intensity, surface normals, or other features which we'll dive into later. Since our goal is to create 3D models, we're focusing on point clouds because they're much easier to work with computationally compared to other 3D formats like meshes. This means all our models need to be tailored specifically to handle point clouds effectively.

Point Cloud Rasterization

Rasterization of point clouds is the process of transforming 3D data, such as a point cloud, into a 2D image that can be displayed on a screen. In other words, it involves converting objects from a 3D scene into pixels on a 2D image. Each point in the point cloud has 3D coordinates (x, y, z), and these points must be projected onto the image plane based on the camera's perspective.

To perform this projection, the camera's extrinsic parameters are used to map the 3D coordinates to 2D pixel locations. After the points are projected, depth sorting is performed to ensure that the closest points to the camera are rendered first, in front of the farther points. This step prevents visual artifacts where distant points may obscure closer ones, ensuring an accurate representation of the scene.

Voxelization

Handling billions of points in a point cloud can be incredibly challenging and time-consuming. This is where voxelization comes in to make things more manageable. Voxelization works by converting the point cloud into a grid of tiny 3D cubes, called voxels (like in fig. 1.2), grouping nearby points into these cubes. It's similar to how pixels work in 2D images, pixels represent small parts of a picture, while voxels represent small volumes in 3D space. This process simplifies the data, making it

easier to process while still preserving the overall shape and structure of the 3D model.

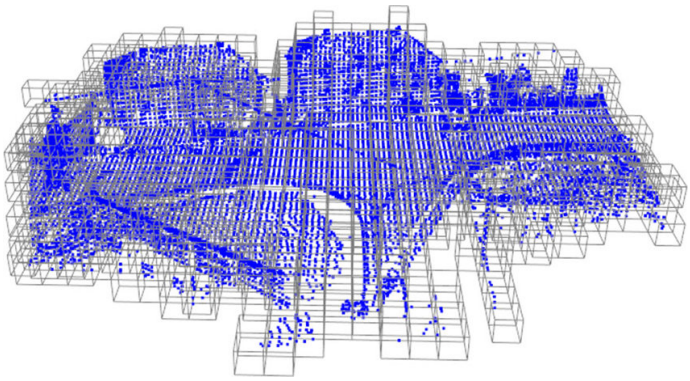


Figure 1.2: Voxel grid representation of point clouds, showcasing visible cubes [35].

1.5.2 Core Methodological Concepts

U-Net

U-Net [36], or "U network," is named for its characteristic U-shaped architecture, as shown in the fig. 1.3.

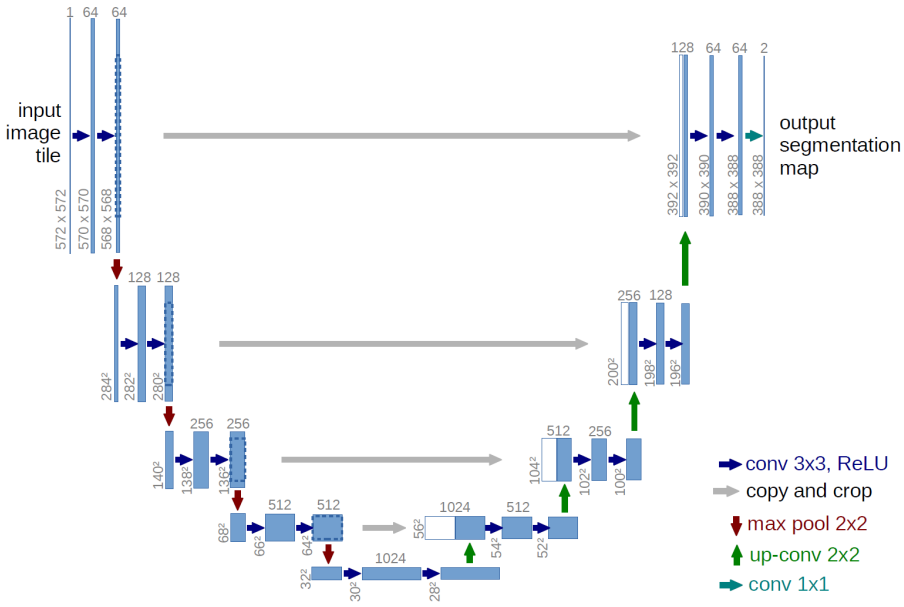


Figure 1.3: U-Net Architecture [36]

This architecture is built on two main components: an encoder and a decoder. The

encoder is responsible for capturing contextual information from the input image, reducing its spatial resolution in the process. This downsampling helps the model focus on identifying essential features, which we call feature maps, by ignoring small, less important details.

Meanwhile, the decoder works to upsample these feature maps to their original resolution, reconstructing the spatial details to generate a high-resolution output. A key aspect of the U-Net design is the use of skip connections between the encoder and decoder, which allow the decoder to directly access high-resolution features from the encoder. This helps the decoder better locate and refine features, resulting in a more precise final output.

Diffusion Models

A diffusion model [28] is a type of generative model capable of producing high-quality data. Its core idea is straightforward: start with a dataset sample, gradually add random Gaussian noise to corrupt the data until it becomes pure noise, and then train the model to reverse this noise, effectively generating new samples. This process is divided into two key stages: the forward process, where noise is progressively added, and the reverse process, where noise is removed step by step to recover or generate the data.

There are various diffusion-based models, each defining the forward and reverse processes in its own way. To understand these differences, we will explore two models, Denoising Diffusion Probabilistic Models (DDPM) [28] and Centred Denoising Probabilistic Models (CDPM) [32] and examine how they approach these processes, including their mathematical formulations.

- **Forward Process (DDPM):**

  Involves the gradual addition of Gaussian noise to clean data x_0, transforming it into a noisy version x_t at each time step t, where $t \in \{1, 2, \ldots, T\}$, with T denoting the total number of diffusion steps. As t progresses, the data becomes

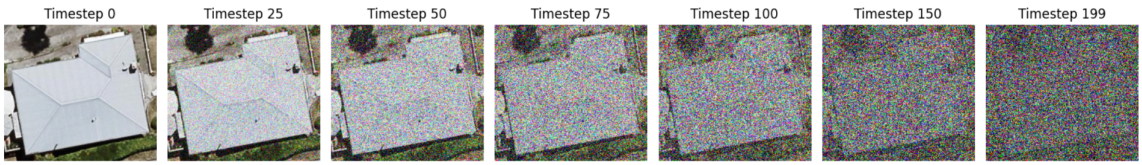


Figure 1.4: Entire forward diffusion process with 200 time steps.

increasingly noisy, ultimately reaching x_T, a fully noisy state resembling a sample from a Gaussian distribution, as presented in fig. 1.4 where $T = 200$.

This is represented by the following transition:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I^{3N}), \tag{1.1}$$

where x_t is the noisy data at time t, α_t is a parameter controlling the noise schedule, and I^{3N} is the identity matrix of size $3N$, where:

- N: the number of points in the point cloud (which is 10000 points in our case),
- Each point has 3 dimensions, typically represented as (x, y, z).

The full forward process can be condensed as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I^{3N}), \tag{1.2}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, and ϵ represents the Gaussian noise added to the data. In this equation, $\mathcal{N}(0, I^{3N})$ denotes a Gaussian distribution with mean 0 and covariance matrix I^{3N}. This process ultimately transforms the clean data x_0 into a fully noisy version x_T, which resembles a sample from a Gaussian distribution.

- **Reverse Process (DDPM):** Aims to reconstruct the clean data by removing the noise, step by step, starting from the noisy data x_T as shown in fig. 1.5. The reverse process is modeled as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(t)) \tag{1.3}$$

where

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \tag{1.4}$$

is the predicted mean at time step t, with $\epsilon_\theta(x_t, t)$ representing the noise predicted by the neural network, and $\beta_t = 1 - \alpha_t$ denoting the noise removal rate. When we remove noise, we don't remove all of the predicted noise, we only remove part of it. This is because the process needs to keep some randomness to ensure the results are diverse and follow the correct data distribution. Removing only part of the noise helps prevent the model from producing overly similar or deterministic outputs.

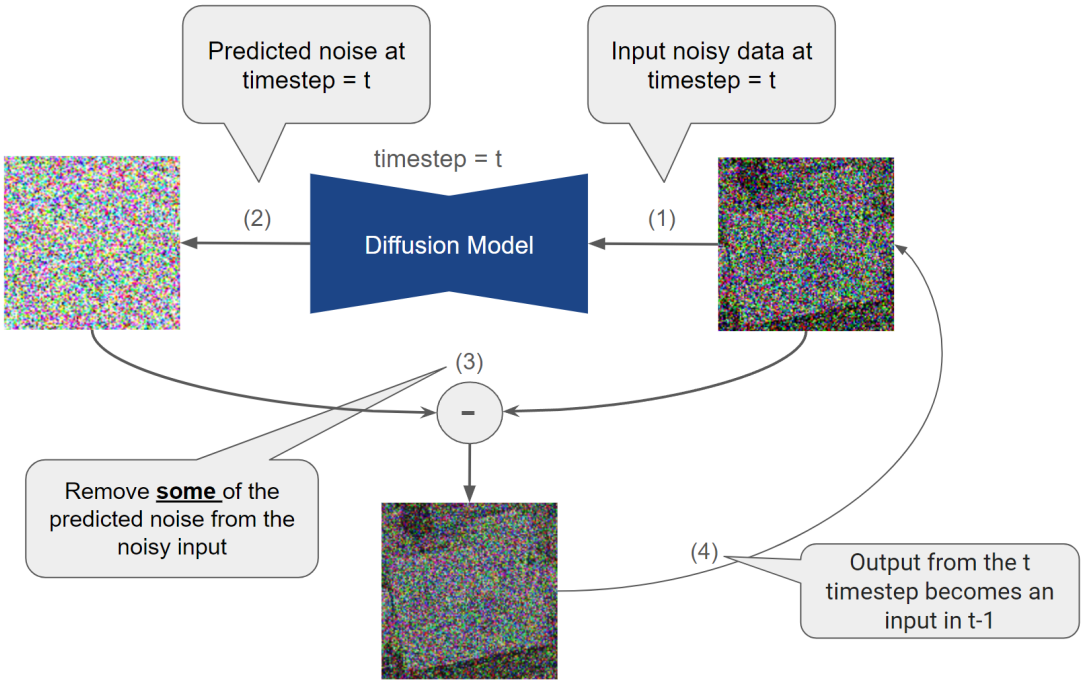


Figure 1.5: The steps of the reverse process start with the input noisy data. This data is passed through the diffusion model to predict the noise. Some of the predicted noise is subtracted, and the resulting image is used as the new input for the next step in the process.

This process works by gradually removing noise at each step to recover the original data, x_0, from its noisy version, x_T. However, when dealing with point clouds, each point is treated separately, without considering how it relates to the other points in space. Because of this, there's no way to ensure that the

center of the point cloud stays in place as the noise is removed. This results in an issue known as **center bias** in DDPM.

- **CDPM:**

  In CDPM, the aim is to fix the center bias that happens during the reverse process in DDPM. In traditional DDPM, the center of the point cloud can move around as the process progresses, which can make 3D reconstructions less accurate. To solve this, CDPM ensures that the denoised point cloud stays centered (zero-mean) at every step, preventing the center from drifting and improving the overall accuracy, as shown in fig. 1.6.

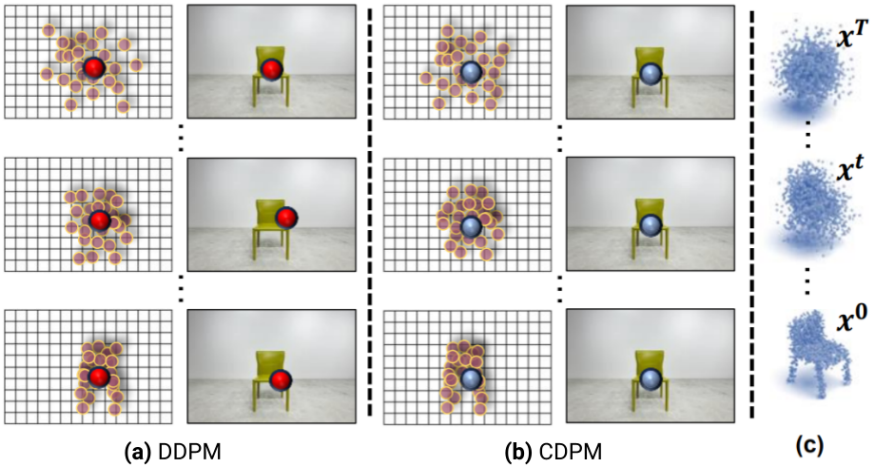


Figure 1.6: Visualizing the differences between DDPM and CDPM (adapted from [32]).

In practice, CDPM ensures that both the added noise during the forward process and the predicted noise during the reverse process are centered at each step by applying a centralizing operation that adjusts the noise by subtracting its mean, effectively centering it around zero, given by $\epsilon = \epsilon - \bar{\epsilon}$ and $\hat{\epsilon} = \hat{\epsilon} - \bar{\hat{\epsilon}}$; similarly, during inference, the point cloud x_{t-1} is centered as $x_{t-1} = x_{t-1} - \bar{x}_{t-1}$.

This centering mechanism ensures that the denoised point cloud remains aligned with its original center throughout the reverse process, improving reconstruction quality.

In summary, while the forward process can generally be defined using mathematical formulas, the reverse process requires a neural network model. In most cases, a U-Net architecture is used for this purpose. The U-Net is typically trained to predict the noise added to the image at each step of the process. Researchers favor U-Net because its architecture effectively captures both global context and local details, making it well-suited for tasks like denoising. However, most U-Nets are designed to work with 2D images as input, predicting the noise directly on these 2D data structures. What we need, instead, is a U-Net tailored for point cloud data, where the input consists of 3D points rather than 2D pixels. This naturally leads us to discuss PVCNN (Point-Voxel Convolutional Neural Network), which adapts the principles of convolutional networks to efficiently process and predict noise in 3D point clouds.

Point-Voxel CNN (PVCNN)

Since we'll be working with diffusion models and point clouds, we need a model capable of predicting the noise added to point clouds. While a 3D U-Net might seem like a natural choice, researchers found that it requires approximately 10 GB of GPU memory even for small inputs, making it impractical for training larger models. To address this limitation, researchers developed an alternative approach: the Point-Voxel CNN (PVCNN) [37].

This model processes a point cloud using two parallel branches: a point-based branch and a voxel-based branch, as demonstrated in fig. 1.7. The voxel-based branch begins by applying voxelization to transform the input point cloud into voxel grids. Once in this format, 3D convolutions are performed on the voxel grids, which is computationally faster compared to operating directly on individual points. In parallel, the point-based branch processes the raw point cloud directly using a multi-layer perceptron (MLP), a type of neural network composed of fully connected layers designed to learn patterns and relationships in the data. Finally, the outputs from the 3D convolutions and the MLP are fused, combining the strengths of both approaches to effectively capture the global and local features of the point cloud.

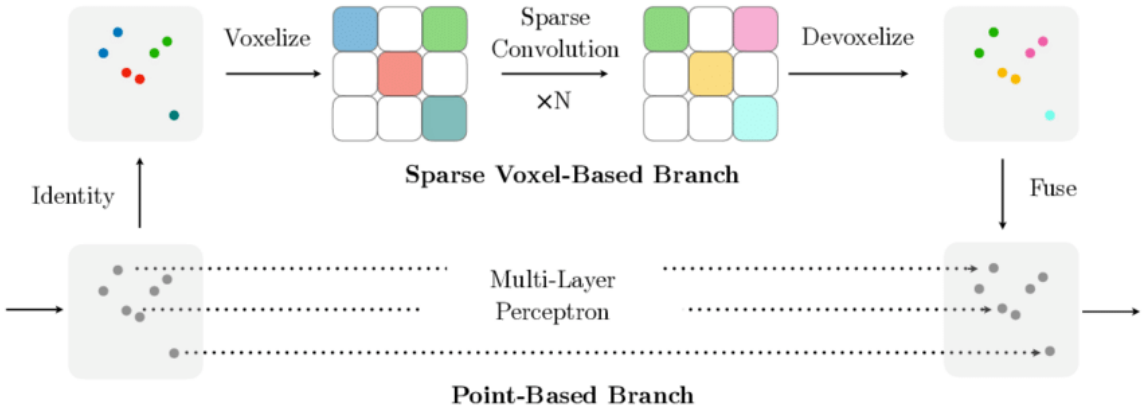


Figure 1.7: PVCNN architecture [37] showcasing two parallel processes, one at the top and the other at the bottom.

Vision Transformer (ViT)

To extract image features from a single input image, we employ the Vision Transformer (ViT) [38] as our feature extractor. The Vision Transformer, originally introduced as a transformer model for computer vision tasks, builds on the transformative success of attention mechanisms in processing text. In natural language processing, transformers enable models to understand relationships between words in a sentence. Similarly, ViT applies this concept to images by identifying relationships between different parts of an image. The core idea behind ViT fig. 1.8 is to divide an image into a series of smaller patches, much like splitting a large picture into smaller tiles. Each patch is then mapped to a vector representation using a linear transformation. To ensure that the model retains information about the original arrangement of these patches, positional encoding is added to each vector. This combination of features is then fed into transformer encoders, which utilize attention mechanisms to learn how patches relate to one another, enabling the model to understand the overall image context.

The architecture of ViT is elegant in its simplicity. By cutting the image into patches and processing them as sequences (akin to words in a sentence), the transformer can create a series of vector embeddings representing different parts of the image. In addition, ViT includes a "classification token" (or global embedding), which serves

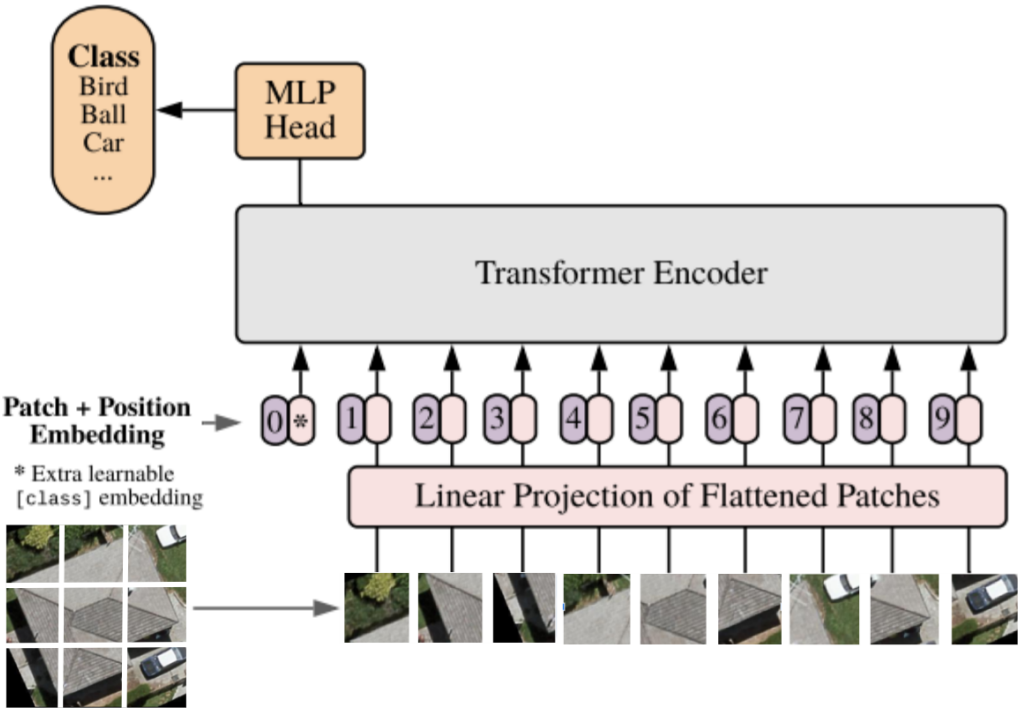


Figure 1.8: Architecture of Vision Transformer [38]

as a holistic representation of the entire image. This global embedding can be used directly for downstream tasks such as classification, detection, or other applications.

In summary, ViT transforms an image into a sequence of feature-rich vectors, enabling a powerful and flexible approach to computer vision tasks.

Conclusion

In conclusion, this chapter lays the groundwork for a deeper understanding of our project. We began by outlining the broader context, including the academic background, before focusing on the specific problem and the key challenges that have guided our research. We also reviewed the literature on 3D reconstruction, covering a progression from traditional methods to learning-based approaches, and finally to generative models for objects and buildings. Additionally, we introduced key concepts such as diffusion models and Point-Voxel CNN, which will play a pivotal role in the subsequent chapters.

Chapter 2

Dataset and Methodological Procedures

Introduction

This chapter focuses on the design of our proposed solution and the process of creating its key components. We begin by detailing the generation of our custom dataset, which forms the foundation for training, as no existing dataset directly meets our requirements. From there, we provide an overview of our approach to the Single-View Reconstruction problem, presenting the pipeline's structure and delving into the design and functionality of its individual components. This comprehensive exploration highlights how our solution was developed and implemented.

2.1 Dataset

The initial dataset, created by [39] in their work, is based on aerial images and uses a method that encodes roof topology through graph structures. It reconstructs entire buildings from single images by adding facade planes along the roof outline and a base plane at the bottom, resulting in complete 3D meshes. The roofs in this dataset are clear, well-maintained, and uniform, making it an excellent ground truth for reconstruction tasks (an example of the dataset can be seen in fig. 2.1).

To use this dataset for specific needs, components such as binary masks, point clouds

(the target of our reconstruction task), and camera parameters can be extracted, as these are essential for our pipeline.

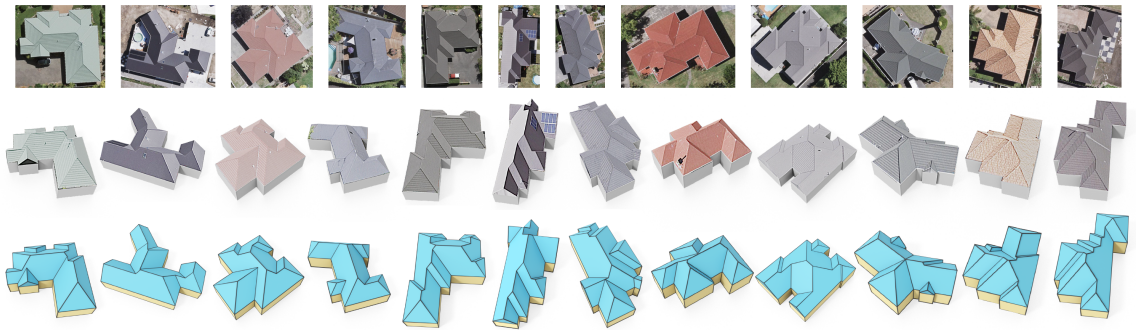


Figure 2.1: Example of the initial dataset [39]: the top row features the RGB image, while the subsequent rows display the mesh files corresponding to the building in the RGB image.

2.1.1 Point Cloud

MeshLab

MeshLab [40] is an open-source software designed for processing 3D meshes (a collection of vertices, edges, and faces that define the shape of a 3D object or surface) is often not used directly in 3D reconstruction because it requires dense, precise surface data, which can be difficult to obtain without detailed scanning or modeling. MeshLab makes it easy to edit, clean, and render 3D models. It offers a wide range

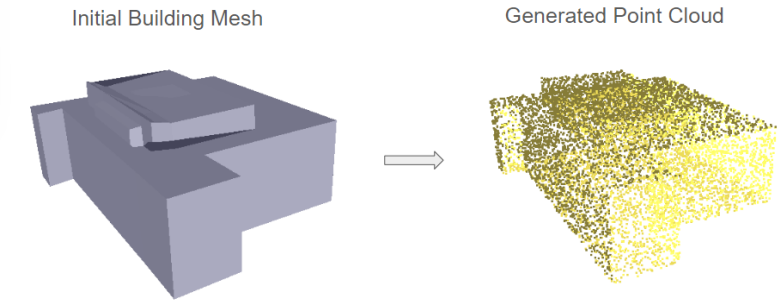


Figure 2.2: Visualization of an example mesh-to-point cloud transformation.

of tools for working with triangular meshes, point clouds, and volumetric data. In

our work, we used this software with a custom script that leverages its tools, particularly the Poisson Disk Sampling method, to convert a 3D mesh into a point cloud (fig. 2.2).

Poisson Disk Sampling method

A mesh is made up of vertices, edges, and faces that define the geometry of a surface. Since this geometry is already well-defined, we have a clear guide for where points can be placed.

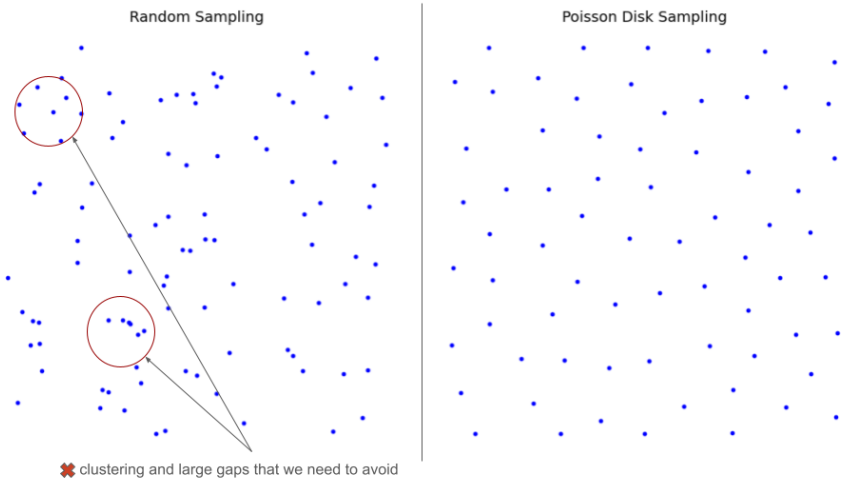


Figure 2.3: Comparison between random sampling and Poisson disk sampling, highlighting the issues to avoid in random sampling.

To create a point cloud from the mesh, we sample points from its surface. The key question is: **what sampling technique should we use?** Ideally, we want a method that ensures the points are evenly distributed across the surface, avoiding clusters or large gaps (see fig. 2.3).

This is exactly what Poisson Disk Sampling [41] achieves, providing a uniform and well-spaced distribution of points that effectively covers the entire mesh.

this method is simple. It begins with the triangular faces of the mesh and defines a minimum distance, r, which determines the required spacing between points. The first point is placed randomly within the region. For each subsequent point, a candidate point is generated within a "ring" between distances r and $2r$ from the current

point, with a random angle for its placement (see fig. 2.4). The candidate point is

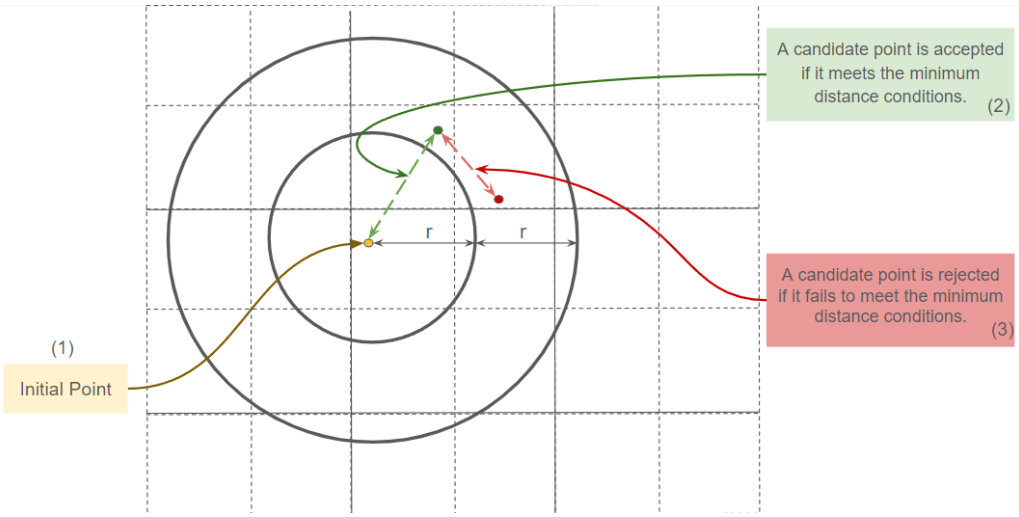


Figure 2.4: Step-by-step Process of Poisson Disk Sampling.

then checked to ensure that it is at least r away from all previously accepted points. If it satisfies this distance requirement, the point is considered accepted and added to the list of accepted points. This ensures that the points are evenly spaced and prevents clustering or large gaps. The process continues until no more valid candidate points can be placed, resulting in a uniform distribution of points across the mesh.

2.1.2 Binary Mask

It's important to highlight the role of binary masks in our pipeline. Binary masks are essential for two key reasons. First, they contribute as conditioned features to the model, a topic that will be discussed in detail in section 2.4.2. Second, and most critically, they are necessary for extracting the camera parameters of the buildings, which will be covered in section 2.2.2.

When generating a binary mask, we explored several approaches. Initially, we tested state-of-the-art models like Segment Anything Model (SAM) [42] and Mask R-CNN [43], which are well-known for segmentation tasks, to segment buildings and generate building masks. As shown in the figure, we evaluated the results from both methods.

We found that SAM, for instance, was not ideal for our needs (see fig. 2.5). It tended to miss parts of buildings due to occlusion from trees, and it sometimes incorrectly segmented a single building into multiple disconnected parts. Based on these observations, we decided to discard SAM. Next, we trained our own Mask

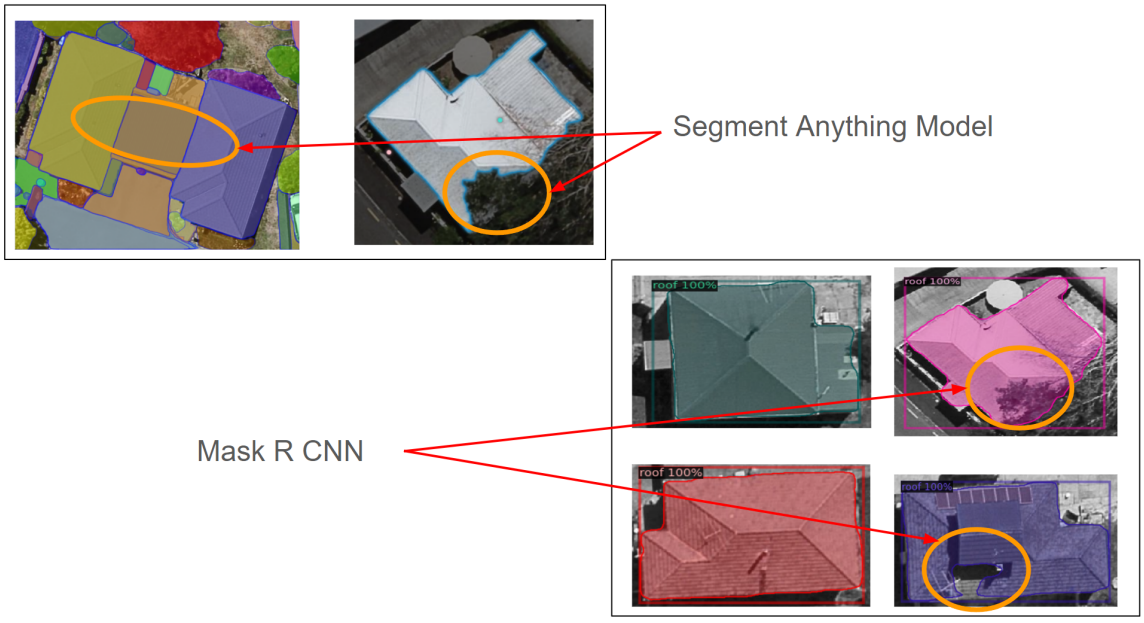


Figure 2.5: State-of-the-art methods for building segmentation, highlighting their challenges and limitations.

R-CNN model after annotating around 700 images for training. This improved the model's ability to detect entire building roofs, even when occluded (see also fig. 2.5). However, there were still occasional issues, such as missing parts of buildings or the segmentation lines not being perfectly aligned, sometimes cutting inside the building. Ultimately, we concluded that Mask R-CNN also wasn't ideal for our task. We then shifted to our proposed method, which uses roof vertices and faces for a more accurate and robust solution. Our method involves using the vertex files and face files. The vertex files store the **image coordinates** of the roof's corner points (see fig. 2.6), while the face files define the sequences of these vertex indices that outline the roof segments. Using this information, we generate a binary mask for each roof, clearly marking the building boundaries and distinguishing the structures from the

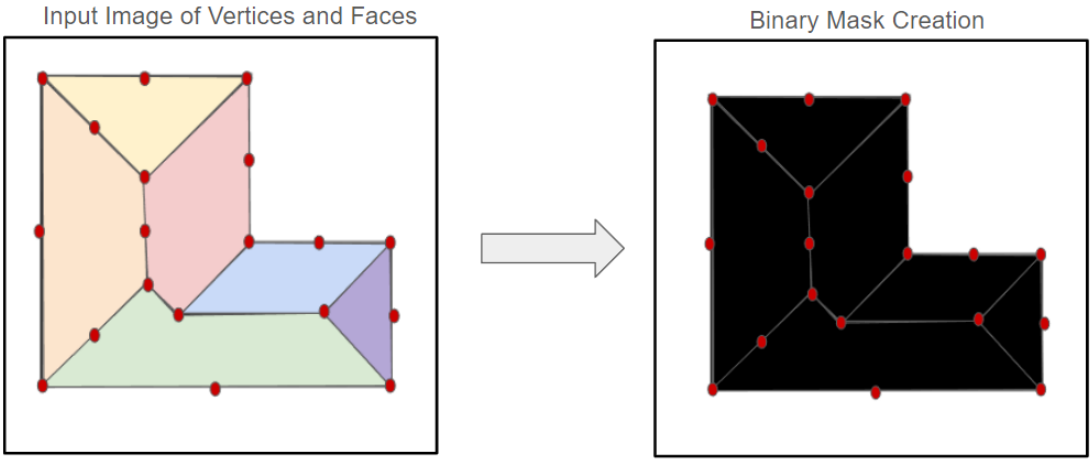


Figure 2.6: Visualization and explanation of how to group the faces together and assign a black value to create a binary mask

surrounding area. This approach ensures precise and well-defined segmentation of roofs in the dataset.

2.1.3 Sobel Edge Maps

As will be discussed later in section 2.4.2, Sobel edge maps are an essential part of our process, and thus, we need to generate them using the Sobel operator.

The Sobel operator [44] is used for edge detection by calculating the gradient of image intensity in both horizontal and vertical directions. It uses two matrices, K_x and K_y, corresponding to each direction.

The **horizontal Sobel kernel** K_x is defined as:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The **vertical Sobel kernel** K_y is defined as:

$$K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

To obtain the final edge-detected image, we calculate the gradient magnitude G by combining the horizontal and vertical gradients G_x and G_y as follows:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

Alternatively, a simplified version of the magnitude can be calculated by:

$$G \approx |G_x| + |G_y| \tag{2.2}$$

This results in an image that highlights the edges based on the strength of intensity changes in both directions.

The challenge here is that applying the Sobel filter to the entire image would result in extra edges from the background, which are not relevant to our needs. To address this, we first applied a binary mask (see fig. 2.7) to isolate the building and remove the background. With the background eliminated, we then used the Sobel operator to detect intensity changes in both horizontal and vertical directions, creating an edge map focused solely on the building's edges.

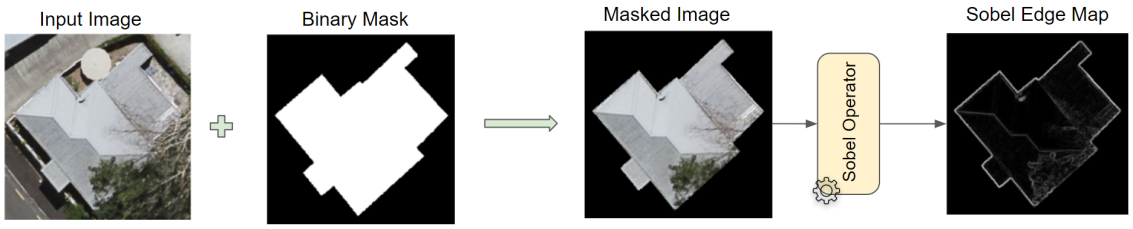


Figure 2.7: Step-by-step process to create a Sobel edge map, excluding background edges.

⚠ **Important:** Several edge detection methods, including Sobel [44], Canny [45],

and Laplacian of Gaussian (LoG) [46], were considered for our task of accurately detecting roof outlines. As shown in fig. 2.8, we evaluated these methods visually and ultimately chose Sobel for several reasons. The Canny edge detector requires parameter tuning (threshold selection) for each image, which is time-consuming and can lead to inconsistencies across datasets. Additionally, Canny can produce edge discontinuities, making it less reliable for detecting continuous roof outlines. On the other hand, the Laplacian of Gaussian (LoG) method tends to overemphasize fine details and amplify noise, capturing textural features of the roof but often missing the clear structural outlines. In contrast, Sobel provided consistent and reliable edge detection without the need for extensive tuning, making it the most suitable choice for our application.

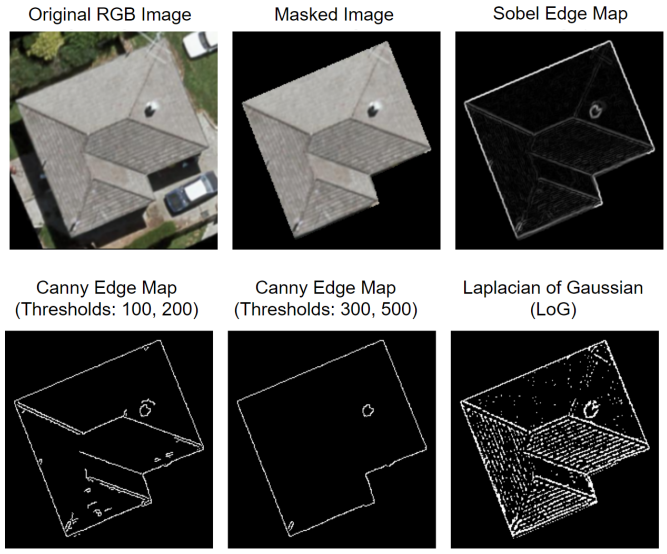


Figure 2.8: Comparison of Sobel, Canny (with thresholds 100, 200 and 300, 500), and Laplacian of Gaussian (LoG) for detecting roof outlines. Sobel proved to be the most consistent, while Canny required threshold tuning, and LoG emphasized fine details but missed the structural outlines.

2.2 Rasterization and Camera Parameters

In order to fit our pipeline and project the features, which will be covered in section 2.4.2, we need to generate the camera parameters. Getting these parameters from a single image can be tricky, as most methods rely on multiple images to estim-

ate them. However, we will explain how we can extract camera parameters from just one image. Before diving into that, we first need to discuss the Powell algorithm, which is essential for optimizing these parameters in our approach.

2.2.1 Powell Algorithm

The Powell algorithm [47] is a method for finding the lowest point of a function without needing to know its derivative, which can be helpful when the function is complex or doesn't have a clear gradient. Unlike methods that rely on gradients to guide the search, Powell's approach takes a different route. It works by exploring multiple directions one by one, testing different step sizes along each direction to find the minimum. It's a clever way to optimize a function without needing detailed slope information, making it ideal for situations where derivatives aren't available or too difficult to compute.

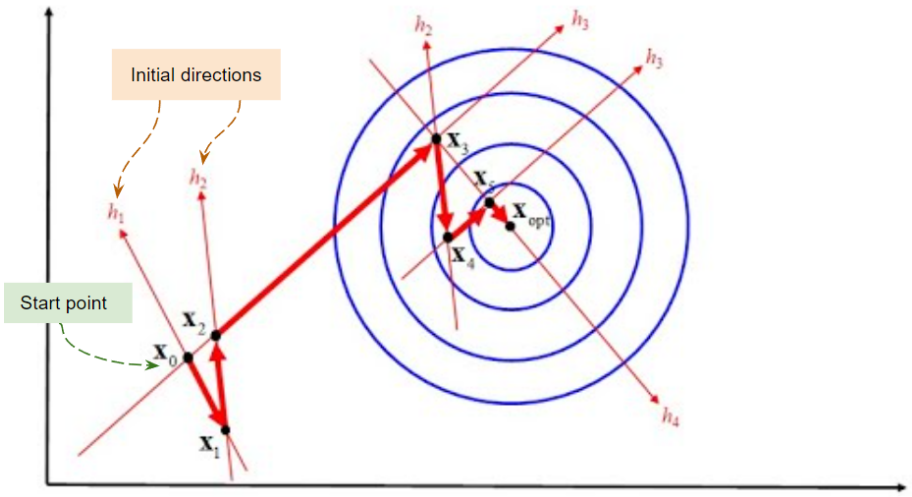


Figure 2.9: Visualization of the Powell algorithm with an example point and the steps taken to reach the optimal point.

Powell's algorithm works in a series of steps (fig. 2.9). First, it picks a few directions to explore, starting randomly. Then, it checks each direction to figure out how far to move in order to get closer to the minimum. To do this, it tries different step sizes and picks the one that reduces the function's value the most. After finding

the best step for each direction, it updates its position by moving along those steps. Next, Powell creates a new direction based on the difference between the current position and the last one, allowing it to explore more and find better paths toward the minimum. This whole process repeats over and over, getting closer to the optimal solution until it converges.

2.2.2 Estimating the Camera Pose

As we need to accurately determine the camera's pose, which involves finding the translation vector and rotation matrix (extrinsic parameters). To do this, We suggest using the Powell optimization method to adjust the translation vector, with the aim of ensuring that the rasterized image of the 3D point cloud aligns perfectly with the ground truth mask. This alignment is important because it allows us to determine the camera's position relative to the 3D scene accurately. The camera's role is to project the 3D point cloud into a 2D plane, and the ground truth mask represents the observed regions of the scene from the camera's perspective. When the rasterized point cloud and the mask align, it confirms that the projection parameters correctly replicate the real-world setup, effectively solving for the camera's spatial position (check fig. 2.10).

Since the 3D models are consistently oriented relative to the input images, there is no need to adjust the rotation matrix, it can simply be set to zero. The translation vector is the critical variable that defines the camera's placement within the 3D space.

We propose a method, described in algorithm 1, to adjust the camera's translation parameters (t_x, t_y, t_z) and align the rasterized image of the 3D point cloud with the ground truth binary mask. The goal is to minimize the misalignment between the bounding box of the building in the rasterized image and its corresponding bounding box in the mask. To achieve this, we use a cost function that measures the difference between these two bounding boxes.

The process starts by centering the building's region in the binary mask along the

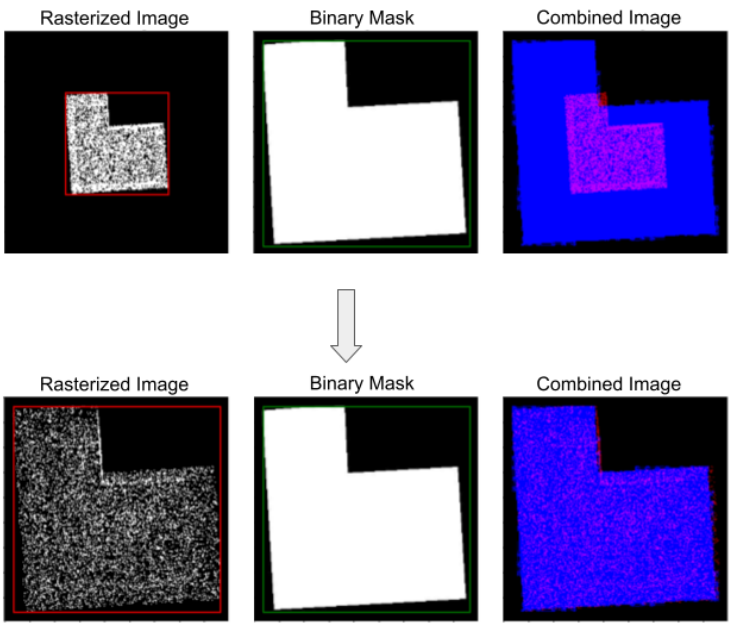


Figure 2.10: Visualization of camera pose estimation using the binary mask as a reference, rasterizing the point cloud until it aligns with the mask, and combining the images to match perfectly on top of each other.

x- and y-axes. This step is crucial for optimizing the depth (z-coordinate), as it ensures that depth adjustments only affect the size of the bounding box, not its position. Without centering, shifts in the x- or y-axes could be mistaken for depth errors. On the other hand, when optimizing the horizontal (x) and vertical (y) translations, centering isn't needed because those parameters directly control the building's position.

We chose Powell's method for this optimization because it's well-suited for problems like this one. The method is designed to handle non-linear and non-differentiable cost functions, such as our bounding box alignment task. Instead of relying on gradients, Powell's method uses line searches to find the best solution, making it a practical and effective choice for this pixel-based optimization problem. The sequential optimization strategy involves:

1. Optimize the z-coordinate (depth).
2. Fix z, then optimize the x-coordinate (horizontal translation).

Algorithm 1 Optimization of Camera Translation Parameters

Input: Point Cloud $\mathcal{P}$, Ground Truth Mask $\mathcal{M}_{gt}$, Initial Camera Translation Parameters $\mathbf{T}_0 = (t_{x0}, t_{y0}, t_{z0})$
Output: Optimized Camera Translation Parameters $\mathbf{T}^* = (t_x^*, t_y^*, t_z^*)$
Initialize camera translation parameters $\mathbf{T} \leftarrow \mathbf{T}_0$
while not converged **do**
    Rasterize point cloud $\mathcal{P}$ using current camera translation parameters $\mathbf{T}$ and fixed rotation parameters $\mathbf{R} = 0$ to generate a rasterized image $\mathcal{M}_r$
    Compute bounding boxes for the ground truth and rasterized images:
        $\text{bbox}_{gt} \leftarrow \text{BoundingBox}(\mathcal{M}_{gt})$
        $\text{bbox}_r \leftarrow \text{BoundingBox}(\mathcal{M}_r)$
    Step 1: Optimize t_z (depth)
    Define the cost function for depth optimization:
        $$\text{cost}_z \leftarrow \sqrt{(x_{\min}^{gt} - x_{\min}^{r})^2 + (x_{\max}^{gt} - x_{\max}^{r})^2 + (y_{\min}^{gt} - y_{\min}^{r})^2 + (y_{\max}^{gt} - y_{\max}^{r})^2}$$
    Optimize t_z using Powell's method to minimize cost_z
    Step 2: Optimize t_x (horizontal translation)
    Define the cost function for horizontal translation optimization:
        $$\text{cost}_x \leftarrow \sqrt{(x_{\min}^{gt} - x_{\min}^{r})^2 + (x_{\max}^{gt} - x_{\max}^{r})^2}$$
    Fix t_z and optimize t_x using Powell's method to minimize cost_x
    Step 3: Optimize t_y (vertical translation)
    Define the cost function for vertical translation optimization:
        $$\text{cost}_y \leftarrow \sqrt{(y_{\min}^{gt} - y_{\min}^{r})^2 + (y_{\max}^{gt} - y_{\max}^{r})^2}$$
    Fix t_z and t_x, and optimize t_y using Powell's method to minimize cost_y
Return optimized camera translation parameters $\mathbf{T}^* = (t_x^*, t_y^*, t_z^*)$

3. Fix both z and x, then optimize the y-coordinate (vertical translation).

This step-by-step approach ensures accurate alignment between the rasterized image of the 3D point cloud and the 2D ground truth mask.

After obtaining the optimized camera translation parameters, we check the Intersection over Union (IoU) between the rasterized image generated with these parameters and the ground truth mask. If the IoU exceeds 93%, we consider it a valid input for further processing. This ensures that only well-aligned configurations are utilized.

2.3 Summary of Our Generated Dataset

In summary, we have created a dataset, as illustrated in fig. 2.11. Each entry includes a single-view RGB image paired with a complete mask, even when there are occlusions. Additionally, the dataset includes the associated Sobel filter image, a

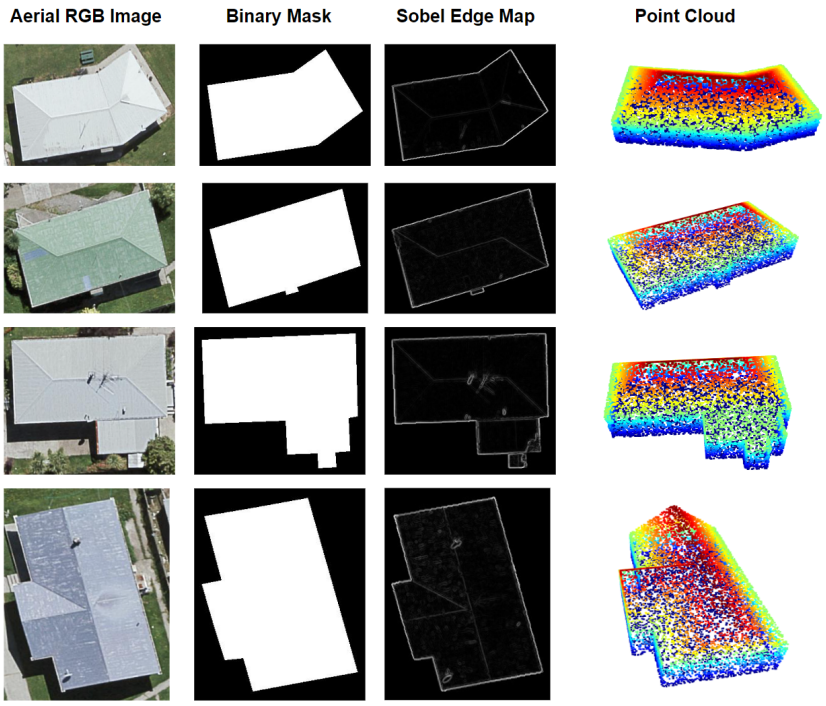


Figure 2.11: Example of the dataset used in our study.

point cloud, and a JSON file containing the camera parameters, such as the x, y, and z coordinates for each image. This dataset will be used as the training data for our pipeline.

2.4 Methodology

Now that the dataset is ready, we can dive into the details of how we built the 3D reconstruction system. To provide better clarity, we will begin with an overview of the overall pipeline and then break it down into its individual components.

2.4.1 Overall Pipeline

Before diving into the details of each part of our method, let us first take a closer look at the overall architecture of the model to set the stage for the discussion ahead. Our approach focuses on reconstructing an entire building from a single digital orthophoto. However, we go a step further by aiming to enhance the reconstruction process to capture finer building details, particularly along the edges, for more pre-

cise and detailed results. In essence, the process begins with an input image that

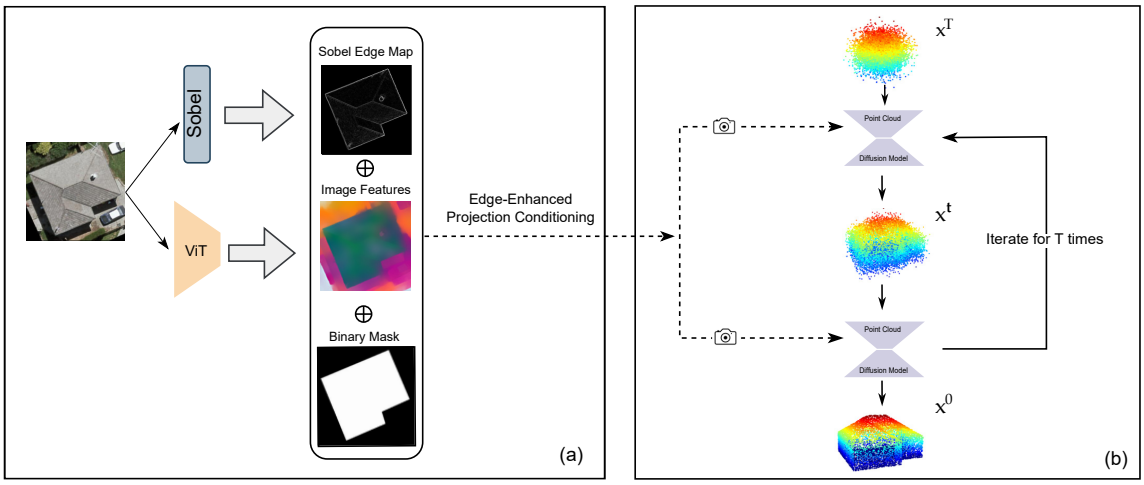


Figure 2.12: The architecture of our method is designed to reconstruct an entire building from a single aerial nadir view image by enhancing edges and capturing finer building details. In Block (a), features are extracted from the input image, including edge details derived from the Vision Transformer (ViT) and Sobel operator, which act as conditioning inputs for the diffusion model. Block (b) represents the reverse diffusion process, where Gaussian noise is progressively refined into a 3D point cloud, guided by the features from Block (a), to reconstruct the building. The connection between the two blocks involves projecting the 2D edge features onto the 3D point cloud, enhancing the reconstruction quality.

is conditioned to enable the reconstruction of the building using a diffusion model. The method is built around three key components:

- **Conditioning Features (block (a) of the figure 2.12)**: This component extracts features from the input image that act as the conditioning elements for the diffusion model. These features are combined and prepared as inputs for the next stage.
- **Generating Process (block (b) of the figure 2.12)**: Representing the reverse process of the diffusion model, this block starts with Gaussian noise and progressively refines a 3D point cloud to reconstruct the building. The refinement is guided by the features generated in block (a).
- **Edge-Enhanced Projection Conditioning**: Serving as the bridge between blocks (a) and (b), this component explains how 2D features can be effectively

projected onto 3D point clouds, which are inherently sparse and lack explicit connectivity between points. This component is referred to as "edge-enhanced" because, in Block (a), we extract edge features that are specifically considered during the projection process to improve the reconstruction quality.

By understanding this overall architecture, we can better appreciate how each part contributes to the goal of detailed and accurate building reconstruction.

2.4.2 Components Details

Conditioning Features

When analyzing an aerial image of a single building, background elements like parked cars or trees can often appear, adding depth and complexity to the image, as shown in fig. 2.13, you can see the unnecessary depth maps generated by these elements, which we aim to remove. To ensure the model focuses exclusively on the building, we use a **binary building mask** as an additional input. This mask is combined with the **image features** extracted using the **Vision Transformer (ViT)** by concatenating the mask values with the ViT features. These combined features are then projected onto the point cloud using the estimated camera pose.

Beyond the binary building mask, roof edges play a vital role in reducing ambiguities during the reconstruction process. By providing clear structural boundaries, roof edges help define the building's silhouette, improving the accuracy and completeness of the reconstruction. This is particularly important for capturing the entire building geometry, where precise roof edge representation significantly impacts the overall quality of the model.

To further enhance edge clarity, we incorporate **Sobel-filtered images** alongside the binary mask and ViT-extracted features fig. 2.14. The Sobel filter detects sharp intensity changes, highlighting key gradients that outline the roof's perimeter. By emphasizing these transitions, this approach makes the model more adept at captur-

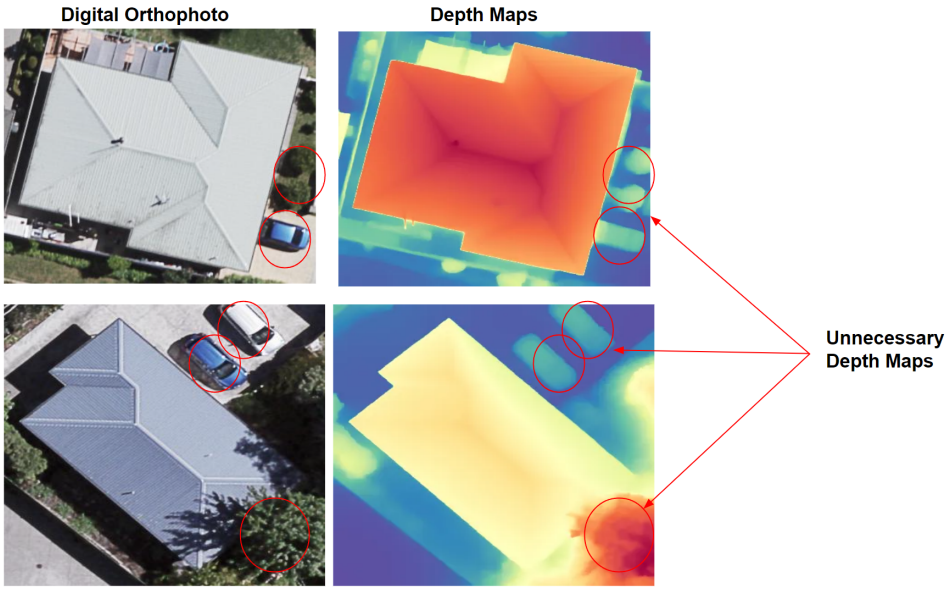


Figure 2.13: Visualizing the unnecessary depth maps that could be learned by our model.

ing the building's structure, minimizing confusion from surrounding elements, and improving the reconstruction's overall accuracy.

Generating Process

In block (b), we present our diffusion model, which operates through the reverse diffusion process. fig. 2.15 illustrates this reverse process, which is central to the model's operation. The process begins with Gaussian noise as the input, generated after the forward diffusion process. In the forward process, we start with the ground truth data and iteratively add noise at each time step until we reach the initial timestamp, resulting in a noisy sample. This noisy sample is denoted as x_T, and it serves as the starting point for the reverse process. The reverse diffusion process aims to iteratively sample x_{t-1} from the reverse distribution, moving from x_T towards the target distribution $q(x_0)$, which represents the clean, denoised data. The model S_θ predicts the offset for each point in the point cloud at time step t. Specifically, it predicts how much each point's noisy position x_t should be adjusted to reach its ideal clean state at time step $t-1$. Thus, at each iteration, the noisy point cloud x_t

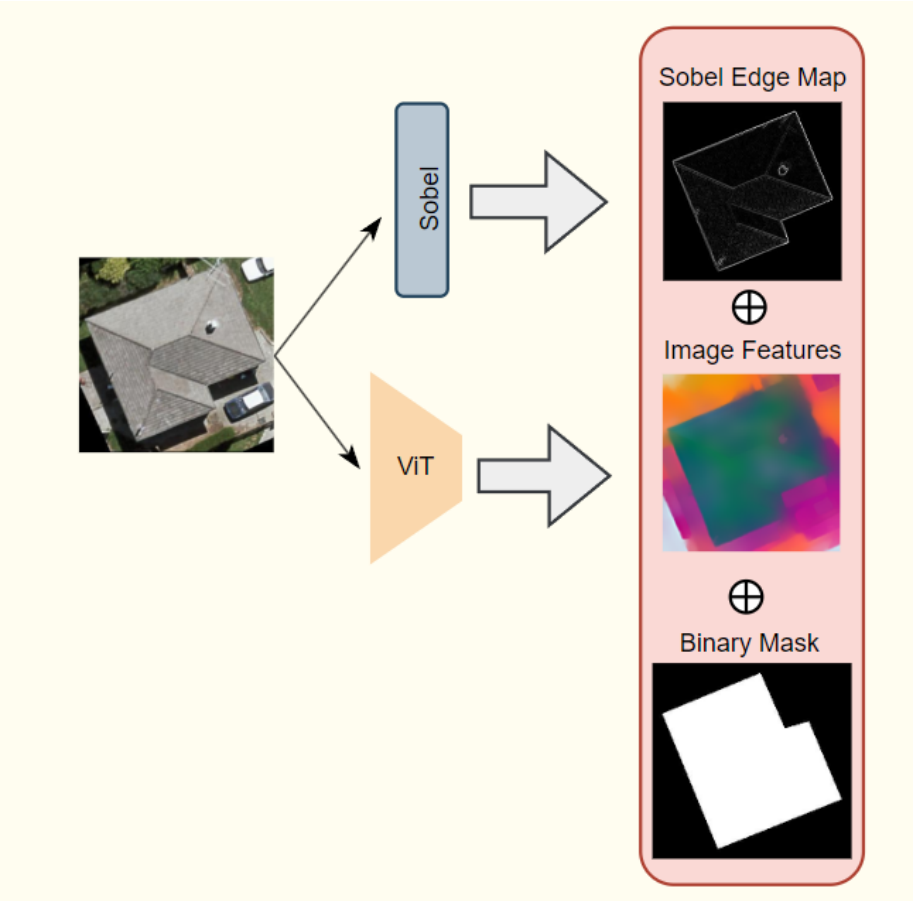


Figure 2.14: Visualization of the first block of our conditioning features.

is updated based on the model's prediction, and we sample x_{t-1}, gradually refining the point cloud until it converges to the target distribution.

During this reverse process, we minimize the L2 loss between the actual noise and the predicted noise at each time step. Mathematically, the L2 loss is expressed as:

$$L_2 = \mathbb{E}_\epsilon \left[\|\epsilon - S_\theta(x_t, t)\|_2^2 \right], \tag{5}$$

where:

- θ: These are the parameters of the model.
- ϵ: This represents the true noise that was added to the image during the forward diffusion process. The model tries to predict this noise during the reverse process.

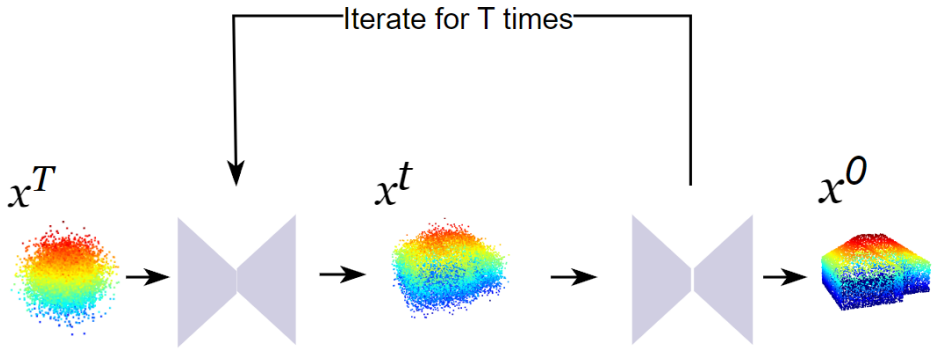


Figure 2.15: Reverse diffusion model visualization of our method.

- x_t: This is the noisy image at time step t.
- t: This is the discrete time step in the diffusion process.
- $\mathbb{E}_\epsilon$: This denotes the expectation over the distribution of the true noise ϵ
- $\|\cdot\|_2^2$: This represents the squared Euclidean norm, which computes the squared difference between the true noise ϵ and the predicted noise $S_\theta(x_t, t)$.

where ϵ represents the actual noise added during the forward process, and $S_\theta(x_t, t)$ is the model's prediction of the noise at time step t. The goal is to reduce the discrepancy between the actual noise ϵ and the predicted noise, allowing the model to effectively denoise the input through the reverse process.

The model used for this process is PVCNN (as described in section 1.5.2). To ensure accurate reconstruction of the point cloud, we apply CDPM, which directs the model to focus on faithfully reconstructing the point cloud's structure rather than simply locating its center. This approach enhances the overall quality of the output.

A critical aspect of our methodology is projection conditioning. During each iteration of the reverse process, we project the image features, enabling the transformation of 2D information into a 3D representation. This projection process will be explained in greater detail in the next component.

Edge-Enhanced Projection Conditioning

Here, we reach the most interesting aspect of our model: projection conditioning. This is where we need to discuss the concept of projection and explore how 2D information can be interpreted in 3D space.

In the context of conditioning PC²[31] proposed a method that involves projecting image features onto partially denoised 3D points during each step of the diffusion process. This is possible because, as discussed in section 1.5.1, the point cloud is represented not only by 3D coordinates (x, y, z) but also by an additional **component c** (x, y, z, **c**), where c represents the image feature. Thus, we extend the traditional 3D point representation to include image features, enabling us to project the relevant image information onto the 3D points during the diffusion process.

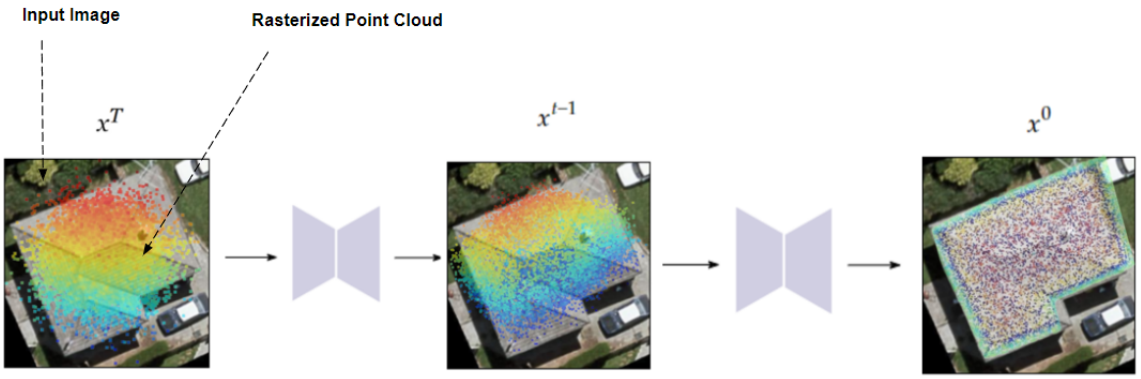


Figure 2.16: Visualization of the projection process, starting with adding features to a noisy point cloud and progressively refining it until it forms a well-organized point cloud with RGB values.

The process is simple: we start with partially denoised 3D points, x_t, and transform them into 2D pixel coordinates based on the camera's perspective, defined by the rotation matrix R and the translation vector T. This transformation projects the 3D points onto the image plane, aligning them with the image's features. Mathematically, this can be expressed as:

$$p_i = \Pi(R \cdot x_{t,i} + T),$$

where p_i represents the 2D pixel coordinates corresponding to the 3D point $x_{t,i}$, and $\Pi(\cdot)$ is the rasterization function that performs the projection.

From the rasterized image, the visible points $V \subseteq \{p_i\}$ are identified. For each visible pixel $p_i \in V$, the image feature $F(I, p_i)$, which represents the feature at pixel p_i in the image I, is assigned to the corresponding 3D point $x_{t,i}$. This process ensures that the features from the visible pixels are accurately associated with the 3D points based on their visibility and mapping through the rasterization (check fig. 2.16). The assignment is defined as:

$$f(x_{t,i}) = F(I, p_i) \quad \text{for} \quad p_i \in V.$$

In our work, this rasterization-based approach is adapted to project image features onto partially denoised 3D points.

Conclusion

In conclusion, We highlighted the key stages of our methodology, focusing on the processes that ensure accurate and effective 3D reconstructions. We began by detailing the estimation of accurate camera poses, a critical component of our pipeline, achieved through optimization techniques using the Powell algorithm. This iterative refinement aligned the camera poses with the scene's geometry, enhancing the system's overall accuracy. We then discussed the extraction of essential features, the projection of 2D features into 3D points to establish a spatial representation, and the use of diffusion models to generate the desired point cloud, ensuring precise and realistic 3D reconstructions. In the following chapter, we will present a series of experiments designed to evaluate the effectiveness and accuracy of the methodology outlined in this chapter.

Chapter 3

Experimental Procedure and Observations

Introduction

In this chapter, we will provide a comprehensive overview of our experimental setup, detailing the steps involved in developing our pipeline. We will also present both qualitative and quantitative results to assess the performance of our model. The evaluation will be carried out using our custom dataset, as well as a separate test dataset, which we will define and describe. Through these results, we aim to demonstrate the effectiveness of our approach in various contexts and highlight its strengths and limitations.

3.1 Implementation Details

To implement the previous chapter, a robust overall environment is essential, and we must define the technical details of the training implementation.

3.1.1 Hardware and Software Environment

Hardware

To carry out this project, we had access to several powerful GPU configurations available locally. For model training, we used a Titan RTX GPU (a graphics pro-

cessing unit designed for high-performance computing) with 24GB of VRAM (Video Random Access Memory, or in other words, GPU memory), running the training process for 24 hours. For inference, we utilized a GeForce RTX 2080 Ti GPU, which has 12GB of VRAM.

Software

- **Programming Language:** Python was chosen as the primary programming language for this project due to its widespread use and reputation as the most accessible language for deep learning. Additionally, most deep learning frameworks are built on Python, making it an ideal choice for this work.
- **Frameworks :** For this project, we relied on two powerful frameworks:
  - PyTorch [48]: This open-source deep learning framework, developed by Facebook AI Research, was the foundation of our work. It's versatile, easy to use, and perfect for building and training machine learning models. The supportive community and resources around PyTorch made troubleshooting and experimenting much smoother.
  - PyTorch3D [49]: Built on top of PyTorch, this library is designed for 3D deep learning tasks. I mainly used it for point rasterization, which was a crucial step in processing and visualizing our 3D data. It simplified a lot of the complexities that come with handling 3D information.

3.1.2 Training Parameters

We trained the diffusion model using a batch size of 8 for a total of 100,000 steps, utilizing 80% of the data for training and 20% for validation. All experiments were conducted on a single GPU to maintain consistent performance.

For dataset preparation, we first optimized the camera parameters. The input images were resized to 224×224 before applying algorithm 1. To detect edges, we used the Sobel operator, a simple yet effective method that emphasizes key structural details

of buildings. This operator applies two 3×3 convolution kernels to estimate gradients in the x- and y-directions, producing edge maps that are crucial for accurate 3D reconstruction.

The point cloud data for each model consisted of 100,000 points, which provided a high level of detail in the reconstructed structures. To ensure uniformity across the dataset, all point clouds were normalized to fit within a unit sphere. This normalization step allowed algorithm 1 to be applied consistently across all buildings without requiring manual adjustment of the initial camera translation parameters (t_{x0}, t_{y0}, and t_{z0}) for each building. However, this approach did result in the loss of the buildings' actual height values.

For Inference: After normalization, the optimized camera parameters (t_x^*, t_y^*, and t_z^*) generally fell within the range of 0 to 1. To simplify inference, we used default camera parameters set to $t_x = 0$, $t_y = 0$, and $t_z = 1$, which simulated a top-down view along the Z-axis. This strategy eliminated the need to specify precise camera parameters for individual buildings, making the reconstruction process more straightforward and scalable.

3.1.3 Optimizer

Optimization was handled by the **AdamW optimizer [50]**, a technique used to improve the training of machine learning models by preventing them from becoming too complex or overfitting to the training data. One way it does this is by **penalizing large weights** through a method called **weight decay**. Weight decay essentially adds a penalty for large weights, helping the model avoid putting too much importance on any single parameter.

What sets AdamW apart from the standard Adam optimizer is how it handles this weight decay. In traditional Adam, weight decay is mixed in with the gradient update, which can sometimes cause issues with the learning process. AdamW, however, **separates** the weight decay from the gradient update. This allows the model to learn

more effectively and makes the training process more stable and efficient. As a result, AdamW often helps the model perform better when tested on new, unseen data.

The update rule for AdamW looks like this:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \right) - \eta \lambda \theta_t \tag{3.1}$$

Where:

- θ_t represents the model's weights (the parameters it learns),
- η is the learning rate (how much we adjust the weights),
- $\hat{m}_t$ and $\hat{v}_t$ are estimates of the gradient and its squared value,
- ϵ is a small constant to avoid errors during calculations,
- λ is the weight decay factor that controls how strongly the penalty is applied to the weights.

3.1.4 Learning Rate

At the start of training, the model's weights are typically random, which can cause the optimizer to make erratic or inconsistent updates. To mitigate this, we apply a learning rate warmup with a cosine scheduler. This means that instead of starting with a large learning rate, we begin with a small value and gradually increase it over time. This gradual increase allows the optimizer to make smaller, more controlled updates during the early stages of training, helping to avoid instability. After approximately 2000 steps, the learning rate starts to decrease, signaling the transition to smaller, more precise updates to help the model converge effectively and stabilize its training as shown in fig. 3.1.

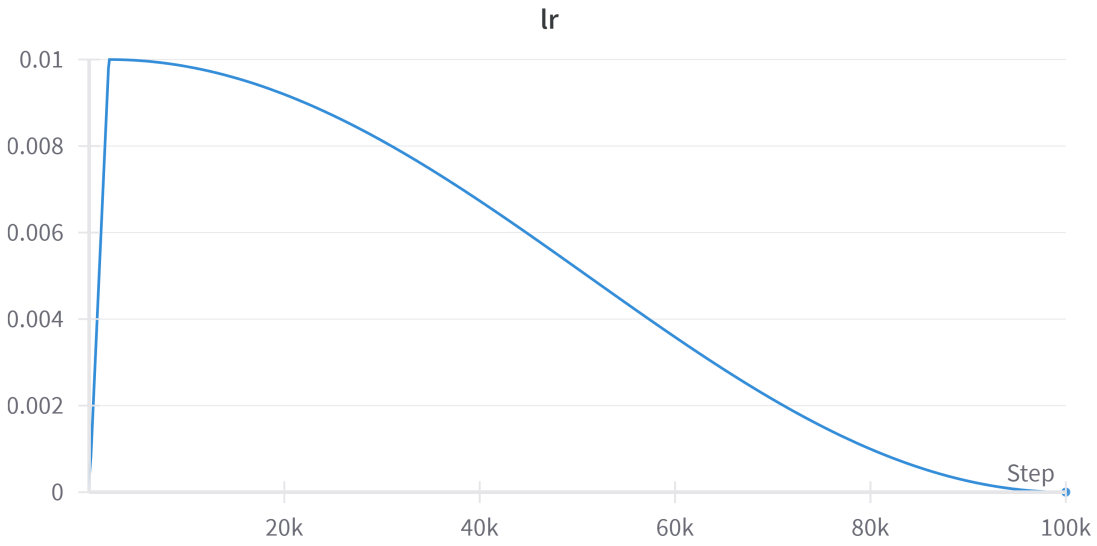


Figure 3.1: Visualization of the learning rate with a cosine schedule and warmup, up to 2000 steps.

3.1.5 Loss Function

For the loss function, we used the standard approach for diffusion models, which is the **Mean Squared Error (MSE)**. This calculates the difference between the actual noise added to the data and the noise predicted by the model, then measures the average of the squared differences. The MSE loss function is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (\hat{\epsilon}_i - \epsilon_i)^2 \tag{3.2}$$

where:

- $\hat{\epsilon}_i$ is the predicted noise for the i-th sample.
- ϵ_i is the actual noise added to the data for the i-th sample.
- N is the total number of samples.

3.2 Quantitative Results

To assess whether our method is performing well, we need quantitative or numerical results that form the basis of our conclusions. These results will be evaluated using specific metrics, which we will define and present along with their exact values.

3.2.1 Our Evaluation Metrics

We evaluate point cloud reconstructions using Chamfer Distance and F-Score@0.001.

Chamfer Distance

Chamfer Distance (CD) measures how similar two point clouds are by calculating the average distance between their points. It works by finding the closest ground truth point for each point in the predicted point cloud, and vice versa, then averaging these minimum distances. This provides an overall score of how well the predicted point cloud aligns with the ground truth point cloud.

The Chamfer Distance is defined as:

$$\mathrm{CD}(P_1, P_2) = \frac{1}{|P_1|} \sum_{p_1 \in P_1} \min_{p_2 \in P_2} \|p_1 - p_2\|_2^2 + \frac{1}{|P_2|} \sum_{p_2 \in P_2} \min_{p_1 \in P_1} \|p_2 - p_1\|_2^2 \tag{3.3}$$

where:

- P_1 is the set of points in the predicted point cloud.
- P_2 is the set of points in the ground truth point cloud.
- p_1 and p_2 are individual points in the respective point clouds.
- $\|p_1 - p_2\|_2^2$ is the squared Euclidean distance between two points p_1 and p_2.

However, CD's sensitivity to outliers (as demonstrated in fig. 3.2) can skew the average distance, leading to potentially misleading assessments of reconstruction quality. Therefore, while CD is a valuable metric for evaluating point cloud accuracy, it is beneficial to use additional metrics for a more comprehensive analysis.

F-Score

In addition to Chamfer Distance, we also use the F-Score (proposed by [51]) to evaluate the quality of the point cloud prediction (as it's more robust to outliers fig. 3.2). The F-Score measures how well the predicted points match the ground truth points, but it only considers a point correctly predicted if it lies within a certain distance threshold from its nearest match. For our point clouds, which contain 10,000 points normalized between 0 and 1, we set this threshold to 0.001. This means that a point is considered correctly predicted only if it is within 0.1% of the distance to its closest ground truth point. This small threshold allows us to detect even subtle differences, providing a more precise evaluation of the model's reconstruction quality.

The F-Score is calculated using two key measures: Precision and Recall. Precision is the proportion of predicted points that are close enough to ground truth points, and Recall is the proportion of ground truth points that are close enough to predicted points.

The formula for the F-Score is as follows:

$$\text{Precision} = \frac{\sum_{d_1 < \text{thr}} 1}{\text{len}(d_1)} \tag{3.4}$$

$$\text{Recall} = \frac{\sum_{d_2 < \text{thr}} 1}{\text{len}(d_2)} \tag{3.5}$$

$$\text{F-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall} + \epsilon} \tag{3.6}$$

Where:

- d_1 represents the distances between the predicted points and their closest ground truth points,

- d_2 represents the distances between the ground truth points and their closest predicted points,
- thr is the threshold value (0.001),
- ϵ is a small value added to avoid division by zero.

⚠ **Important:** F-Score@0.001 (or 0.01) refers to the F-score metric calculated using a threshold of 0.001 (or 0.01).

This method provides a balanced evaluation of how well the predicted and ground truth point clouds align, considering both how many predicted points are correctly located (precision) and how many ground truth points are successfully matched (recall).

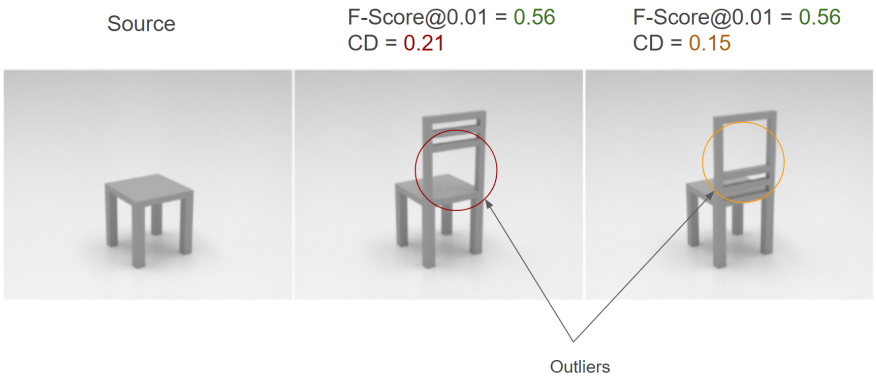


Figure 3.2: Visualization of how Chamfer distance is sensitive to outliers in a 3D object. The outliers are highlighted, and when comparing metrics across different 3D models, the Chamfer distance value changes, while the F-score remains consistent despite the presence of various outliers [51].

3.2.2 Comparison

In table 3.1, we highlight the performance of our method compared to baseline approaches, focusing on achieving the best average across key metrics. Our method delivers notable improvements over the baseline models. Specifically, it achieves a 21.87% higher F-Score than the PC² method and a 4.83% improvement over the CCD-3DR model. For Chamfer Distance (CD), our approach outperforms PC² by

10.44% and CCD-3DR by 3.40%. These results underscore the effectiveness of our method in generating point cloud reconstructions that are both more accurate and reliable.

Table 3.1: Quantitative results for different methods are presented under Chamfer Distance (CD, reported as $\times 10^{-3}$) and F-Score@0.001, demonstrating that our method outperforms the others.

Method	**F-Score@0.001** ↑	**CD** ($\times 10^{-3}$) ↓
PC²	0.535	3.172
CCD-3DR	0.622	2.941
Ours	**0.652**	**2.841**

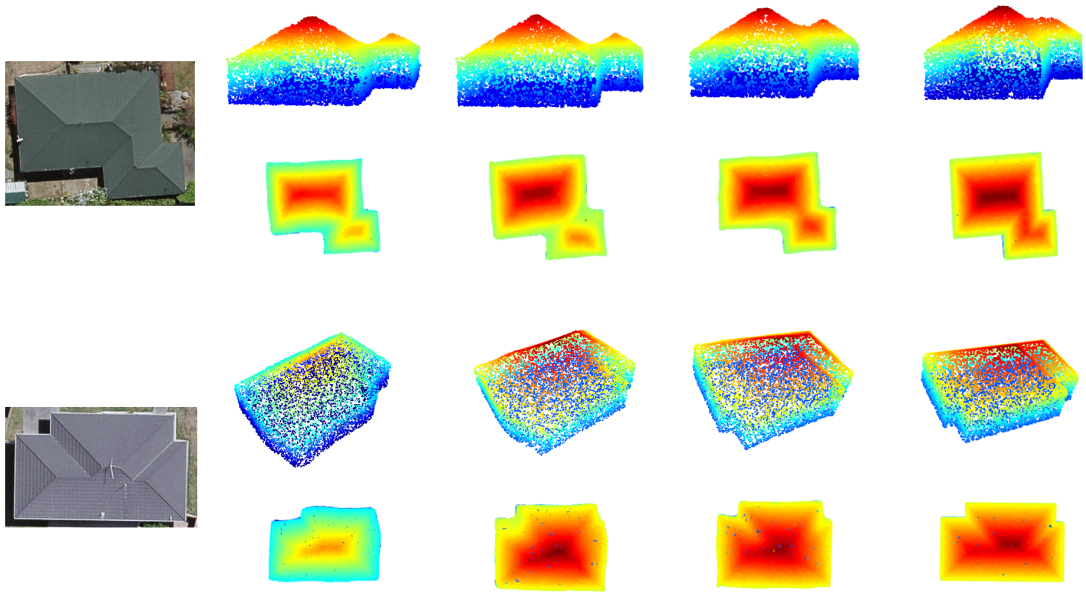
Note: Due to the probabilistic nature of the models, each prediction can yield different results. To address this, we generated seven separate predictions for each model and selected the best outcome from them for evaluation.

3.3 Qualitative Results

The best way to assess the performance of our edge-aware reconstruction is by visually comparing the results with those from other methods.

3.3.1 Comparison

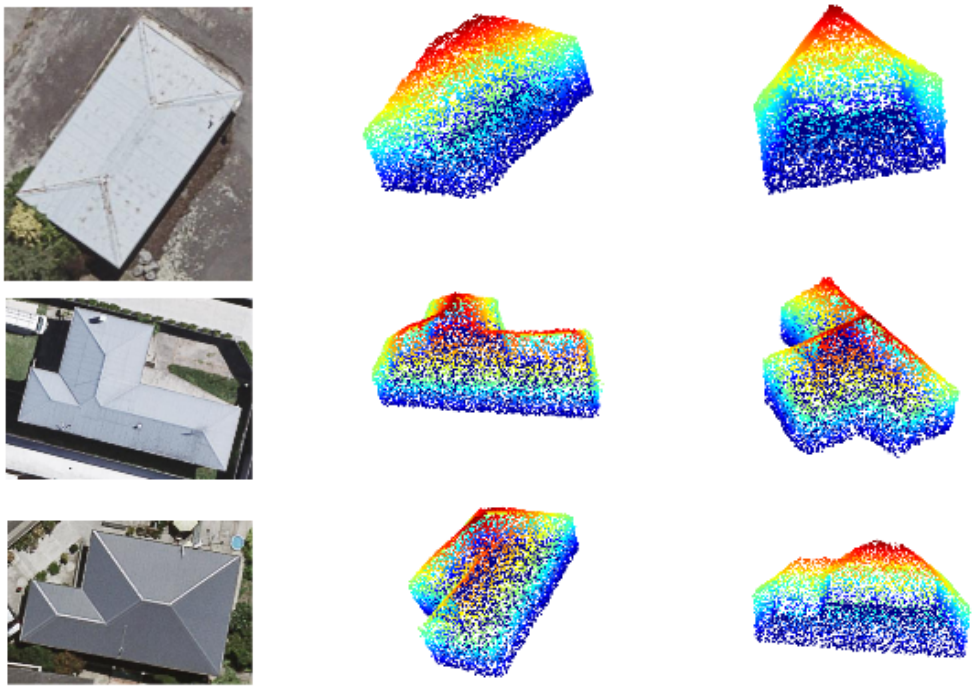
Figure 3.3 shows a visual comparison of the PC² [31], CCD-3DR [32], and our model, with point clouds displayed from both a top-down view and an additional angle to give a fuller perspective. Figure 3.4 showcases the visual outcomes of our approach, demonstrating its ability to reconstruct 3D models with intricate structural details. Unlike conventional methods that focus on a single viewpoint, our approach generates fully realized buildings from multiple angles, revealing well-reconstructed walls and roofs. This results in a more thorough and lifelike representation of the structure. The color mapping, which represents height variations along the Z-axis, is especially important as it helps us visualize and compare the predicted heights against the



(a) Input Image (b) PC² (c) CCD-3DR (d) Ours (e) GT

Figure 3.3: A qualitative comparison from both a top-down and an alternative viewpoint, showcasing our model alongside baseline models. The comparison begins with (3.3a), which shows the single aerial input image, followed by (3.3b) illustrating the reconstruction using PC² , and (3.3c) showing the reconstruction with CCD-3DR. Next, (3.3d) highlights the reconstruction generated by our method, and (3.3e) shows the ground truth. This comparison clearly demonstrates the strengths of our approach in accurately capturing the roof's shape. From the top view, it's evident that our method produces a point cloud with a color distribution that closely matches the ground truth. On the other hand, PC² tends to underestimate the roof height and misses several sections, while CCD-3DR provides incomplete roof reconstructions with important details missing. These shortcomings become even more noticeable when viewed from different perspectives, highlighting the structural and edge issues that our method successfully addresses.

actual ground truth. This color coding makes it easier to assess how each model performs. When we compare our approach with the others, it's clear that our method captures more intricate details, especially around roof edges, which have a big impact on the overall structure of the building. Additionally, our method provides more precise elevation values, giving us a better and more accurate representation of complex surfaces. Even though the point clouds are normalized, the visual differences in



(a) Single Input Image (b) View 1 (c) View 2

Figure 3.4: The qualitative results of two newly reconstructed views (3.4b) and (3.4c), generated by from a single input aerial image (3.4a), showcase the strength of our approach. These results highlight our solution's ability to accurately reconstruct the full point cloud of the entire building.

elevation representation between the models become much clearer, making it easier to see the advantages of our approach.

3.4 Generalization

We evaluated our method using a dataset from Tallinn City, Estonia [52]. The dataset consists of digital orthophotos with a 20 cm Ground Sampling Distance (GSD) and 3D building meshes from the Building3D dataset [15]. The Building3D dataset provides detailed 3D models for each building. The orthophotos in this dataset are of lower spatial resolution compared to our training data, which presents additional challenges, such as higher levels of occlusion and more intense shadows that can obscure parts of the buildings. To address these challenges, we processed the digital

orthophotos through our pipeline using the default top-view camera parameters for inference. The results, comparing our method with CCD-3DR, are presented in Table 3.2. The variation in metrics when comparing the test dataset arises from the

Table 3.2: Quantitative results for CCD-3DR and our method, showcasing how our method outperforms the baseline model on Tallinn city dataset different from the training data .

Method	**F-Score@0.001** ↑	**CD** ($\times 10^{-3}$) ↓
CCD-3DR	0.324	11.669
Ours	**0.334**	**10.164**

fact that the Building3D test dataset includes more detailed roof structures and a variety of roof types that our model has not been trained on previously.

Qualitatively speaking, our goal was to evaluate the generalization of our method by comparing it to the ground truth. After observing the results, particularly in

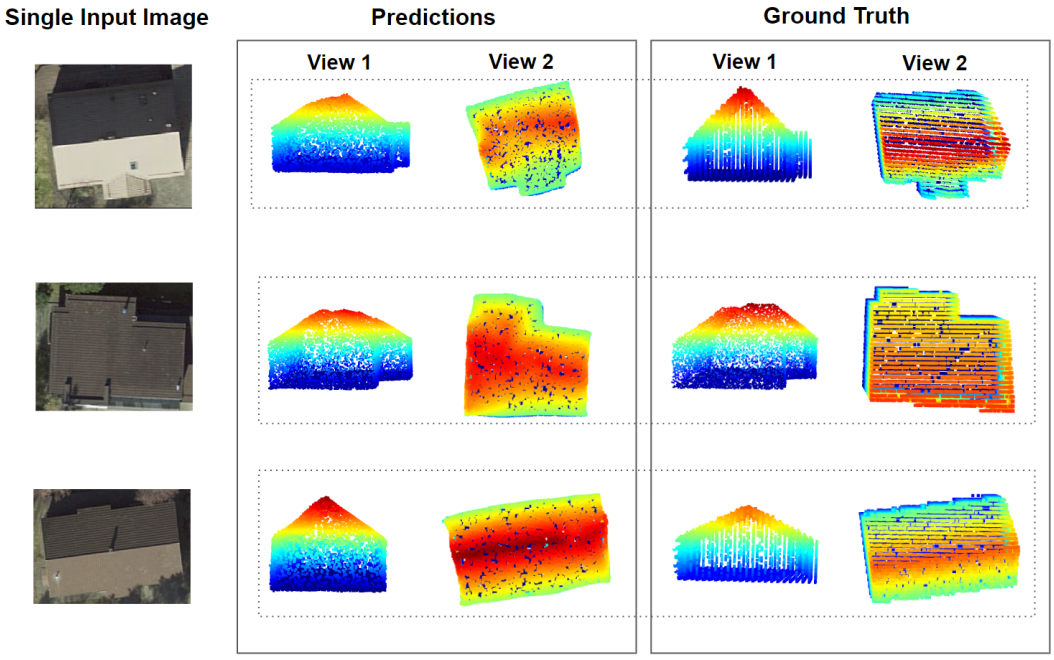


Figure 3.5: Showcasing the difference between our method's prediction and the ground truth from two different views: top view and side view, for the same input image from Tallinn City, Estonia.

fig. 3.5, we can confidently affirm that our method demonstrates strong generalization capabilities, effectively handling new and unseen data from the Tallin City dataset.

Conclusion

To conclude, we presented our hardware and software setup, detailing the specific concepts and techniques used. We also introduced the metrics employed and highlighted key differences in these metrics. Using these metrics, we conducted a quantitative comparison with our baseline models, demonstrating that our model outperforms them. Additionally, we showcased qualitative results to further validate our model's superior performance. Finally, we conducted a generalizability study to assess whether our method could be applied effectively to a different dataset.

General Conclusion

In our research journey, we achieved a groundbreaking milestone: reconstructing complete 3D building models from a single-view digital orthophoto, an accomplishment that, to our knowledge, has not been previously realized. This innovative work has been submitted to the esteemed Geospatial Week Conference 2025 in Dubai and is currently under review. Our efforts began with an in-depth examination of the broader research landscape, identifying the complexities of the problem and the key challenges that propelled this project forward. We explored the academic framework surrounding our internship, analyzed the contributions of our host organizations, and reviewed existing studies to understand previous attempts at solving similar challenges. Two foundational papers, PC^2 and CCD-3DR , served as a basis for our approach, upon which we introduced critical innovations to advance the field. A significant initial step was creating essential datasets, including estimating camera parameters for aerial images using optimization algorithms. Building on this foundation, we developed a comprehensive methodology designed to address the limitations of existing techniques. Our method emphasized refining 3D point cloud generation with advanced feature extraction, integrating edge maps and additional characteristics to improve accuracy and detail capture. By seamlessly aligning these features with the generated point clouds, we achieved a precise reconstruction process that outperformed baseline models. Extensive experiments validated the effectiveness of our approach, showing significant improvements both quantitatively and qualitatively. As we reflect on this journey, we recognize the challenges overcome, the insights gained, and the meaningful contributions made, all of which set the stage for future advancements in 3D reconstruction technologies.

Despite the progress made, significant advancements in 3D building reconstruction are still on the horizon. Our research has laid the groundwork for these developments, identifying key areas for further refinement. This includes the reintegration of height information lost during point cloud normalization, which will enhance model accuracy. We are also exploring methods for camera-free reconstruction, enabling the generation of 3D models without relying on specific camera parameters. Another crucial development is the ability to reconstruct multiple buildings from a single image, moving beyond the current single-building approach. The addition of realistic textures for roofs, walls, and ground surfaces aims to create more detailed and lifelike models. Finally, by capturing finer architectural details like doors and windows, we seek to make point clouds richer and more precise. Collectively, these advancements promise to improve the accuracy, flexibility, and real-world applicability of 3D reconstruction techniques.

References

[1] Filip Biljecki et al. 'Applications of 3D City Models: State of the Art Review'. In: *ISPRS International Journal of Geo-Information* 4.4 (2015), pp. 2842–2889. ISSN: 2220-9964. DOI: 10.3390/ijgi4042842. URL: https://www.mdpi.com/2220-9964/4/4/2842 (cit. on p. 1).

[2] T.L. Haithcoat, W. Song and J.D. Hipple. 'Building footprint extraction and 3-D reconstruction from LIDAR data'. In: *IEEE/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas (Cat. No.01EX482)*. 2001, pp. 74–78. DOI: 10.1109/DFUA.2001.985730 (cit. on p. 1).

[3] Liuyun Duan and Florent Lafarge. 'Towards Large-Scale City Reconstruction from Satellites'. In: vol. 9909. Oct. 2016, pp. 89–104. ISBN: 978-3-319-46453-4. DOI: 10.1007/978-3-319-46454-1_6 (cit. on p. 1).

[4] Yiwei Jin, Diqiong Jiang and Ming Cai. '3d reconstruction using deep learning: a survey'. In: *Communications in Information and Systems* 20.4 (2020), pp. 389–413 (cit. on p. 1).

[5] Mehmet Buyukdemircioglu, Sultan Kocaman and Martin Kada. 'Deep learning for 3D building reconstruction: A review'. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2022), pp. 359–366 (cit. on p. 1).

[6] Weijia Li et al. '3D Building Reconstruction from Monocular Remote Sensing Images'. In: Oct. 2021, pp. 12528–12537. DOI: 10.1109/ICCV48922.2021.01232 (cit. on p. 1).

[7] Stephan Reichelt et al. 'Depth cues in human visual perception and their realization in 3D displays'. In: vol. 7690. Apr. 2010. DOI: 10.1117/12.850094 (cit. on p. 1).

[8] DLR. *DLR Introduction*. URL: https://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-5277/8858_read-15912/?fbclid=IwAR07o6bK0oydVGthaex-_MNYLu33kM8EchVwfvmPhvV-eVsdscS6zCbq79k (cit. on p. 3).

[9] CRNS. *CRNS Introduction*. URL: https://crns.rnrt.tn/ (cit. on p. 3).

[10] German Academic Exchange Service (DAAD). *German Academic Exchange Service (DAAD)*. 2024. URL: https://www.daad.de/en/ (cit. on p. 3).

[11] Marek Kulawiak. 'A Cost-Effective Method for Reconstructing City-Building 3D Models from Sparse Lidar Point Clouds'. In: *Remote Sensing* 14.5 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14051278. URL: https://www.mdpi.com/2072-4292/14/5/1278 (cit. on p. 4).

[12] Zhiwei Hu et al. 'Delay performance comparison across seven Low-Earth-Orbit (LEO) satellite constellations'. In: *Ninth Symposium on Novel Photoelectronic Detection Technology and Applications*. Ed. by Junhao Chu, Wenqing Liu and Hongxing Xu. Vol. 12617. International Society for Optics and Photonics. SPIE, 2023, 126171T. DOI: 10.1117/12.2664563. URL: https://doi.org/10.1117/12.2664563 (cit. on p. 4).

[13] Bing Zhang et al. 'Progress and Challenges in Intelligent Remote Sensing Satellite Systems'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), pp. 1814–1822. DOI: 10.1109/JSTARS.2022.3148139 (cit. on p. 4).

[14] Yoones Rezaei and Stephen Lee. *sat2pc: Estimating Point Cloud of Building Roofs from 2D Satellite Images*. 2022. arXiv: 2205.12464 [cs.CV]. URL: https://arxiv.org/abs/2205.12464 (cit. on pp. 5, 7).

[15] Hongxin Yang Ruisheng Wang Shangfeng Huang. 'Building3D: An Urban-Scale Dataset and Benchmarks for Learning Roof Structures from Point Clouds'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023 (cit. on pp. 5, 50).

[16] Berthold Horn. 'Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View'. In: (Oct. 2004) (cit. on p. 6).

[17] David Knill. 'Discrimination of planar surface slant from texture: Human and ideal observers compared'. In: *Vision research* 38 (July 1998), pp. 1683–711. DOI: 10.1016/S0042-6989(97)00325-8 (cit. on p. 6).

[18] Eno Töppe et al. 'Silhouette-Based Variational Methods for Single View Reconstruction'. In: *Video Processing and Computational Video: International Seminar, Dagstuhl Castle, Germany, October 10-15, 2010. Revised Papers*. Ed. by Daniel Cremers et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 104–123. ISBN: 978-3-642-24870-2. DOI: 10.1007/978-3-642-24870-2_5. URL: https://doi.org/10.1007/978-3-642-24870-2_5 (cit. on p. 6).

[19] Sander Oude Elberink and George Vosselman. 'Building Reconstruction by Target Based Graph Matching on Incomplete Laser Data: Analysis and Limitations'. In: *Sensors* 9.8 (2009), pp. 6101–6118. ISSN: 1424-8220. DOI: 10.3390/s90806101. URL: https://www.mdpi.com/1424-8220/9/8/6101 (cit. on p. 6).

[20] Vivek Verma, Rakesh Kumar and Stephen Hsu. '3D Building Detection and Modeling from Aerial LIDAR Data'. In: vol. 2. Feb. 2006, pp. 2213–2220. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.12 (cit. on p. 6).

[21] Ali Ozgun Ok, Caglar Senaras and Baris Yuksel. 'Automated Detection of Arbitrarily Shaped Buildings in Complex Environments From Monocular VHR Optical Satellite Imagery'. In: *IEEE Transactions on Geoscience and Remote Sensing* 51.3 (2013), pp. 1701–1717. DOI: 10.1109/TGRS.2012.2207123 (cit. on p. 6).

[22] Adrian Johnston et al. 'Scaling CNNs for High Resolution Volumetric Reconstruction from a Single Image'. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 930–939. DOI: 10.1109/ICCVW.2017.114 (cit. on p. 7).

[23] Christopher B. Choy et al. *3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction*. 2016. arXiv: 1604.00449 [cs.CV]. URL: https://arxiv.org/abs/1604.00449 (cit. on p. 7).

[24] K O'Shea. 'An introduction to convolutional neural networks'. In: *arXiv preprint arXiv:1511.08458* (2015) (cit. on p. 7).

[25] Anh-Duc Nguyen et al. *GraphX-Convolution for Point Cloud Deformation in 2D-to-3D Conversion*. 2019. arXiv: 1911.06600 [cs.CV]. URL: https://arxiv.org/abs/1911.06600 (cit. on p. 7).

[26] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114 (cit. on p. 8).

[27] Ian Goodfellow et al. 'Generative Adversarial Nets'. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (cit. on p. 8).

[28] Jonathan Ho, Ajay Jain and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239 (cit. on pp. 8, 13).

[29] Jiajun Wu et al. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*. 2017. arXiv: 1610.07584 [cs.CV]. URL: https://arxiv.org/abs/1610.07584 (cit. on p. 8).

[30] Yao Wei, George Vosselman and Michael Ying Yang. *BuilDiff: 3D Building Shape Generation using Single-Image Conditional Point Cloud Diffusion Models*. 2023. arXiv: 2309.00158 [cs.CV]. URL: https://arxiv.org/abs/2309.00158 (cit. on p. 8).

[31] Luke Melas-Kyriazi, Christian Rupprecht and Andrea Vedaldi. *PC^2: Projection-Conditioned Point Cloud Diffusion for Single-Image 3D Reconstruction*. 2023. arXiv: 2302.10668 [cs.CV]. URL: https://arxiv.org/abs/2302.10668 (cit. on pp. 9, 38, 48).

[32] Yan Di et al. *CCD-3DR: Consistent Conditioning in Diffusion for Single-Image 3D Reconstruction*. 2023. arXiv: 2308.07837 [cs.CV]. URL: https://arxiv.org/abs/2308.07837 (cit. on pp. 9, 13, 16, 48).

[33] Wikipedia contributors. *Orthophoto*. https://en.wikipedia.org/wiki/Orthophoto. 2024 (cit. on p. 10).

[34] Surveying Group. *What is an Orthophoto?* 2024. URL: https://www.surveyinggroup.com/what-is-orthophoto/ (cit. on p. 10).

[35] Zhizhong Kang et al. 'Voxel-Based Extraction and Classification of 3-D Pole-Like Objects From Mobile LiDAR Point Cloud Data'. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11 (Nov. 2018), pp. 4287–4298. DOI: 10.1109/JSTARS.2018.2869801 (cit. on p. 12).

[36] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597 (cit. on p. 12).

[37] Zhijian Liu et al. *Point-Voxel CNN for Efficient 3D Deep Learning*. 2019. arXiv: 1907.03739 [cs.CV]. URL: https://arxiv.org/abs/1907.03739 (cit. on pp. 17, 18).

[38] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929 (cit. on pp. 18, 19).

[39] Jing Ren et al. 'Intuitive and efficient roof modeling for reconstruction and synthesis'. In: *arXiv preprint arXiv:2109.07683* (2021) (cit. on pp. 20, 21).

[40] Paolo Cignoni et al. 'MeshLab: an Open-Source Mesh Processing Tool.' In: vol. 1. Jan. 2008, pp. 129–136. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136 (cit. on p. 21).

[41] Robert L. Cook. 'Stochastic sampling in computer graphics'. In: *ACM Trans. Graph.* 5.1 (Jan. 1986), pp. 51–72. ISSN: 0730-0301. DOI: 10.1145/7529.8927. URL: https://doi.org/10.1145/7529.8927 (cit. on p. 22).

[42] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643 (cit. on p. 23).

[43] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV]. URL: https://arxiv.org/abs/1703.06870 (cit. on p. 23).

[44] Irwin Sobel and Gary Feldman. *An Isotropic 3x3 Image Gradient Operator*. Aug. 2015. DOI: 10.13140/RG.2.1.1912.4965 (cit. on pp. 25, 26).

[45] John Canny. 'A Computational Approach To Edge Detection'. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (Dec. 1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851 (cit. on p. 26).

[46] D. Marr and E. Hildreth. 'Theory of Edge Detection'. In: *Proceedings of the Royal Society of London. Series B, Biological Sciences* 207.1167 (1980), pp. 187–217 (cit. on p. 27).

[47] M. J. D. Powell. 'An efficient method for finding the minimum of a function of several variables without calculating derivatives'. In: *The Computer Journal* 7.2 (Jan. 1964), pp. 155–162. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.155. eprint: https://academic.oup.com/comjnl/article-pdf/7/2/155/959784/070155.pdf. URL: https://doi.org/10.1093/comjnl/7.2.155 (cit. on p. 28).

[48] PyTorch Team. *PyTorch*. https://pytorch.org/ (cit. on p. 41).

[49] PyTorch3D Team. *PyTorch3D*. https://pytorch3d.org/ (cit. on p. 41).

[50] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101 (cit. on p. 42).

[51] Maxim Tatarchenko et al. *What Do Single-view 3D Reconstruction Networks Learn?* 2019. arXiv: 1905.03678 [cs.CV]. URL: https://arxiv.org/abs/1905.03678 (cit. on pp. 46, 47).

[52] Maaamet. *Orthophotos*. https://geoportaal.maaamet.ee/eng/spatial-data/orthophotos-p309.html (cit. on p. 50).