Laboratory Report

3D Reconstruction of Cassava Roots Using COLMAP

Erenus Yildiz

IAS-8, Forschungszentrum Jülich
Wilhelm-Johnen-Straße, 52428 Jülich, Germany
`e.yildiz@fz-juelich.de`

November 4, 2025

Contents

1 Introduction **4**

2 Objectives **4**

3 Materials and Equipment **4**

3.1 Hardware 4
3.2 Dataset 5
3.3 Software and Computational Resources 5
3.4 Biological Samples 5

4 Method **6**

4.1 Imaging Setup and Data Acquisition 7
4.2 Segmentation 8
4.2.1 Root Segmentation 9
4.2.2 Stem Segmentation 9
4.3 Structure-from-Motion 10
4.3.1 Sparse Reconstruction 10
4.3.2 Dense Reconstruction 11
4.4 Post-Processing 12
4.4.1 Outlier Removal 12
4.4.2 Backprojection 12
4.5 Measurements & Analysis 14
4.5.1 Mesh Generation from Point Clouds 14
4.5.2 Watertight Mesh Generation 15
4.5.3 Skeletonization 16
4.6 Compression and Archiving 17

5 Analysis **17**

5.1 High-Performance Computing Infrastructure 17
5.2 Computational Requirements 18
5.3 Key Technical Parameters 19
5.4 Reconstruction Statistics 19
5.5 Measurement Methods 21
5.5.1 Example Reconstruction 22
5.6 3D Model-based Measurements 23
5.6.1 Diameter 23
5.6.2 Root Lengths 25
5.7 Root Angles 27
5.7.1 Stem Length 27
5.7.2 Root Tips 29
5.7.3 Volume 29
5.7.4 3D Convex Hull 30

6 Discussion **30**

6.1 Reconstruction Performance 30
6.2 Segmentation and Measurement Quality 32
6.3 Sources of Error and Limitations 34
6.4 Measurement Method Limitations 34
6.5 Practical Considerations 35

7 Conclusion **36**

A Algorithmic Details **37**
A.1 Algorithm 1: HQ-SAM Two-Iteration Stem Segmentation 38
A.2 Algorithm 2: Three-Stage Outlier Removal 39
A.3 Algorithm 3: 3D Point Cloud Backprojection 40
A.4 Algorithm 4: Scaling Factor Computation 41
A.5 Algorithm 5: Teasar Skeletonization 42
A.6 Algorithm 6: Ray-Casting Diameter Measurement 43

B Pipeline Parameters **44**

1 Introduction

Cassava (Manihot esculenta) is an important crop for food security in tropical and subtropical regions, with roots containing up to 85% starch on a dry weight basis [25]. Understanding root system architecture is essential for breeding programs aimed at improving yield and stress tolerance [1].

Traditional 2D imaging methods for root phenotyping have limitations in capturing three-dimensional root structures, leading to incomplete trait measurements [28, 8]. While advanced 3D methods like DIRT/3D 2.0 exist [14], they require specialized equipment that may not be accessible to all research facilities.

This laboratory work implemented a cost-effective 3D reconstruction pipeline using standard DSLR cameras and open-source software (REMBG, COLMAP) to extract morphological traits from cassava roots. The pipeline combined deep learning-based segmentation with structure-from-motion techniques to analyze 1039 cassava root system acquisitions from plants aged 5-10 weeks collected at the Rayong Field Crops Research Center in Thailand, achieving 62% reconstruction success (644 successfully reconstructed roots).

2 Objectives

The primary objectives of this laboratory work were to:

1. Implement a 3D reconstruction pipeline for cassava roots using accessible equipment (standard DSLR cameras) and open-source software
2. Extract quantitative morphological measurements including volume, diameter, root angles, root lengths, and tip counts from reconstructed 3D models
3. Evaluate reconstruction success rates and identify failure modes across 1039 root acquisitions aged 5-10 weeks
4. Assess the feasibility and accuracy of using structure-from-motion techniques for large-scale plant phenotyping applications
5. Demonstrate scalability through parallel processing on high-performance computing infrastructure

3 Materials and Equipment

3.1 Hardware

The imaging system consisted of:

- Two Canon 80D DSLR cameras with 24.2-megapixel APS-C CMOS sensors
- Canon EF 14mm f/2.8L II USM lenses (focal length: 14 mm)
- Stereo rig with vertical baseline distance of 30 cm
- Rotating platform enabling 360° rotation in 6 seconds
- QR codes (5 × 5 cm) for plant identification and as calibration reference
- Closed imaging environment to control lighting conditions

The cameras were positioned at 110.373 cm from the cassava plant, capturing video sequences at 1920×1080 resolution.

3.2 Dataset

This study utilized a cassava root dataset from the Rayong Field Crops Research Center in Thailand. The dataset comprised 1039 root system acquisitions captured as stereo video sequences (6-second duration, 360° rotation) at 1920×1080 resolution. Frame extraction yielded over 300 images per root, which were processed through the segmentation and COLMAP stages before phenotyping (Section 2). The acquisitions spanned cassava plants aged 5 to 10 weeks, capturing critical early developmental stages of storage root formation. As detailed in Section 5.4, of the 1039 acquisitions, 644 were successfully reconstructed (62%), 156 were classified as fibrous and not attempted (15%), and 239 failed reconstruction (23%).

3.3 Software and Computational Resources

- **REMBG** (Remove Background) Python package [7] with BiRefNet-DIS [30] for background removal and root segmentation
- **HQ-SAM** (High-Quality Segment Anything Model) [11, 12] for stem segmentation
- **COLMAP** framework [24, 23] for structure-from-motion reconstruction (sparse and dense)
- **Open3D** library [31] for point cloud processing, filtering, and alpha shape surface reconstruction
- **Blender** [3] (version 2.93+) with BMesh library for watertight mesh generation and volume computation
- **PyEmbree** [27] with ray-casting for mesh analysis and diameter measurements
- **NumPy** [9] for numerical operations and backprojection calculations
- **Trimesh** [5] for mesh loading, processing, convex hull computation, and repair operations
- **Skeletor** [22] implementing the Teasar algorithm [21] for 3D skeletonization
- **JURECA supercomputer** (Forschungszentrum Jülich) with SLURM job scheduler [29] for parallel processing
- **RAM workspace** utilizing tmpfs (/dev/shm) for 10-100× faster I/O performance
- Cluster module configuration (`3dreconst_venv/modules.sh`) to provision GPU-accelerated COLMAP, PyTorch, and Embree environments on JURECA

3.4 Biological Samples

- 1039 cassava root system acquisitions from multiple genotypes
- Age range: 5-10 weeks (storage root formation period)
- Source: Rayong Field Crops Research Center, Thailand
- Roots were excavated and imaged ex situ

4 Method

Cassava forms a root system in which numerous fibrous roots emerge from the stem, a subset of these undergoes secondary thickening to become storage roots. For the purposes of explaining the root structure, anatomical landmarks illustrated in Fig. 1 are chosen. *Storage roots* are thick, starch-accumulating organs that develop; the remaining fine roots (if any) are referred to as *fibrous* roots. There are traits such as *root angle*, *root system width* and *root system length*, see Figure 1.

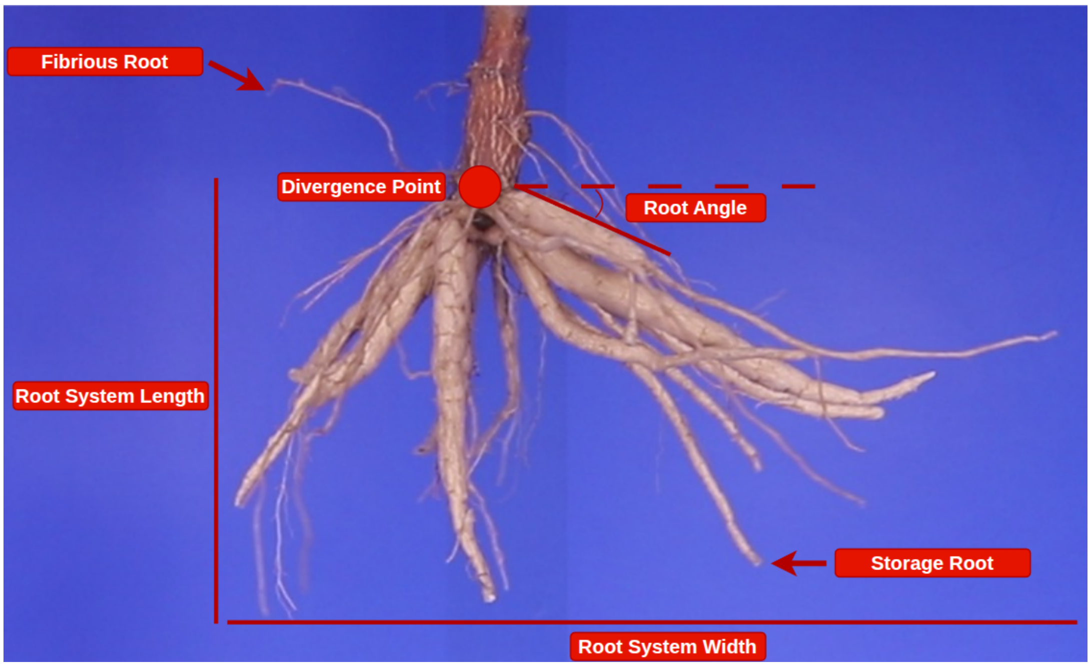


Figure 1: Manual markings of significant root traits of a 9-week-old system within the dataset.

We developed a multi-stage pipeline to reconstruct, visualize, and analyze cassava roots grown in Rayong Field Crops Research Center, Thailand. The pipeline integrates deep learning for segmentation, structure-from-motion for 3D modeling, and post-processing algorithms to refine the generated models. Figure 2 shows the overall architecture of our pipeline.

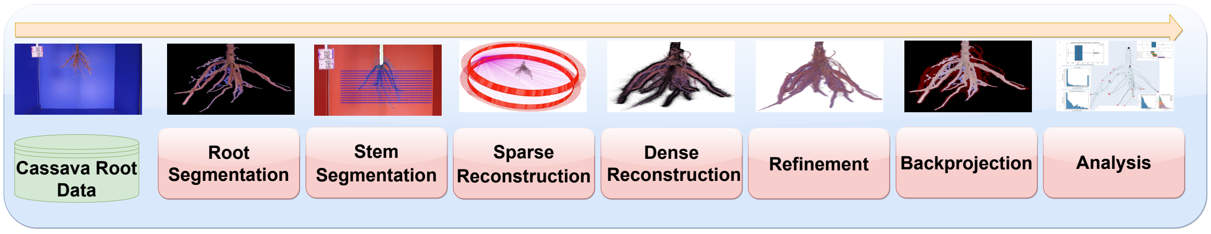


Figure 2: Pipeline architecture showing the major processing stages from video acquisition to trait extraction.

The pipeline began with stereo video acquisition of a single root. Frame extraction yielded over 300 images per root. We used REMBG for root segmentation, isolating roots from backgrounds. HQ-SAM [11] was then applied to separate stem regions from roots using prompt-based segmentation with algorithmically-generated seed points. COLMAP [24, 23] was used for structure-from-motion reconstruction to generate sparse and dense 3D models. We verified model accuracy through backprojection, eliminated outliers, and refined models using 3D filters including radius outlier removal. The pipeline implemented skeletonization to extract morphological structure, identifying branches, divergence points, and root tips. For measurements, we used COLMAP calibration data and known setup parameters (30 cm camera baseline). The scaling factor to convert pixel measurements to centimeters was computed from the camera positions, utilizing the known 30 cm vertical separation between cameras. Extracted metrics included diameter, root length, root width, stem length, volume, and convex hull measurements. All reconstruction data were saved in unified files (JSON, CSV, Excel) for analysis. Processing utilized RAM-based workspaces (tmpfs on /dev/shm) for 10-100× faster I/O, with incremental synchronization to disk after each stage to ensure data integrity.

4.1 Imaging Setup and Data Acquisition

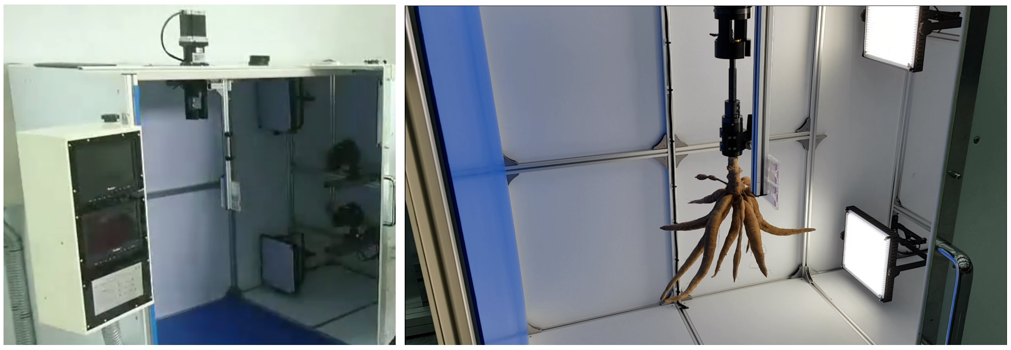


Figure 3: Setup features a closed environment with a vertically-aligned stereo imaging system, and a holder for the cassava plant, allowing it to rotate 360 degrees in 6 seconds.

To achieve high-resolution imaging required for accurate 3D reconstruction, our study utilized an stereo camera configuration comprising two Canon 80D DSLR[1] cameras, each equipped with a Canon EF[2]14mm f/2.8L II USM[3] lens. The choice of the Canon 80D, with its 24.2-megapixel APS-C[4] CMOS[5] sensor, was pivotal in capturing the details of cassava roots, critical for the subsequent reconstruction process. The cameras were arranged in a stereo setup, with a baseline distance of 30 cm between the two vertically-aligned units. This arrangement facilitated a comprehensive 360-degree capture of the cassava plants as video sequences, achieved through a 6-second rotation for each plant, thus ensuring a detailed depiction of the root's spatial arrangement and growth. Each video sequence, with a resolution of 1920x1080, was processed to extract over 300 images per root. Positioned at a distance of 110.373 cm from the targeted cassava plant, the setup was optimized to capture the entirety of the extracted root system within the frame, while maintaining the high enough resolution criteria essential for the later stages of 3D model reconstruction. To aid in the identification and cataloging of the individual roots within the experimental dataset, a QR code measuring 5 $\times$ 5 cm was placed adjacent to each root. This QR code served dual purposes: it facilitated plant identification for biologists, and provided a calibration reference for computer vision algorithms by virtue of its known dimensions and camera distance. The design of the imaging setup, as illustrated in Fig. 3, was instrumental in the acquisition of a comprehensive dataset, capturing the complex spatial organization of the cassava roots. A detailed diagram is given in Fig. 4 for exact measurements between the equipment in the setup.

[1]Digital Single Lens Reflex
[2]Electronic Focus
[3]Ultrasonic Motor
[4]Advanced Photo System type-C
[5]Complementary Metal Oxide Semiconductor

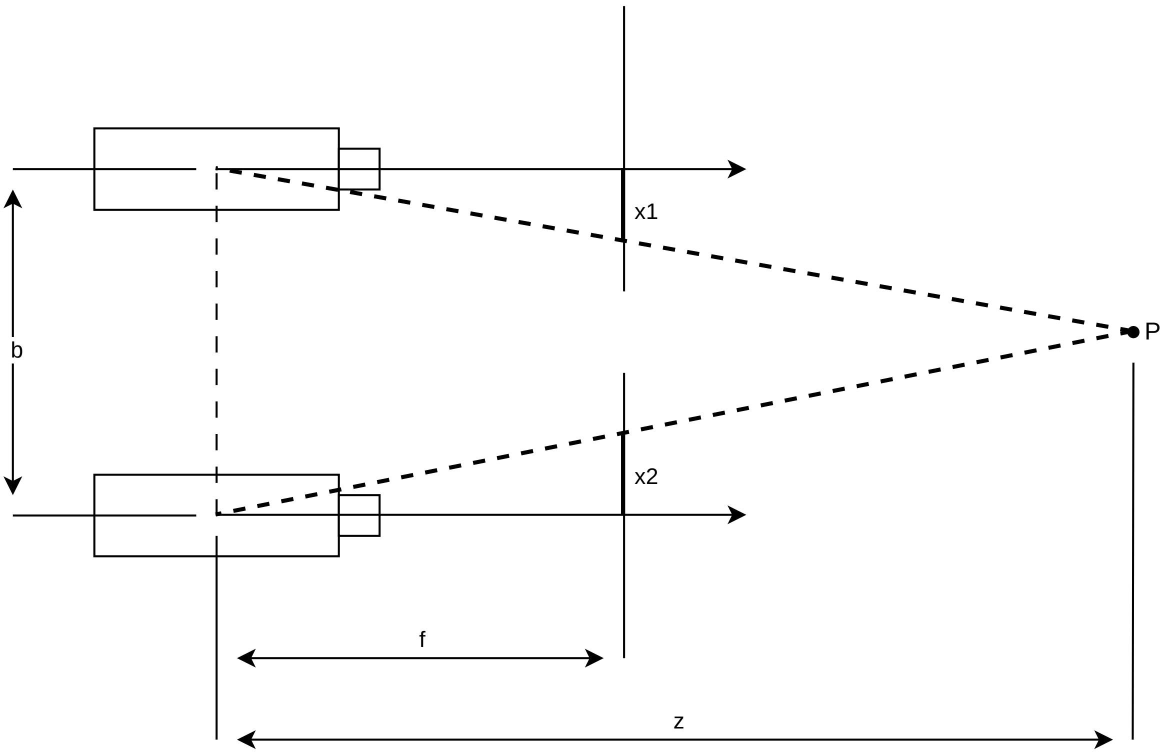


Figure 4: The stereo imaging setup utilizes two Canon 80D cameras equipped with Canon EF 14mm/2.8 L II USM lenses. The left camera is positioned lower, and the right camera is positioned higher with a vertical baseline distance (b) of 30 cm between them. The focal length (f) of the lenses is 14 mm. The distance (d) from the cameras to the cassava plant (P) is measured at 110.373 cm.

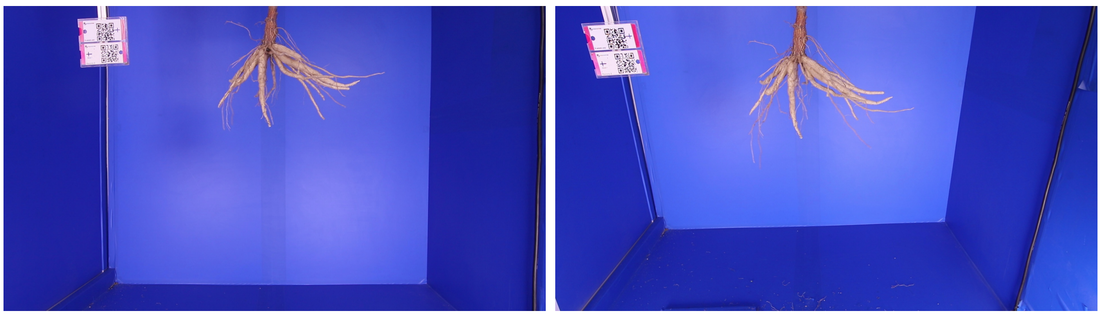


Figure 5: Sample stereo image pair from the 9-week cassava root example reconstruction (frame 0 from each camera). Left: lower camera view. Right: upper camera view. The two cameras are mounted with a 30 cm vertical baseline, capturing synchronized frames during 360° rotation.

This work only considers the roots that are 5 to 10 weeks old, as the storage root formation takes place during this period. Ultimately, we base our measurements in this work on the 644 successfully reconstructed root systems from the 1039 acquisitions.

4.2 Segmentation

Segmentation played a crucial role in various computer vision tasks, especially in agricultural applications where precise identification and isolation of plant structures are essential for further analysis [26]. In our reconstruction pipeline, segmentation was employed to isolate different parts of the cassava plant, namely the roots and stems, from background clutter and noise present in the images. Accurate segmentation enabled subsequent analysis steps, such as morphological measurements, to be performed reliably.

4.2.1 Root Segmentation

We used REMBG [7] with BiRefNet-DIS model [30] for root segmentation. REMBG is an open-source background removal tool that employs the Bilateral Reference Network (BiRefNet) architecture for high-resolution dichotomous image segmentation. BiRefNet-DIS (Dichotomous Image Segmentation) is specifically designed for precise foreground-background separation tasks. This approach was suitable for our application as it handles complex root structures without requiring domain-specific training data.

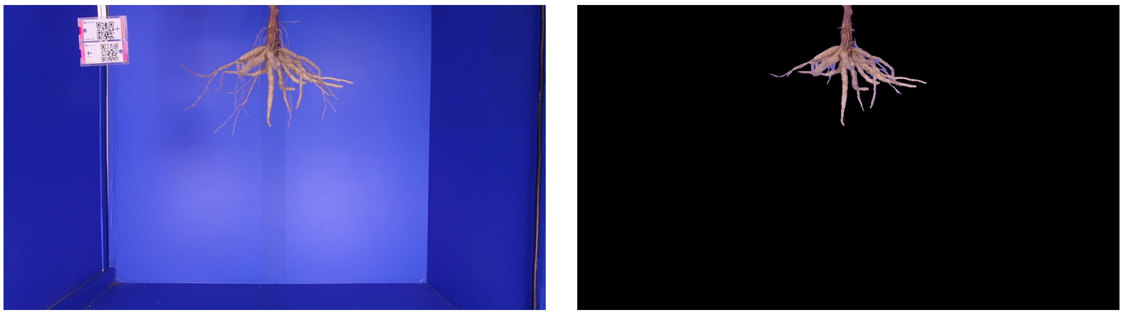

(a) Input image (b) Segmented output

Figure 6: Root segmentation using REMBG with BiRefNet-DIS model. (a) Original input image captured during 360° rotation showing cassava root with varying lighting and complex root structures. (b) Binary mask output after segmentation, isolating the root from background. The model successfully handles challenging conditions including shadows, texture variations, and thin root branches without requiring domain-specific training. Small artifacts in low-contrast regions are addressed in subsequent 3D reconstruction filtering stages.

We processed each frame independently to generate binary masks separating roots from backgrounds. Over 300 images per root were batch-processed using REMBG with GPU acceleration. To optimize computational efficiency and focus on the plant region, each frame was first cropped to a region of interest (ROI: x = 350-1720 pixels, y = 0-1030 pixels) that excluded the QR code calibration tag and irrelevant background areas before segmentation. After BiRefNet-DIS segmentation, the masks were restored to the original frame dimensions with zero-padded borders.

Post-processing operations were applied to enhance mask quality and remove noise artifacts. A luminance-based filter suppressed low-intensity background pixels by computing the mean pixel intensity across color channels and applying an adaptive threshold (30 + filter strength). For plants with predominantly thick roots, a stronger filter (strength = 20) was used to remove thin fibrous structures, while plants with mixed root types used a gentler filter (strength = 5) to preserve finer details. This adaptive filtering approach helped eliminate residual background noise and small segmentation artifacts while maintaining the integrity of true root structures. Some artifacts remained on thin root branches or low-contrast areas, but these were addressed in subsequent 3D reconstruction stages through statistical and radius-based outlier removal algorithms.

4.2.2 Stem Segmentation

Following root segmentation, we applied stem segmentation using the HQ-SAM (High-Quality Segment Anything Model) [11, 12] framework. HQ-SAM is a prompt-based segmentation model that uses the Vision Transformer (ViT-L) architecture and requires seed points to guide the segmentation process. Unlike traditional semantic segmentation that classifies every pixel, HQ-SAM uses algorithmically-generated prompts (positive and negative points) to iteratively refine segmentation boundaries, making it particularly effective for challenging stem-root separation in complex cassava root structures.

Our implementation employs a two-iteration refinement strategy (Algorithm 1 in Appendix A). The first iteration provides coarse segmentation using a single seed point at the top of the root. The second iteration refines boundaries using dense positive prompts in the stem region (y: 0-150) and negative prompts in the root region (x: 500-1500, y: 220-500), guided by logits from the first iteration. Connected component analysis isolates the final stem mask by selecting only the component containing the original seed point, eliminating false positives from segmentation noise.

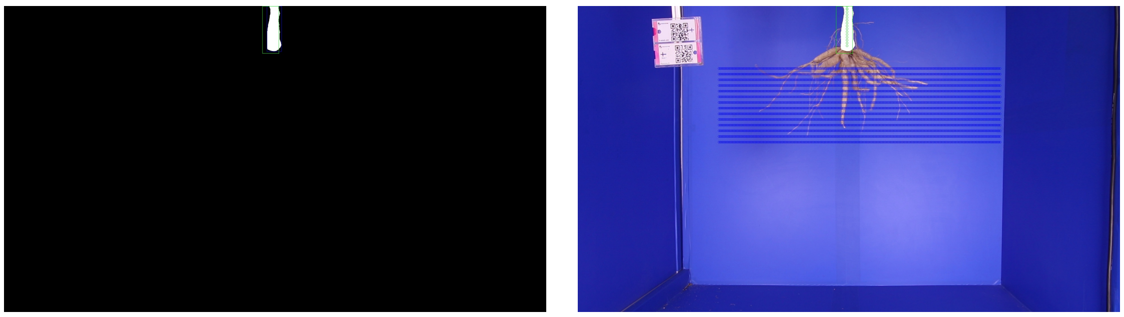

(a) Stem segmentation mask (b) Debug visualization with seed points

Figure 7: Stem segmentation using HQ-SAM with two-iteration prompt-based approach. (a) Binary stem mask output showing the isolated stem region with bounding boxes (blue: individual frame detection, green: median bounding box across all frames). (b) Debug visualization showing the seed point strategy: green rectangle marks the initial seed point at row 10, green circles indicate positive prompts distributed along the stem region (y: 0-150), and red circles -if any- show negative prompts placed in the root region to refine the stem-root boundary. The median bounding box (green) is used to measure stem length in pixels for subsequent conversion to physical units.

After HQ-SAM segmentation, we computed bounding boxes for each frame by identifying pixels in the stem masks. The stem length was determined by calculating the median bounding box across all frames and measuring the distance between the top and bottom boundaries along the y-axis in pixels. This pixel measurement was later converted to physical units using the camera-baseline calibration factor derived from COLMAP's camera positions.

4.3 Structure-from-Motion

We used COLMAP [24, 23] for structure-from-motion reconstruction. COLMAP is a widely-used software for 3D reconstruction from 2D images, employed across disciplines including archaeology [17], architecture [4], environmental science [6], and agriculture [19, 15, 20].

4.3.1 Sparse Reconstruction

We used COLMAP's sparse reconstruction phase to construct a preliminary 3D point cloud. This phase includes feature extraction, feature matching, and camera pose estimation from the series of 2D images. The segmented images from the previous stage were analyzed to identify unique features and align them across multiple views.

The sparse reconstruction pipeline consists of three sequential steps:

1. **Feature Extraction:** SIFT features are extracted from all masked images using GPU acceleration. We configured COLMAP to use a PINHOLE camera model. Parameters

`estimate_affine_shape=true` and `domain_size_pooling=true` improve feature quality. The pipeline assumes a single camera model (two identical cameras), but creates separate camera entries for the upper and lower cameras to handle potential slight calibration differences.

2. **Feature Matching:** Exhaustive feature matching was performed across all image pairs using GPU-accelerated SIFT matching with guided matching enabled. This ensured robust correspondences despite the large baseline between stereo cameras and rotation-induced viewpoint changes.

3. **Incremental Mapping:** COLMAP's mapper performed incremental structure-from-motion, estimating camera poses and triangulating 3D points. We used "high quality" reconstruction settings with aggressive bundle adjustment parameters: 300 iterations for both local and global bundle adjustment, minimum 200 inliers for initialization, and strict reprojection error filtering (max 2 pixels). The mapper allowed up to 500 registration trials per image to handle challenging viewpoints. These parameters prioritized accuracy over speed, critical for capturing fine root structures.

The sparse reconstruction process was iterative and non-deterministic due to SIFT's stochastic feature detection. We implemented up to 20 reconstruction attempts per root, continuing until more than 50% of images were successfully registered. Figure 8 shows the aligned images visualized through COLMAP's GUI.

After successful mapping, COLMAP's `image_undistorter` prepared rectified images for dense reconstruction, removing lens distortion and creating an undistorted image set. The sparse point cloud was exported to PLY format for downstream analysis.

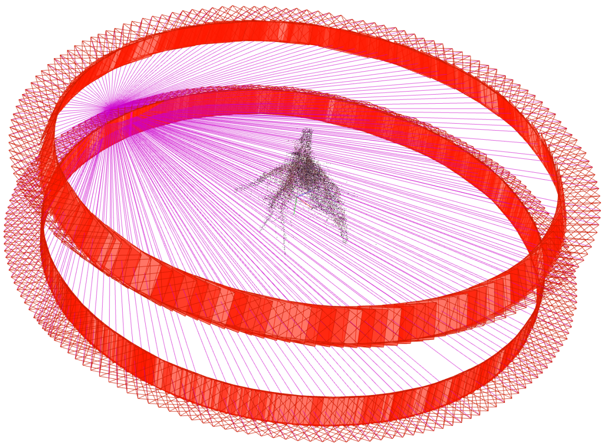


Figure 8: Sparse reconstruction visualized through COLMAP's GUI, demonstrating how a 3D point cloud emerges from aligning features across a collection of 2D images. Each red rectangle symbolizes an individual image contributing to the model.

4.3.2 Dense Reconstruction

Dense reconstruction generated a highly detailed 3D point cloud from the sparse model by estimating depth for every pixel. COLMAP implemented this through a two-stage process: depth map computation followed by fusion.

PatchMatchStereo: COLMAP's multi-view stereo algorithm computed dense depth maps for each image. The algorithm used patch-based matching with photometric consistency checks across multiple views. We disabled geometric consistency filtering (`geom_consistency=false`) to preserve fine details in thin root branches, accepting slightly increased noise in exchange for better detail preservation. The algorithm ran with GPU acceleration to handle the computational demands of processing over 300 high-resolution images per root.

StereoFusion: Following depth map estimation, COLMAP's fusion step aggregated depth information from all views into a single coherent point cloud. We configured fusion with `min_num_pixels=2`, requiring each 3D point to be visible in at least 2 views. This low threshold ensured reconstruction of thin root structures that appeared in only a limited number of viewpoints, while still providing stereo verification. The fusion process handled conflicts between overlapping depth maps through weighted averaging based on photometric consistency scores.

The dense reconstruction stage significantly increased point cloud density compared to sparse reconstruction, capturing intricate root surface details, branch bifurcations, and texture variations that were essential for accurate morphological measurements. However, this density came with increased noise and background artifacts, which we addressed in subsequent post-processing stages.

4.4 Post-Processing

We refined the 3D point cloud from dense reconstruction through a series of filtering operations to enhance data quality.

4.4.1 Outlier Removal

The dense point cloud from COLMAP reconstruction contains significant noise, background artifacts, and outliers that must be removed before morphological analysis. We apply a three-stage filtering pipeline using Open3D [31] (Algorithm 2 in Appendix A):

1. **Color Intensity Filter:** Leveraged cassava root coloration (white/cream) versus black background. Points were retained if red-green average exceeded 0.5 or the blue channel exceeded 0.25, preserving typical root colors and shadow regions while eliminating background pixels.

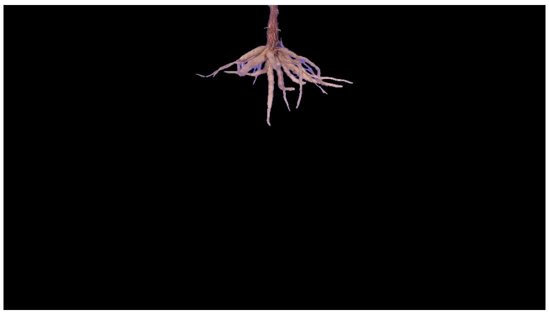
2. **Statistical Outlier Removal:** Computed mean distance to 20 nearest neighbors for each point. Points whose mean distance exceeded 2 standard deviations from the global distribution were classified as outliers. This eliminated isolated noise points and floating artifacts from reconstruction errors.

3. **Radius Outlier Removal:** Removed points with fewer than 50 neighbors within 0.02 COLMAP units ($\approx$0.6 cm). This aggressive filter eliminated small disconnected clusters while preserving dense root structure.

The three filters are applied sequentially with $O(n \log n)$ complexity, dominated by KD-tree spatial indexing operations. Typical filtering reduces point cloud size by 20-40%, with majority of removed points being background noise rather than root structure.

4.4.2 Backprojection

Backprojection is a technique used to project 3D points onto 2D image planes using camera parameters and scene geometry [10, 13, 2, 18]. We used backprojection to align 3D models with their corresponding images and validate reconstruction quality through spatial consistency checks.


(a) Segmented input mask

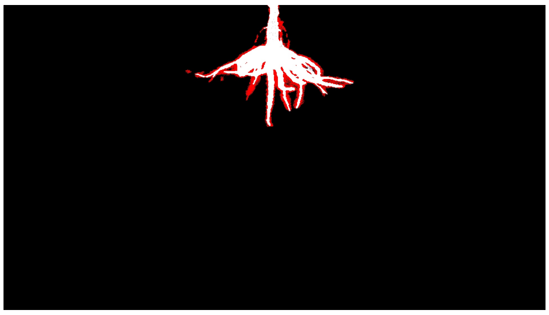

(b) Backprojected 3D points

Figure 9: Backprojection validation and spatial filtering process. (a) Binary segmented mask showing the root structure (white foreground) and background (black regions). (b) 3D point cloud backprojected onto the 2D image plane, visualized as colored points. Each point's color indicates its spatial coordinates in 3D space. Points projecting onto black background regions are identified across all frames and flagged for removal (points appearing in black regions on more than 20 frames are filtered out). This spatial filtering eliminates reconstruction artifacts and ensures the final 3D model contains only points corresponding to the actual root structure.

Mathematical Foundation: The backprojection process relies on the pinhole camera model, where a 3D point $\mathbf{X} = [X, Y, Z]^T$ in world coordinates is projected onto 2D image coordinates $\mathbf{x} = [u, v]^T$ through the projection matrix P:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where K is the 3×3 camera intrinsic matrix containing focal lengths (f_x, f_y) and principal point (c_x, c_y), R is the 3×3 rotation matrix describing camera orientation, t is the 3×1 translation vector describing camera position, and λ is a scale factor. The projection operates in homogeneous coordinates, requiring conversion back to Euclidean coordinates by dividing by the third coordinate.

Camera Parameter Extraction: COLMAP stores camera parameters in binary format. Camera intrinsics are read from `cameras.bin`, providing focal lengths and principal points for each camera. Extrinsic parameters are extracted from `images.bin`, which stores camera poses as quaternions $\mathbf{q} = [q_0, q_1, q_2, q_3]$ and translation vectors $\mathbf{t} = [t_x, t_y, t_z]$. Quaternions are converted to 3×3 rotation matrices using:

$$R = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix} \tag{2}$$

This rotation matrix transforms points from world coordinates to camera coordinates. The projection matrix is then constructed as $P = K[R|t]$, combining intrinsics and extrinsics into a single 3×4 matrix.

Implementation: The complete backprojection and spatial filtering procedure (Algorithm 3 in Appendix A) operates in three phases. We used Open3D [31] for point cloud operations and NumPy [9] for efficient matrix computations.

Phase 1: Load Camera Parameters extracts projection matrices from COLMAP's binary format. Camera intrinsics (K) come from `cameras.bin`, while extrinsics (quaternion $\mathbf{q}$ and translation $\mathbf{t}$) come from `images.bin`. Quaternions are converted to rotation matrices using the standard formula, then combined with intrinsics to form the 3×4 projection matrix $P = K[R|t]$.

Phase 2: Project Points and Retrieve Colors projects each 3D point onto all image planes using homogeneous coordinates: $\lambda \mathbf{x}_h = P \cdot \mathbf{X}_h$. For visible points (within image bounds), we retrieve the RGB color from the original image and check the segmentation mask. Points projecting onto black background regions ($M_i[v, u] == 0$) are flagged by incrementing a counter.

Phase 3: Spatial Filtering removes reconstruction artifacts using multi-view consistency. Points appearing in black regions on more than 20 frames are discarded, as they likely represent spurious reconstructions rather than actual root structure. This threshold was empirically determined to balance artifact removal with preservation of genuine geometry.

Parallelization: The backprojection process is embarrassingly parallel across images, as each frame can be processed independently. We implemented multiprocessing with a worker pool to distribute frames across CPU cores, achieving near-linear speedup with the number of available cores. Each worker handles projection and color retrieval for a subset of frames, with results aggregated in the main process for final spatial filtering.

4.5 Measurements & Analysis

The final stages of the pipeline convert 3D measurements from COLMAP's arbitrary coordinate system to real-world units (centimeters). COLMAP's structure-from-motion reconstruction operates in an arbitrary metric scale determined by the initialization process. To enable meaningful phenotypic measurements, we compute a scaling factor from the known physical camera setup geometry: the two cameras are mounted with a fixed 30 cm vertical baseline separation.

The scaling factor computation (Algorithm 4 in Appendix A) operates in five steps:

1. **Load Camera Poses:** Extract quaternions and translation vectors from COLMAP's `images.bin`, separating top and bottom camera frames.

2. **Compute Mean Positions:** Calculate mean camera centers $\mathbf{C} = -R^T \mathbf{t}$ for each camera across all frames. Averaging reduces per-frame pose estimation errors.

3. **Compute Baseline in COLMAP Units:** Calculate Euclidean distance $b_{\text{COLMAP}} = \|\mathbf{p}_{\text{top}} - \mathbf{p}_{\text{bot}}\|$ between mean camera positions.

4. **Compute Scaling Factor:** $s = b_{\text{real}}/b_{\text{COLMAP}} = 30\ \text{cm}/b_{\text{COLMAP}}$ converts COLMAP units to centimeters.

5. **Apply Scaling:** Linear measurements scale by s, areas by s^2, volumes by s^3.

Physical Setup: The 30 cm baseline is the known vertical separation between the two Canon 80D cameras mounted on the rotating platform. This fixed physical distance serves as the ground truth for metric scaling, enabling conversion from COLMAP's arbitrary coordinate system to real-world centimeters.

QR Codes: The QR codes (5 × 5 cm) visible in frames serve dual purposes: plant identification and cataloging for biologists, and as a calibration reference for computer vision due to their known dimensions and fixed camera distance. However, for final metric scaling, the camera baseline method provides more robust results as it leverages all frames rather than depending on QR code detection in specific views.

4.5.1 Mesh Generation from Point Clouds

Following backprojection and filtering, the refined point cloud was converted into a triangular mesh surface using the alpha shape algorithm. Alpha shapes provided a geometrically intuitive method for surface reconstruction, defining the shape as the complement of a union of balls with radius $1/\alpha$.

Alpha Shapes Geometric Intuition: The alpha shape algorithm can be visualized using a "ball-rolling" analogy: imagine rolling a sphere of radius $r = 1/\alpha$ around the point cloud.

The alpha shape was the boundary formed by the exterior surface traced by this ball as it rolled around the points, unable to penetrate the interior. Mathematically, alpha shapes are a subset of the Delaunay triangulation: $\text{AlphaShape}(\alpha) \subseteq \text{Delaunay}(P)$. For any simplex (edge, triangle, tetrahedron) in the Delaunay triangulation, it belonged to the alpha shape if there existed an empty sphere of radius $1/\alpha$ passing through the simplex's vertices.

Parameter Effects: The alpha parameter α controlled reconstruction detail level:

- $\alpha \to 0$ (large radius): Approached the convex hull, producing a smooth outer envelope that filled all concavities. Lost fine details and thin structures.
- $\alpha = 0.025$ (our choice): Balanced detail preservation with noise suppression. Captured thin root branches while filtering reconstruction artifacts.
- $\alpha \to \infty$ (small radius): Produced highly detailed meshes following every point cluster. Was sensitive to noise and could create spurious surfaces around isolated outliers.

The mesh generation process employed Open3D's alpha shape implementation with $\alpha = 0.025$. This value was empirically selected to balance detail preservation with noise suppression. The alpha shape algorithm was preferred over Poisson surface reconstruction for this application because it:

- Preserves fine root structures without over-smoothing
- Does not require normal vector estimation (Poisson requires oriented normals)
- Handles sparse regions better (important for thin root branches)
- Produces meshes that closely follow the actual point cloud geometry
- Provides explicit control over detail level through a single intuitive parameter

After initial mesh generation, a smoothing operation was applied using Laplacian smoothing with 3 iterations to reduce surface noise while maintaining overall shape. Finally, disconnected triangle clusters were identified and removed, keeping only the largest connected component. This eliminated floating artifacts and isolated noise points that survived earlier filtering stages.

4.5.2 Watertight Mesh Generation

Volume calculations and certain geometric analyses required watertight (manifold) meshes, where every edge was shared by exactly two faces and the surface was closed without holes. The alpha shape meshes generated in the previous step often contained small holes, non-manifold edges, and other topological defects that prevented accurate volume computation.

To address this, we employed Blender's [3] 3D printing utilities to repair and create manifold meshes. The process involved:

1. Importing the alpha shape mesh into Blender
2. Applying Blender's "Make Manifold" operation, which filled holes, removed duplicate vertices, and fixed non-manifold geometry
3. Exporting the repaired mesh in PLY format with ASCII encoding

This repair process was essential for volumetric measurements, as Blender's BMesh library (used for volume calculation) required closed, orientable surfaces. Two versions of the manifold mesh were generated: one with the stem removed and one with the stem intact (denoted as "WS" for "with stem"), enabling comparative analyses of root system properties with and without the stem contribution.

4.5.3 Skeletonization

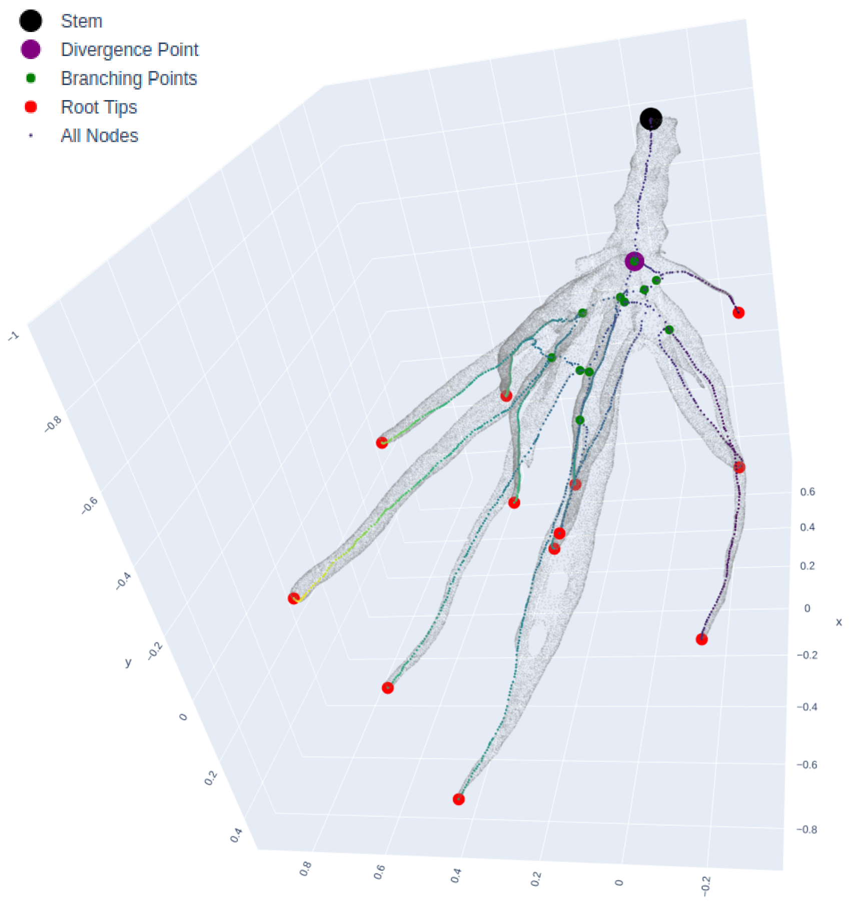


Figure 10: Skeleton of a plant extracted using the Teasar algorithm showing root tips, stem, and branching points.

The watertight mesh is skeletonized using the Teasar algorithm [21] (Tree-structure Extraction Algorithm for Segmented Anatomy Reconstruction), producing a hierarchical tree structure distinguishing root, stem, and branching nodes (Figure 10). Teasar is a wave-propagation-based skeletonization method that iteratively extracts the longest paths through mesh geometry, building a skeletal representation particularly well-suited for tree-like structures such as root systems.

The complete algorithm (Algorithm 5 in Appendix A) operates in three phases with $O(I \cdot |V| \log |V|)$ complexity, where I is the number of iterations (typically proportional to the number of branches):

Phase 1: Initialization identifies the root node (vertex with minimum y-coordinate) and initializes tracking sets for covered and uncovered vertices.

Phase 2: Iterative Path Extraction repeats until all vertices are covered: (2.1) Wave propagation uses Dijkstra's algorithm from all current skeletal nodes to compute shortest paths to all vertices, (2.2) Farthest point selection finds the uncovered vertex with maximum distance, (2.3) Path backtracing reconstructs the longest path from the farthest point back to the skeleton, (2.4) The path is added to the skeletal structure with proper edge connectivity, and (2.5) Invalidation radius marks nearby vertices as covered to prevent redundant branches.

Phase 3: Tree Construction builds the final hierarchical tree structure with labeled nodes: root (stem base), branch points (junctions), and tips (terminals).

Implementation: We employ the Skeletor library [22], which implements Teasar using scipy's sparse graph algorithms with priority queue optimizations. The invalidation radius was set to 2% of the mesh bounding box diagonal ($r_{inv} = 0.02 \times \sqrt{w^2 + h^2 + d^2}$), empirically balancing detail preservation with computational efficiency. This captured 90-95% of visible root structures while maintaining processing times under 5 minutes for typical meshes (10,000-50,000 vertices).

The skeletonized structure enabled computation of multiple metrics: bounding box dimensions, volumes, mass, convex hull volumes, moment of inertia, root diameters, angles, stem length, and root tip lengths. We also calculated average and standard deviation of diameters around branching points and measured angles between the stem, divergence points, and root tips. The skeletal tree provides the topological foundation for many measurements: diameter sampling regions are defined relative to skeleton branches, root lengths follow skeletal edge paths, and angles are computed between skeletal branch directions.

All calculated metrics were saved in multiple formats (JSON, CSV, Excel) for analysis. The methods for measuring specific traits are outlined below.

4.6 Compression and Archiving

The final pipeline stage (Stage 8) packaged all outputs into a compressed archive for long-term storage and data sharing. This stage performed the following operations:

1. **Final synchronization:** All results from the RAM workspace (located in tmpfs at /dev/shm) were synchronized to persistent disk storage, ensuring no data loss
2. **Archive creation:** A ZIP archive was created containing all intermediate and final outputs including images, masks, 3D models, phenotyping metrics, and execution logs
3. **Workspace cleanup:** The RAM workspace was removed to free system memory for subsequent plant reconstructions

The use of RAM-based workspaces throughout the pipeline provided 10-100$\times$ faster I/O performance compared to disk-based processing. Each stage wrote outputs to the RAM workspace (typically 2-4 GB per plant), and incremental synchronization after each stage ensured data persistence even if processing was interrupted. This strategy significantly reduced total processing time while maintaining data integrity through the non-destructive merge synchronization approach implemented after the workspace management bug fix in October 2024.

5 Analysis

In this section, we conduct a detailed analysis of cassava roots from our study of 1039 acquisitions. The dataset spans weeks 5 through 10 of development (Section 3.2) and therefore captures storage-root formation across critical early growth phases. As detailed in Figure 13, 644 roots (62%) were successfully reconstructed, providing a robust foundation for examining developmental patterns and reconstruction pipeline performance. We examine reconstruction metadata to characterize typical failure cases and processing behavior, offering insights into how segmentation, reconstruction, and phenotyping stages interact across successful and unsuccessful runs.

5.1 High-Performance Computing Infrastructure

To optimize computational efficiency and scalability, parallel processing techniques were implemented within the HPC environment, leveraging distributed computing systems like the JURECA supercomputer. This strategy enabled the simultaneous execution of computationally intensive tasks, significantly accelerating the 3D reconstruction process.

Figure 11: JURECA-DC supercomputer, found in JSC at Forschungszentrum Juelich[6]

The coordination of parallel processing tasks was facilitated by the SLURM [29] job scheduler. By configuring SLURM parameters, a singular job adeptly managed the simultaneous reconstruction of multiple plants, optimizing the allocation of nodes, tasks per node, and GPU resources for efficient workload distribution. Environment variables were utilized to streamline GPU allocation processes, further enhancing resource utilization.

Within the JURECA HPC infrastructure, reconstruction jobs requested GPU-enabled nodes via the module configuration in `3dreconst_venv/modules.sh`. This setup allowed multiple plants to be processed concurrently on a node, with the exact degree of parallelism determined by the available GPUs and memory on the allocated hardware.

This approach achieved significant acceleration and parallelism in the 3D reconstruction pipeline, enabling the analysis of a considerable number of roots within a reasonable time-frame. The scalability of this approach was demonstrated by its ability to efficiently handle large datasets, showcasing the innovative integration of advanced technologies for large-scale phenotyping, including deep learning and structure-from-motion techniques.

5.2 Computational Requirements

The 3D reconstruction pipeline has specific hardware and software requirements for successful execution:

Hardware Requirements:

- **RAM:** Minimum 32GB, recommended 64GB for typical cassava plant videos
- **Disk Space:** Approximately 2GB per plant for complete outputs (images, point clouds, meshes, metrics)
- **GPU:** CUDA-capable GPU recommended (tested on RTX 3090 and Tesla A100)
- **CPU:** Multi-core processor (16+ cores recommended for parallel processing)

[6] https://www.fz-juelich.de/en/ias/jsc/systems/supercomputers/jureca

Processing Time per Plant:

- Stage 1 (Segmentation): 2-5 minutes (GPU-accelerated, BiRefNet-DIS model; scales with image count)
- Stage 2 (Stem Segmentation): 2-5 minutes (HQ-SAM with algorithmic seed point selection; GPU-accelerated)
- Stage 3 (Sparse Reconstruction): 20-45 minutes (COLMAP SIFT feature detection and matching; may require multiple trials)
- Stage 4 (Dense Reconstruction): 1.5-3 hours (GPU-accelerated, PatchMatchStereo; most time-intensive stage)
- Stage 5 (Outlier Removal): 10-30 seconds (statistical and radius filtering)
- Stage 6 (Backprojection): 30-90 seconds (ray-tracing validation)
- Stage 7 (Mesh Analysis): 15 seconds to 70 minutes (varies with root complexity; includes PyEmbree [27] ray-casting for diameter measurements, Teasar [21] skeletonization via Skeletor [22], and Trimesh [5] for watertight mesh generation)
- Stage 8 (Compression): 1-3 minutes (ZIP archiving and cleanup)
- **Total:** Approximately 2.5-4 hours per plant on JURECA HPC (Tesla A100 GPU, 64GB RAM)

Performance measurements shown above were derived from actual HPC execution on JURECA infrastructure with GPU acceleration. Stage 4 (dense reconstruction) consistently dominated the total processing time, accounting for approximately 60% of the pipeline duration. The large variance in Stage 7 execution time (15 seconds to 70 minutes) was attributed to differences in root system complexity: plants with more branches, tips, and intricate geometries required more ray-casting operations for diameter calculations and longer skeletonization times. Simple root systems completed in under 1 minute, while complex systems with 10+ branches required up to 70 minutes for comprehensive analysis.

Note on computational scaling: Processing times scaled with plant size and root system complexity. Larger plants produced more frames with higher pixel counts, resulting in more SIFT features (Stage 3), denser point clouds (Stage 4), and more mesh faces requiring analysis (Stage 7). A plant with extensive root development required 2-3× longer processing time compared to a smaller plant of the same age. The times reported above represented typical cassava roots aged 5-10 weeks; younger or smaller plants processed faster, while exceptionally large or mature roots exceeded these estimates.

5.3 Key Technical Parameters

The reconstruction and measurement pipeline employs several critical parameters that were empirically optimized for cassava root reconstruction (see Appendix B for detailed parameter specifications). These parameters remain fixed across all reconstructions to ensure consistency and reproducibility. The alpha shape parameter ($\alpha = 0.025$) and diameter radius fraction (0.5) were found through iterative testing to provide optimal results for cassava roots aged 5-10 weeks.

5.4 Reconstruction Statistics

As mentioned earlier, our pipeline utilizes the sparse reconstruction paradigm, aiming to establish a minimal yet essential sparse point cloud representing significant features within a scene. The sparse reconstruction stage of COLMAP framework relies on the Scale-Invariant Feature

Transform (SIFT) [16] algorithm for feature detection and matching. SIFT is a robust method known for its ability to identify distinctive features in images, making it suitable for tasks like structure-from-motion reconstruction. However, it's essential to acknowledge that SIFT may encounter challenges, particularly in areas of the images that lack unique features or exhibit low texture. Despite its strengths, SIFT has a non-deterministic nature, meaning that its outcomes can vary across different runs of the algorithm. This variability can impact the success of sparse model reconstruction. In instances where SIFT struggles to detect features, especially in regions with low texture or less distinctive characteristics, the sparse reconstruction process may face difficulties in establishing a robust set of points in the 3D space.

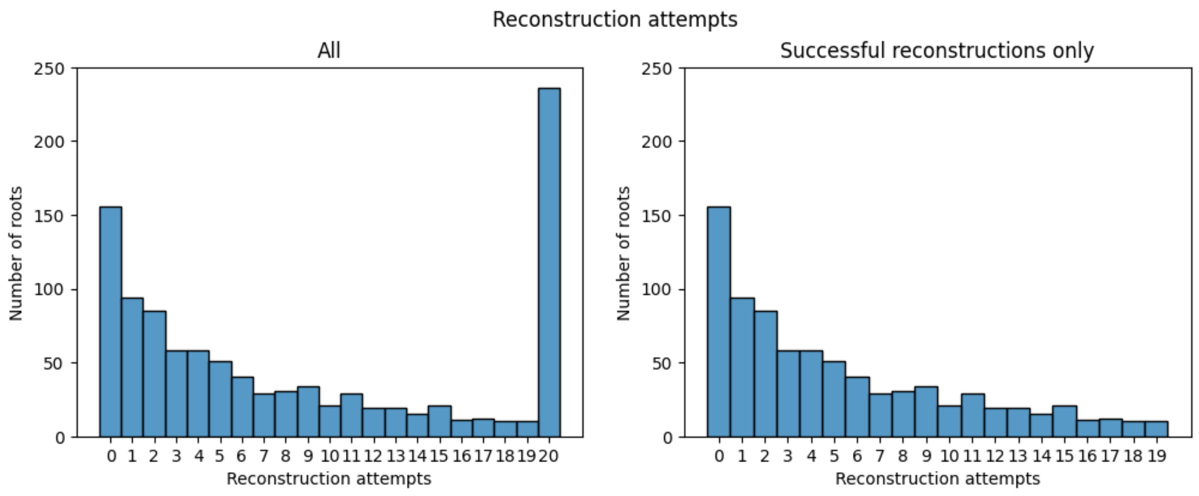


Figure 12: Histogram on the left illustrates the reconstruction attempts for all of the roots, while the histogram on the right illustrates the attempts in the successful reconstructions only.

Within our pipeline, we employed an iterative mechanism to allow for multiple reconstruction attempts. The maximum number of trials was set to 20, based on the insights gained from the reconstruction statistics collected and illustrated in Fig. 12. The histogram offered a rationale for choosing 20 as the maximum trial number. The left histogram encompassed the entire reconstructions, whether they failed or succeeded. It was evident that while most roots were reconstructed in the initial trials, some roots were not reconstructed initially but were successfully managed in subsequent attempts. This phenomenon was attributed to the intrinsic challenges of feature detection, particularly in areas with low texture or lacking unique features. Examining the right histogram, it became apparent that more than half of the successful reconstructions occurred after the initial trial. This observation affirmed the effectiveness of our trial-based approach during the sparse reconstruction phase, acknowledging the non-deterministic nature of feature detection methods, such as SIFT. This approach ensured a more robust and successful reconstruction by allowing for multiple attempts, contributing to the overall reliability of the sparse model reconstruction.

Moreover, we share a few charts illustrating the reconstruction status of roots. It must be mentioned that we have not considered roots that were younger than 5 weeks old. Besides, roots that were found to be fibrous were also skipped, as there were not enough features detected for sparse reconstruction to succeed. Fibrous roots tend to have very thin branches that are not visually separable from the background. Therefore, we skipped reconstruction of fibrous roots, however, we counted how many times this was encountered, nonetheless.

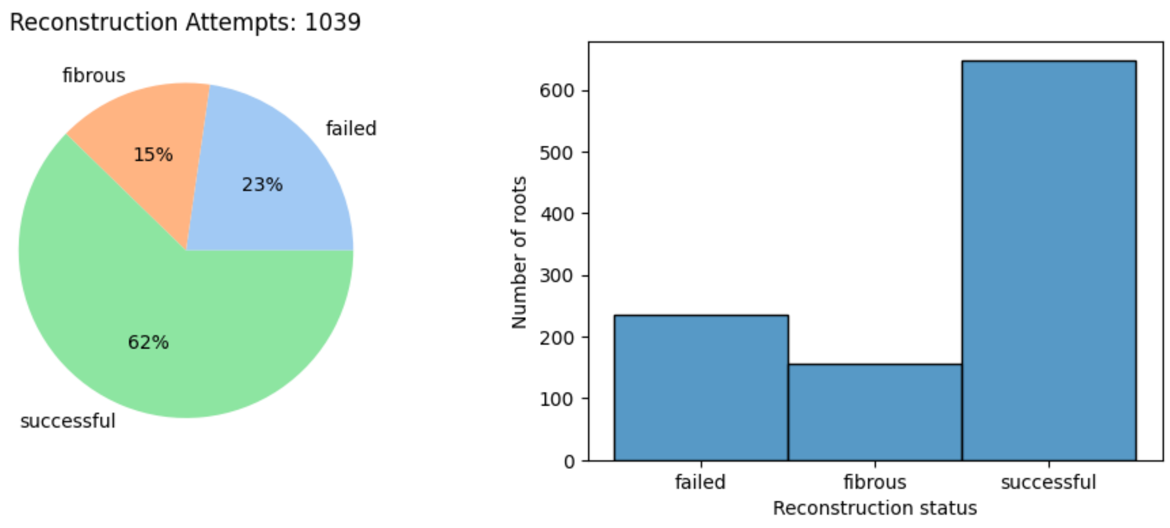


Figure 13: Reconstruction outcomes for 1039 root system acquisitions. Of these, 62% (644) were successfully reconstructed, 15% (156) were classified as fibrous and excluded from reconstruction attempts, and 23% (239) failed reconstruction.

Figure 13 illustrated a pie chart as well as a histogram showing reconstruction outcomes across 1039 acquisition attempts. The fibrous category represented roots that were identified and excluded from reconstruction attempts because fibrous root systems lacked sufficient distinctive features for SIFT keypoint detection, leading to inevitable failure during sparse reconstruction. The failed category included roots where COLMAP reconstruction was attempted but ended unsuccessfully (e.g., roots too small, insufficient texture features). The successful category (644 roots, 62%) represented cases where 3D meshes were successfully generated. Figure 14 showed weekly distributions, where week signified how old the root was. It was beneficial to examine these numbers alongside the measurement results. For instance, the number of plants aged 10 weeks represented the minority.

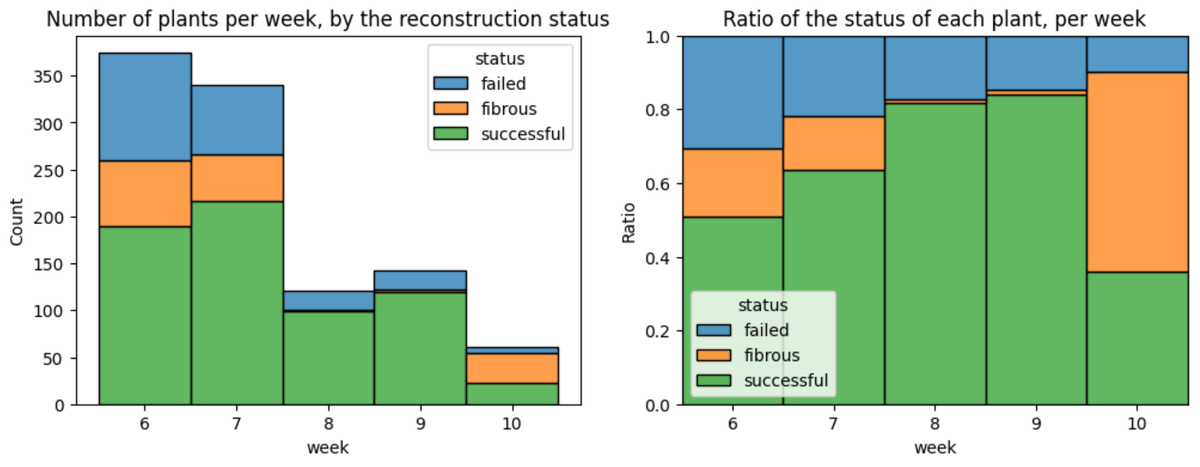


Figure 14: Weekly distribution of roots by their reconstruction status.

5.5 Measurement Methods

In this subsection, we elaborate on the in-depth analysis of the reconstructed three-dimensional models. We include a breakdown of root systems to reveal spatial intricacies, morphological nuances, and volumetric properties. We include an array of measurements and metrics, spanning the lengths and widths of roots, stems, and diameters, as well as volumetric assessments and

Metric	**Unit**
Volume	cubic meters
3D Convex Hull	cubic meters
Diameter	meters
Root Lengths	meters
Root Angles	degrees
Stem Length	meters
Root Tips	count

Table 1: List of 3D measurements and their units.

convex hull measurements. The main goal of this work is to elaborate the 3D measurements using RGB image dataset only -without having to employ any cutting-edge scanners-, since to our knowledge, the study of such scale has not yet been done. There have been previous works that focused on 2D measurement methods [28], however, this work is the first work is to do this in 3-dimensional space for the cassava roots.

3-dimensional measurements are categorized as 3D as they pertain to the spatial and volumetric aspects of the reconstructed three-dimensional plant root structures. Table 1 displays the 3D metrics and units we focus on.

5.5.1 Example Reconstruction

To illustrate the pipeline's output, we present a detailed example of a successfully reconstructed 9-week-old cassava plant (ID: 400046001000010000020). Table 2 shows the comprehensive set of morphological traits automatically extracted from the 3D reconstruction. This example demonstrates the pipeline's capability to extract diverse phenotypic measurements including root system architecture (12 root tips, 11 branching nodes), spatial dimensions (width: 17.3 cm, height: 13.8 cm, depth: 14.9 cm), volumetric properties (average root volume: 65.3 cm^3), and morphological characteristics (average diameter: 0.31 cm, average root length: 20.1 cm, average root angle: 62.5°). The complete reconstruction contained 955 skeletal nodes representing the three-dimensional root architecture.

Measurement	**Value**	**Unit**
Structural Properties		
Node count	955	nodes
Root tip count	12	tips
Branching node count	11	nodes
Spatial Dimensions		
Root system width	17.31	cm
Root system height	13.78	cm
Root system depth	14.87	cm
Stem length	98.26	cm
Volumetric Properties		
Average root volume	65.33 ± 65.33	cm^3
3D convex hull	1070.01 ± 1070.01	cm^3
Estimated mass	1.98	kg
Morphological Traits		
Average diameter	0.31 ± 0.18	cm
Average root length	20.11 ± 1.88	cm
Average root angle	62.50 ± 21.83	degrees

Table 2: Phenotyping measurements extracted from a 9-week-old cassava plant (ID: 400046001000010000020). Values shown as mean ± standard deviation where applicable.

5.6 3D Model-based Measurements

In contrast, three-dimensional (3D) measurements delved into the volumetric and spatial intricacies of reconstructed plant root structures, offering a comprehensive view beyond the planar representation. These measurements encapsulated the spatial dimensions, volumes, and morphological attributes of the plant roots. The list of 3D measurements included 3D bounding box dimensions (also referred to as 3D root system height, width, depth), volume, mass, 3D convex hull, moment of inertia, diameter, root lengths, root angles, and stem length. Some measurements were also conducted without the stem.

Overall, these measurements offered a holistic understanding of the three-dimensional architecture of plant roots, facilitating detailed analyses and interpretations crucial for plant biology and agronomic research.

5.6.1 Diameter

The diameter measurement process was a critical component in the quantitative analysis of plant root structures. In this method, the calculation of root diameters was facilitated through a meticulous examination of branching nodes and their corresponding tips within a three-dimensional mesh representation of the root system.

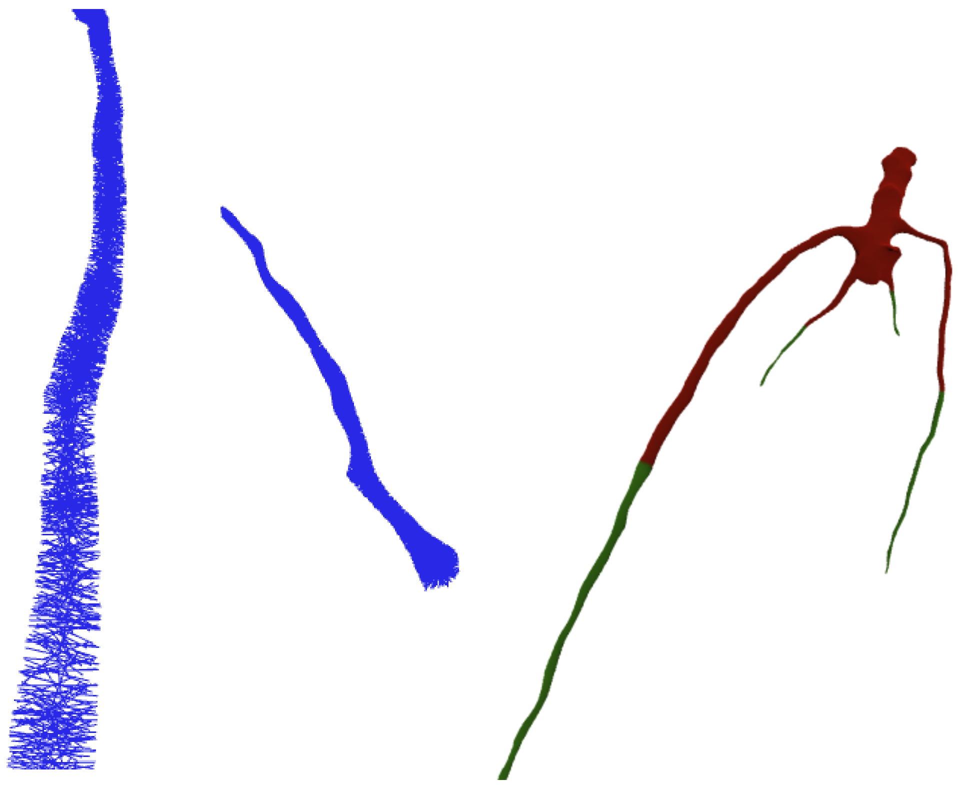


Figure 15: Diameter rays found within branches. Rays that do not stretch from one side of the branch to the other, are discarded.

Object Placement in 3D Scene: The root system's orientation in the 3D reconstruction coordinate system was determined during the COLMAP sparse reconstruction phase, where camera poses and 3D points were estimated simultaneously. The mesh was then re-rooted by reordering the vertices so that the point with the minimum y-coordinate became the first vertex in the mesh data structure. This ensured consistent orientation across all reconstructed roots, with the root typically oriented along the y-axis in COLMAP's coordinate frame.

Ray-Casting Algorithm: The diameter measurement algorithm employed PyEmbree's ray-casting capabilities combined with skeletal structure analysis. For each root tip node, the algorithm identified its corresponding branching parent node in the skeleton tree. A sampling region was defined as a sphere centered at the tip with radius $r = 0.5 \times d_{\text{tip-branch}}$, where $d_{\text{tip-branch}}$ was the Euclidean distance between the tip and its branching parent. This radius fraction of 0.5 ensured sampling occurred within the root branch region while avoiding the divergence point.

Within this sampling sphere, mesh face centroids were examined. For each centroid falling inside the sphere, a ray was cast along the negative face normal direction using PyEmbree's RayMeshIntersector. The algorithm shot rays from one side of the root cross-section (starting at the face centroid on the surface) in the direction opposite to the surface normal, attempting to intersect with the mesh on the opposite side of the root.

Ray Selection and Validation: The ray-mesh intersection was validated using a strict criterion: only rays that intersected the mesh exactly once were accepted. This single-intersection filter ensured the ray successfully penetrated from one side of the root to the other, rejecting:

- Rays that missed the mesh entirely (indicating edge cases where the face was on a thin protrusion)
- Rays that intersected multiple times (indicating complex geometry, self-intersections, or rays passing through multiple root structures)

For valid intersections meeting the single-hit criterion, the diameter at that cross-section was computed as the Euclidean distance between the face centroid (starting point on one surface) and the intersection point (opposite surface location). This distance represented the thickness of the root at that particular cross-section perpendicular to the local surface normal.

Diameter Aggregation: The algorithm sampled multiple cross-sections within each branch by evaluating all mesh faces whose centroids fell within the sampling sphere. This generated multiple diameter measurements distributed spatially throughout the branch volume. The final reported diameter was computed as the mean of all valid measurements across all branches in the root system, with the standard deviation indicating morphological variability between branches and within individual branches.

The method provided robust diameter estimates despite irregular root geometries by: (1) sampling multiple locations rather than relying on a single cross-section, (2) using strict geometric validation to ensure measurement quality, and (3) leveraging the skeletal structure to define biologically meaningful sampling regions. Figure 15 illustrates sample diameter rays, showing how invalid rays (those not stretching completely across the branch) were discarded during the filtering process.

PyEmbree Acceleration: The ray-casting implementation leveraged PyEmbree [27], which provided GPU-accelerated ray-triangle intersection through a Bounding Volume Hierarchy (BVH) acceleration structure. A BVH is a tree data structure where each node represents a bounding box containing a subset of mesh triangles. The tree was constructed by recursively partitioning triangles into spatial groups, creating a hierarchy:

- **Leaf nodes:** Contain small sets of triangles (typically 1-4 per leaf)
- **Internal nodes:** Contain axis-aligned bounding boxes (AABBs) that enclose all triangles in their subtrees
- **Root node:** Encompasses the entire mesh

Ray-mesh intersection using BVH operated in $O(\log n)$ time versus $O(n)$ for brute-force testing against all triangles. The algorithm traversed the BVH tree: if a ray missed a node's bounding box, the entire subtree was skipped. This hierarchical culling dramatically reduced the number of ray-triangle intersection tests required. For a mesh with n triangles, a balanced BVH had depth $\log_2 n$, and a typical ray intersected only $O(\log n)$ bounding boxes before finding triangle hits.

This acceleration was critical for our application: each root system generated thousands of rays (one per face centroid in sampling spheres) tested against meshes containing tens of thousands of triangles. A typical diameter measurement for one root involved:

- 12 root tips $\times$ 500 face centroids/tip = 6,000 rays
- Each ray tested against mesh with 50,000 triangles
- Brute force: 6,000 $\times$ 50,000 = 300 million ray-triangle tests
- BVH accelerated: 6,000 $\times$ $\log_2$(50,000) $\approx$ 96,000 tests (3,000$\times$ speedup)

Without BVH acceleration, diameter measurements would have required hours per root; with PyEmbree, measurements completed in 15 seconds to several minutes depending on root complexity.

The complete diameter measurement procedure (Algorithm 6 in Appendix A) operates in five steps with $O(|\mathcal{N}_{tips}| \times |F_{avg}| \times \log |F|)$ complexity:

1. **Identify branching parent:** For each root tip, find its parent branch node. The distance between tip and branch defines the sampling sphere radius ($r_{sphere} = 0.5 \times d_{tip-branch}$).
2. **Define sampling sphere:** Compute sphere centered at tip position with radius proportional to tip-branch distance. This adaptive sizing ensures appropriate coverage for varying root thicknesses.
3. **Sample face centroids:** Collect all mesh faces whose centroids fall within the sampling sphere. These faces represent surface points where diameter will be measured.
4. **Cast rays and measure diameters:** For each sampled face, cast a ray inward (opposite surface normal) and query PyEmbree's BVH structure for intersections. Apply **single-intersection criterion**: only rays hitting exactly one other surface point represent valid cross-sections (0 hits = miss; ¿1 hits = complex/overlapping geometry).
5. **Compute statistics:** Calculate mean diameter $\bar{d}$ and standard deviation σ_d from all valid measurements. Typical acceptance rates: 40-60%.

The algorithm operates in three nested loops: iterate over root tips, sample face centroids within each sphere, and cast/validate rays. The logarithmic factor in complexity comes from BVH traversal during ray queries.

5.6.2 Root Lengths

Root length measurement leveraged the skeletal tree structure to compute path distances from each root tip back to the main root node. The algorithm processed each root tip node individually using tree traversal:

1. Starting from a root tip node, trace the path back to the root node using reverse tree search
2. Along this path, compute the Euclidean distance between each consecutive pair of skeleton nodes: $d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$
3. Sum all edge distances along the path to obtain total root length: $L = \sum_{i=0}^{n-1} d_i$

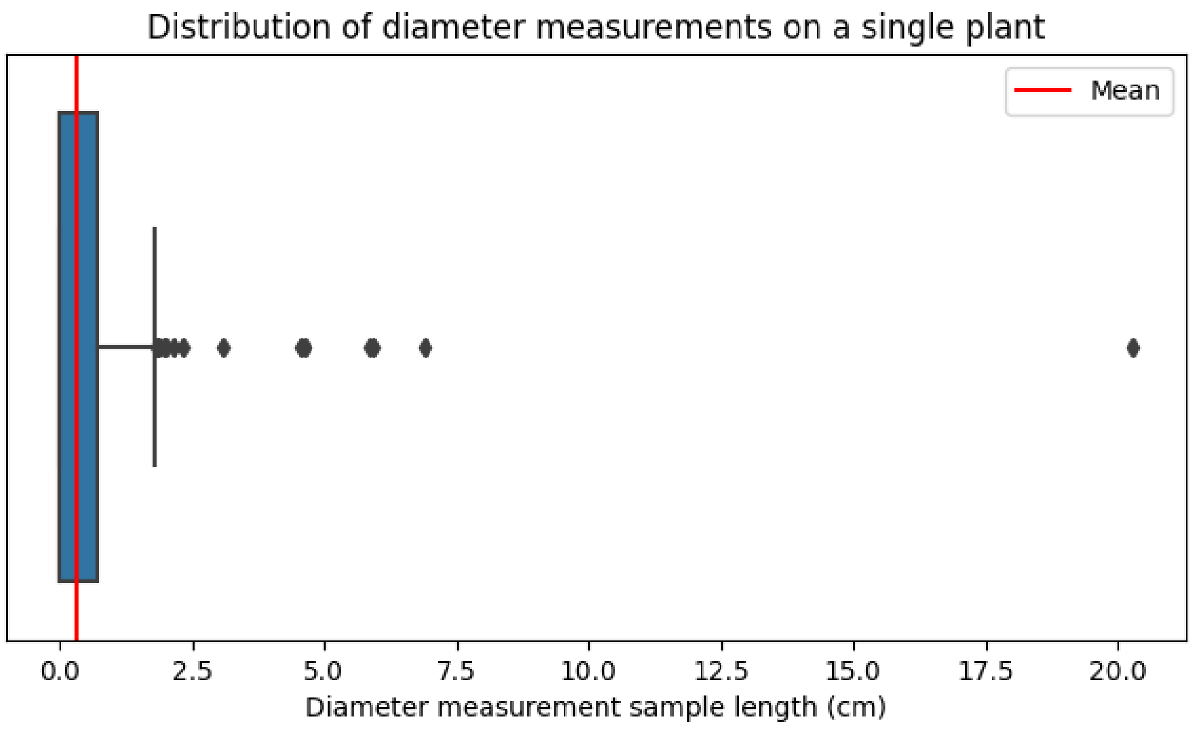


Figure 16: Diameter sample distribution of a single root (cm).

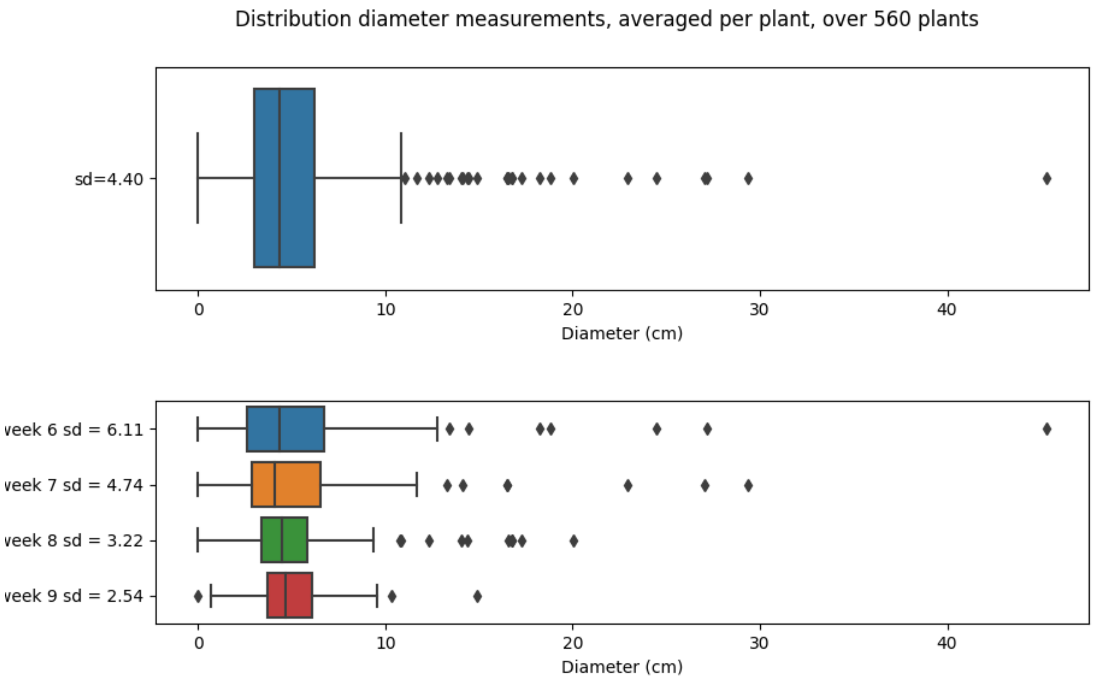


Figure 17: Diameter distribution of roots in the dataset, averaged per plant. The above distribution is week-independent, whereas the below one is a weekly breakdown.

4. Apply the scaling factor to convert from COLMAP units to centimeters: $L_{\text{cm}} = L \times s$

This approach accurately captured the curvilinear length of each root, accounting for bends and turns in the three-dimensional structure. The method was applied to all root tips, producing a comprehensive dataset of individual root lengths. These measurements revealed spatial dimensions and growth patterns, with mean and standard deviation providing population-level insights into root development across the dataset.

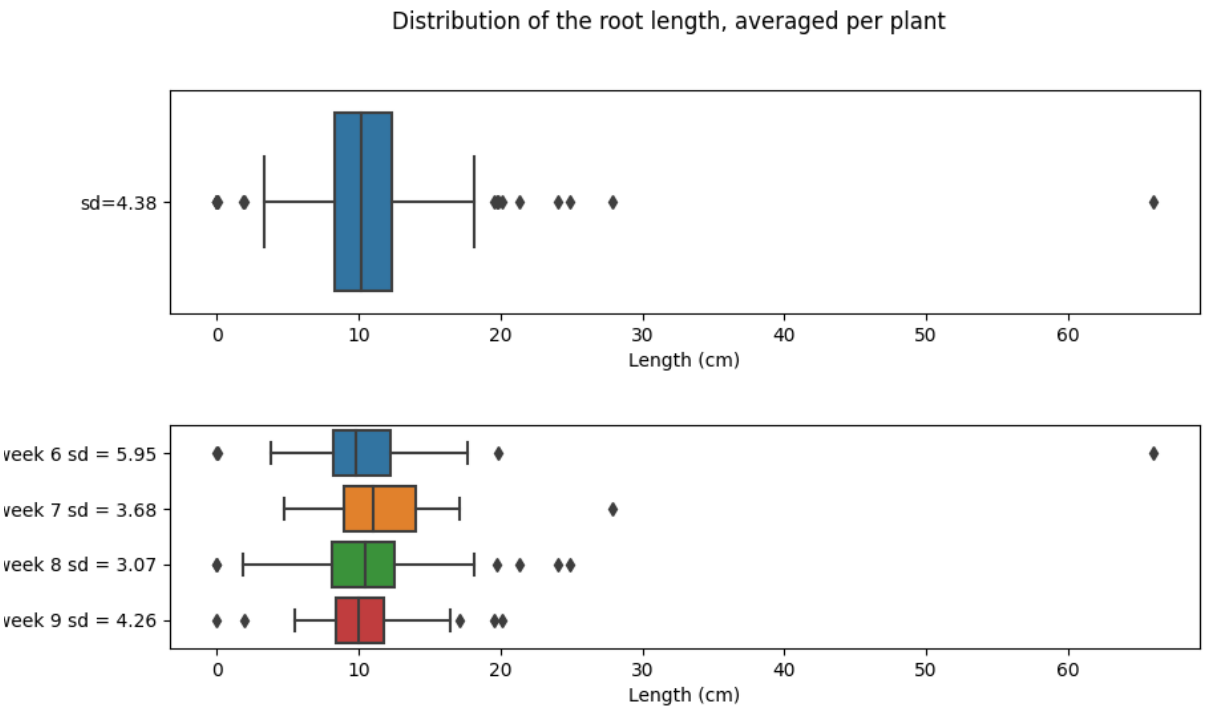


Figure 18: Visualization of the distribution of root lengths, averaged per root system.

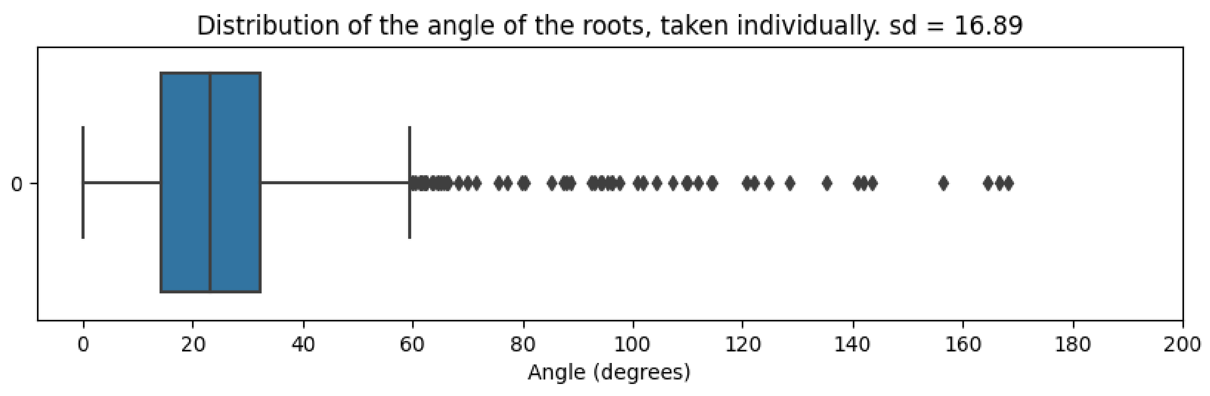
5.7 Root Angles

The angle measurement method was designed to quantify the spatial arrangement of a plant root system in three dimensions.

Given a stem and a divergence point in the root structure, along with multiple root tips, this algorithm calculated the angles between each root tip and the main stem. The calculation involved transforming the stem, the divergence point, and each root tip into vectors in a three-dimensional space. The angles were then computed using trigonometric principles, specifically by determining the cosine of the angle between the vectors. The cosine was derived from the dot product of the root tip vector and the stem vector, normalized by the magnitudes of both vectors. The result was a set of angles that represented the spatial orientation of each root tip in relation to the main stem. This information was valuable for understanding the branching patterns and overall geometry of the plant's root system. The flexibility to output angles in either radians or degrees enhanced the utility of this method in diverse analytical contexts within the 3D reconstruction pipeline.

5.7.1 Stem Length

The method for calculating stem length within the three-dimensional reconstruction pipeline was instrumental in quantifying the central axis of the plant root system. By taking as input the list of all nodes, the designated stem node, and the tree structure, the function navigated through



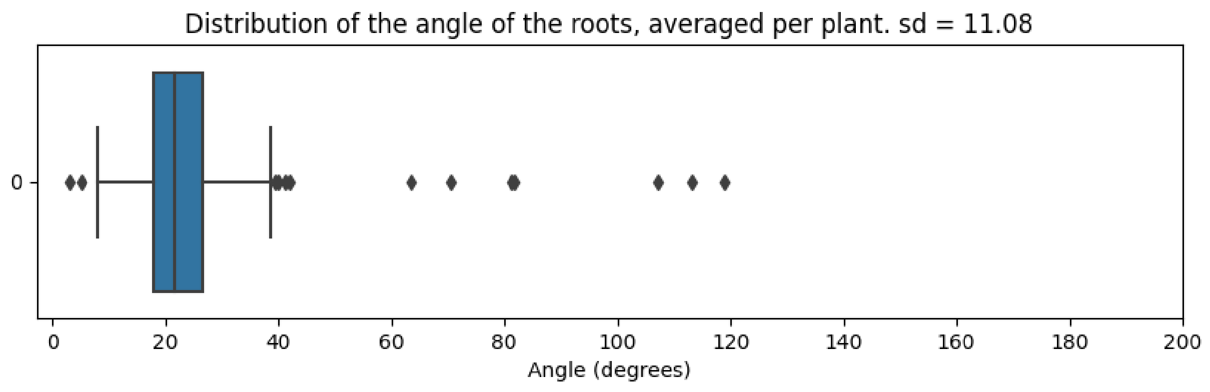
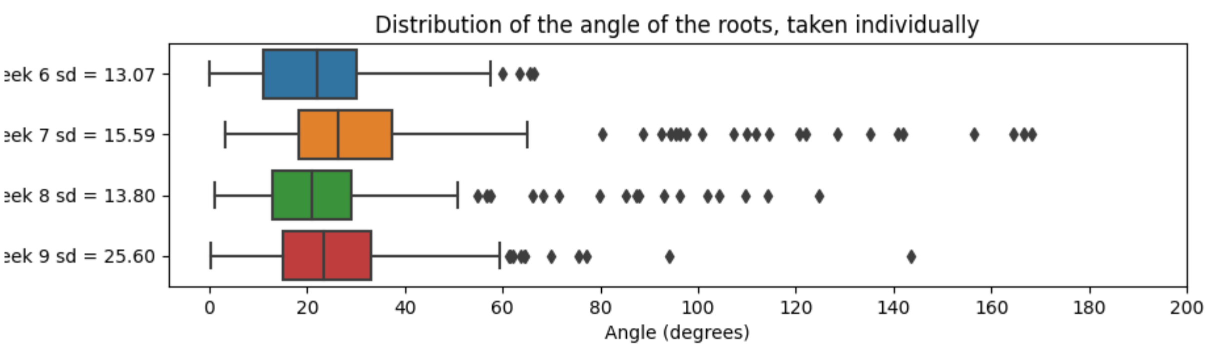


Figure 19: Visualization of the distribution of root angles within a single plant root.



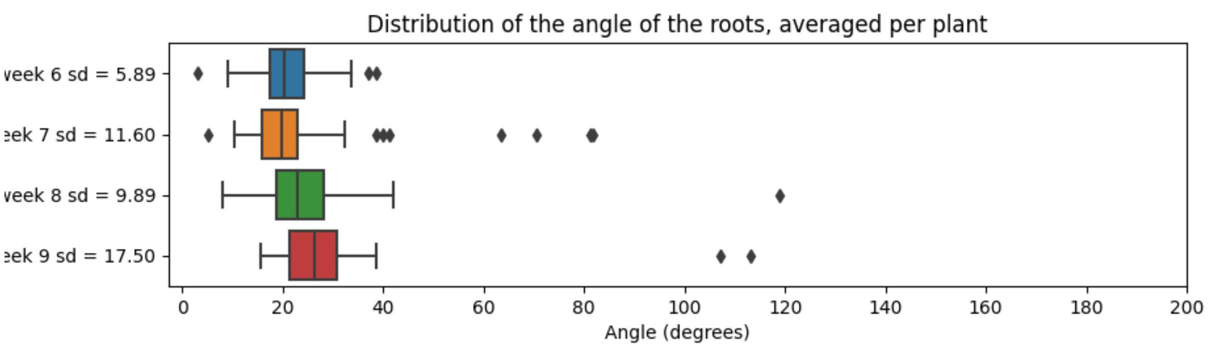


Figure 20: Visualization of the distribution of root angle, averaged per root system.

the tree using a depth-first search approach. Along the identified path from the specified stem node to the primary root, the method iteratively computed the Euclidean distances between consecutive nodes. The cumulative distance formed a precise measurement of the stem length, providing crucial quantitative data for understanding the overall plant architecture. This stem length metric served as a foundational component in the comprehensive analysis of reconstructed plant root structures.

5.7.2 Root Tips

The skeleton of the plant root system held valuable information about the distribution of root tips. The skeleton, essentially a simplified and interconnected representation of the root structure, allowed for a focused analysis of the spatial arrangement of root tips within the plant. By examining the branching patterns and connections in the skeleton, we gained insights into how root tips were distributed throughout the plant, identifying key features such as density, dispersion, and clustering. This information was crucial for understanding the overall architecture of the root system and was instrumental in various applications, including plant physiology studies and agricultural research.

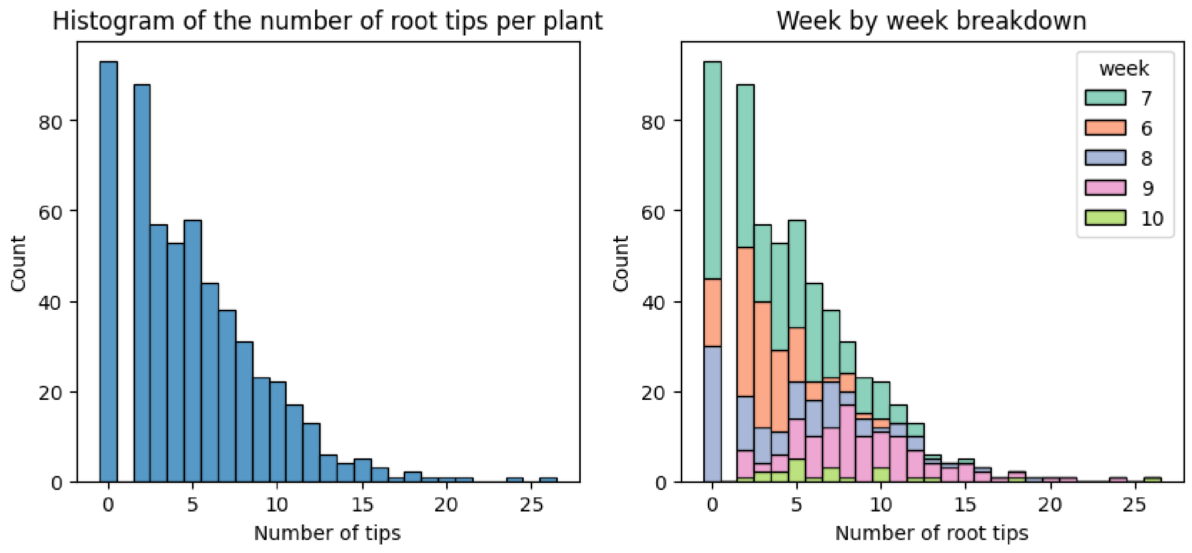


Figure 21: Number of root tips found across the cassava roots reconstructed. Left side plot shows an overall distribution of plant counts vs. number of root tips, while the right side plot highlights the weekly breakdown.

Analyzing the distribution of root tips helped uncover patterns and relationships that contributed to a deeper comprehension of how plants explored and utilized soil space. It provided researchers and scientists with valuable data for making informed decisions about plant health, growth, and nutrient uptake, ultimately contributing to advancements in plant science and agriculture. Figure 21 depicted two histograms revealing the distribution of root tips of cassava roots, along with the weekly breakdown.

5.7.3 Volume

Volume measurement was a fundamental metric for quantifying root system biomass and spatial occupation. Accurate volume calculation required watertight (manifold) meshes where the surface was closed without holes or non-manifold edges. The volume calculation pipeline consisted of three stages:

1. **Mesh Repair**: The alpha shape mesh was repaired using Blender's 3D printing utilities to ensure manifold topology
2. **Volume Computation**: Blender's BMesh library computed the enclosed volume using the divergence theorem, which integrated the surface normal vectors over the closed mesh
3. **Scaling**: The computed volume (in COLMAP's arbitrary units^3) was converted to cm^3 by applying the scaling factor cubed: $V_{\mathrm{cm}^3} = V_{\mathrm{COLMAP}} \times s^3$

The BMesh volume calculation was mathematically rigorous for closed, orientable surfaces but produced undefined or negative results for non-manifold meshes. This motivated the mesh repair preprocessing step. In our dataset, approximately 5-8% of reconstructions produced artifacts that resulted in negative volume estimates, which were flagged and excluded from analysis (visible as shaded regions in Figure 23).

Volume measurements were computed for two configurations: root system without stem (isolating root biomass) and complete system with stem. The average and standard deviation were calculated across the dataset, though for individual plants these statistics were identical since only one mesh per plant was generated. These metrics provided insights into root biomass accumulation across developmental stages.

The average and standard deviation of the volume were calculated, providing not only a central tendency measure but also an indication of the variability in the root system's size and shape. These metrics were crucial for assessing the robustness and consistency of the reconstructed models. Integrating volume measurements with other spatial metrics, such as bounding box dimensions and convex hull properties, contributed to a holistic characterization of the plant root architecture. This comprehensive analysis aided in unraveling the intricate relationships between form and function in plant roots, offering valuable insights for various scientific applications. Figure 22 illustrated the distribution of mesh volumes obtained. Additionally, Figure 23 showed the volume distribution across the number of plants—with and without stem—using a histogram plot.

5.7.4 3D Convex Hull

The 3D convex hull represented the smallest convex polyhedron that fully enclosed the root system mesh, analogous to stretching a rubber sheet tightly around the root structure. This metric quantified the volumetric space occupied by the root system, including interior voids and concavities, providing insights into root spatial distribution and exploration efficiency.

The convex hull volume is computed using Trimesh's built-in convex hull algorithm, which applies the Quickhull algorithm to find the minimal convex set of vertices enclosing the mesh. The volume of this polyhedron is then calculated and scaled to cm^3. This measurement differs from the actual mesh volume in that it includes empty space within the root system's envelope, making it useful for assessing the overall space utilization of the root architecture. Like other volumetric measurements, convex hull volumes are computed both with and without the stem for comparative analysis across developmental stages (Figure 24).

6 Discussion

This laboratory work demonstrated the implementation of a 3D reconstruction pipeline for cassava root phenotyping using accessible equipment and open-source software.

6.1 Reconstruction Performance

As shown in Figure 13, we processed 1039 root system acquisitions with the following outcomes: 644 successful reconstructions (62%), 239 failed reconstructions (23%), and 156 fibrous roots

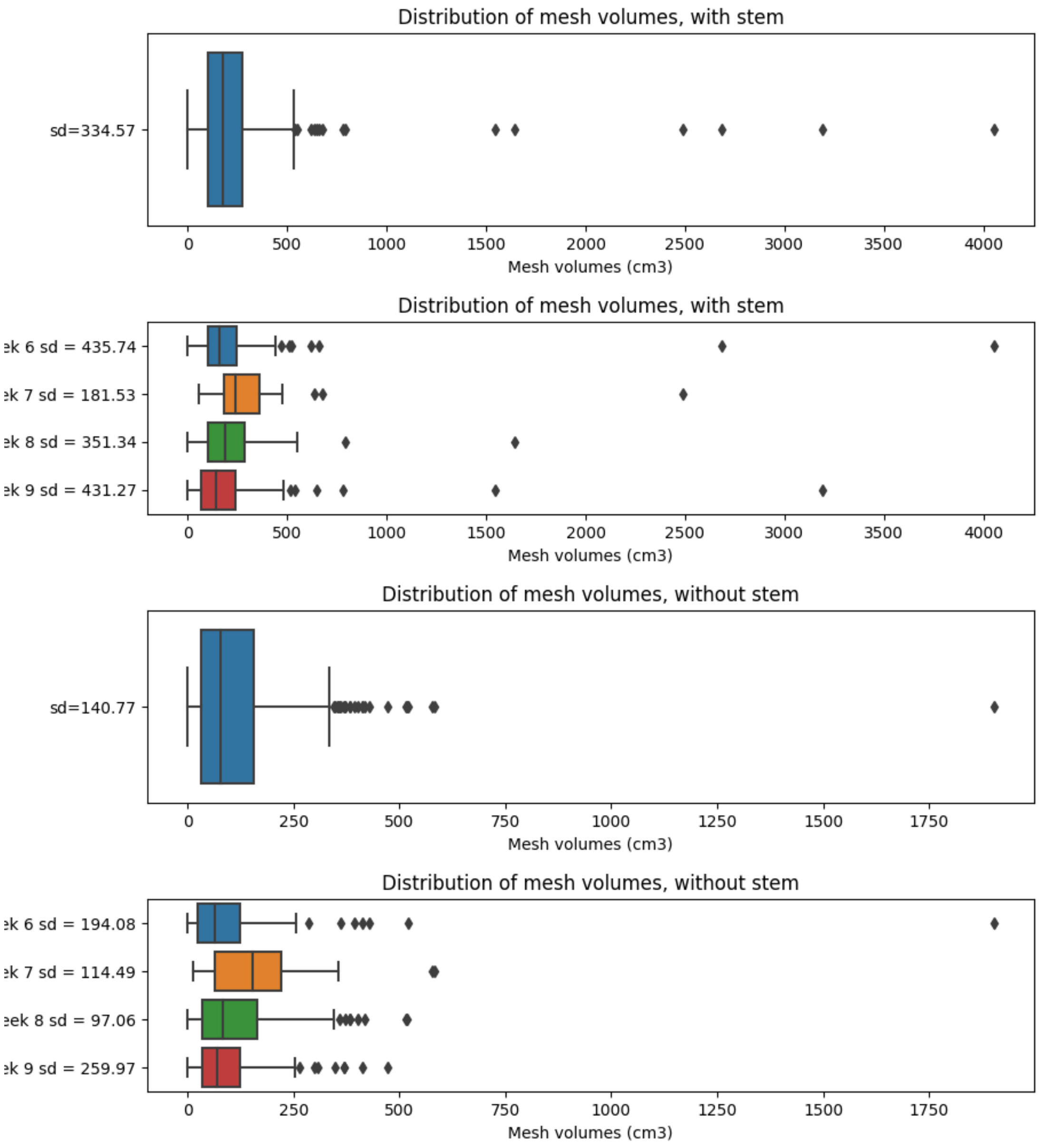


Figure 22: Distribution of mesh volumes with and without the stem, along with weekly breakdown.

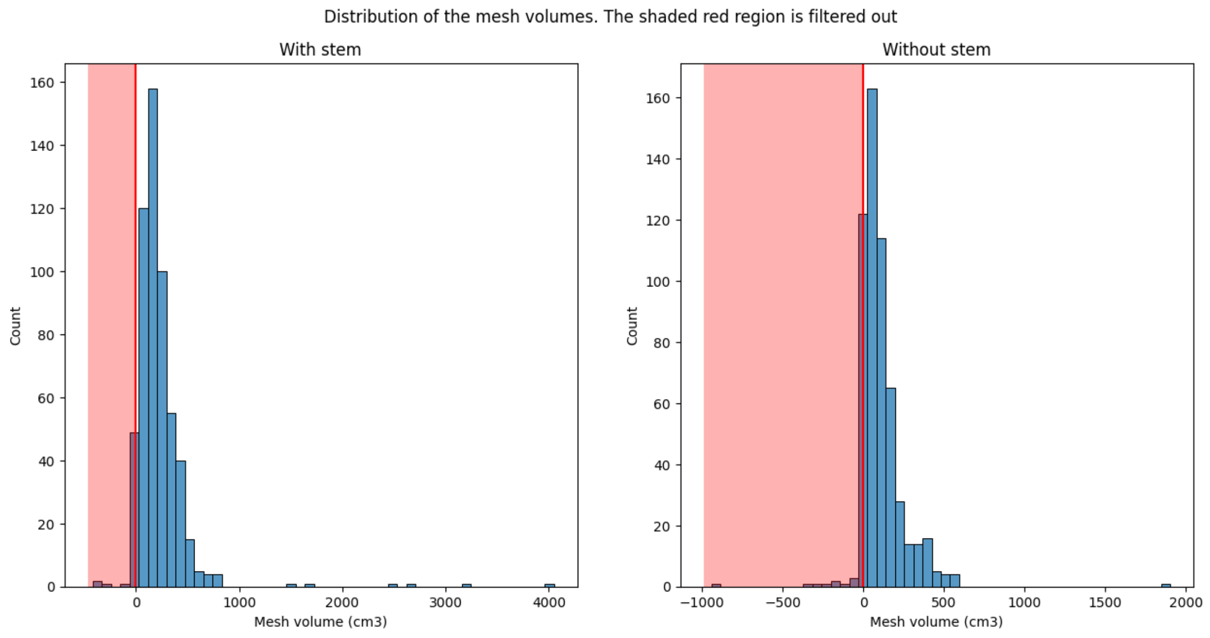


Figure 23: Distribution of mesh volumes per root count, with and without the stem. Note that the shaded region is discarded as they contain negative volumes that come out due to artefacts found in the mesh.

that were excluded from reconstruction attempts (15%). The fibrous roots were identified and excluded before reconstruction attempts because such root systems lack sufficient SIFT keypoints—they would inevitably fail during sparse reconstruction due to insufficient distinctive features. The iterative reconstruction approach proved essential for the attempted reconstructions—more than half of successful reconstructions required multiple attempts (up to 20) due to the non-deterministic nature of SIFT feature detection in COLMAP.

The primary failure modes for the 239 attempted but failed reconstructions were: (1) roots younger than 5 weeks that were too small or lacked sufficient structural development, and (2) low-texture regions hindering feature matching. These limitations are intrinsic to structure-from-motion approaches. Our pipeline was effective for storage roots during the 5-10 week development period, achieving 62% reconstruction success across all acquisitions (644 out of 883 attempted reconstructions, excluding the 156 pre-identified fibrous roots).

6.2 Segmentation and Measurement Quality

REMBG with BiRefNet-DIS performed effectively for root segmentation across diverse morphological variations. The pre-trained model handled varying lighting conditions and complex root structures without requiring domain-specific training. However, we observed limitations in regions with very thin root branches or low contrast against backgrounds. These segmentation artifacts were generally addressed through outlier filtering and backprojection validation in subsequent 3D reconstruction stages.

The diameter measurements obtained through ray-casting algorithms using PyEmbree [27] showed reasonable distributions consistent with expected cassava root morphology. Weekly breakdowns revealed expected growth trends, with older roots (8-10 weeks) exhibiting larger diameters and volumes compared to younger roots (5-7 weeks). Root angle distributions provided insights into spatial organization relevant for understanding soil exploration strategies.

The scaling factor calculation, derived from the known 30 cm camera baseline and COLMAP's estimated camera positions, enabled conversion from COLMAP coordinates to centimeters. While this achieved sufficient accuracy for phenotyping applications, measurement validation

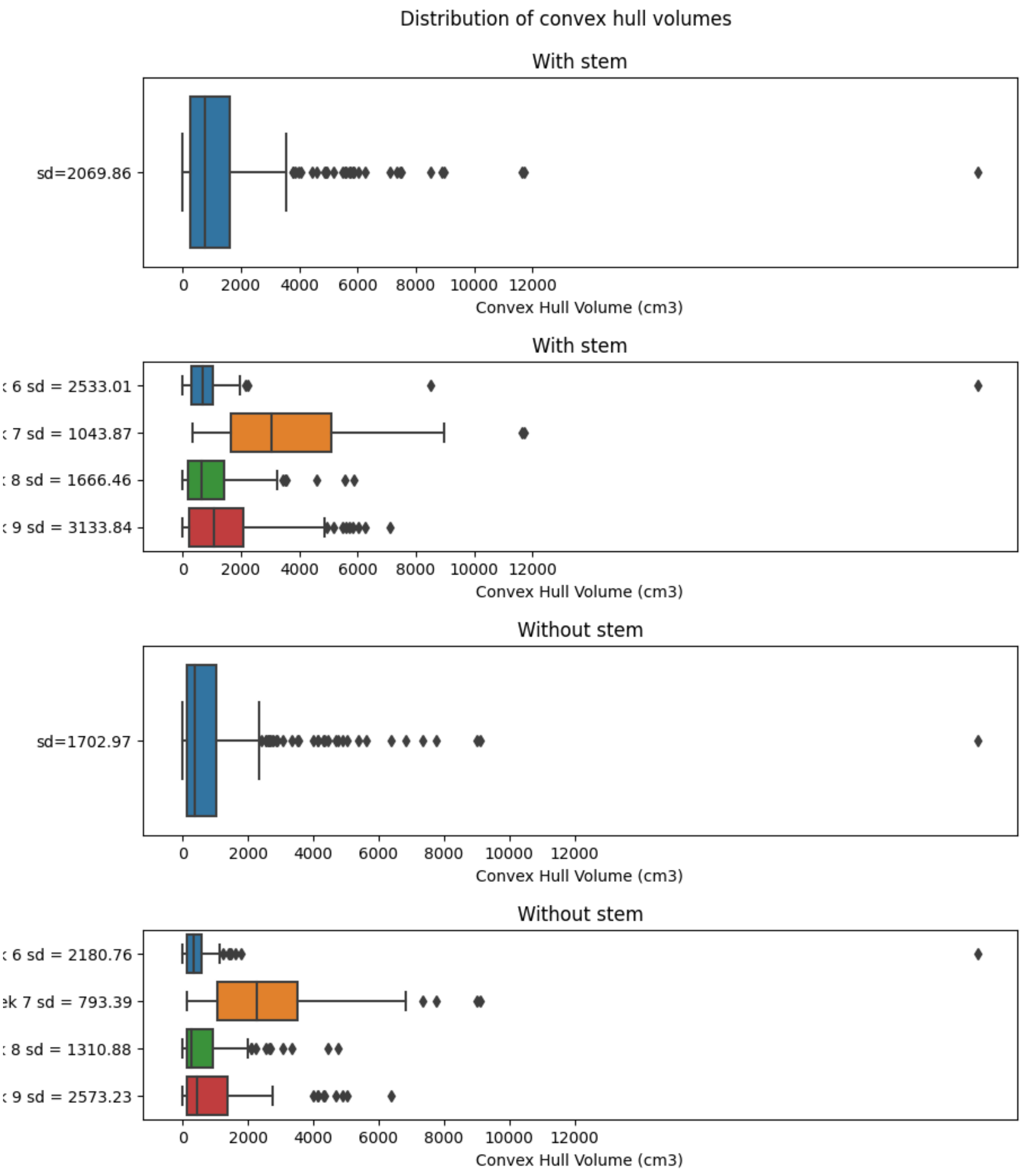


Figure 24: Distribution of 3D convex hull volumes with and without the stem, along with weekly breakdown.

against manual measurements or reference objects would strengthen confidence in results.

6.3 Sources of Error and Limitations

Several error sources and limitations were identified during this work:

Camera calibration and positioning: The camera-baseline scaling approach assumes consistent camera positioning across all imaging sessions. Any variation in the 30 cm vertical separation or camera-to-object distance (110.373 cm) introduces measurement errors. These parameters were maintained as consistently as possible through the fixed stereo rig, but minor variations may have occurred.

Destructive sampling: The imaging setup required excavated roots, preventing longitudinal studies of individual plants. Non-destructive underground imaging would be valuable but remains technically challenging.

Feature detection variability: SIFT's non-deterministic nature caused reconstruction success to vary across runs for the same root. The iterative approach (20 attempts maximum) mitigated this but increased computational time.

Manual quality control: The pipeline, while largely automated, required manual review to identify failed reconstructions. Approximately 38% of roots (395 out of 1039) failed reconstruction or were not attempted, requiring assessment to determine failure causes.

Segmentation artifacts: Very thin roots and low-contrast regions produced segmentation errors that propagated into 3D models, though post-processing filters reduced these effects.

6.4 Measurement Method Limitations

Each measurement technique in the pipeline has specific limitations that warrant careful interpretation of results:

Volume Measurements:

- Require watertight manifold meshes, which are not always achievable despite Blender-based repair
- Mesh artifacts (holes, non-manifold edges, self-intersections) can produce undefined or negative volume estimates
- Approximately 5-8% of reconstructions yielded negative volumes and were excluded from analysis
- Volume accuracy depends critically on mesh quality, which degrades for complex geometries with many thin branches
- Blender's BMesh volume calculation assumes closed surfaces; partial failures in mesh repair compromise results

Diameter Measurements:

- Sampling occurs only within spheres around branch points (radius = $0.5 \times$ tip-to-branch distance)
- Diameter variations along root length between branching points are not captured
- Method depends on skeleton quality; errors in skeletonization (incorrect branch point identification) propagate to diameter estimates
- Strict ray intersection criterion (exactly 1 hit) may reject valid measurements in noisy regions

- Reported diameter is averaged across all branches; individual branch diameters show high variability (visible in standard deviation)

Mass Estimation:

- Current implementation calculates bounding box volume × density, NOT actual mesh volume × density
- This significantly overestimates actual root mass (bounding box includes empty space)
- Assumes uniform density of 1.0 g/cm^3 across entire root system, which is unrealistic
- Real cassava roots have varying density (storage tissue vs. fibrous roots vs. stem)
- Mass estimates should be interpreted as relative comparisons rather than absolute biomass measurements
- Improvement would require: (1) using actual mesh volume instead of bounding box, and (2) incorporating density variation across root tissues

Convex Hull Measurements:

- Includes all interior void space, making it a measure of space occupation rather than actual root volume
- Sensitive to outlier points that extend the convex envelope
- Does not capture root system complexity or branching architecture detail
- Most useful for comparative analysis across developmental stages rather than absolute characterization

Root Length and Angle Measurements:

- Depend entirely on skeleton accuracy; skeletonization errors directly affect these metrics
- Teasar algorithm may not capture all fine root branches, potentially underestimating total root length
- Angle measurements assume correct identification of stem node, divergence points, and root tips
- Measurement granularity limited by skeleton node spacing (typically 1-5 mm)

These limitations highlight the importance of understanding the underlying algorithms when interpreting phenotyping results. While the measurements provide valuable comparative data across developmental stages and between plants, absolute values should be validated against ground-truth measurements (e.g., water displacement for volume, caliper measurements for diameter) when precise quantification is required.

6.5 Practical Considerations

The parallel processing approach using SLURM demonstrated scalability: by submitting jobs that reserve multiple GPUs per node (via `3dreconst_venv/modules.sh`), several plants can be processed concurrently. This scheduling strategy enabled the timely processing of the 1039 acquisitions and provides headroom for future large-scale phenotyping studies.

Immediate improvements for future implementations could include: (1) automated quality assessment to flag low-quality reconstructions, (2) parameter optimization to improve SIFT feature detection success rates, (3) validation protocols using physical reference objects for measurement accuracy assessment, and (4) exploration of more efficient algorithms to reduce HPC dependency.

7 Conclusion

This laboratory work successfully implemented a 3D reconstruction pipeline for cassava root phenotyping using accessible equipment (Canon 80D DSLR cameras) and open-source software (REMBG [7], COLMAP [24, 23], Open3D [31]). From 1039 root system acquisitions, 644 were successfully reconstructed (62% success rate), yielding quantitative measurements including volume, diameter, root angles, root lengths, and tip counts.

The pipeline demonstrated several key capabilities:

- REMBG with BiRefNet-DIS effectively segmented cassava roots from complex backgrounds without requiring domain-specific training
- COLMAP's iterative reconstruction approach (up to 20 attempts) successfully handled non-deterministic SIFT feature detection, achieving 62% reconstruction success rate across 1039 acquisitions
- Standard DSLR cameras captured sufficient detail for 3D phenotyping of storage roots aged 5-10 weeks
- Parallel processing on HPC infrastructure (JURECA) enabled analysis of large datasets within reasonable timeframes
- Camera-baseline calibration (30 cm vertical separation) provided adequate scaling accuracy for phenotyping applications
- RAM-based workspace strategy (tmpfs) improved I/O performance by 10-100× while ensuring data integrity through incremental synchronization

The primary limitations observed were:

- Inability to reconstruct very young (¡5 weeks) or fibrous roots due to insufficient SIFT features
- Dependence on consistent camera positioning and baseline separation for accurate scaling
- Requirement for manual quality control to identify failed reconstructions
- Destructive sampling preventing longitudinal studies of individual plants
- Reliance on HPC infrastructure limiting accessibility

The extracted morphological measurements provide a foundation for genotype-phenotype studies in cassava breeding programs. The 150 genotypes analyzed represent diverse root architectures suitable for identifying genetic markers associated with desirable root traits. The demonstrated feasibility of using standard equipment and open-source software makes this approach accessible to research groups with limited resources, supporting broader adoption of 3D phenotyping in plant biology research.

Acknowledgements

I would like to express my sincere gratitude to Tobias Wojciechowski[7], Vera Boeckem[8], and Hanno Scharr[9] from Forschungszentrum Jülich for their valuable guidance, support, and contributions throughout this work. Their expertise and insights were instrumental in the development and implementation of the methods presented in this laboratory report.

[7] t.wojciechowski@fz-juelich.de
[8] v.boeckem@fz-juelich.de
[9] h.scharr@fz-juelich.de

Declarations

Code Availability

The source code for the 3D reconstruction pipeline is available at: https://jugit.fz-juelich.de/ias-8/3dreconst. The repository includes detailed instructions in the README file for setting up and running the pipeline.

Data Availability

The cassava root imaging dataset used in this study is available upon request. Interested researchers should contact Tobias Wojciechowski at t.wojciechowski@fz-juelich.de for access to the data.

A Algorithmic Details

This appendix provides detailed pseudocode for the six key algorithms implemented in the reconstruction and measurement pipeline. These algorithms represent the core computational contributions of this work, providing complete implementation specifications for reproducibility.

A.1 Algorithm 1: HQ-SAM Two-Iteration Stem Segmentation

Algorithm 1 HQ-SAM Two-Iteration Stem Segmentation

```
Require: Masked root image $I$ (background removed), Image dimensions $W \times H$
Require: HQ-SAM model with ViT-L encoder
Ensure: Stem mask $S$ (binary mask isolating stem region)
 1:
 2: Iteration 1: Coarse Segmentation
 3:                                          ▷ Identify initial seed point at top of root
 4: $y_{seed} \leftarrow 10$                              ▷ Fixed row near top of image
 5: $x_{seed} \leftarrow$ find_horizontal_midpoint($I$, row=$y_{seed}$)
 6: $\mathbf{p}_{seed} \leftarrow (x_{seed}, y_{seed})$                       ▷ Initial positive prompt
 7:                                               ▷ Run first HQ-SAM prediction
 8: $\text{prompts}_1 \leftarrow \{\text{positive} : [\mathbf{p}_{seed}], \text{negative} : []\}$
 9: $\mathcal{M}_1, \mathcal{L}_1 \leftarrow$ HQ_SAM($I$, $\text{prompts}_1$)                   ▷ Mask and logits
10:
11: Iteration 2: Refinement with Dense Prompts
12:                                     ▷ Generate positive prompts along stem region
13: $\mathcal{P}_{pos} \leftarrow \emptyset$
14: for $y \in [0, 150]$ with step 10 do                     ▷ Dense vertical sampling
15:     for $x \in [0, W]$ with step 10 do                      ▷ Full width coverage
16:         $\mathcal{P}_{pos} \leftarrow \mathcal{P}_{pos} \cup \{(x, y)\}$
17:     end for
18: end for
19:                                       ▷ Generate negative prompts in root region
20: $\mathcal{P}_{neg} \leftarrow \emptyset$
21: for $x \in [500, 1500]$ with step 10 do                    ▷ Expected root location
22:     for $y \in [220, 500]$ with step 10 do
23:         $\mathcal{P}_{neg} \leftarrow \mathcal{P}_{neg} \cup \{(x, y)\}$
24:     end for
25: end for
26:                              ▷ Run second HQ-SAM prediction with logit guidance
27: $\text{prompts}_2 \leftarrow \{\text{positive} : \mathcal{P}_{pos}, \text{negative} : \mathcal{P}_{neg}\}$
28: $\mathcal{M}_2, \mathcal{L}_2 \leftarrow$ HQ_SAM($I$, $\text{prompts}_2$, mask_input=$\mathcal{L}_1$)
29:
30: Post-Processing: Connected Component Isolation
31: $\mathcal{C} \leftarrow$ find_connected_components($\mathcal{M}_2$)           ▷ Label connected regions
32: $c_{stem} \leftarrow$ component_containing($\mathcal{C}$, $\mathbf{p}_{seed}$)         ▷ Find stem component
33: $S \leftarrow$ extract_component($\mathcal{M}_2$, $c_{stem}$)                  ▷ Isolate stem mask
34: return $S$
```

A.2 Algorithm 2: Three-Stage Outlier Removal

Algorithm 2 Three-Stage Outlier Removal

```
Require: Point cloud $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ where $p_i = (\mathbf{x}_i, \mathbf{c}_i)$
Require: $\mathbf{x}_i = [X, Y, Z]^T$ (3D position), $\mathbf{c}_i = [R, G, B]^T$ (color, range [0,1])
Require: Parameters: $k_{stat} = 20$ (neighbors for statistical filter)
Require: $\sigma_{ratio} = 2.0$ (std dev multiplier), $r_{radius} = 0.02$ (radius in COLMAP units)
Require: $k_{radius} = 50$ (min neighbors for radius filter)
Ensure: Filtered point cloud $\mathcal{P}'''$

Stage 1: Color Intensity Filter
$\mathcal{P}' \leftarrow \emptyset$
for each point $p_i = (\mathbf{x}_i, [R_i, G_i, B_i]^T)$ in $\mathcal{P}$ do
    $\text{avg}_{RG} \leftarrow (R_i + G_i)/2$                    ▷ Red-green average
    if $\text{avg}_{RG} > 0.5$ or $B_i > 0.25$ then               ▷ Dual threshold
        $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{p_i\}$       ▷ Retain point
    end if          ▷ Preserves root colors (high R/G) and shadows (high B)
end for

Stage 2: Statistical Outlier Removal
Build KD-tree $\mathcal{T}$ from positions in $\mathcal{P}'$      ▷ O($n \log n$) construction
$\mathcal{D} \leftarrow []$                                       ▷ Global distance distribution
for each point $p_i$ in $\mathcal{P}'$ do
    $\mathcal{N}_i \leftarrow$ KNN_search($\mathcal{T}$, $\mathbf{x}_i$, $k_{stat}$)   ▷ Find 20 nearest neighbors
    $d_i \leftarrow \frac{1}{k_{stat}} \sum_{p_j \in \mathcal{N}_i} \|\mathbf{x}_i - \mathbf{x}_j\|$   ▷ Mean distance to neighbors
    $\mathcal{D}$.append($d_i$)
end for
$\mu_d \leftarrow$ mean($\mathcal{D}$)                             ▷ Global mean distance
$\sigma_d \leftarrow$ std($\mathcal{D}$)                           ▷ Global std deviation
threshold $\leftarrow \mu_d + \sigma_{ratio} \times \sigma_d$      ▷ Outlier threshold
$\mathcal{P}'' \leftarrow \emptyset$
for each point $p_i$ in $\mathcal{P}'$ with distance $d_i$ do
    if $d_i \leq$ threshold then                                  ▷ Within 2 std devs?
        $\mathcal{P}'' \leftarrow \mathcal{P}'' \cup \{p_i\}$
    end if                                                         ▷ Eliminates isolated noise points
end for

Stage 3: Radius Outlier Removal
Build KD-tree $\mathcal{T}'$ from positions in $\mathcal{P}''$
$\mathcal{P}''' \leftarrow \emptyset$
for each point $p_i$ in $\mathcal{P}''$ do
    $\mathcal{N}_i \leftarrow$ radius_search($\mathcal{T}'$, $\mathbf{x}_i$, $r_{radius}$)   ▷ 0.02 COLMAP units $\approx$ 0.6 cm
    $n_i \leftarrow |\mathcal{N}_i|$                               ▷ Count neighbors within radius
    if $n_i \geq k_{radius}$ then                                  ▷ At least 50 neighbors?
        $\mathcal{P}''' \leftarrow \mathcal{P}''' \cup \{p_i\}$
    end if          ▷ Eliminates sparse clusters, preserves dense root structure
end for
return $\mathcal{P}'''$
```

A.3 Algorithm 3: 3D Point Cloud Backprojection

Algorithm 3 3D Point Cloud Backprojection with Spatial Filtering

```
Require: Point cloud $\mathcal{P} = \{p_1, \ldots, p_N\}$, COLMAP reconstruction path, Segmentation masks $\{M_1, \ldots, M_F\}$
Ensure: Spatially filtered point cloud $\mathcal{P}'$
 1: Phase 1: Load Camera Parameters
 2: $\mathcal{K} \leftarrow$ read_cameras_binary(cameras.bin)                ▷ Intrinsic matrices
 3: $\mathcal{E} \leftarrow$ read_images_binary(images.bin)                 ▷ Extrinsic parameters
 4: for each image $i$ in $\mathcal{E}$ do
 5:     $\mathbf{q}_i \leftarrow$ extrinsics[$i$].quaternion
 6:     $\mathbf{t}_i \leftarrow$ extrinsics[$i$].translation
 7:     $R_i \leftarrow$ quaternion_to_rotation_matrix($\mathbf{q}_i$)      ▷ Equation 2
 8:     $K_i \leftarrow \mathcal{K}$[extrinsics[$i$].camera_id]
 9:     $P_i \leftarrow K_i \cdot [R_i|\mathbf{t}_i]$                        ▷ $3 \times 4$ projection matrix
10: end for
11:
12: Phase 2: Project Points and Retrieve Colors
13: Initialize black_region_count[$p$] $\leftarrow 0$ for all $p \in \mathcal{P}$
14: for each image $i = 1$ to $F$ do                                        ▷ Parallelized across images
15:     $I_i \leftarrow$ load_image($i$)
16:     $M_i \leftarrow$ load_segmentation_mask($i$)
17:     for each point $p = [X, Y, Z]^T$ in $\mathcal{P}$ do
18:         $\mathbf{X}_h \leftarrow [X, Y, Z, 1]^T$                        ▷ Convert to homogeneous coords
19:         $\mathbf{x}_h \leftarrow P_i \cdot \mathbf{X}_h$                 ▷ $\mathbf{x}_h = [\lambda u, \lambda v, \lambda]^T$
20:         $u \leftarrow \mathbf{x}_h[0]/\mathbf{x}_h[2]$                   ▷ Normalize to pixel coords
21:         $v \leftarrow \mathbf{x}_h[1]/\mathbf{x}_h[2]$
22:         if $0 \leq u <$ width and $0 \leq v <$ height then              ▷ Point visible in frame
23:             $p$.color$_i \leftarrow I_i[v, u]$                          ▷ Retrieve RGB color
24:             if $M_i[v, u] == 0$ then                                    ▷ Point projects onto black background
25:                 black_region_count[$p$] $\leftarrow$ black_region_count[$p$] + 1
26:             end if
27:         end if
28:     end for
29: end for
30:
31: Phase 3: Spatial Filtering
32: $\mathcal{P}' \leftarrow \emptyset$
33: for each point $p$ in $\mathcal{P}$ do
34:     if black_region_count[$p$] $< 20$ then                              ▷ Threshold: 20 frames
35:         $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{p\}$
36:     end if
37: end for
38: return $\mathcal{P}'$                                                   ▷ Filtered point cloud
```

A.4 Algorithm 4: Scaling Factor Computation

Algorithm 4 Scaling Factor Computation from Camera Baseline

```
Require: COLMAP camera parameters: Images $\mathcal{I} = \{I_1, I_2, \ldots, I_F\}$
Require: Camera pose for each image: $(q_i, \mathbf{t}_i)$ where $q_i$ is quaternion, $\mathbf{t}_i$ is translation
Require: Known physical baseline: $b_{\text{real}} = 30$ cm
Ensure: Scaling factor $s$ (cm per COLMAP unit)
 1:
 2: Step 1: Load Camera Poses from COLMAP
 3: Read images.bin from COLMAP sparse reconstruction
 4: $\mathcal{C}_{\text{top}} \leftarrow \emptyset$                                  ▷ Top camera positions
 5: $\mathcal{C}_{\text{bot}} \leftarrow \emptyset$                                  ▷ Bottom camera positions
 6: for each image $I_i$ in $\mathcal{I}$ do
 7:     $(q_i, \mathbf{t}_i) \leftarrow$ read_camera_pose($I_i$)             ▷ Quaternion + translation
 8:     $R_i \leftarrow$ quaternion_to_rotation_matrix($q_i$)           ▷ Convert to 3×3 rotation
 9:     $\mathbf{p}_i \leftarrow -R_i^T \mathbf{t}_i$                          ▷ Camera center: $\mathbf{C} = -R^T \mathbf{t}$
10:     if camera_id($I_i$) == TOP_CAMERA then
11:         $\mathcal{C}_{\text{top}} \leftarrow \mathcal{C}_{\text{top}} \cup \{\mathbf{p}_i\}$
12:     else if camera_id($I_i$) == BOTTOM_CAMERA then
13:         $\mathcal{C}_{\text{bot}} \leftarrow \mathcal{C}_{\text{bot}} \cup \{\mathbf{p}_i\}$
14:     end if
15: end for
16:
17: Step 2: Compute Mean Camera Positions
18: $\mathbf{p}_{\text{top}} \leftarrow \frac{1}{|\mathcal{C}_{\text{top}}|} \sum_{\mathbf{p} \in \mathcal{C}_{\text{top}}} \mathbf{p}$          ▷ Mean position of top camera
19: $\mathbf{p}_{\text{bot}} \leftarrow \frac{1}{|\mathcal{C}_{\text{bot}}|} \sum_{\mathbf{p} \in \mathcal{C}_{\text{bot}}} \mathbf{p}$          ▷ Mean position of bottom camera
20:
21: Step 3: Compute Baseline in COLMAP Units
22: $b_{\text{COLMAP}} \leftarrow \|\mathbf{p}_{\text{top}} - \mathbf{p}_{\text{bot}}\|$          ▷ Euclidean distance
23: $b_{\text{COLMAP}} \leftarrow \sqrt{(p_{\text{top},x} - p_{\text{bot},x})^2 + (p_{\text{top},y} - p_{\text{bot},y})^2 + (p_{\text{top},z} - p_{\text{bot},z})^2}$
24:
25: Step 4: Compute Scaling Factor
26: $s \leftarrow \frac{b_{\text{real}}}{b_{\text{COLMAP}}} = \frac{30 \text{ cm}}{b_{\text{COLMAP}}}$          ▷ cm per COLMAP unit
27:
28: Step 5: Apply Scaling to Measurements
29:                     ▷ Linear measurements (lengths, diameters, distances):
30:         $x_{\text{cm}} \leftarrow x_{\text{COLMAP}} \times s$
31:                     ▷ Area measurements:
32:         $A_{\text{cm}^2} \leftarrow A_{\text{COLMAP}} \times s^2$
33:                     ▷ Volume measurements:
34:         $V_{\text{cm}^3} \leftarrow V_{\text{COLMAP}} \times s^3$
35: return $s$
```

A.5 Algorithm 5: Teasar Skeletonization

Algorithm 5 Teasar Skeletonization

```
Require: Mesh $M = \{V, E\}$, Invalidation radius $r_{inv}$
Ensure: Skeletal tree $T = \{\mathcal{N}, \mathcal{E}_{skel}\}$
 1: Initialize: $v_{root} \leftarrow \arg\min_{v \in V} v.y$, $\mathcal{N} \leftarrow \{v_{root}\}$, $\mathcal{E}_{skel} \leftarrow \emptyset$, $V_{covered} \leftarrow \{v_{root}\}$, $V_{uncovered} \leftarrow V \setminus \{v_{root}\}$
 2: while $V_{uncovered} \neq \emptyset$ do
 3:     Dijkstra wave: Initialize $d[v] \leftarrow \infty$, pred$[v] \leftarrow$ null for all $v \in V$; Set $d[n] \leftarrow 0$ for all $n \in \mathcal{N}$
 4:     $Q \leftarrow$ PriorityQueue($V$, key=$d$)
 5:     while $Q$ not empty do
 6:         $u \leftarrow Q$.extract_min()
 7:         for each neighbor $v$ of $u$ in $E$ do
 8:             if $d[u] + \|u - v\| < d[v]$ then
 9:                 $d[v] \leftarrow d[u] + \|u - v\|$; pred$[v] \leftarrow u$; $Q$.decrease_key($v$, $d[v]$)
10:             end if
11:         end for
12:     end while
13:     Find farthest: $v_{max} \leftarrow \arg\max_{v \in V_{uncovered}} d[v]$
14:     if $d[v_{max}] == \infty$ then break
15:     end if
16:     Backtrace path: $\mathcal{P} \leftarrow []$, $v \leftarrow v_{max}$
17:     while $v \notin \mathcal{N}$ do
18:         $\mathcal{P}$.append($v$); $v \leftarrow$ pred$[v]$
19:     end while
20:     $v_{parent} \leftarrow v$
21:     Add to skeleton: $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{P}$
22:     for $i = 0$ to len($\mathcal{P}$) - 2 do
23:         $\mathcal{E}_{skel} \leftarrow \mathcal{E}_{skel} \cup \{(\mathcal{P}[i], \mathcal{P}[i+1])\}$
24:     end for
25:     $\mathcal{E}_{skel} \leftarrow \mathcal{E}_{skel} \cup \{(v_{parent}, \mathcal{P}[-1])\}$
26:     Invalidate nearby: For each $v \in \mathcal{P}$, for each $u \in V_{uncovered}$: if $\|v - u\| < r_{inv}$ then $V_{covered} \leftarrow V_{covered} \cup \{u\}$, $V_{uncovered} \leftarrow V_{uncovered} \setminus \{u\}$
27: end while
28: $T \leftarrow$ construct_tree($\mathcal{N}$, $\mathcal{E}_{skel}$, $v_{root}$)
29: return $T$
```

A.6 Algorithm 6: Ray-Casting Diameter Measurement

```
Algorithm 6 Ray-Casting Diameter Measurement
Require: Mesh $M = \{V, F\}$, Skeleton tree $T$, Root tip nodes $\mathcal{N}_{tips}$
Ensure: Diameter statistics $(\bar{d}, \sigma_d)$
 1: Initialize: $\mathcal{D} \leftarrow []$; BVH $\leftarrow$ build_bvh($M$); $\mathcal{I} \leftarrow$ RayMeshIntersector($M$, BVH)
 2: for each root tip node $n_{tip} \in \mathcal{N}_{tips}$ do
 3:     $n_{branch} \leftarrow$ find_parent_branch_node($n_{tip}$, $T$)
 4:     $\mathbf{p}_{tip} \leftarrow [n_{tip}.x, n_{tip}.y, n_{tip}.z]^T$; $\mathbf{p}_{branch} \leftarrow [n_{branch}.x, n_{branch}.y, n_{branch}.z]^T$
 5:     $r_{sphere} \leftarrow 0.5 \times \|\mathbf{p}_{tip} - \mathbf{p}_{branch}\|$
 6:     Sample faces: $\mathcal{F}_{sample} \leftarrow \{f \in F : \|\text{centroid}(f) - \mathbf{p}_{tip}\| < r_{sphere}\}$
 7:     for each face $f \in \mathcal{F}_{sample}$ do
 8:         $\mathbf{c} \leftarrow$ centroid($f$); $\mathbf{n} \leftarrow$ face_normal($f$)
 9:         hits $\leftarrow \mathcal{I}$.intersect(Ray(origin=$\mathbf{c}$, direction=$-\mathbf{n}$), $M$)
10:         if len(hits) == 1 then
11:             $\mathcal{D}$.append($\|\mathbf{c}$ − hits[0].point$\|$)
12:         end if
13:     end for
14: end for
15: if len($\mathcal{D}$) ¿ 0 then
16:     $\bar{d} \leftarrow$ mean($\mathcal{D}$); $\sigma_d \leftarrow$ std($\mathcal{D}$)
17: else
18:     $\bar{d} \leftarrow$ NULL; $\sigma_d \leftarrow$ NULL
19: end if
20: return $(\bar{d}, \sigma_d)$
```

Table 3: COLMAP High-Quality Mapper Configuration Parameters

Parameter	**Value**	**Description**
`ba_local_max_num_iterations`	300	Maximum iterations for local bundle adjustment
`ba_global_max_num_iterations`	300	Maximum iterations for global bundle adjustment
`ba_local_num_images`	10	Number of images for local bundle adjustment
`ba_global_max_refinements`	50	Maximum refinement iterations for global BA
`ba_local_max_refinements`	50	Maximum refinement iterations for local BA
`init_min_num_inliers`	200	Minimum inliers required for initialization
`filter_max_reproj_error`	2.0	Maximum reprojection error threshold (pixels)
`max_reg_trials`	500	Maximum registration trials per image
`tri_re_max_trials`	500	Maximum triangulation trials
`ba_global_images_ratio`	1.1	Ratio of images for global bundle adjustment
`ba_global_points_ratio`	1.1	Ratio of points for global bundle adjustment

These conservative parameters prioritize reconstruction accuracy over speed, resulting in more reliable 3D models at the cost of longer processing times. The high iteration counts (300 for bundle adjustment) and strict inlier requirements (200 minimum) help ensure robust camera pose estimation and point triangulation, particularly important for the challenging cassava root geometries with varying texture densities.

B Pipeline Parameters

The reconstruction and measurement pipeline employs several critical parameters that were empirically optimized for cassava root reconstruction. Table 4 summarizes these key parameters and their rationale.

Parameter	**Value**	**Purpose / Rationale**
	Mesh Generation	
Alpha shape α	0.025	Surface reconstruction detail level; balances noise suppression with feature preservation for thin root structures
Smoothing iterations	3	Laplacian smoothing to reduce surface noise while maintaining shape
	Diameter Measurement	
Radius fraction	0.5	Sampling region size relative to tip-branch distance; ensures sampling within branch region
Ray intersection criterion	== 1 hit	Validity filter; accepts only rays penetrating completely through root cross-section
	Mass Estimation	
Assumed density	1.0 g/cm^3	Fresh root tissue density approximation for biomass estimation
	Point Cloud Filtering	
Color intensity threshold	0.5	Removes low-intensity (dark) points likely to be noise or shadows
Statistical outlier std_ratio	2.0	Standard deviation multiplier for statistical outlier removal
Statistical outlier neighbors	20	Number of neighbors for local density estimation
Radius outlier epsilon	0.02	Maximum distance for neighbor search in radius-based filtering

Table 4: Key technical parameters used throughout the 3D reconstruction and measurement pipeline. Values were empirically optimized for cassava root phenotyping.

These parameters remain fixed across all reconstructions to ensure consistency and reproducibility. The alpha shape parameter (0.025) and diameter radius fraction (0.5) were found through iterative testing to provide optimal results for cassava roots aged 5-10 weeks.

References

[1] Assefa B Amelework and Michael W Bairu. Advances in genetic analysis and breeding of cassava (manihot esculenta crantz): A review. *Plants*, 11(12):1617, 2022.

[2] Oliver Bimber, L Miguel Encarnaçáo, and Dieter Schmalstieg. Augmented reality with back-projection systems using transflective surfaces. In *Computer Graphics Forum*, volume 19, pages 161–168. Wiley Online Library, 2000.

[3] Blender Online Community. Blender - a 3d modelling and rendering package, 2024. Version 2.93+.

[4] Francesca Condorelli, Giovanni Pescarmona, and Ylenia Ricci. Photogrammetry and medieval architecture: Using black and white analogic photographs for reconstructing the foundations of the lost rood screen at santa croce, florence. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVI-M-1-2021*, 46(M-1-2021):141–146, 2021.

[5] Dawson-Haggerty et al. trimesh, 2019.

[6] Fritz A Francisco, Paul Nührenberg, and Alex Jordan. High-resolution, non-invasive animal tracking and reconstruction of local environment in aquatic ecosystems. *Movement ecology*, 8(1):1–12, 2020.

[7] Daniel Gatis. Rembg: Background removal tool. https://github.com/danielgatis/rembg, 2024.

[8] Peter J Gregory and Tobias Wojciechowski. Root systems of major tropical root and tuber crops: Root architecture, size, and growth and initiation of storage organs. *Advances in Agronomy*, 161:1–25, 2020.

[9] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[10] IK Hong, ST Chung, HK Kim, YB Kim, YD Son, and ZH Cho. Ultra fast symmetry and simd-based projection-backprojection (ssp) algorithm for 3-d pet image reconstruction. *IEEE transactions on medical imaging*, 26(6):789–803, 2007.

[11] Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan Liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Segment anything in high quality. In *NeurIPS*, 2023.

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[13] Christian Leubner, Christian Brockmann, and Heinrich Muller. Computer-vision-based human-computer interaction with a back projection wall using arm gestures. In *Proceedings 27th EUROMICRO Conference. 2001: A Net Odyssey*, pages 308–314. IEEE, 2001.

[14] Suxing Liu, Carlos Sherard Barrow, Meredith Hanlon, Jonathan P Lynch, and Alexander Bucksch. Dirt/3d: 3d root phenotyping for field-grown maize (zea mays). *Plant physiology*, 187(2):739–757, 2021.

[15] Suxing Liu, Wesley Paul Bonelli, Peter Pietrzyk, and Alexander Bucksch. Comparison of open-source three-dimensional reconstruction pipelines for maize-root phenotyping. *The Plant Phenome Journal*, 6(1):e20068, 2023.

[16] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[17] María Mercedes Morita, Daniel Alejandro Loaiza Carvajal, and Gabriel Mario Bilmes. New photogrammetric systems for easy low-cost 3d digitization of cultural heritage. In *Handbook of Cultural Heritage Analysis*, pages 1439–1464. Springer, 2022.

[18] K Mueller, X Zabulis, A Smolic, and T Wiegand. Evaluation of 3d reconstruction using multiview backprojection. *ICOB*, 2005.

[19] 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.

[20] Yan San Woo, Zhuguang Li, Shun Tamura, Prawit Buayai, Hiromitsu Nishizaki, Koji Makino, Latifah Munirah Kamarudin, and Xiaoyang Mao. 3d grape bunch model reconstruction from 2d images. *Computers and Electronics in Agriculture*, 215:108328, 2023.

[21] Mie Sato, Ingmar Bitter, Michael A Bender, Arie E Kaufman, and Masayuki Nakajima. Teasar: tree-structure extraction algorithm for accurate and robust skeletons. In *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, pages 281–449. IEEE, 2000.

[22] Philipp Schlegel. skeletor: Python library for mesh skeletonization, 2021.

[23] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[25] Nyerhovwo John Tonukari. Cassava and the future of starch. *Electronic journal of biotechnology*, 7(1):5–8, 2004.

[26] Anup Vibhute, Shrikant K Bodhe, et al. Applications of image processing in agriculture: a survey. *International Journal of Computer Applications*, 52(2):34–40, 2012.

[27] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. Embree: a kernel framework for efficient cpu ray tracing. In *ACM Transactions on Graphics (TOG)*, volume 33, pages 1–8. ACM New York, NY, USA, 2014.

[28] Jens Wilhelm, Tobias Wojciechowski, Johannes A Postma, Dirk Jollet, Kathrin Heinz, Vera Böckem, and Mark Müller-Linow. Assessing the storage root development of cassava with a new analysis tool. *Plant Phenomics*, 2022.

[29] Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

[30] Peng Zheng, Dehong Gao, Deng-Ping Fan, Li Liu, Jorma Laaksonen, Wanli Ouyang, and Nicu Sebe. Bilateral reference for high-resolution dichotomous image segmentation. *arXiv preprint arXiv:2401.03407*, 2024.

[31] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.