Reconstructing Tree Skeletons in Agricultural Robotics: A Comparative Study of Single-View and Volumetric Methods

Xinyu Wang
CMU-RI-TR-25-06
Mar. 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Oliver Kroemer, *chair*
George Kantor, *chair*
Shubham Tulsiani
John Kim

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Accurate 3D reconstruction is essential for robotic applications that require interaction with complex structures. Traditional methods relying on sensor-based point cloud data often struggle with noise and occlusion, leading to incomplete or imprecise reconstructions. This thesis explores and compares two approaches to 3D reconstruction: Image to 3D reconstruction, which predicts a full 3D model from a single image, and 3D to 3D reconstruction, which reconstructs a complete object from a partial 3D input.

Image to 3D reconstruction is inherently challenging due to the difficulty of inferring missing depth information from a single viewpoint. Our experiments demonstrate that an encoder-decoder architecture trained on 3D skeleton data can reconstruct meaningful structures, but the accuracy is highly sensitive to thresholding. At lower confidence thresholds, the model captures more details but introduces false positives, while higher thresholds suppress errors at the cost of missing finer structures.

3D to 3D reconstruction, by contrast, benefits from richer spatial information provided by an incomplete 3D input. The results show that this method achieves consistently higher F1-scores and IoU across all thresholds, making it significantly more robust to occlusions. Architectural choices also play a crucial role; models with skip connections outperform deeper or more complex variations, highlighting the importance of feature retention in volumetric reconstruction. Additionally, our study of loss functions confirms that Weighted Binary Cross-Entropy (WBCE) provides the best reconstruction accuracy at lower thresholds, whereas F1 Loss offers more stable performance across varying levels of occlusion.

The findings of this thesis suggest that 3D to 3D reconstruction is the preferred method for scenarios with significant occlusions, while Image to 3D reconstruction remains viable in settings where only 2D inputs are available. Future work should explore hybrid models that combine 2D and 3D data, develop adaptive thresholding techniques to improve reconstruction confidence, and conduct real-world validation to bridge the gap between experimental and practical applications.

Acknowledgments

This work would not have been possible without the support, guidance, and kindness of many wonderful people (and creatures).

First and foremost, I want to thank my thesis committee—Prof. Oliver Kroemer, Prof. George Kantor, Prof. Shubham Tulsiani, and John Kim—for your unwavering support, insightful feedback, and encouragement throughout this journey. Your expertise and mentorship have deeply shaped this work and my growth as a researcher.

To my lab mates and collaborators: thank you for the late-night debugging sessions, idea exchanges, and all the little moments that made this experience rich and rewarding. Your camaraderie and brilliance kept me motivated, especially during the most challenging phases of this thesis.

To my dear friends—near and far—thank you for your constant encouragement, for being sounding boards, for making sure I touched grass, and for reminding me that there is a life outside of voxel grids and neural networks.

To my parents, thank you for your unconditional love, belief in me, and for all the sacrifices you've made to support my path. Your strength and resilience are the roots that ground me.

And lastly, to Mocha and Mochi—thank you for your snuggles, purrs, and constant curiosity about what I was doing on my keyboard at 3 a.m. You two have been the perfect writing companions.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

In agricultural robotics, tasks like inspection or harvesting often require interaction with trees. A detailed 3D model of a tree enables the robot to reason about interaction dynamics, such as how to pick a fruit or push aside an occluding branch. However, generating accurate 3D models in the field is challenging due to environmental factors like heavy occlusion and noise. State-of-the-art agricultural manipulation methods often bypass 3D modeling altogether, relying instead on heuristics for contact dynamics. This can lead to suboptimal interactions, such as damaging the tree or failing to reach the target fruit.

Existing approaches to 3D reconstruction in agricultural settings predominantly rely on point cloud data obtained via 3D sensors such as Light Detection and Ranging (LiDAR) or depth cameras [9]. While these methods can generate coarse 3D representations, they frequently incorporate heuristic assumptions, making them prone to errors under conditions of heavy noise and occlusion. To address these limitations, there is a growing interest in Image to 3D reconstruction methods, which leverage 2D images to infer 3D structures. Multi-view approaches dominate this space but often require complex setups and are less practical for dynamic or large-scale agricultural environments. By contrast, single-view methods, although more challenging due to the limited input data, offer a scalable and efficient alternative. In this thesis, we explore single-view 3D reconstruction to simplify the process while addressing the challenges of occlusion and noise.

Single-view 3D (SV3D) reconstruction in agricultural robotics presents a unique

challenge: generating a 3D representation of a tree's skeletal structure from a single RGB image of a tree with dense foliage. This is critical for tasks like pruning or fruit harvesting, where accurately identifying branches and the trunk without interference from leaves is essential. For instance, a 3D model that excludes foliage can significantly improve the precision of robotic arms, reducing the risk of damage and increasing efficiency. In this work, we propose an encoder-decoder architecture that learns to infer 3D volumes from single RGB images, assuming the availability of ground-truth 3D labels. By training the model to focus on the skeletal structure rather than foliage, we aim to generate accurate reconstructions that are robust to occlusion.

However, achieving this requires addressing an open-ended research question: given a heavily occluded image of a tree, can a model reliably reconstruct a 3D representation that excludes foliage and captures only the skeletal structure? To further advance this field, we also investigate 3D to 3D reconstruction methods, where the input is a partially occluded 3D voxel grid, and the output is a complete 3D voxel grid of the tree structure. This volumetric approach can provide more accurate and detailed reconstructions by directly processing 3D data. By comparing Image to 3D and 3D to 3D reconstruction methods, this thesis evaluates their effectiveness in reconstructing accurate 3D tree models under varying conditions of occlusion.

The remainder of this thesis is structured as follows: Chapter 2 reviews related work in 3D reconstruction and its applications in agricultural robotics. Chapter 3 details the dataset generation process, including the creation of synthetic tree models and datasets for both Image to 3D and 3D to 3D reconstruction. Chapter 4 describes the methodologies, including model architectures and loss functions. Chapter 5 presents experimental results, comparing reconstruction methods and analyzing their performance. Finally, Chapter 6 concludes the thesis, discussing the findings and offering suggestions for future work.

Chapter 2

Related Work

2.1 Learning-Based Methods for 3D Reconstruction in Robotics

Recent advancements in learning-based methods have significantly enhanced the capabilities of 3D reconstruction, particularly in shape completion and occluded area reconstruction. This section explores several key contributions that utilize deep learning techniques to address these challenges, focusing on applications that enhance robotic perception and interaction.

2.1.1 Shape Completion in Robotics

Traditional 3D reconstruction methods often rely on dense sampling and favorable viewing conditions, which are not always feasible in unstructured environments. Varley et al. (2017) develop a deep learning framework for robotic grasp planning using a 3D Convolutional Neural Network (CNN). Their method predicts complete geometry from partial point clouds, significantly enhancing grasp accuracy in unstructured environments [14]. Similarly, Dai et al. (2017) propose a 3D-Encoder-Predictor CNN architecture for volumetric shape completion. Their approach efficiently completes shapes from sparse and incomplete point clouds, improving robotic interaction capabilities [4]. Han et al. (2017) focus on high-resolution shape completion, emphasizing the

inference of fine structures in 3D scans. This work is particularly crucial for modeling organic shapes, such as tree limbs, in agricultural robotics [7]. While these methods demonstrate significant advancements in reconstructing hand-sized or moderately complex objects, they are not directly designed to handle the geometric and topological complexity of full tree structures. Unlike smaller objects, trees present unique challenges due to their intricate branching patterns, varying densities, and occluded regions caused by leaves or other environmental factors. This level of structural detail is critical in applications such as precision pruning, fruit harvesting, and biomass estimation, where the reconstruction of fine branches and overall topology plays a pivotal role.

2.1.2 Handling Occluded Areas

The ability to infer occluded areas in reconstruction for agricultural robotics is especially important for robotic control and planning. Sock et al. (2018) integrate RGB-D data for real-time occlusion-aware 3D object reconstruction. Their method effectively leverages the fusion of depth and color information to infer the geometry of occluded surfaces, providing a comprehensive 3D model for robotic navigation and manipulation [12]. Firman et al. (2016) develop a technique using sparse depth inputs from LiDAR sensors to predict dense depth maps, excelling in reconstructing occluded areas and enhancing the operational safety and efficiency of autonomous vehicles and robotic systems [5]. These methods integrate real-time data processing with deep learning to reconstruct occluded surfaces effectively, leveraging advancements in sensor technologies and computational models. Kim et al. (2023) present a novel approach to extracting the skeleton of self-occluded tree canopies by estimating the unobserved structures of the tree. This method utilizes an instance segmentation network to detect visible trunks, branches, and twigs and then builds a custom 3D likelihood map in the form of an occupancy grid. This grid hypothesizes the presence of occluded skeletons through minimum-cost path searches, effectively addressing the occlusion challenge in heavily foliated areas [9]. This technique is particularly tailored for agricultural robotics, where the precise modeling of tree structures is crucial for tasks such as pruning and harvesting. Various data representations for 3D reconstruction have been explored. Park et al. (2019) propose DeepSDF, a signed

distance function learned via a deep neural network, enabling the reconstruction of complex shapes from sparse and noisy data [10]. Choy et al. (2016) introduce the 3D-R2N2 network, which utilizes a 3D occupancy grid that allows for incremental updates to the reconstruction as more data becomes available, a significant departure from methods that require complete and clean datasets [3].

2.2 3D Reconstruction of Plants

The application of three-dimensional (3D) reconstruction technologies has become increasingly prominent in agriculture and environmental science, marking a significant leap forward in how plant models are created and analyzed. While traditional methods such as photogrammetry and structured light scanning have laid the groundwork by offering detailed but time-intensive reconstructions, the integration of multiview stereo (MVS) techniques and deep learning has heralded a new era. These advancements promise to dramatically reduce computational demands while enhancing the scalability and detail of 3D plant models. In agricultural settings, 3D reconstruction has primarily been applied to broad-scale tasks like field mapping and crop inspection. For example, Potena et al. (2019) and Chebrolu et al. (2017) made significant contributions to aerial and ground collaborative mapping in precision farming [2, 11]. However, recent studies have begun to address the more intricate challenges of individual plant modeling, addressing specific needs within agricultural research and practice. Tabb and Medeiros (2017) designed a learning-based system capable of autonomously measuring tree traits in field settings; however, this is computationally intensive, making it unrealistic for field applications [13]. Marin et al. (2015) and Millan et al. (2019) explore image-based segmentation techniques for structural recovery and pruning weight assessment in vineyards. These studies advance our understanding of plant structure through image segmentation and 3D modeling but underscore the inherent challenges in achieving high precision, particularly in identifying branching points and segmenting plants against dynamic outdoor backgrounds. A recent study by Freeman et al. (2022) focused on the 3D reconstruction of sorghum panicles, utilizing seeds as semantic landmarks. This approach improves upon traditional methods in seed count and weight estimation, and introduces novel metrics for assessing point cloud quality [6].

2.3 Contribution

Key contributions in this project include a custom dataset comprising tree meshes with and without leaves, which can be used to render images and obtain ground truth 3D volumes, as well as a study on appropriate encoder-decoder model architecture and ablations on various loss metrics. Specifically:

- **Data Acquisition:** We used a tree growth simulator capable of providing 3D models of the same tree with and without leaves. A dataset of 5000 3D tree meshes and corresponding images from multiple viewpoints was collected.
- **Baseline Single View Image to 3D:** Given a single RGB image of a tree without leaves, we trained a model that outputs a 3D volumetric representation of the tree structure. We evaluated three different loss functions to train the model, including Chamfer Distance, Earth Mover's Distance, and Density-Aware Chamfer Distance [15], to determine the appropriate metric.
- **Single View 3D Voxels to 3D:** Given a partially occluded voxel grid from a single view, we trained a model that outputs the complete 3D volumetric representation of the tree structure. We used masked loss on unknown areas, Binary Cross Entropy (BCE) Loss, and Weighted Binary Cross Entropy (WBCE) Loss to train the model.

Chapter 3

Dataset Generation

This chapter details the procedures involved in generating and preprocessing the dataset used in our study. To train networks capable of learning complete tree structures from a single view, we created two datasets. The first dataset D_{2D} uses 2D images as input $X_{i,j}$ and outputs in three different 3D formats: point clouds $Y_{i,j}^{\text{pc}}$, meshes $Y_{i,j}^{\text{mesh}}$, and voxel grids $Y_{i,j}^{\text{voxel}}$. The second dataset D_{3D} uses augmented 3D partial voxel grids $X_{i,j}$ as input and complete voxel grids $Y_{i,j}$ as output. The first dataset enables the network to learn how to reconstruct different 3D representations from 2D images. The second dataset focuses on refining the network's ability to complete partial 3D voxel grids, thereby enhancing its understanding and reconstruction of occluded or incomplete structures.

Obtaining high-resolution datasets of real-world tree canopies, especially with occluded areas, is often time-consuming and inaccurate. To address this, we propose a pipeline to generate high-fidelity synthetic dataset of trees, both with and without leaves, generated using Blender. Our approach includes creating a custom dataset of 3D tree models, preprocessing these models into a format suitable for 3D reconstruction, and implementing dataloaders for efficient data handling.

3.1 Tree Model Generation

This section outlines the steps involved in generating the Blender Tree Model Dataset, which is a dataset comprised of meshes of synthetic trees of 5 different species with

and without leaves.

Reconstructing 3D representation of trees requires an understanding of the branches' physical structure and consideration of complex occlusions from leaves and twigs. Due to the uniqueness of trees' structures and shapes, we made several considerations for the data generation process: 1) to ensure that our samples adequately represent real-world trees, we generated a wide variety of tree models. This diversity is crucial for developing robust reconstruction algorithms capable of generalizing to different tree species and shapes; 2) For each generated tree skeleton mesh, we also created a corresponding mesh with leaves. This dual representation allows us to better understand and manage occlusions and unknown areas during the reconstruction process. In this section, we discussed the detailed steps and designs for the dataset generation procedure.

We built our custom multithreaded script to allow simultaneous generation and exporting trees, leveraging Grove 3D [1], a sophisticated tree-growing simulator plugin for Blender and Blender's Python API. The script enables simultaneous generation and export of multiple trees in batches, significantly speeding up the data creation process. We managed threading by assigning tree generation and export tasks to separate threads, ensuring efficient utilization of computational resources and minimizing idle time.

The Grove 3D allows for the procedural generation of detailed tree structures, which we utilized to create a comprehensive dataset of 5,000 unique tree skeleton meshes. Additionally, we generated 5,000 meshes of the same tree skeletons with leaves, saved in '.obj' format. This dataset includes an equal distribution of five different tree species—Olive, Maple, Walnut, Pine, and Linden—with each species represented by 1,000 tree models without leaves and 1,000 with leaves. Figure 3.1 shows an example of a tree model generated using The Grove 3D. Parameters during the tree generation were carefully selected and randomized to enhance the robustness of the dataset, as detailed in Table 3.1.

Unconstrained generation often resulted in unrealistic clustering of excessively short or dense branches. Therefore, we incorporated a dynamic pruning function from The Grove 3D, which automatically adjusts pruning parameters for each tree model. Branches growing below 0.4 unit height are pruned. In addition, we limited the tree flush number (i.e.,) between 3 and 7 to prevent cluttering and excessive

Table 3.1: Parameters Used in Generating Tree Models

Parameter	Value
Tree Scale	0.6
Flush Number	3-7
Pruning Height	0.4 units
Randomization	Yes

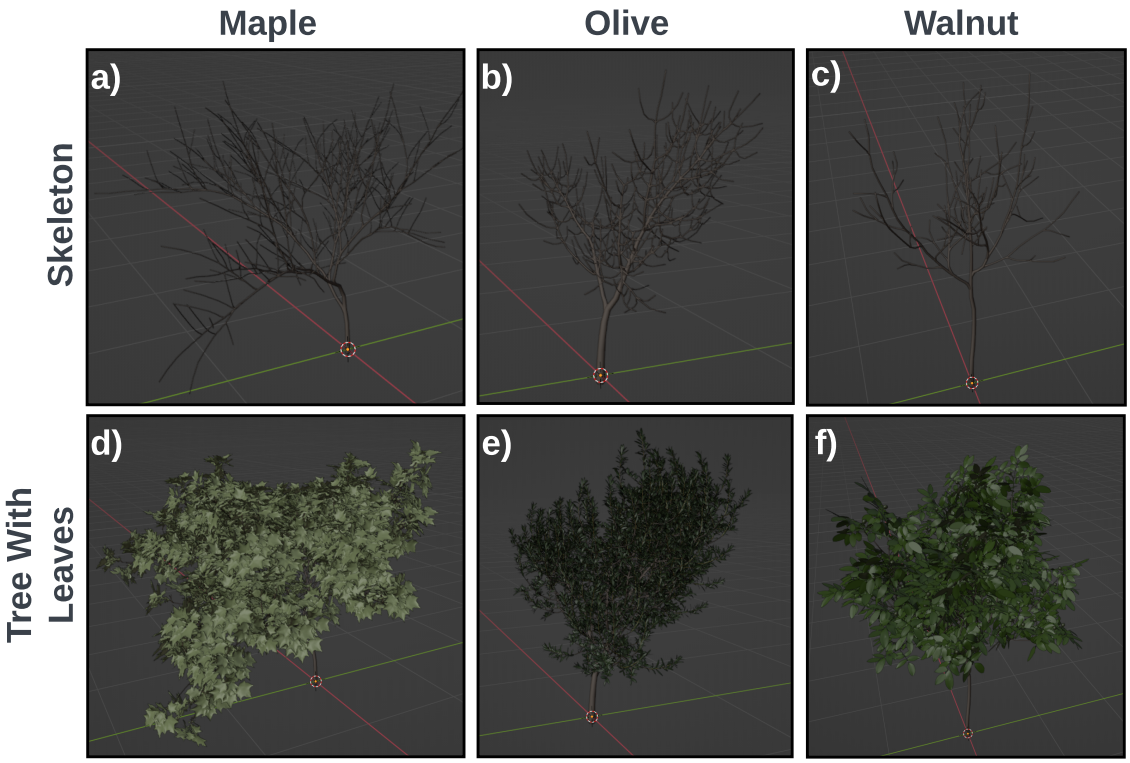

Figure 3.1: Examples of tree models generated with The Grove[1], a-c) the skeletons of the trees, d-f) the models of the associated trees with leaves.

growth, which could otherwise compromise realism.

To standardize our models, we configured the plugin settings to maintain uniformity in size across the generated trees. We limited parameters such as the scale/size of the tree, and the number of tree flushes. Specifically, we set the scale to 0.6 to limit the tree to grow in a unit cube. Note that this does not mean the model will only be able to adapt to trees of certain sizes, but instead input images of trees may still be rescaled to fit our mode.

After generation, we performed post-processing to ensure consistent alignment and orientation of the tree models. Overall, we obtained 5000 samples of diverse tree samples in 5 categories. These samples will allow the model to generalize well to a number of tree categories, orientations, sizes, and shapes.

3.2 Image to 3D Reconstruction Dataset Generation

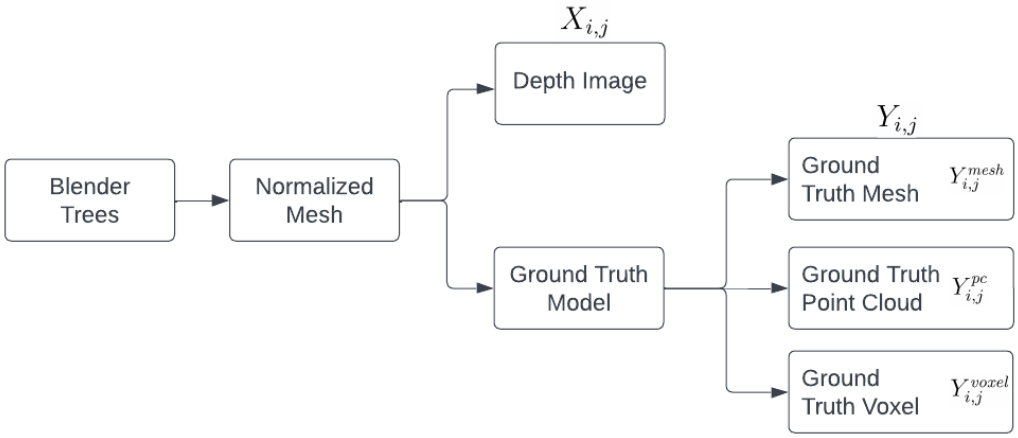

Figure 3.2: Image to 3D reconstruction dataset generation pipeline.

To implement a supervised learning pipeline for image reconstruction of trees, we created a dataset consisting of paired 2D images and their corresponding 3D ground truth objects. The dataset is defined as $D_{2D} = \{(X_{i,j}, Y_{i,j})\}_{i=1,j=1}^{n,m}$, where:

- $X_{i,j}$ represents the j-th 2D image of the i-th sample, generated from one of 24 distinct viewpoints.

- $Y_{i,j}$ includes three types of 3D model formats:
  - Point clouds $Y_{i,j}^{pc}$,
  - Meshes $Y_{i,j}^{mesh}$,
  - Voxel grids $Y_{i,j}^{voxel}$.
- $m = 24$ is the number of images generated from equidistant viewpoints for each tree mesh.
- n is the number of samples in D_{2D}, which is 5000.

Each instances $X_{i,j}$ is an image sampled from the space of 2D images $\mathbb{R}^{w*h}$, and each label $Y_{i,j}$ from the combined space of 3D structures. This arrangement captures multiple perspectives of each sample, enhancing the dataset's utility for training models that need to recognize and reconstruct 3D structures from 2D inputs.

A custom data loader was developed to facilitate the training process. It performs the following functions: Given tree mesh files, the loader selects m^{th} input image $X_{i,j}$ for each mesh from the pool of 24 distinct viewpoints. It also applies view-aligned, object-centric rotations to the ground truth 3D models, converting them into the desired formats.

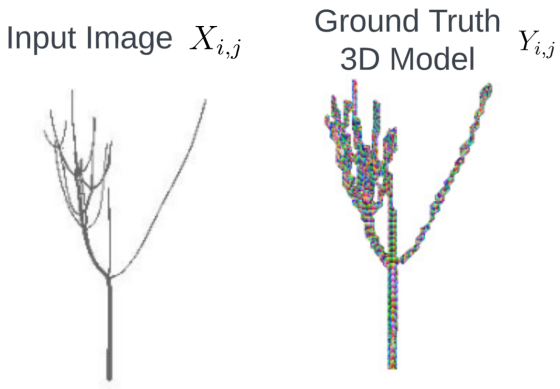

Figure 3.3: Examples of the input and ground truth of the D_{2D} dataset.

We used Pytorch3D's Renderer to render images $X_{i,j}$ in RGB, RGBD, and depth formats. The images were generated with 15-degree increments around the tree's z-axis to ensure comprehensive coverage of all possible views. In PyTorch3D, rendering is achieved through the combined action of a rasterizer and a shader. For rasterization, we employed Pytorch3D's MeshRasterizer to handle the geometric complexity of

the trees, and for shading, we used the HardPhongShader which provides detailed lighting and surface reflections. We pre-processed the given mesh files to extract the point clouds and voxel representations of the model. The point clouds $Y_{i,j,\mathrm{pc}}$ were derived by sub-sampling 1000 points from the input mesh models, providing a sparse representation of the objects' full surface geometry. Meanwhile, to obtain 3D data in $Y_{i,j,\mathrm{voxel}}$, we voxelized the mesh with Open3D and stored the voxelized tree in a binary format. This storage method simplifies the interaction with neural networks by providing a uniform input structure.

Through the processes outlined above, we compiled a dataset $D_{2D} = \{(X_{i,j}, Y_{i,j})\}_{i=1,j=1}^{n,m}$, where ($X_{i,j}$ denotes a list of 2D images, and $Y_{i,j}$) represents the corresponding 3D ground truth objects in various formats such as point clouds $Y_{i,j,\mathrm{pc}}$, meshes $Y_{i,j,\mathrm{mesh}}$, and voxels $Y_{i,j,\mathrm{voxel}}$.

3.3 3D to 3D Reconstruction Dataset Generation

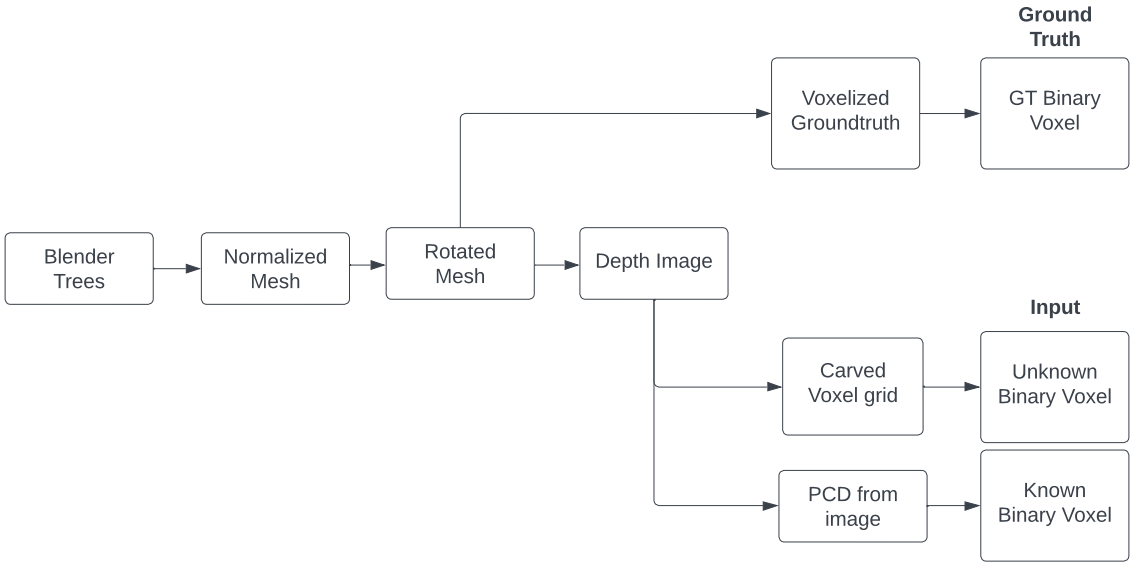

Figure 3.4: 3D to 3D Reconstruction Dataset generation pipeline.

We generated input $X_{i,j}$ and ground truth $Y_{i,j}$ dataset pairs for 3D reconstructions using voxel grids as the second dataset, $D_{3D} = \{(X_{i,j}, Y_{i,j})\}_{i=1,j=1}^{n,m}$. Throughout this thesis, "voxel grid" refers to a 64^3 or 128^3 cube with voxels occupying the space, and

"voxel" refers to a single cube within the voxel grid. Both the input and ground truth are represented as voxel grids at resolutions of 64^3 and 128^3. The input voxel grid $X_{i,j}$ represents an occluded view of the tree from a single perspective $Y_{i,j}$, while the ground truth voxel grid is obtained by voxelizing the entire Blender-generated tree mesh. We chose voxel representations for both the input and ground truth to facilitate the use of skip connections in our network architecture.

To generate the input voxel grid $X_{i,j}$—which represents known occupied areas, known unoccupied areas, and unknown areas using binary voxel grids—and the ground truth voxel grids $Y_{i,j}$, we followed the procedure outlined in Figure 3.4. The high resolutions (64^3 and 128^3) were selected to capture detailed information about thin tree branches while also allowing for efficient computation. We defined all voxel grids were initially 1 unit length and calculated a voxel edge length for each voxel grid resolution.

3.3.1 Input Voxel Grid Generation

Depth Image Acquisition

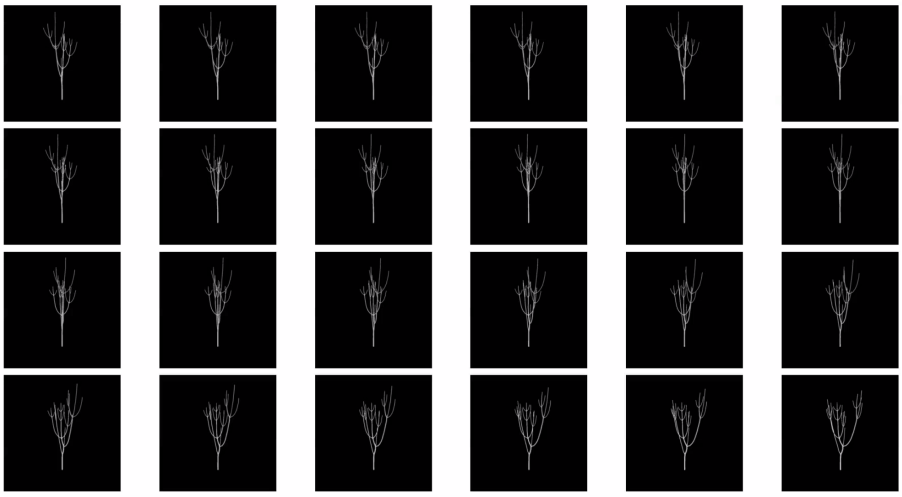

Figure 3.5: Example of the generated depth images from 24 different views of the same tree model.

Depth images were generated from 24 distinct viewpoints around the z-axis of the tree mesh, ensuring even coverage. We maintained the position of the tree mesh and

the camera's position and orientation consistently but rotated the tree around its own z-axis by 15 degrees per increment to ensure uniform voxel grid generation. The tree and camera were separated by 1.5 meters to ensure a full capture of the tree. We used Open3D's default camera with a FOV of 90 to simulate realistic camera distortion and a depth scale of 1000 to convert depth values from the native units of the depth sensor to millimeters. Offscreen rendering capabilities were used for efficient image generation. The depth image was then used for point cloud generation and voxel carving.

3.3.2 Dataset Generation Overview

Known Occupied Voxel Grid

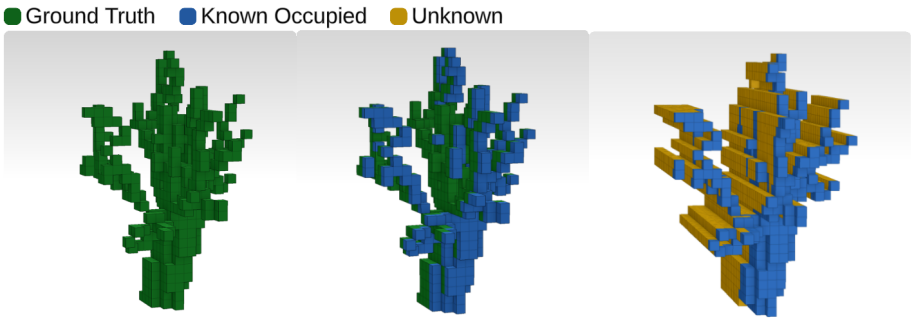

Figure 3.6: Voxel representation of tree model that shows the ground truth voxels, known occupied voxels and unknown voxels.

The depth images were converted into point clouds using the `create_point_cloud_from_depth_image` function from Open3D. We obtained the camera's intrinsic and extrinsic parameters. The point clouds were then voxelized with the calculated voxel edge length to create the known occupied areas of the input voxel grid.

Occluded Area Representation

A dense unit voxel grid centered at the origin of the tree model was generated first. The voxel size in the voxel grid was the same as the previously calculated voxel edge length. Voxel carving was applied from the camera's viewpoint using the generated

depth image, effectively carving out the view frustum from the dense grid. From this, we obtained a binary voxel grid that covers both the known occupied area and the occluded area.

3.3.3 Ground Truth Voxel Grid Generation

To ensure a consistent grid, the entire tree model was voxelized from various viewpoints to serve as the ground truth. This process ensured comprehensive coverage of the tree structure. The resulting binary voxel grids for each tree model from every view were stored in .binvox format for efficient data storage.

3.3.4 Custom Data Loader Implementation

We implemented a custom data loader to efficiently manage the loading of input and ground truth *.binvox* files. The data loader employs a view-aligned, object-centric coordinate frame for representing both the ground truth tree voxel grid and the inputs. Furthermore, we merged the binary-valued voxel grid from the point cloud and the carved voxel grid into a single voxel grid. In this grid, a value of 1 indicates occupied, 0.5 denotes unknown, and 0 signifies unoccupied spaces. We also generated a mask voxel grid to target the unknown area during the training process. For this masked voxel grid, occupied voxels in both known and unknown areas are marked as 1.

Chapter 4

Method

In this section, we describe our methods in approaching tree 3D reconstruction in details. First, we adopt a conventional method in taking 2D images as input to reconstruct the 3D geometry of trees. Theses models map input tree images with image encoders and decode to various 3D formats such as voxel, point clouds, and mesh. Moreover, as trees consist of complex 3D structures which may not be fully disclosed by a single 2D image, we present a 3D-to-3D reconstruction method that offers additional capacity in encoding the 3D structures of trees.

4.1 Image to 3D Reconstruction (M_{2D})

4.1.1 Model Architecture

In this section, we introduce the model architecture we used to perform 3D reconstruction using given input tree images. For brevity, we omit the subscript i, j in this section in describing the model process on a single instance in the dataset. The proposed approach aims to establish a baseline 3D model by training an encoder-decoder model, as illustrated in Figure 4.2 using the D_{2D} dataset we generated previously. The goal is to learn a function M_{2D} that can map the input 2D image X to 3D formats Y of trees, formally represented as $Y' = M_{2D}(X)$.

Here we describe the architecture of the Image to 3D model M_{2D}. The encoder leverages the ResNet18 architecture to encode the input into a latent feature vector,

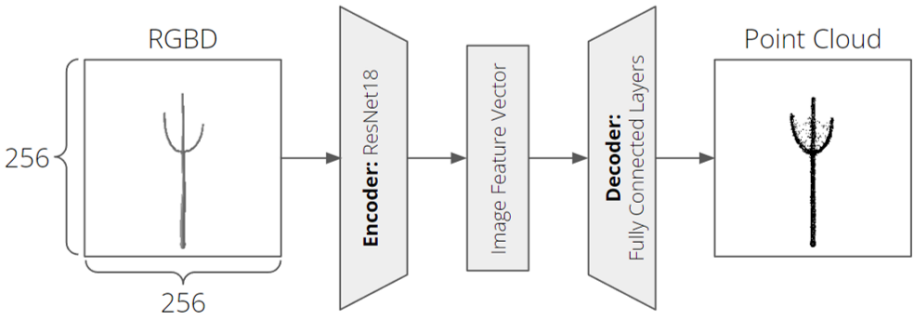

Figure 4.1: Our model architecture. The encoder is ResNet18 [8] with 3 channel input to encode 256x256 RGBD image into a latent feature vector and the decoder is 5 fully connected linear layers $(512, 1024, 2048, 4096, 8192, 3 * n_points)$ with RELU activation except at the final layer.

while the decoder consists of five fully connected layers with a ReLU activation function, except for the last layer. This model generates a 3D representation of a tree structure using a RGB, a depth(*.png*), or a RGBD image of a leafless tree. Overall, the model consists of the following components:

- **Encoder**: The encoder uses the ResNet18 architecture to process the input image. It extracts hierarchical features through a series of convolutional layers and outputs a latent feature vector. The encoder is pre-trained on a large dataset and fine-tuned on our specific task.
  - `ResNet18` [8]: Pre-trained model used to encode the input image into a 512-dimensional latent feature vector.
- **Decoder**: Depending on the type of 3D representation (voxel grid or point cloud), the decoder has different structures:
  - `Voxel Decoder (`$\mathtt{M}_{2D}^{v}$`)`: For voxel grid output, the decoder consists of fully connected layers that reshape the latent feature vector into a 64x64x64 voxel grid.
    - Linear layers with sizes: 512, 1024, 2048, 4096, 4096, 8096, 64x64x64
  - `Point Cloud Decoder (`$\mathtt{M}_{2D}^{pc}$`)`: For point cloud output, the decoder consists of fully connected layers that reshape the latent feature vector into $n_{\text{points}} \times 3$ format.
    - Linear layers with sizes: 512, 1024, 2048, 4096, 4096, 8096, $n_{\text{points}} \times 3$

- `Mesh Decoder` ($\texttt{M}_{2D}^{m}$): For mesh output, the decoder typically reconstructs the 3D surface geometry by predicting the vertex positions and face connectivity. The decoder consists of fully connected layers that output the vertex coordinates, followed by additional layers or operations to predict the mesh topology.
  - Linear layers with sizes: 512, 1024, 2048, 4096, 8192, $n_{\text{vertices}} \times 3$
  - Additional processing includes:
    - Vertex Prediction: Outputs the 3D coordinates for each vertex in the mesh.
    - Mesh Topology: Can be handled by pre-defining the face connectivity or predicting it using methods like a graph neural network (GNN) or other specialized layers that generate the mesh faces from vertices.

- **Activation Functions**: ReLU activation functions are used after each fully connected layer to introduce non-linearity, except for the final layer in the point cloud decoder.
- **Normalization**: Input images are normalized using pre-defined mean and standard deviation values to match the distribution of the pre-trained ResNet18 model.

4.1.2 Loss Functions

The effectiveness of each 3D reconstruction format is closely tied to the loss function used during training. Here, we describe the loss functions tailored to each format and their respective strengths and weaknesses:

- **Chamfer Distance ($\mathcal{L}_{\text{CD}}$) for Point Cloud Reconstruction:**
  - Sensitive to noise and provides a measure of the average distance between points in the predicted and target sets.
  - Effective for assessing the overall alignment of the point clouds.

- Equation:

$$\mathcal{L}_{\text{CD}}(Y', Y) = \frac{1}{|Y'|} \sum_{y' \in Y'} \min_{y \in Y} \|y' - y\|^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{y' \in Y'} \|y - y'\|^2 \quad (4.1)$$

- **Earth Mover's Distance ($\mathcal{L}_{\text{EMD}}$) for Mesh Reconstruction:**
  - Measures the minimum cost of transforming one point cloud Y' into another Y, represented as $\phi : Y' \rightarrow Y$.
  - Useful for evaluating overall shape and structure similarity.
  - Equation:

$$\mathcal{L}_{\text{EMD}}(Y', Y) = \min_{\phi: Y' \rightarrow Y} \sum_{y' \in Y'} |y' - \phi(y')|^2 \quad (4.2)$$

- **Binary Cross-Entropy Loss ($\mathcal{L}_{\text{BCE}}$) for Voxel Grid Reconstruction:**
  - Measures the difference between predicted probabilities and actual binary outcomes.
  - Effective for voxel grid representations, where accurate voxel classification is crucial for model fidelity.
  - Equation:

$$\mathcal{L}_{\text{BCE}} = -\sum_{i} (y_i \log(x_i) + (1 - y_i) \log(1 - x_i)) \quad (4.3)$$

4.2 3D-to-3D Reconstruction (M_{3D})

4.2.1 Model Architecture

This section focuses on the task of 3D-to-3D reconstruction, where the goal is to take an input 3D voxel grid and reconstruct it into the same or higher resolution while maintaining structural integrity and detail. The proposed approach is based on a primary model architecture using an encoder-decoder scheme, which we call "Voxelto3D" (M_{64}). This model is designed to handle volumetric data effectively, with several variations to explore the impact of different architectural choices.

For 3D to 3D reconstruction, we developed a primary model architecture based

on an encoder-decoder scheme, leveraging the capabilities of a pre-trained network adapted from the ShapeNet architecture. The primary model, named "Voxelto3D" (M_{64}), integrates a pre-trained vision model to serve as its encoder, designed to extract features from 3D volumetric data. This framework processes the input voxel grid through the encoder, and the decoder then reconstructs this feature representation back into the original volumetric format. This primary model reconstructs the input into a 64x64x64 voxel grid, using skip connections for better reconstruction quality. The primary objective is to learn a mapping function f_{3D} such that $y'_{i,j} = f(x_{i,j})$ where

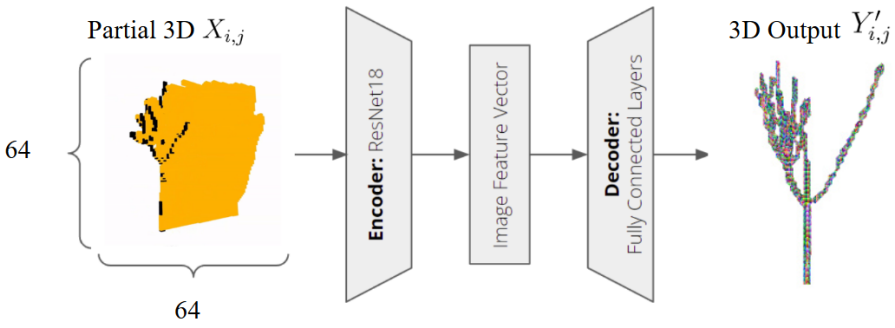

Figure 4.2: Our 3D to 3D reconstruction model architecture. The encoder consists of three 3D convolutional layers with ReLU activation and max-pooling, transforming the input voxel grid into a compressed representation. The decoder uses transposed 3D convolution layers with skip connections, reconstructing the voxel grid from the encoded features. The final output is obtained using a sigmoid activation function to predict voxel occupancy.

$x_{i,j}$ is a 3D voxel grid input and $y'_{i,j}$ is the reconstructed 3D output using the D_{3D} dataset we generated. This model aims to accurately recreate the volumetric structure from a given input, reflecting detailed features and maintaining the structural integrity of the original model.

The architecture of the primary Voxelto3D model (M_{64}) consists of the following components:

- **Encoder**: The encoder comprises a series of 3D convolutional layers. It starts with an initial 3D convolution layer followed by subsequent layers to extract hierarchical features from the input voxel grid. Each convolution layer is followed by a ReLU activation function and a 3D max-pooling layer to reduce the spatial dimensions and retain essential features.

  - `enc_conv1`: 3D convolution layer with 1 input channel and 64 output channels.
  - `enc_conv2`: 3D convolution layer with 64 input channels and 128 output channels.
  - `enc_conv3`: 3D convolution layer with 128 input channels and 256 output channels.
- **Decoder**: The decoder reconstructs the encoded features back to the original voxel grid resolution. It uses transposed 3D convolution layers with skip connections to combine features from the corresponding encoder layers, ensuring better reconstruction quality.
  - `dec_conv1`: Transposed 3D convolution layer with 256 input channels and 128 output channels.
  - `dec_conv2`: Transposed 3D convolution layer with 384 input channels (256 from `dec_conv1` + 128 from `enc_conv3`) and 64 output channels.
  - `dec_conv3`: Transposed 3D convolution layer with 192 input channels (128 from `dec_conv2` + 64 from `enc_conv2`) and 32 output channels.
  - `dec_conv4`: 3D convolution layer with 96 input channels (64 from `dec_conv3` + 32 from `enc_conv1`) and 1 output channel.
- **Activation Functions**: ReLU activation functions are used after each convolution and transposed convolution layer to introduce non-linearity. The final output is passed through a sigmoid activation function to obtain voxel occupancy probabilities.

The forward pass of the primary model (M_{64S}) involves processing the input through the encoder layers to obtain a compressed representation, followed by decoding this representation back to the original 3D format using the decoder layers with skip connections for better feature retention.

4.2.2 Model Variations

In addition to the primary model architecture described, we experimented with several variations to explore their impact on reconstruction quality:

- **Model without Skip Connections** (M_{64}^{NS}): This model variation excludes the skip connections between the encoder and decoder layers, relying solely on the latent representation for reconstruction.
- **Deeper Model** (M_{64}^{D}): This model includes additional layers in both the encoder and decoder to increase the network's capacity and capture more complex features.

4.2.3 Loss Functions

To train the model, we utilized different loss functions to optimize the reconstruction quality:

- **Binary Cross-Entropy Loss ($\mathcal{L}_{\mathbf{BCE}}$):**

$$\mathcal{L}_{\text{BCE}} = -\sum_{i} \left(y_i \log(x_i) + (1 - y_i) \log(1 - x_i)\right) \tag{4.4}$$

  Applied directly to compare the predicted voxel occupancy (x_i) with the target (y_i), this loss function is particularly effective for datasets where the classes are approximately balanced.

- **Weighted Binary Cross-Entropy Loss ($\mathcal{L}_{\mathbf{WBCE}}$):**

$$\mathcal{L}_{\text{WBCE}} = -\left(w_{\text{occupied}} \sum_{i \in \text{occupied}} \log(x_i) + w_{\text{unoccupied}} \sum_{i \in \text{unoccupied}} \log(1 - x_i)\right) \tag{4.5}$$

  This loss function manages class imbalance by adjusting the loss contribution of each voxel based on its class frequency. This method emphasizes learning from underrepresented classes, thus enhancing model sensitivity to these classes.

- **Masked L1 Loss ($\mathcal{L}_{1_m}$):**

$$\mathcal{L}_{1_m} = \frac{\sum_i m_i \cdot |x_i - y_i|}{\sum_i m_i} \tag{4.6}$$

  This loss function applies a mask (m_i) to selectively weight the absolute differences between the predicted (x_i) and target (y_i) voxel values, emphasizing errors in specific regions. The mask m_i is derived from the input voxel grid

and assigns a weight to each voxel based on its importance for reconstruction. m_i is set to 1 for unknown or uncertain areas and to 0 for regions that are already known to be unoccupied. This approach ensures the model prioritizes reconstructing critical and uncertain areas within the voxel grid, while still considering known regions to maintain overall reconstruction accuracy.

Thresholding for Voxel-Based Classification

Since the output of the reconstruction models is a probabilistic voxel grid, a threshold t is applied to determine whether a voxel is classified as occupied (1) or empty (0). For a given voxel prediction $p(x)$, the final binary output is determined as:

$$\hat{y} = \begin{cases} 1, & p(x) \geq t \\ 0, & p(x) < t \end{cases} \tag{4.7}$$

where t is varied to analyze its impact on **precision and recall**.

Lower values of t increase **recall** by accepting more uncertain voxels, which ensures that more tree structures are captured but may introduce false positives. Conversely, higher values improve **precision** by requiring stronger confidence in occupied voxels, leading to a more conservative reconstruction that avoids false positives but risks missing smaller branches.

Chapter 5

Experiments

In this section, we validate our approach proposed in section 4 to understand the reconstruction capability as well as to compare the performance across various model and input configurations. Specifically, we designed our experiments to answer the following questions:

- **RQ1**: Does M_{3D} offer superior performance in tree reconstruction than M_{2D}?
- **RQ2**: What model architecture and configurations of M_{3D} is most suitable for tree reconstruction?

The rest of the section is then organized as follows. We begin by presenting the reconstruction results using M_{2D} to predict the 3D reconstruction using a 2D image (5.1). Subsequently, we presented the reconstruction results of M_{3D} using various model architectures (**??**) and loss functions (**??**). Finally, we carefully compared and analyzed the performance of M_{3D} vs. M_{2D} and summarized the findings (**??**).

5.1 Experiment on Image to 3D Reconstruction

In this section, we evaluate the performance of M_{2D} methods for reconstructing 3D representations from 2D image inputs.

Experiment Setup. The experiments cover models that output distinctive 3D geometry output formats: voxel M_{2D}^{v}, point cloud M_{2D}^{pc}, and mesh M_{2D}^{m}. Three different loss functions were tested: Chamfer Distance ($\mathcal{L}_{\mathrm{CD}}$), Earth Mover's Distance

($\mathcal{L}_{\text{EMD}}$), and Density-aware Chamfer Distance ($\mathcal{L}_{\text{DCD}}$). The details of the model architecture and loss function definitions are described in Section 4.1. The models were trained on a Tesla V100 for 2 hours, and the training parameters are listed in Table 5.1.

Hyperparameter	**Value**
Learning Rate	5×10^{-5}
Optimizer	Adam
Batch Size	16
Maximum Iterations	10,000
Weight Decay	1×10^{-5}
Validation Frequency	100 iterations
Checkpoint Frequency	1000 iterations

Table 5.1: Summary of Hyperparameters for Model Training

Evaluation Metrics. The evaluation of the reconstructed models was conducted using the F1 score. The F1 score provides a measure of a model's accuracy by considering both precision and recall, which is particularly useful for imbalanced datasets and detailed structure evaluation.

5.1.1 Result

We present a thorough examination of the qualitative outcomes obtained from our model trained using diverse metrics and input formats for tree model analysis. Figure 5.1 showcases the comparative evaluation of these results. Our focus lies on investigating the effects of employing different metrics and input configurations, specifically considering tree models with simple configurations (without sub-branches) and complex configurations (with sub-branches) as input images. Our findings indicate that models trained with Earth Mover's Distance loss L_{EMD} exhibit the most visually compelling effects, while models trained with Density-Aware Chamfer Distance L_{DCD} yield the least desirable outcomes. Notably, models with L_{DCD} only manage to describe the main tree trunk, which coincides with the densest regions of the point clouds.

In addition to the qualitative analysis, we conducted a quantitative evaluation by comparing the F1 scores for each metric across various threshold values. The results

of this comparison are summarized in Figure **??**.

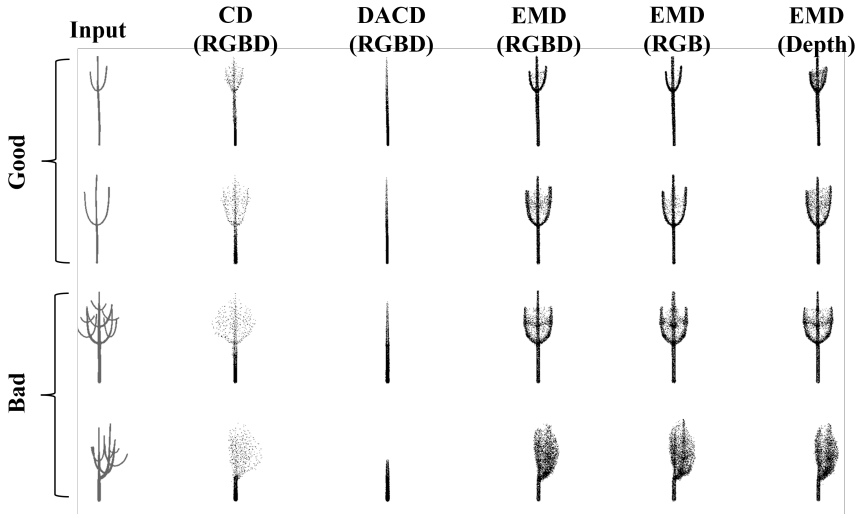

Figure 5.1: Qualitative results of example inputs trained with different point cloud distance metrics.

5.1.2 Discussion

5.2 Image to 3D Reconstruction vs. 3D to 3D Reconstruction

In this experiment, we compare the performance of image to 3D reconstruction (M_{2D}) using voxels with 3D to 3D reconstruction (M_{3D}). The goal is to evaluate which approach yields better reconstruction accuracy and robustness under occlusions.

5.2.1 Experiment Setup

To compare M_{2D} and M_{3D}, we trained models using the same dataset and evaluated their reconstruction accuracy across different confidence thresholds $(0.3, 0.4, 0.5)$.

- **Image to 3D Reconstruction (M_{2D})**: Converts a single 2D image into a 3D voxel representation using an encoder-decoder structure.

- **3D to 3D Reconstruction (M_{3D})**: Uses an incomplete 3D voxel input (e.g., partial tree structure) and reconstructs the missing regions.

The models were trained using Binary Cross-Entropy (BCE) Loss and evaluated using F1-score and IoU, which assess the precision-recall balance and volumetric similarity.

5.2.2 Results

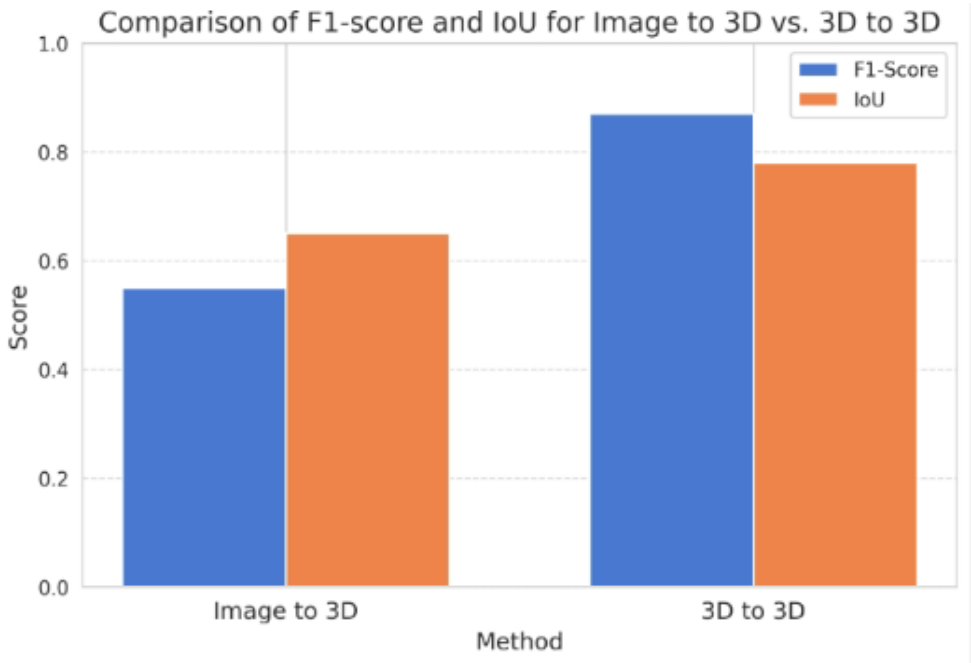

Figure 5.2: Comparison of F1-score and IoU for image to 3D vs. 3D to 3D reconstruction.

Key Observations

1. **3D to 3D Reconstruction (M_{3D}) Achieves Higher Overall F1-score** - M_{3D} maintains an F1-score of 0.87** at $t = 0.3$, compared to 0.55 for M_{2D}. - This suggests that 3D input significantly improves reconstruction accuracy.
2. **3D to 3D Reconstruction is More Robust to Occlusion** - IoU remains relatively stable for M_{3D} (0.78 to 0.7 as threshold increases). - In contrast,

IoU for M_{2D} increases at lower thresholds but struggles at higher ones. - This indicates that M_{3D} better reconstructs missing structures, making it **more reliable under occlusions**.

3. **Image to 3D Reconstruction (M_{2D}) Benefits from Lower Thresholds** - M_{2D}'s IoU improves significantly from 0.3 to 0.65 as the threshold increases. - This suggests that lower confidence thresholds allow it to capture more details, though at the risk of false positives.
4. **Trade-offs Between the Two Methods** - M_{3D} is superior in occluded environments, where partial 3D data provides stronger priors. - M_{2D} may still be preferable in less occluded scenarios, as it does not require partial 3D input.

5.2.3 Conclusion

These results indicate that:

- **3D to 3D reconstruction provides higher F1-score and IoU**, making it more robust for tree reconstruction under occlusion.
- **Image to 3D reconstruction is more sensitive to thresholding**, showing improvements in IoU at lower confidence levels.

5.3 Model Variations for Occluded 3D to 3D Reconstruction

5.3.1 Model Architectures

We experimented with the following three model variations, as detailed in Section 4.2.2:

- **Primary Model (64^3 Voxel Grid with Skip Connections) (M_{64S})**: This model reconstructs a $64 \times 64 \times 64$ voxel grid using **skip connections** between the encoder and decoder to enhance feature retention.
- **Model without Skip Connections (M_{64NS})**: This variant removes skip connections, forcing the decoder to reconstruct the volume solely from the latent feature representations.

- **Deeper Model (M_{64D})**: This model introduces additional layers in both the encoder and decoder to increase model capacity and capture finer structural details.

Thresholding in Model Evaluation: Since the output of these models is a probabilistic voxel grid, a threshold must be applied to classify each voxel as either occupied (1) or empty (0). To assess the robustness of each architecture, we evaluate performance across multiple thresholds $(0.3, 0.4, 0.5, 0.55)$, balancing recall (capturing more voxels) and precision (avoiding false positives).

5.3.2 Results and Discussion

We evaluate model performance across multiple thresholds $(0.3, 0.4, 0.5, 0.55)$ to analyze its impact on F1-score and IoU.

- **Lower thresholds (0.3, 0.4)**: More permissive in considering voxels as occupied, increasing recall at the cost of more false positives.
- **Higher thresholds (0.5, 0.55)**: More conservative, leading to higher precision but lower recall.

Figure 5.3 shows that the normal model maintains the best balance, while deeper models struggle with stricter thresholds.

The **performance comparison** of these models was evaluated using **F1-score** and **Intersection over Union (IoU)** across different **thresholds** $(0.3, 0.4, 0.5, 0.55)$. The results are summarized in Figure 5.3.

Key Observations

1. **The Normal Model (M_{64S}) Achieves the Best Performance**
   - Maintains the **highest F1-score** ($\sim 0.85 - 0.9$) across all thresholds.
   - **IoU is consistently higher** than the other models, indicating better spatial alignment with the ground truth.
   - However, both **F1-score and IoU drop sharply at 0.55**, suggesting the model struggles under high-confidence thresholding.
2. **Removing Skip Connections (M_{64NS}) Leads to Lower Performance**

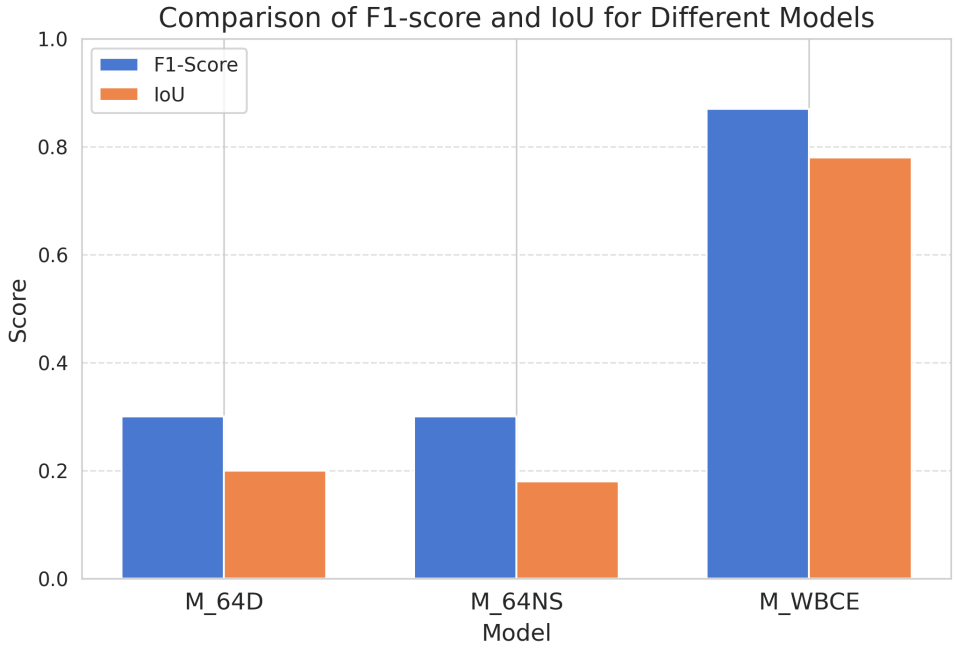

Figure 5.3: Comparison of F1-score and IoU for different models across thresholds.

- The F1-score is **lower than the Normal Model** across all thresholds.
- **IoU is significantly worse**, showing that the model struggles with precise voxel reconstructions.

3. **The Deeper Model (M_{64D}) Does Not Improve Accuracy**
   - **Adding more layers does not enhance performance**—F1-score remains low ($\sim 0.2 - 0.3$) across all thresholds.
   - **IoU is the lowest** among all models, suggesting that increasing depth does not improve volumetric reconstruction.
   - This may indicate **overfitting** or **inefficient feature extraction** in deeper layers.
4. **IoU Scores Are Lower Than F1-Scores Across All Models**
   - IoU is a **stricter metric**, as it penalizes both false positives and false negatives more heavily.
   - The **Normal Model (M_{64S}) maintains the best balance**, achieving significantly higher IoU compared to the other models.

5.3.3 Conclusion

From these experiments, we conclude that:

- **Skip connections significantly enhance reconstruction accuracy**, as shown by the superior performance of the **Normal Model** (M_{64S}).
- **Simply making the model deeper does not improve performance**, suggesting that **structural enhancements (skip connections) are more beneficial than increased depth**.

5.4 Varying Loss Functions for 3D to 3D Reconstruction

This experiment aims to evaluate the effectiveness of different loss functions on the reconstruction quality of the 3D models. The loss functions tested include:

5.4.1 Experiment Setup

The models were trained using the same dataset, D_{3D}, with each model trained separately using one of the three loss functions mentioned above. The training parameters were kept consistent across all experiments to ensure a fair comparison, with the hyperparameters summarized in Table 5.1.

5.4.2 Training Procedure

The training procedure for each model used the corresponding loss function to optimize the reconstruction quality, as detailed in Section 4.3. The loss functions were:

- **Binary Cross-Entropy Loss (BCE)**: Treats voxel occupancy as a classification task, applying equal weighting to all voxels.
- **Weighted Binary Cross-Entropy (WBCE)**: Adjusts class weights to compensate for imbalanced voxel occupancy distributions.
- **F1 Loss**: Optimizes for the F1-score directly, balancing precision and recall.

5.4.3 Evaluation Metrics

To assess model performance, we evaluated reconstruction quality using two key metrics:

- **F1-score**: Balances precision and recall, making it suitable for evaluating voxel classification.
- **Intersection over Union (IoU)**: A stricter metric that penalizes false positives and false negatives equally, offering a holistic view of volumetric accuracy.

Since the model outputs a probabilistic voxel grid, a threshold is required to determine whether a voxel is classified as occupied (1) or empty (0). We evaluated each loss function across multiple thresholds $(0.3, 0.4, 0.5, 0.55)$ to assess its robustness to confidence variations.

5.4.4 Results

Figure 5.4 shows the performance of different loss functions evaluated using F1-score and IoU, while Table 5.2 provides the quantitative breakdown.

Table 5.2: F1 and IoU Scores for Different Loss Functions at Various Thresholds

Threshold	**F1 Loss**		**WBCE Loss**		**BCE Loss**	
	F1	IoU	F1	IoU	F1	IoU
0.30	0.80	0.65	0.87	0.78	0.30	0.18
0.40	0.77	0.60	0.86	0.75	0.30	0.20
0.50	0.73	0.55	0.84	0.70	0.42	0.30
0.55	0.72	0.50	0.00	0.00	0.03	0.10

5.4.5 Discussion

1. Weighted BCE (WBCE) Loss Outperforms Other Losses: - WBCE achieves the highest F1-score (0.87 at $t = 0.3$) and IoU (0.78 at $t = 0.3$) across all thresholds, demonstrating that class balancing improves volumetric reconstruction. - The sharp drop at $t = 0.55$ suggests that WBCE models rely on probability scores closer to 0.5, making them more sensitive to strict thresholding.

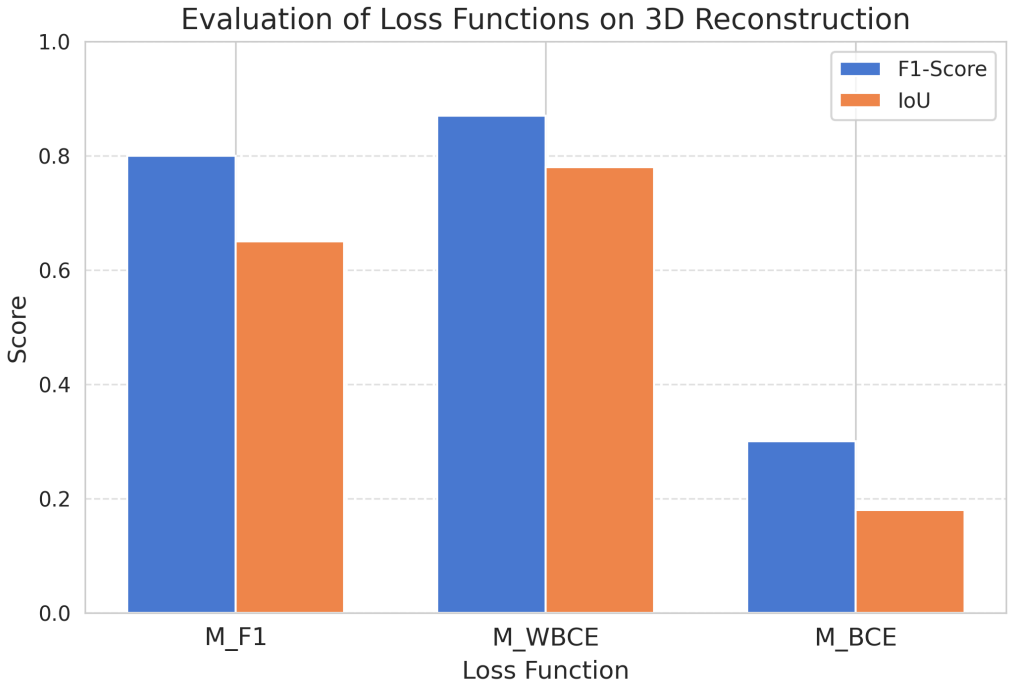

Figure 5.4: Evaluation of Different Loss Functions on 3D Reconstruction using F1-score and IoU.

2. F1 Loss Provides Consistent Performance: - The F1 loss consistently scores above 0.7 in F1-score and 0.5 in IoU, making it the most stable loss across thresholds. - Unlike WBCE, F1 loss does not experience a catastrophic drop at $t = 0.55$, indicating greater robustness to threshold variations.

3. BCE Loss Performs the Worst: - BCE struggles across all thresholds, with F1-scores remaining below 0.42. - IoU scores are significantly lower than both WBCE and F1 loss, reinforcing that treating all voxels equally results in poor reconstruction.

4. The Effect of Thresholding: - At lower thresholds ($t = 0.3, 0.4$), WBCE and F1 Loss maintain high scores, but BCE lags behind. - At higher thresholds ($t = 0.55$), WBCE drops to near zero, while F1 loss retains moderate performance.

5. Application-Specific Trade-offs: - If recall is more critical (e.g., branch detection in pruning analysis), lower thresholds with WBCE perform best. - If precision is the priority (e.g., avoiding false detections in fruit harvesting), F1 loss with a moderate threshold ($t = 0.5$) offers the best balance.

5.4.6 Conclusion

These results indicate that:

- **WBCE is the best-performing loss function**, but it is highly sensitive to thresholding.
- **F1 loss is more stable**, making it a good choice for applications requiring consistent performance.
- **BCE is the weakest loss function**, confirming that simple binary classification is insufficient for accurate reconstruction.
- Future research should explore hybrid loss functions, such as combining WBCE and F1 loss, to maximize performance.

Chapter 6

Conclusions

This thesis explored and compared different 3D reconstruction approaches for reconstructing tree structures in agricultural robotics, with a focus on handling occlusions. Through a comprehensive set of experiments, we investigated the effectiveness of Image to 3D and 3D to 3D reconstruction methods, analyzed model variations for occlusion handling, and evaluated the impact of different loss functions on reconstruction accuracy.

Our experiments demonstrated that 3D to 3D reconstruction significantly outperforms Image to 3D reconstruction in heavily occluded environments. The results indicate that 3D inputs provide structural priors that improve reconstruction quality, with 3D to 3D reconstruction maintaining higher F1-score and IoU across different confidence thresholds. In contrast, Image to 3D reconstruction is more sensitive to thresholding and benefits from lower confidence thresholds, which allow it to capture finer details at the cost of increased false positives.

The study of different model architectures showed that skip connections play a crucial role in improving reconstruction accuracy. Models with skip connections consistently outperformed variations without them, demonstrating better retention of structural features. Deeper models did not show improvements in reconstruction quality, suggesting that increased depth may lead to overfitting or ineffective feature extraction. Higher confidence thresholds negatively affected all models, leading to performance drops due to stricter voxel classification.

The evaluation of different loss functions revealed that Weighted Binary Cross-

Entropy (WBCE) achieves the highest F1-score and IoU at lower thresholds, highlighting the importance of class balancing in reconstruction tasks. F1 Loss provided the most stable performance across different thresholds, making it a good choice for applications requiring consistency. Binary Cross-Entropy (BCE) underperformed in all cases, reinforcing the need for loss functions that address class imbalance in voxel-based reconstructions. Higher thresholds negatively impacted WBCE, as its reliance on probability scores close to 0.5 made it more sensitive to strict classification.

These findings have significant implications for 3D reconstruction applications in robotics and automation. Accurate 3D models are essential for tasks that involve interacting with complex structures, and the results suggest that 3D to 3D reconstruction is the preferred method for occlusion-heavy environments. Hybrid approaches that combine 2D and 3D inputs could further enhance reconstruction quality by leveraging complementary strengths from both modalities. Exploring loss functions that dynamically adjust based on occlusion density and optimizing model architectures for structural retention could yield additional improvements.

Future research should focus on integrating multimodal data sources, such as depth and multispectral imaging, to enhance model robustness. Further work on occlusion-aware learning strategies and adaptive thresholding methods could refine voxel classification and improve reconstruction fidelity. Finally, real-world validation of these models in operational environments will be critical for transitioning from experimental results to practical applications.

This study has demonstrated that 3D to 3D reconstruction methods provide superior accuracy and robustness compared to Image to 3D methods. The choice of model architecture, loss function, and thresholding strategy all contribute to overall performance, and continued advancements in these areas will be crucial for improving 3D reconstruction in real-world applications.

Bibliography

[1] 3d tree growing software. https://www.thegrove3d.com/. Accessed: 2023-01-17. (document), 3.1, 3.1

[2] Nived Chebrolu, Philipp Lottes, Alexander Schaefer, Wera Winterhalter, Wolfram Burgard, and Cyrill Stachniss. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *The International Journal of Robotics Research*, 36(10):1045–1052, 2017. doi: 10.1177/0278364917720510. 2.2

[3] Christopher Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2.1.2

[4] Angela Dai, Charles R Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2.1.1

[5] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.1.2

[6] Harry Freeman, Eric Schneider, Chung Hee Kim, Moonyoung Lee, and George Kantor. 3d reconstruction-based seed counting of sorghum panicles for agricultural inspection. *arXiv*, arXiv:2211.07748, 2022. doi: 10.48550/arXiv.2211.07748. URL https://doi.org/10.48550/arXiv.2211.07748. arXiv:2211.07748 [cs.RO]. 2.2

[7] Xiao Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. *Proceedings of the IEEE International Conference on Computer Vision*, pages 85–93, 2017. 2.1.1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (document), 4.1, 4.1.1

[9] Chung Hee Kim and George Kantor. Occlusion reasoning for skeleton extraction

of self-occluded tree canopies. *arXiv preprint arXiv:2301.08387*, 2023. 1, 2.1.2

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 2.1.2

[11] Ciro Potena, Raghav Khanna, Juan Nieto, Roland Siegwart, Daniele Nardi, and Alberto Pretto. Agricolmap: Aerial-ground collaborative 3d mapping for precision farming. *IEEE Robotics and Automation Letters*, 4(2):1085–1092, 2019. doi: 10.1109/LRA.2019.2894468. 2.2

[12] Jeonghyun Sock, Kinam Kim, and Jan Kautz. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 37(6), 2018. 2.1.2

[13] Amy Tabb and Henry Medeiros. A robotic vision system to measure tree traits. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page Please add the page range, 2017. doi: 10.1109/IROS.2017.8206497. URL https://doi.org/10.48550/arXiv.1707.05368. 2.2

[14] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2017. 2.1.1

[15] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702*, 2021. 2.3