EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

LeafFit: Plant Assets Creation from 3D Gaussian Splatting

Chang Luo and Nobuyuki Umetani

The University of Tokyo, Japan

Data size: 11.9 Mb
#Primitives: 47k
FPS: 936
(a) Input 3D Gaussian splatting model

Data size: 26.2 Mb
#Vertices: 623k
FPS: 4,384
(b) Mesh extracted using implicit surface

Data size: 0.75 Mb
#Vertices: 2.0k
FPS: 14,694
(c) Our templated mesh model

Figure 1: *Leveraging the repetition of leaf shapes, LeafFit reuses a single template leave across all the leaves in a plant. From an input 3D Gaussian splatting (3DGS) capture (a) to produce single-sided, instanced meshes that preserve thin-leaf detail (c) while drastically reducing storage and boosting frame per second. Our mesh is more compact than the mesh from implicit surface conversion of input 3DGS (b). The results are game-ready, fully editable assets, where textures and per-leaf geometry can be calculated in real time.*

Abstract

We propose LeafFit, a pipeline that converts 3D Gaussian Splatting (3DGS) of individual plants into editable, instanced mesh assets. While 3DGS faithfully captures complex foliage, its high memory footprint and lack of mesh topology make it incompatible with traditional game production workflows. We address this by leveraging the repetition of leaf shapes; our method segments leaves from the unstructured 3DGS, with optional user interaction included as a fallback. A representative leaf group is selected and converted into a thin, sharp mesh to serve as a template; this template is then fitted to all other leaves via differentiable Moving Least Squares (MLS) deformation. At runtime, the deformation is evaluated efficiently on-the-fly using a vertex shader to minimize storage requirements. Experiments demonstrate that LeafFit achieves higher segmentation quality and deformation accuracy than recent baselines while significantly reducing data size and enabling parameter-level editing. Our source code is publicly available at https://github.com/netbeifeng/leaf_fit.

CCS Concepts

• ***Computing methodologies*** → *Shape modeling; Graphics systems and interfaces;*

1. Introduction

Plants are ubiquitous and diverse in natural environments. However, their assets in virtual scenes are still largely created by hand, which is a slow and labor-intensive process that requires significant expertise. In particular, the rich morphological variations, such as stem height and leaf sizes, make replicating each plant manually difficult to handle. 3D Gaussian Splatting (3DGS) [KKLD23] offers high-fidelity capture with efficient rendering, and it is par-

ticularly effective for vegetation with thin leaves and intricate silhouettes. Despite their visual quality, Gaussian reconstructions of plants see limited practical adoption in asset production. First, representing complex leaf structures faithfully using 3DGS requires large memory storage. This size challenge can be significantly mitigated by exploiting the repetition of leaf instances within an individual plant. However, similar leaves within the same reconstruction are neither identified nor reused. Secondly, production workflows and game engines are typically optimized for meshes and textures. Therefore, raw 3DGS primitives cannot natively benefit from standard hardware acceleration pipelines. Conversion of Gaussian splatting into mesh with texture is possible using implicit surfaces [GL24, HYC*24, YSG24]. However, these methods using implicit surfaces are not suitable for thin structures, the implicit representation tends to thicken thin sheets, cannot represent the front and back sides of the mesh with different textures, and produces overly dense geometry (see Fig. 1-b). This difficulty motivates a representation that factors a plant into leaf instances, aligns them non-rigidly to a shared template, and extracts a lightweight surface that preserves per-leaf variation while avoiding redundancy. Instance-retrieval approaches that ground to external models can segment repeated objects [VMG*25]. Still, they are effective mainly when instances are nearly identical and do not account for subtle geometric differences that commonly appear across leaves of the same species.

To address these limitations, we propose a pipeline that factors plants into leaf instances and reuses a single explicit template. Our system comprises three components: (i) automatic and manual segmentation of Gaussian splatting of a plant into leaf instances; (ii) non-rigid registration of leaves to preserve their shape variation; and (iii) extraction of a thin, lightweight template mesh and texture from Gaussian primitives. Technically, we present a robust leaf segmentation method based on the geodesic distances computed on the Gaussian primitives. This approach is fast and robust across various plant specimens, as it does not require a training dataset. Furthermore, we propose a differentiable moving least squares method that allows fitting a 3D Gaussian splatting of a leaf to another.

We evaluate our approach qualitatively by comparing leaf instance overlays and renderings against manual annotations and recent Gaussian-to-mesh baselines. Quantitatively, our method improves segmentation accuracy and deformation fidelity while producing lighter meshes that better preserve thin structures, and it supports real-time online evaluation from compact per-leaf parameters. Our technical contributions include:

- **Instance-aware leaf segmentation:** a robust workflow that separates leaves into instances, combining automatic geodesic-based segmentation.
- **Template-driven non-rigid alignment:** a differentiable MLS formulation that registers each leaf to a shared template; per-leaf alignment parameters are optimized and stored once, then evaluated online to generate deformed geometry at render time.
- **Lightweight surface extraction and appearance transfer:** a template mesh tailored to thin leaves via point-based reconstruction with seamless texture mapping, whose geometry and appearance are reused across leaves to avoid redundancy while preserving variation.

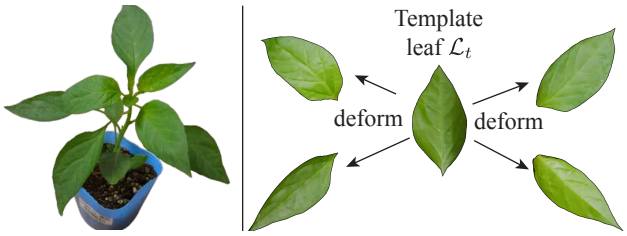


Figure 2: *No two leaves belongs to a plant are exactly the same, but they are similar. We compress the data by replacing the leaves by deformed template leaf.*

2. Related Works

2.1. Game Asset Creation for Plants

Game asset creation for vegetation has traditionally relied on manual modeling and sculpting in digital content creation tools (e.g., Maya, Blender [Aut, Com18]), which is highly labor-intensive and demands substantial artistic expertise. To reduce this burden, computer-aided approaches such as procedural methods [PL12, IOI06, SPK*14, GXY*18, TLL*11] and dedicated toolchains [Spe17, PBN*09] have been widely developed and adopted. Notably, Lee et al. [LLB23] leverage a Transformer-based architecture for controllable tree generation, whereas template-based approaches [ZLB*25] reconstruct forest scenes by retrieving and ranking models from a database. Although these methods can automatically generate lightweight, well-structured polygon meshes that are suitable for real-time rendering and can be seamlessly integrated into game engines, most of these approaches primarily focus on generating branch structures, while leaves are often only randomly scattered across the skeleton. To compactly represent plant's leaves, parametric templates using Bézier curves are presented [BNB13, CB17]. However, handling deformations between two Bézier-parametrized leaves remains challenging, as the direct deformation is often infeasible due to mismatches in control point configurations, making it difficult to establish a valid mapping without resampling. More recently, [YDH*25] leverages neural networks to learn latent embeddings of leaf shape, texture, and deformation, enabling high-quality reconstructions but requiring training on large-scale leaf datasets. In contrast, our work avoids the need for training feedforward networks on massive datasets.

2.2. 3D Representations for Plant Phenotyping

A wide range of 3D representations have been explored for plant reconstruction, particularly in the context of plant phenotyping and digital twin creation [PSB*12, WHW*13, Pau19]. For comprehensive surveys, we refer readers to [Oku22] for general 3D plant modeling and reconstruction, and to [SWWG25] specifically for point cloud segmentation and organ identification. Explicit methods such as meshes and point clouds are widely adopted in industry, where dense reconstructions can be obtained via Structure-from-Motion (SfM) [SF16] and Multi-View Stereo (MVS) [SZPF16]. While classic pipelines can tackle with large-scale reconstructions, they often suffer from geometric noise, require careful multi-camera setups, and struggle with thin structures such as leaves [NST*16,

ACFQ*18, LOL*20]. Manual modeling, in contrast, achieves high-quality results but is extremely labor-intensive [PBN*09, Spe17]. More recently, implicit neural methods such as Neural Radiance Fields (NeRF) [MST*20] have shown impressive capabilities in synthesizing photorealistic views by representing scenes as continuous volumetric functions [HYP*24, LQN*25, MGSS24]. However, NeRFs are slow to train and memory-intensive, the implicit nature makes it difficult to extract explicit structures or support fine-grained editing [HTE*23]. Accelerations such as hash grid [MESK22] improve performance but do not fundamentally resolve these limitations.

Most recently, 3D Gaussian splatting (3DGS) [KKLD23] has emerged as a breakthrough representation that combines the visual fidelity of implicit methods with the efficiency and explicitness required for downstream use. By rasterizing anisotropic Gaussian primitives with learned color, opacity, and covariance parameters, 3DGS achieves real-time rendering quality that was previously out of reach. Building on its success, subsequent works have extended Gaussian kernel with more diverse kernel formulations [HYC*24, HLS*25], enabled direct editing [WFZ*24], repetition localization and replacement [VMG*25], and explored integration with game asset generation pipelines [LRX*24].

Unlike implicit neural fields, the discrete and spatially localized nature of Gaussian splatting makes them inherently well-suited for selection, manipulation, and user interaction. These properties are particularly advantageous for plant modeling, where fine-scale, repetitive structures need to be accurately extracted, aligned, and instanced [OLMS24, SJD*25]. In this work, we adopt Gaussian Splatting as the foundation for representing plants due to its efficiency, editability, and compatibility with both real capture pipelines and lightweight game asset workflows.

2.3. Interactive Editing of Gaussian Representations

For editing or interacting with Gaussian-reconstructed scenes, selection and segmentation form the foundation. A number of recent efforts [WFZ*24, YLS*24, XCH*25] explore scene editing guided by user prompts, where text input is used to assist object selection or deletion within the scene. Similarly, [JMG24] employs sparse point clicks or sketches in combination with a graph-cut construction over Gaussians to achieve interactive selection. In contrast, [PS25] provides more concrete manual selection tools, offering screen-space pickers such as lasso and brush, as well as 3D pickers such as sphere and box selection, enabling accurate control. However, these approaches can be time-consuming, as users often need to rotate the scene from multiple angles to complete the selection. None of the above-mentioned methods are specialized for plant leaf segmentation. In our work, we introduce an automatic segmentation algorithm based on the heat distance method [CWW17], leveraging the strong prior of leaf shape. We also provide a fallback interactive manual segmentation tool, also guided by the heat distance method, to handle challenging cases where automatic segmentation may fail.

Once Gaussian primitives or object parts have been reliably selected, the next step in interactive editing is to deform or align them to achieve the desired geometry. Several efforts explore Gaussian deformation through cage-based control [HXYL24], sketch input [XABP24], or physics-inspired constraints [JYX*24], but these methods mainly focus on in-place deformation. More recent studies investigate registration of Gaussian representations, either by directly exploiting Gaussian parameters for fast alignment [CXL*24] or by extending iterative closest point (ICP) [SHT09] into SLAM systems [HYY24], spline arc length parameterized non-rigid deformation [PHT*25]. In the broader context of point-to-mesh alignment, hierarchical strategies combined with As-Rigid-As-Possible (ARAP) energy have been proposed to robustly register meshes to unstructured point clouds [BCDD22]. While these works demonstrate the feasibility of Gaussian-based manipulation, our method leverages differentiable moving least squares (MLS) [SMW06, ZG07] deformation, enabling robust inter-leaf alignment and providing a basis for mesh extraction tailored to vegetation assets.

2.4. Surface Reconstruction from Gaussian Splatting

Although learned Gaussians can be directly imported into a game scene, the representation is heavy and redundant due to its unstructured point-based nature. Despite strong visual fidelity, Gaussians remain image-space oriented for vegetated scenes. With lacking explicit topology, do not encode semantic part structure (e.g., individual leaves), and offering limited support for repetition and instancing that are common in plants. As a result, direct deployment as game assets is hindered by memory footprint, platform support, and the absence of mesh-level controls required in asset pipelines.

Existing pipelines [GL24, HYC*24, YSG24] obtain meshes from Gaussians by binding them to implicit fields and applying isosurface extraction. Typically, Gaussian opacity or density is converted into a continuous signed distance field [PFS*19], and a surface is then extracted with marching cubes [LC98]. This approach is general but often produces overly dense and thick geometry for thin sheets such as leaves, and is prone to staircasing or blobby artifacts. Subsequent decimation can reduce fidelity, and the outputs remain too heavy for real-time use. Moreover, implicit extraction is highly sensitive to query resolution, and struggles with thin structures, frequently generating inflated surfaces around the true geometry. In contrast, we incorporate shape priors and employ the ball pivoting algorithm (BPA) [BMR*02] to reconstruct thin and lightweight template leaf meshes directly from Gaussians. This preserves per-leaf individuality while supporting instancing, yielding game-ready assets that are both efficient and faithful to observed structures, surpassing implicit-field isosurfaces in compactness and procedural synthesis in realism.

3. Methods

Input and Output The overall workflow of our method is shown in Figure 3. Let $\mathcal{G} = \{g_i\}_{i=1}^{|\mathcal{G}|}$ be an input 3D Gaussian splatting scene [KKLD23] of a single plant, where each Gaussian primitive g_i stores a 3D center, covariance, opacity, and view-dependent color coefficients. Note that this scene contains only a single plant, as we trim out the background beforehand. Then, we segment the input Gaussian primitives $\mathcal{G}$ into per-leaf groups $\{\mathcal{L}_j\}_{j=1}^{|\mathcal{L}|}$ as described in Sec. 3.1. The user chooses one group as the template $\mathcal{L}_t$. The outputs are: (i) a template mesh $M = (V, F)$ reconstructed from $\mathcal{L}_t$, (ii)

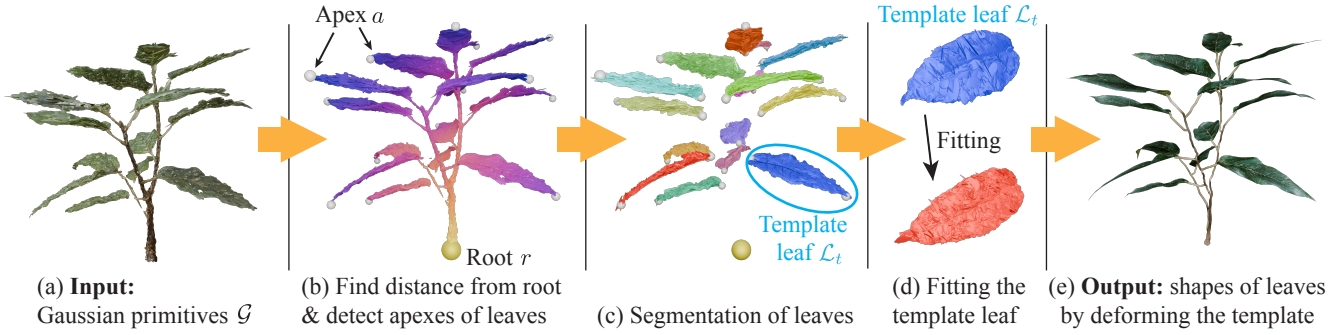


Figure 3: ***Workflow.*** *(a) Input: a pre-trained set of Gaussian primitives $\mathcal{G}$. (b) The user selects a root primitive r; we compute geodesic distances from r, detect apexes a as local maxima, and build a tree. The tree graph is used to find leaf petiole. (c) Using petiole and apex cues, we segment leaves and let the user choose a template leaf $\mathcal{L}_t$. (d) Other leaves are fitted to $\mathcal{L}_t$ via Moving Least Squares deformation. (e) Output: an efficient, editable instanced mesh suitable for game assets.*

position of the control points for all the leaves $\{C_j\}_{j=1}^{|\mathcal{L}|}$. By moving the control points of the template leaf C_t to target leaf $C_j, \{j \neq t\}$, we can define the moving least squares (MLS) deformation field Φ_j that moves the vertices of the template mesh such that the template mesh fits the shape of the leaf (see Sec. 3.2). The mesh extraction procedure is described in Sec. 3.3. We do not store per-leaf meshes; instead, they are generated on the fly by applying the MLS deformation induced by C_j to V at loading time or in real-time using a vertex shader.

3.1. Leaf Instance Segmentation

We denote each segmented leaf as $\mathcal{L}_j$, and the remaining Gaussian primitives that do not belong to any leaf segment are grouped as the stem $\mathcal{S}$. Together they form a partition of the plant Gaussians $\mathcal{G} = \{\mathcal{L}_j\} \cup \mathcal{S}$. Segmentation is achieved by combining automatic detection based on geodesic distances with optional manual refinement.

Distance from Root From the Gaussian splatting of a scene $\mathcal{G}$, the user selects the Gaussian r that is closest to the root. This root Gaussian is the source for the distance field computation using heat propagation [CWW17]. Note that we regard a Gaussian splat as a point cloud using only the center coordinates of the Gaussian primitives. As a result, we obtain the geodesic distance defined at each Gaussian $D_r[i]$, where $D_r[r] = 0$ and values increase smoothly towards the leaf apex (i.e., leaf tip).

Leaf Apexes To segment the leaf, we first detect leaf apexes by computing the local maxima of distance from the root $D_r[\cdot]$. However, because each leaf is represented by many non-uniformly distributed Gaussian primitives, directly identifying local maxima on the dense input is prone to noise. Hence, to stabilize the computation, we first apply farthest point sampling to sample the Gaussian primitives uniformly (see Fig. 4). We denote the set of sampled Gaussian primitives as $\bar{\mathcal{G}} \subset \mathcal{G}$. The sample size $N_s = |\bar{\mathcal{G}}|$ is chosen empirically to ensure sufficient density for robust topological analysis. Note that farthest point sampling tends to preserve points at the boundaries, effectively retaining potential leaf tips. We further compute N_k nearest neighbors [FBF77] among the sampled Gaussian primitives $\bar{\mathcal{G}}$ using the Euclidean distance. A sampled Gaussian i is registered as a candidate apex if it is a strict local geodesic maximum, while all its N_k Euclidean neighbors j satisfy $D_r[j] < D_r[i]$.

Tree Construction A leaf can have multiple local maxima, so a single apex does not directly correspond to a unique leaf. To analyze the connectivity of each leaf, we construct a tree that connects the root to the apexes, where its branches consist of sampled Gaussian primitives. We trace a path connecting the sparsely sampled Gaussians $\bar{\mathcal{G}}$ from the apex a to the root r, denoted as $\mathcal{P}_a = \{p_k \mid p_k \in \bar{\mathcal{G}}, p_1 = a, p_{|\mathcal{P}|} = r\}$. The path follows the steepest descent of the geodesic distance D_r:

$$p_{k+1} = \arg\min_{g \in \mathcal{N}(p_k)} D_r[g]. \tag{1}$$

When paths from different apexes intersect, their shared Gaussian is marked as a junction of the tree. To avoid premature joins caused by noise (e.g., paths accidentally touching at boundaries), we adopt a *deferred-merge rule*. Specifically, if the neighborhood of the current point p_k contains a node labeled as visited by another path, we do not terminate immediately. Instead, we continue one step further along the steepest descent to p_{k+1}; we merge the paths only if the

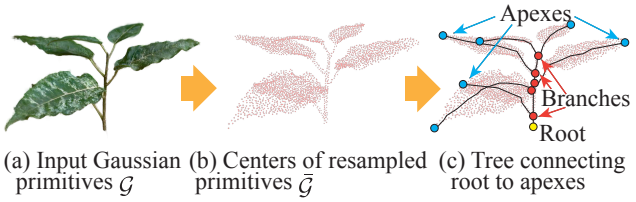


Figure 4: ***Tree-graph construction.*** *(a) Start from the full set of Gaussians $\mathcal{G}$. (b) Downsample to a sparse subset $\bar{\mathcal{G}}$ to reduce the cost of detecting apexes via local maxima on the geodesic field (from the root r). (c) For each apex a, trace a rootward path by greedy descent on geodesic distance; when it meets an existing path, attach it as a branch to complete the tree.*

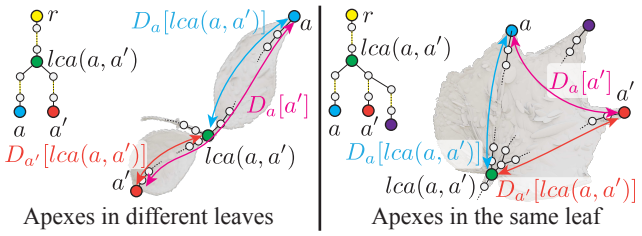


Figure 5: ***Separating apex pairs via triangle inequality.*** *Given two apexes a and a' with lowest common ancestor* $\mathrm{lca}(a,a')$ *on the rootward tree, let $D_a[\cdot]$ and $D_{a'}[\cdot]$ denote geodesic distances from a and a', respectively.* ***Left:*** *Apexes on different leaves satisfy* $D_a[a'] \approx D_a[\mathrm{lca}(a,a')] + D_{a'}[\mathrm{lca}(a,a')]$. ***Right:*** *Apexes on the same leaf yield a strictly shorter direct path,* $D_a[a'] < D_a[\mathrm{lca}(a,a')] + D_{a'}[\mathrm{lca}(a,a')] - \tau$. *We use this criterion to group apexes into leaves.*

neighborhood of p_{k+1} also contains visited vertices. This ensures that the merge occurs well within the shared branch structure rather than at a noisy interface.

Grouping Apexes of a Leaf For complex leaves such as maple, multiple apexes exist on a single leaf (see Fig. 5-right). Hence, we group apexes that belong to the same leaf. First, we compute the geodesic distance from each apex $D_a[\cdot]$. Similar to the distance from root D_r, the distance from apex is computed for all the Gaussian primitives $\mathcal{G}$ using the heat distance method [CWW17]. To check if two apexes a and a' belong to the same leaf, we compare the direct geodesic distance $D_a[a']$ against the path through their common tree structure. First, we identify the lowest common ancestor (LCA) node in the tree, denoted as $\mathrm{lca}(a,a') \in \bar{\mathcal{G}}$. According to the triangle inequality, the direct path is shorter or equal to the path via the LCA. Specifically, we consider two apexes to be on the *same* leaf if the direct geodesic distance is strictly shorter than the tree-based path by a margin τ:

$$D_a[a'] < D_a[\mathrm{lca}(a,a')] + D_{a'}[\mathrm{lca}(a,a')] - \tau, \tag{2}$$

where τ is a distance margin parameter. Otherwise, if the left and right hand sides are nearly equal (within τ), the path must traverse the stem, implying the apexes belong to different leaves. At the end, we construct an adjacency graph where edges connect apexes satisfying Eq. (2), and final leaf instances are obtained as the connected components of this graph.

Leaf Segmentation Once leaf apexes are grouped into leaf instances, the leaf petioles can be determined. For each leaf instance, we designate the apex with the largest geodesic distance from the root as the primary tip a. To locate the leaf petiole, we traverse the pre-computed path $\mathcal{P}_a$ from the tip a towards the root. At each path point $p_k \in \mathcal{P}_a$, we estimate the local leaf diameter by analyzing a geodesic slice. Specifically, we query the set of Gaussian primitives whose geodesic distance from the apex, $D_a[\cdot]$, falls within a narrow iso-geodesic ring of width δ centered at $D_a[p_k]$. We denote this local band $\mathcal{G}_{p_k}$ as:

$$\mathcal{G}_{p_k} = \left\{ g \mid |D_a[g] - D_a[p_k]| \le \frac{\delta}{2} \right\}. \tag{3}$$

The local diameter d_{p_k} is defined as the maximum Euclidean distance between the path node p_k and the retrieved primitives in the set $\mathcal{G}_{p_k}$, formulated as:

$$d_{p_k} = \max_{g \in \mathcal{G}_{p_k}} \|p_k - g\|_2. \tag{4}$$

As we march rootward, the iteration over path will be terminated when the diameter d_{p_k} drops below a threshold ϵ, indicating the petiole. Crucially, to prevent premature termination at narrow leaf tips (e.g., in elongated leaves), we apply an *early protection window* controlled by a ratio ρ: base detection is disabled within the initial fraction ρ of the path segment from a to the first branching junction in the tree graph. All primitives with $D_a[\cdot]$ smaller than the determined base distance are assigned to the leaf.

Manual Segmentation In addition to the automatic segmentation described above, we provide an interactive tools with two selection methods, *drag* and *brush*, to refine the results (see Figure 7). These tools are particularly effective for correcting segmentation in noisy regions or for handling structurally complex plants where automatic methods may fail.

For the *drag tool*, the user clicks on a leaf apex and drags the cursor toward the base. Upon the initial button press, we identify the source Gaussian $p \in \mathcal{G}$ via ray-casting. This point serves as the source for a real-time geodesic distance field computation $D_p[\cdot]$. As the mouse is dragged, we continuously update the target Gaussian $d \in \mathcal{G}$ under the cursor. Any Gaussian primitive g is highlighted and selected if it lies within the geodesic radius defined by the current cursor position, i.e., $D_p[g] \le D_p[d]$.

On the other hand, by using the *brush tool*, users select Gaussian primitives within an adjustable geodesic radius r centered at the Gaussian d under the cursor. Specifically, we compute the heat geodesic distance from d and select any primitive g satisfying $D_d[g] \le r$. To ensure real-time performance for these ray-Gaussian intersections, we utilize a Bounding Volume Hierarchy (BVH) structure constructed on the Gaussian primitives. Once all leaf instances are segmented, the remaining unassigned primitives are grouped as the stem segment $\mathcal{S}$.

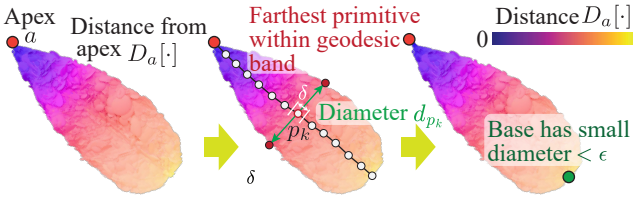
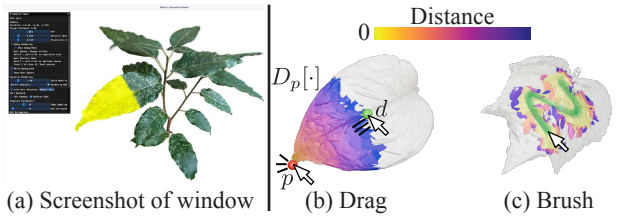


Figure 6: ***Leaf base determination.*** *For each apex a, we compute the geodesic distances $D_a[\cdot]$ and traverse the rootward path $\mathcal{P}_a$. At each path point p_k, we query the set of primitives $\mathcal{G}_{p_k}$ within an iso-geodesic band of width δ to compute the local diameter d_{p_k}. The traversal terminates when d_{p_k} drops below the threshold ϵ, marking the petiole.*



(a) Screenshot of window (b) Drag (c) Brush

Figure 7: ***Manual segmentation tools.*** *(a) GUI for editing Gaussian primitives; selected primitives are highlighted in yellow. (b)* ***Drag selection:*** *Selection expands based on the geodesic distance from the apex. (c)* ***Brush selection:*** *Primitives within a fixed geodesic radius around the cursor are selected.*

3.2. Leaf Registration using Moving Least Squares

Once all the leaves $\{\mathcal{L}_j\}$ are segmented, we first denoise the leaf point clouds using a moving least squares (MLS) projection. Specifically, we locally fit a plane to the center of the Gaussian primitives and remove outliers that lie far from this surface [AK04]. This filtering yields a clean leaf point set for stable rendering and downstream alignment. The user then selects a template leaf $\mathcal{L}_t$ to serve as the template leaf for alignment.

Rasterization of Leaves We employ the 2D rasterizer from the Gaussian splatting pipeline [KKLD23] to render each segmented leaf and extract its depth and color in image space. To maximize projection quality, each leaf is first PCA-aligned so that its principal plane coincides with the xy-plane and its normal points along the positive z-axis. This preprocessing improves both the depth computation for registration and the subsequent texture sampling of the template mesh.

We use farthest point sampling on each leaf's Gaussian primitives $\mathcal{L}_j$ to generate a sparse set of control points $C_j \in \mathbb{R}^{K\times 3}$, where K is the sampled control points amount per leaf. Given the two sampled control points set $C_t \subset \mathcal{L}_t$ (for template leaf) and $C_j \subset \mathcal{L}_j$ (for target leaf), we need firstly establish a one-to-one correspondence between their control points C_t and C_j, as an initial guessed correspondence is required for starting the optimization of MLS deformation. This initial correspondence assignment is solved by minimizing the global correspondence cost using the Jonker-Volgenant algorithm [Cro16].

From this initialization, we optimize the target control point positions C_j to fit the template leaf to the target leaf. The MLS deformation field $\Phi_j : \mathbb{R}^3 \to \mathbb{R}^3$ is parameterized by the source C_t and target C_j control handles. First, we compare the depth images of the deformed template and the target leaf rendered from the PCA-aligned view. Let D_{Φ_j} and $D_{\mathcal{L}_j}$ denote the depth maps of $\Phi_j(\mathcal{L}_t)$ and $\mathcal{L}_j$, respectively. The depth loss is defined as:

$$L_{\text{depth}} = \frac{1}{|\Omega|}\sum_{u\in\Omega}\left|D_{\Phi_j}(u) - D_{\mathcal{L}_j}(u)\right|^2, \tag{5}$$

where Ω is the set of foreground pixels. Second, we regularize the alignment in 3D space using a bidirectional Chamfer distance:

$$\begin{aligned} L_{\text{chamfer}} = &\frac{1}{|\mathcal{L}_t|}\sum_{p\in\mathcal{L}_t}\min_{q\in\mathcal{L}_j}\|\Phi_j(p)-q\|^2 \\ &+\frac{1}{|\mathcal{L}_j|}\sum_{q\in\mathcal{L}_j}\min_{p\in\mathcal{L}_t}\|\Phi_j(p)-q\|^2. \end{aligned} \tag{6}$$

Finally, we solve for the optimal control points C_j by minimizing the composite objective:

$$\min_{C_j}\; L_{\text{depth}}(C_j) + \lambda L_{\text{chamfer}}(C_j), \tag{7}$$

where L_{depth} enforces depth-map consistency in the PCA-aligned view and L_{chamfer} promotes 3D geometric agreement.

GPU instancing and memory footprint. At runtime, we store only the template mesh (V,F), its texture, and per-leaf kernel data of size $K\times N$. MLS deformation is evaluated on the fly in a vertex shader, which writes deformed positions to a GPU buffer consumed by the vertex stage. This keeps CPU cost negligible and avoids duplicating geometry; memory scales as $O(|V|)$ shared $+O(KN)$ per plant, while the per-leaf compute is $O(|V|K)$ and parallelizable.

3.3. Template Leaf Mesh Extraction

The deformation computed in the Gaussian domain can be directly reused to deform mesh vertices, enabling a consistent representation across both modalities. Since Gaussian splats are not yet widely supported in many content creation pipelines and game engines, we extend our approach to produce explicit meshes. Only the template leaf needs to be meshed; all other leaves are then obtained by applying the previously estimated MLS deformations to this template.

Surface Topology Reconstruction Because leaves have thin, sheet-like structures, implicit isosurface extraction methods often fail to capture their geometry reliably, tending to produce inflated artifacts. Instead, we downsample the template leaf Gaussians using farthest point sampling to obtain a representative point cloud, and reconstruct a watertight surface with the ball pivoting algorithm (BPA) [BMR*02]. While BPA is known at recovering thin surfaces, it is sensitive to the non-uniform density of Gaussian primitives, which can result in small topological gaps. We therefore employ post-processing to repair and fill holes [Lie03], yielding a clean triangular mesh $M = (V,F)$ suitable for further deformation.

Texture Extraction For texture computation, we first align the leaf using PCA, similar to the depth image computation. Then, we render the leaf from the back and front sides. To support standard game engine rendering, each original triangle is duplicated into a front–back pair with separate UV patches. Vertices are also duplicated per side with split normals to prevent UV conflicts. This construction ensures spatial consistency between the Gaussian appearance model and the extracted mesh textures, allowing for seamless transfer of visual detail from 3D Gaussian splatting to conventional mesh-based pipelines.

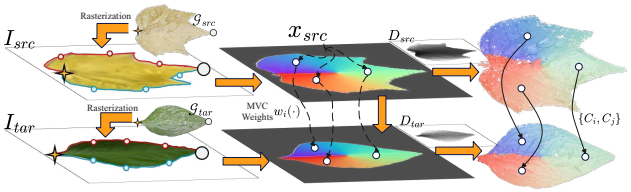


Figure 8: ***Correspondence for leaf retargeting across species.*** *Orthographic renders* $(I_{\text{src}}, I_{\text{tar}})$ *with depths* $(D_{\text{src}}, D_{\text{tar}})$ *produce silhouettes that define source/target cages. Interior samples are transferred by MVC and back-projected with depth to form dense 3D pairs* $\{(C_i, C_j)\}$ *that drive MLS retargeting.*

Stem Generation As mentioned in Sec. 3.1, we construct a tree by connecting the resampled Gaussian primitives $\tilde{\mathcal{G}}$ from apexes to the root r. We generate the stem geometry by replacing the edges of the tree that do not belong to the leaf with cylinders. In our current implementation, we determine the radius of the cylinders using Leonardo's formula [Son88], i.e., the cross-sectional area of a trunk is equal to the sum of the cross-sectional areas of its branches. The user specifies the radius of the root and that of the petioles to ground the interpolation via GUI. This paper focuses on the instantiation of the leaves, and a more accurate reconstruction of the stem based on Gaussian primitives is future work.

3.4. Post Editing of Plant

Our scene is represented by a *single* template mesh driven by a sparse set of control points via MLS deformation. This design decouples appearance from geometry, allowing both to be easily edited while remaining efficient.

Texture editing Since every leaf is a deformation of the same template mesh, we can edit the template UV texture once and propagate the change to all instances through the MLS warp. In practice, we apply masked inpainting or image-generation models to the template texture (standard 2D editing), which is significantly more efficient than operating directly on a dense Gaussian field. We demonstrate two editing workflows in Fig. 9: (1) manual composition for input mesh (a), where users overlay external images or decals onto the texture, and (2) full texture regeneration for input mesh (c), leveraging generative AI services to completely alter the plant's appearance style.

Geometry retargeting across species Straightforward 3D correspondence matching via farthest point sampling can be brittle for large shape gaps, so we adopt a 2D cage–transfer strategy (see Fig. 8). We orthographically render source and target to obtain images $I_{\text{src}}, I_{\text{tar}}$ and depths $D_{\text{src}}, D_{\text{tar}}$. Using the segmentation results, we trace the leaf silhouettes from the apex to the petiole base. By uniformly sampling along these left and right boundaries, we construct a 2D cage surrounding each leaf. Interior samples x_{src} are drawn inside the source cage (via farthest point sampling); their mean value coordinates (MVC) [Flo03] $w_i(x_{\text{src}})$ are evaluated w.r.t. the source cage and used to transfer them to the target cage:

$$x_{\text{tar}} = \frac{\sum_i w_i(x_{\text{src}})\, b_i^{\text{tar}}}{\sum_i w_i(x_{\text{src}})}. \tag{8}$$

Finally, back-projecting $(x_{\text{src}}, x_{\text{tar}})$ with their respective depth values $(D_{\text{src}}, D_{\text{tar}})$ yields dense 3D correspondence pairs $\{(C_i, C_j)\}$, which we use to fit an MLS deformation for robust retargeting.

4. Results

We evaluate our approach across four axes: (i) instance segmentation, (ii) mesh quality and compactness, (iii) non-rigid deformation accuracy, and (iv) editability demonstrated via texture or geometry editing. Quantitative metrics and qualitative visualizations are provided in the following.

Segmentation Evaluation We evaluate across diverse species and leaf morphologies. Manual annotations serve as ground truth; qualitative comparisons are in Fig. 10 and quantitative results in Tab. 1. Segmentation comparisons show clean separation of overlapping leaves and suppression of stray stem regions. We report Accuracy (Acc), mean IoU (mIoU), mean F1 (mF1), Panoptic Quality (PQ), and wall-clock time (mean $\pm$ std across plants). PQ [KHG*19] measures joint detection+mask quality and, when the stem is a class, specifically penalizes leaf–stem confusions. Baselines include: (i) a *training-free* Euclidean-density heuristic on a k-NN graph, which isolates the stem based on low density ρ and high gradient $\|\nabla\rho\|$ (identifying the base at the last pre-minimum along the rootward path); and (ii) a *learned* Point Transformer v3 [WJW*24], trained on the PLANesT-3D dataset [MŞS*24] and evaluated on our data.

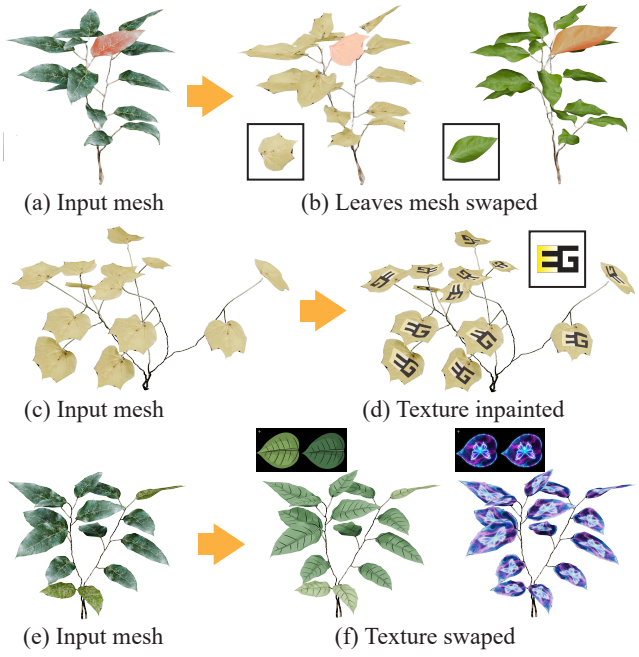


Figure 9: ***Editing and retargeting.*** *Geometry retargeting from a reconstructed rubber-tree leaf to target leaves (top), and texture edits that propagate to all instances, including image inpainting overlay (middle) and full texture replacement (bottom).*

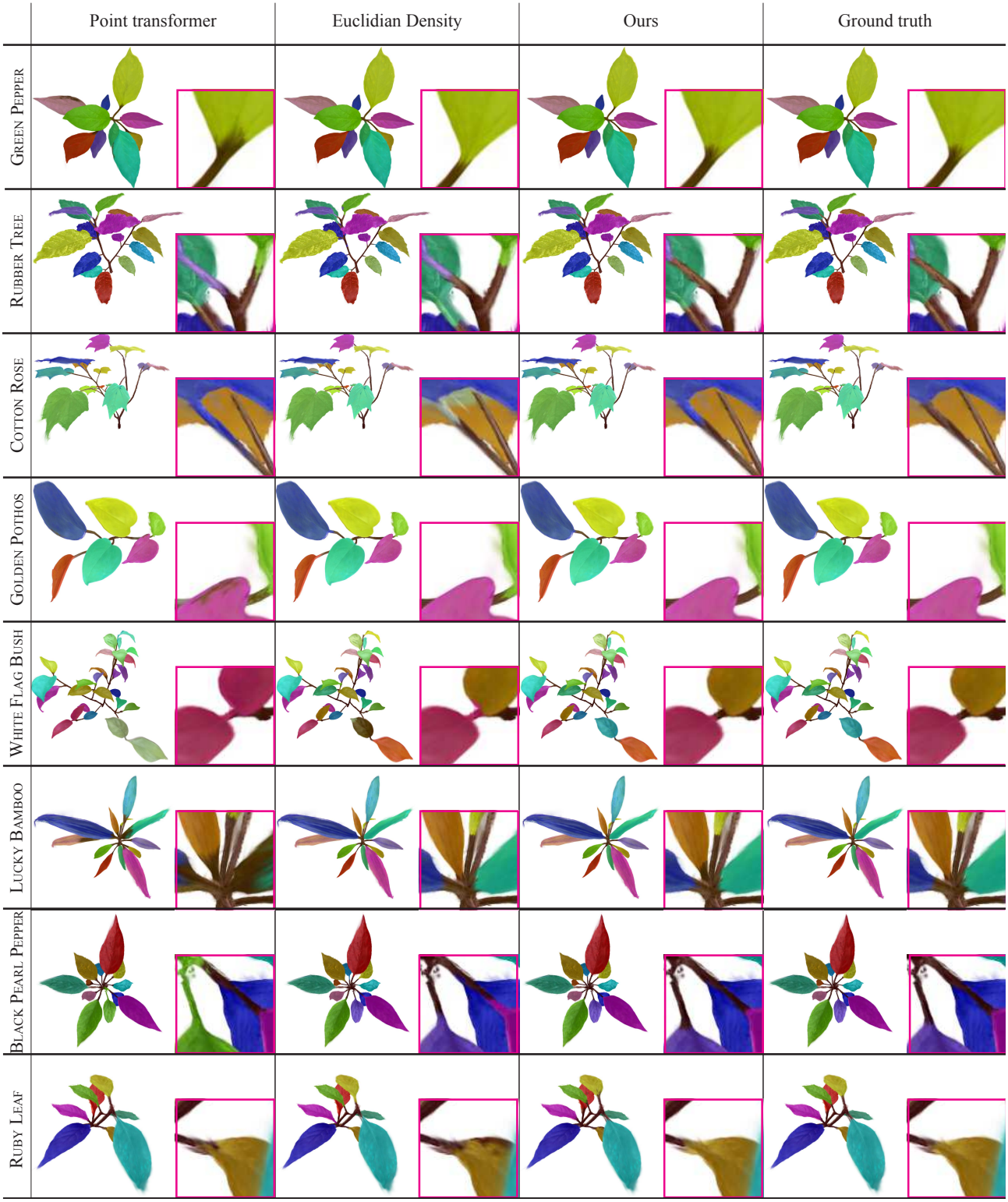


Figure 10: ***Segmentation qualitative comparison.*** *We compare our automatic segmentation against a learned baseline (Point Transformer [WJW*24]) and a Euclidean-density heuristic. Ground truth is manually annotated. Instance correspondences are matched via Hungarian assignment (IoU≥0.5). Despite being training-free, our method achieves cleaner separation of overlapping leaves compared to the learned baseline, which struggles with unseen plant structures.*

	Acc ↑		mIoU ↑		mF1 ↑		PQ ↑		Time (s) ↓		*Training-free*
Method	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	
Point Transformer [WJW*24]	94.52	4.37	86.88	9.26	89.54	8.29	89.36	7.90	16.534	10.062	✗
Euclidean Density	97.26	1.92	95.16	2.92	97.35	1.75	94.96	3.39	**3.786**	**2.334**	✓
Ours	**98.95**	**0.58**	**98.20**	**0.95**	**99.08**	**0.49**	**98.20**	**0.95**	4.456	2.724	✓

Table 1: ***Segmentation quantitative comparison.*** *Best scores are bolded. Metrics are reported as mean (± std) over all plants.* Ours *consistently outperforms both the learned and heuristic baselines across all metrics (Acc, mIoU, mF1, PQ), demonstrating robust generalization without requiring training data.*

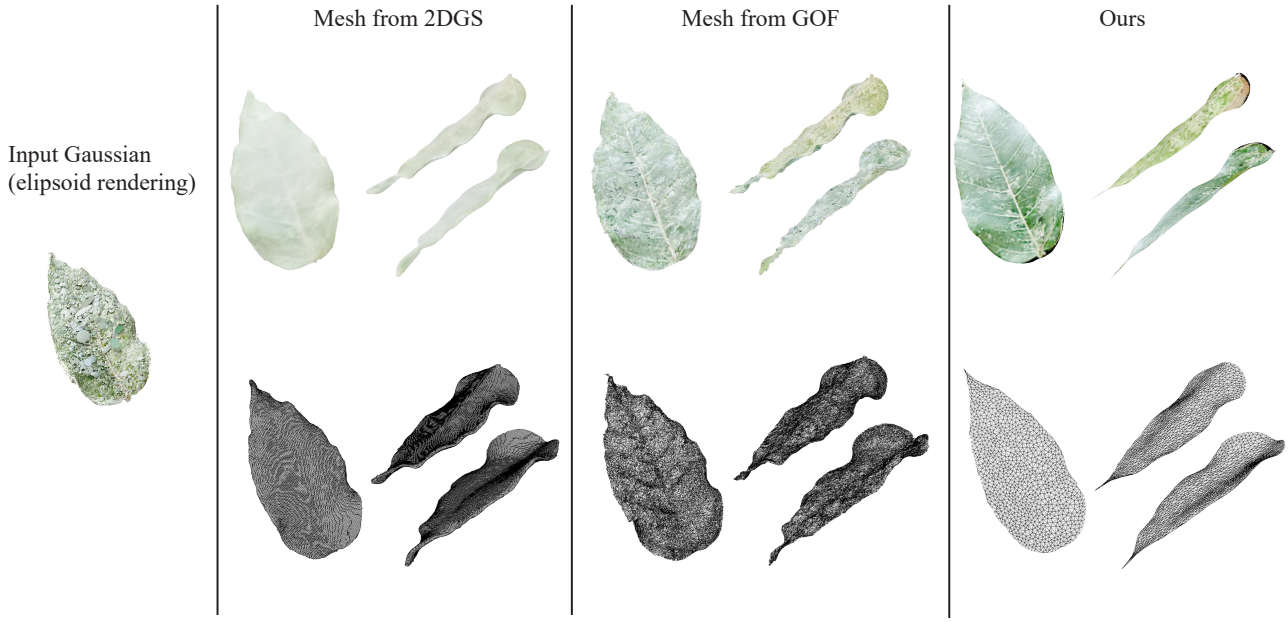


Figure 11: ***Mesh extraction qualitative comparison*** **(RUBBER TREE).** *Left: Input Gaussian primitives. Middle: Meshes extracted from implicit baselines (2DGS [HYC*24], GOF [YSG24]). Right: Meshes from our conversion. Implicit methods suffer from artificial thickening (double walls) and topological noise on thin leaf structures. In contrast, our method faithfully recovers thin, sharp leaf blades with compact triangulation and clean textures.*

Mesh extraction evaluation We evaluate surface reconstruction for both fidelity and compactness at two scales: plant and leaf. In Tab. 2, we report vertex and face counts, file size, and frames per second (FPS) for real-time rendering (plant only). As implicit baselines, we include 2DGS [HYC*24] and GOF [YSG24]. Specifically, 2DGS fits depth-aware 2D Gaussians in screen space and lifts them to an implicit surface, whereas GOF directly optimizes a 3D Gaussian opacity field. In both cases, the resulting fields are converted to meshes via Marching Cubes [LC98] for comparison. We compare their rendered quality, model size, and extracted meshes to assess representation accuracy, rendering speed, and storage requirements. As summarized in the table, the plant-level mesh attains the highest FPS with the smallest storage footprint, while the leaf-level mesh maintains a fixed 2,048-vertex budget and the smallest file size (per-leaf FPS is not applicable). Qualitatively (Fig. 11, RUBBER TREE), the extracted mesh preserves thin blades and sharp tips, recovers clear, high-frequency texture from Gaussian splatting, and exhibits uniformly distributed vertices with well-shaped triangles. Compared with implicit baselines, our thin template surface reveals reduced thickening and clean sharpness along edges, while maintaining low triangle counts.

Deformation evaluation We assess non-rigid alignment by deforming a single template to each target leaf and reporting Corr-ℓ_2 (mean nearest-neighbor error), symmetric Chamfer distance (CD) [Bor86], and Hausdorff distance (HD) [HKR93]. We compare PCA [Pea01], NR-ICP [ARV07], BCPD [Hir23], and ours, using per-leaf meshes reconstructed from target Gaussians as ground truth. Regarding the non-rigid baselines, NR-ICP [ARV07] extends standard ICP by solving for a smooth deformation field via a stiffness-regularized least squares objective, whereas BCPD [Hir23] treats registration as a probabilistic inference problem within a Gaussian Mixture Model, estimating coherent motion without requiring explicit point correspondences. Qualitatively (Fig. 12), the MLS-warped template (red) follows target contours with reduced stretching and texture distortion. Quantita-

Method	Representation	FPS ↑	Size ↓	Verts ↓
2DGS [HYC*24]	Gaussian (Plant)	653.93	19.08	76,943
	Mesh (Plant)	6,741.25	38.40	741,502
	Mesh (Leaf)	—	7.26	218,245
GOF [YSG24]	Gaussian (Plant)	375.80	20.76	77,996
	Mesh (Plant)	2,934.32	46.69	1,107,809
	Mesh (Leaf)	—	9.90	253,576
Ours	Mesh (Plant)	**11,980.11**	**1.13**	**2432**
	Mesh (Leaf)	—	**0.334**	**2048**

Table 2: ***Rendering and storage efficiency.*** *Best scores are bolded. Plant-level rows report FPS, storage size (MB), and vertex count. Our explicit mesh representation achieves superior rendering speed and compression rates compared to implicit baselines (2DGS, GOF), utilizing a minimal vertex budget (sum of template vertices and control points).*

Method	Corr ↓	CD ↓	HD ↓
PCA [Pea01]	0.2950	0.0368	1.5486
NR-ICP [ARV07]	0.1358	0.0145	1.0586
BCPD [Hir23]	0.1186	0.0061	0.7554
Ours (w/o optim.)	0.1115	0.0050	0.6512
Ours (full)	**0.0823**	**0.0022**	**0.4669**

Table 3: ***Deformation quantitative comparison.*** *Best scores are bolded. We report correspondence error (Corr), Chamfer distance (CD), and Hausdorff distance (HD) against ground-truth meshes. Our full optimization pipeline yields the lowest error across all metrics. All values are scaled by* $\times 10$ *for readability.*

tively (Tab. 3), our full model attains the lowest errors on all metrics.

Editing results A single template mesh is instanced across species to match diverse target leaf shapes. In Fig. 9 (a) and (b), we showcase the result of changing the leaves of a tree into the ones from another species using MVC-based control point retargeting (see Sec. 3.4). Leaf instances share a consistent UV layout, so edits to a texture of a template mesh propagate across all the leaves as (c) and (d) show the inpainting of the EG logo on the leaves. In other words, changing appearance requires only swapping texture files, with no geometry edits (as demonstrated in Fig. 9 (e) and (f).

4.1. Implementation Details

Our prototype is primarily in Python with a lightweight GUI for fallback manual selection and live deformation previews. Performance-critical components (BVH construction and ray–AABB tests) are in C++: for interactive picking we build an AABB-BVH [Wal07] over per-Gaussian primitives, using a truncation scale of 3 to bound each primitive. For MLS optimization and texture baking, we follow the 3DGS rasterizer [KKLD23] and render with an orthographic projection to produce UV textures for the extracted meshes. Geodesic distances are computed via `geometry-central` [SC*19]. The linear assignment algorithm [Cro16] mentioned in Sec. 3.2 is implemented by `SciPy` library. All experiments run on an AMD Ryzen 9 9950X, an NVIDIA RTX 3090 (24 GB), and 64 GB RAM.

4.2. Experimental Setup

Dataset and training We evaluate eight plants captured with an iPhone 12 mini (wide lens). The dataset comprises six plants with roughly elliptic leaves: GREEN PEPPER (9), RUBBER TREE (14), GOLDEN POTHOS (5), BLACK PEARL PEPPER (11), RUBY LEAF (7), and WHITE FLAG BUSH (27); and two with complex morphology: COTTON ROSE (multi-lobed, 12) and LUCKY BAMBOO (thin and elongated, 11). The number in parentheses denotes the leaf count. For reconstruction, 3DGS training enables Gaussian Opacity Fields (GOF) [YSG24] to suppress floaters. Plants are isolated from full scene reconstructions using SuperSplat's interactive selection [PS25] (or any semantic 3D-Gaussian method [YDYK24, CFY*25]). We will release the full dataset and code upon acceptance.

Parameter settings We perform global downsampling to $N_s = 8{,}192$, which empirically ensures sufficient density (> 100 primitives) per leaf. For efficiency, the neighbor count for graph construction is set to $N_k = 512$. Regarding segmentation thresholds, we set the triangle inequality margin $\tau = 0.5$ for Eq. 2. For petiole determination in Eq. 3, we use a geodesic band height $\delta = 0.01$ (normalized, approx. 1 cm) to aggregate sufficient local primitives for robust estimation. We set the petiole detection threshold $\epsilon = 0.05$ (normalized) to effectively identify the transition from the broader leaf blade to the stem. Crucially, we apply an early protection ratio $\rho = 0.25$ to bypass naturally narrow leaf tips, preventing premature termination. We use $K = 32$ control points per leaf, which we identified as the optimal trade-off between accuracy and efficiency (see Tab. 4). Before BPA, raw Gaussians are denoised by MLS with $\sigma = 0.1$; the same σ serves as the MLS kernel width in deformation. Optimization runs for 200 steps with a learning rate of 7×10^{-3} and Chamfer weight $\lambda = 0.7$. All hyperparameters are fixed and robust across all tested plants.

Ablation Studies We conduct ablation studies to investigate the influence of the control point count K and to validate the necessity of the MLS optimization stage. Increasing K adds degrees of freedom to the MLS warp, allowing it to capture finer bending or twisting; however, compute and memory costs grow linearly with K. Beyond a moderate value, gains plateau, and the risk of overfitting increases. We therefore adopt K=32 as an accuracy–efficiency

K	Corr ↓	CD ↓	HD ↓
16	0.0970	0.0034	0.6030
32 (default)	0.0823	0.0021	0.4670
32 (w/o optim.)	0.1115	0.0050	0.6512
64	**0.0810**	**0.0020**	**0.4361**

Table 4: ***Ablation studies on number of control points*** K ***and optimization.*** *Increasing* K *reduces error but raises training cost. Metrics scaled by* $\times 10$.

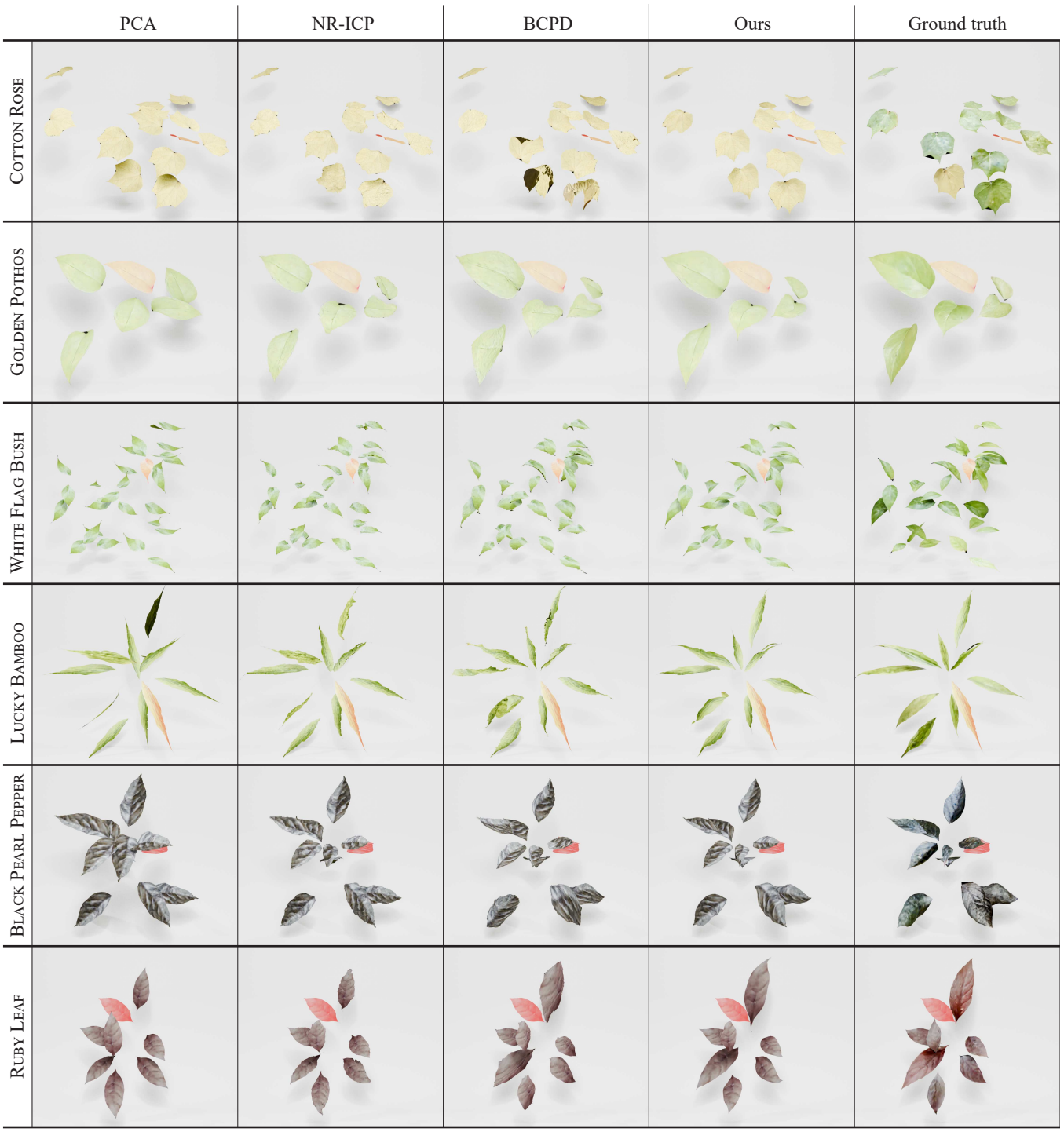
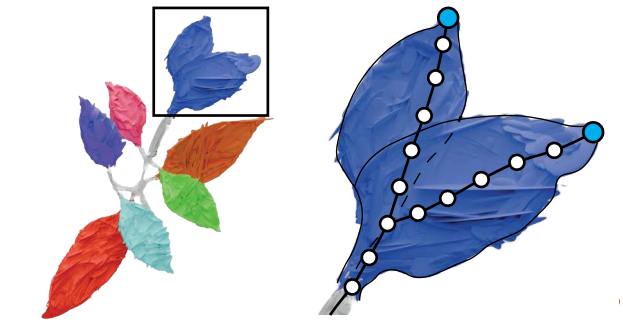


Figure 12: ***Deformation qualitative comparison.*** *Visual results across six species comparing PCA [Pea01], NR-ICP [ARV07], BCPD [Hir23], and Ours. Target geometries are reconstructed from input Gaussians. While texture details naturally differ (since we warp a single template texture to match target geometry), our method exhibits the least geometric distortion and boundary misalignment compared to baselines, preserving the natural curvature of the leaves.*

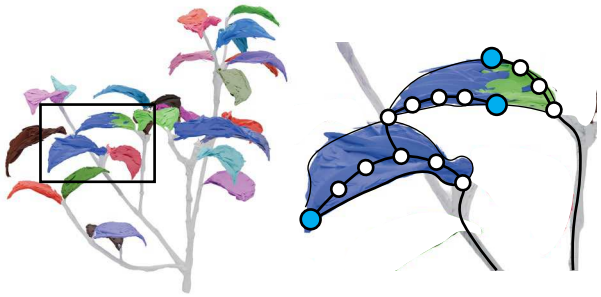
sweet spot. As shown in Tab. 4, increasing to K=64 offers only marginal improvements over K=32, whereas reducing to K=16 or removing the optimization step (at K=32) significantly degrades all metrics, confirming that our optimization is essential for accurate alignment.

5. Limitations

Our method requires a cleaned up Gaussian model as the input and steps require manual intervention to complete the segmentation. Besides, our method struggles with topological ambiguity, a frequent issue in dense foliage with overlapping leaves. Since the Heat Method [CWW17] computes the Point Laplacian based on point proximity, overlapping leaves introduce ambiguous relations that corrupt the geodesic distance field. Fig. 9 highlights two specific failure cases, while the detected apexes are highlighted as blue colored circles. In horizontal overlapping, although the apexes are correctly detected, the grouping fails because the two paths share a large overlap in the ambiguous region cause the leafs wrongly segmented as one leave with multiple apexes. In vertical overlapping, the apex of a top leaf is spatially close to a bottom leaf, which causes the geodesic estimation around the upper leaf apex highly unreliable, as the points around are going to take the bottom leaf as a short cut to the root. Local maxima are found where the length of the short-cut equals the normal path (the upper two detected apexes), which leads to the incorrect segmentation at the end.



(a) Horizontally overlapping



(b) Vertically overlapping

Figure 13: ***Failure in overlapping cases.*** *The blue colored circles represent the found local minima (apexes). (a) Horizontal overlapping leads to merging paths. (b) Vertical overlapping causes the short-cut, resulting in incorrect segmentation of the upper leaf.*

We also acknowledge that the diversity of our dataset is limited. Most plant types (except COTTON ROSE and LUCKY BAMBOO) have simple ellipsoid-shaped leaves. However, since the ellipsoid is the most common leaf shape, we focus on this typical class in this paper. Handling more complex shapes remains a valuable direction for future work. For instance, in Ginkgo leaves, the local geodesic maximum often does not align with the biological apex, which leads to incorrect orientation estimation and segmentation failures which can benefit from exploiting the leaf shape symmetry. Finally, since the leaves are deformed from a single template, the texture is shared across all leaves. This lack of unique texture variation for individual leaves is also a limitation.

6. Conclusion

We present LeafFit, a framework that converts plant reconstructions using dense 3D Gaussian splats to *editable and instanced meshes*. By establishing a parametric link between 3DGS and mesh assets via differentiable MLS retargeting, LeafFit reuses a single template with lightweight per-leaf controls, preserving thin-leaf detail while compressing the representation by orders of magnitude. A geodesic-guided segmentation stage yields clean instances; meshes with seamless UVs support texture and geometry edits that propagate consistently across leaves. Evaluated in a shader at runtime, the assets deliver high FPS and small memory footprints without sacrificing visual fidelity. LeafFit integrates seamlessly with standard digital content creation tools and real-time game engines through standard mesh and texture assets, enabling artists to author, retarget, and iterate at scale.

Looking ahead, we find that the intrinsic symmetry of leaves is a valuable geometric cue that could be leveraged to enhance both segmentation and apex-petiole detection. Additionally, incorporating cage-based refinement offers a promising avenue for recovering sharper and cleaner leaf silhouettes. Finally, we anticipate that joint optimization with the Gaussian representation would benefit from a learning-based approach, where fully observed leaves serve as guidance to regularize leaves hidden by occlusion.

Acknowledgements

This work was supported by JST SPRING, Grant Number JPMJSP2108. We would like to thank Haato Watanabe and Yudi Wu for their assistance with data collection, as well as the anonymous reviewers for their constructive comments which helped improve the manuscript.

References

[ACFQ*18] ANDÚJAR D., CALLE M., FERNÁNDEZ-QUINTANILLA C., RIBEIRO Á., DORADO J.: Three-dimensional modeling of weed plants using low-cost photogrammetry. *Sensors 18*, 4 (2018), 1077. 2

[AK04] AMENTA N., KIL Y. J.: Defining point-set surfaces. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 264–270. URL: https://doi.org/10.1145/1015706.1015713, doi:10.1145/1015706.1015713. 6

[ARV07] AMBERG B., ROMDHANI S., VETTER T.: Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8. doi:10.1109/CVPR.2007.383165. 9, 10, 11

[Aut] AUTODESK, INC.: Maya. URL: https://autodesk.com/maya. 2

[BCDD22] BOURQUAT P., COEURJOLLY D., DAMIAND G., DUPONT F.: Hierarchical mesh-to-points as-rigid-as-possible registration. *Computers & Graphics 102* (2022), 320–328. 3

[BMR*02] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics 5*, 4 (2002), 349–359. 3, 6

[BNB13] BRADLEY D., NOWROUZEZAHRAI D., BEARDSLEY P.: Image-based reconstruction and synthesis of dense foliage. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 2

[Bor86] BORGEFORS G.: Distance transformations in digital images. *Computer vision, graphics, and image processing 34*, 3 (1986), 344–371. 9

[CB17] CHAURASIA G., BEARDSLEY P.: Editable parametric dense foliage from 3d capture. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 5305–5314. 2

[CFY*25] CEN J., FANG J., YANG C., XIE L., ZHANG X., SHEN W., TIAN Q.: Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2025), vol. 39, pp. 1971–1979. 10

[Com18] COMMUNITY B. O.: *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org. 2

[Cro16] CROUSE D. F.: On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems 52*, 4 (2016), 1679–1696. 6, 10

[CWW17] CRANE K., WEISCHEDEL C., WARDETZKY M.: The heat method for distance computation. *Commun. ACM 60*, 11 (Oct. 2017), 90–99. URL: http://doi.acm.org/10.1145/3131280, doi:10.1145/3131280. 3, 4, 5, 12

[CXL*24] CHANG J., XU Y., LI Y., CHEN Y., FENG W., HAN X.: Gaussreg: Fast 3d registration with gaussian splatting. In *European Conference on Computer Vision* (2024), Springer, pp. 407–423. 3

[FBF77] FRIEDMAN J. H., BENTLEY J. L., FINKEL R. A.: An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS) 3*, 3 (1977), 209–226. 4

[Flo03] FLOATER M. S.: Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1 (Mar. 2003), 19–27. 7

[GL24] GUÉDON A., LEPETIT V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR* (2024). 2, 3

[GXY*18] GUO J., XU S., YAN D.-M., CHENG Z., JAEGER M., ZHANG X.: Realistic procedural plant modeling from multiple view images. *IEEE transactions on visualization and computer graphics 26*, 2 (2018), 1372–1384. 2

[Hir23] HIROSE O.: Geodesic-based bayesian coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence 45*, 5 (2023), 5816–5832. doi:10.1109/TPAMI.2022.3214191. 9, 10, 11

[HKR93] HUTTENLOCHER D., KLANDERMAN G., RUCKLIDGE W.: Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 9 (1993), 850–863. doi:10.1109/34.232073. 9

[HLS*25] HUANG Y.-H., LIN M.-X., SUN Y.-T., YANG Z., LYU X., CAO Y.-P., QI X.: Deformable radial kernel splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (2025), pp. 21513–21523. 3

[HTE*23] HAQUE A., TANCIK M., EFROS A. A., HOLYNSKI A., KANAZAWA A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF international conference on computer vision* (2023), pp. 19740–19750. 3

[HXYL24] HUANG J., XU S., YU H., LEE T.-Y.: Gsdeformer: Direct, real-time and extensible cage-based deformation for 3d gaussian splatting. *arXiv preprint arXiv:2405.15491* (2024). 3

[HYC*24] HUANG B., YU Z., CHEN A., GEIGER A., GAO S.: 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers* (2024), Association for Computing Machinery. doi:10.1145/3641519.3657428. 2, 3, 9, 10

[HYP*24] HU K., YING W., PAN Y., KANG H., CHEN C.: High-fidelity 3d reconstruction of plants using neural radiance fields. *Computers and Electronics in Agriculture 220* (2024), 108848. 3

[HYY24] HA S., YEON J., YU H.: Rgbd gs-icp slam. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXXVI* (Berlin, Heidelberg, 2024), Springer-Verlag, p. 180–197. URL: https://doi.org/10.1007/978-3-031-72764-1_11, doi:10.1007/978-3-031-72764-1_11. 3

[IOI06] IJIRI T., OWADA S., IGARASHI T.: The sketch l-system: Global control of tree modeling using free-form strokes. In *International symposium on smart graphics* (2006), Springer, pp. 138–146. 2

[JMG24] JAIN U., MIRZAEI A., GILITSCHENSKI I.: Gaussiancut: Interactive segmentation via graph cut for 3d gaussian splatting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems* (2024). 3

[JYX*24] JIANG Y., YU C., XIE T., LI X., FENG Y., WANG H., LI M., LAU H., GAO F., YANG Y., JIANG C.: Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663* (2024). 3

[KHG*19] KIRILLOV A., HE K., GIRSHICK R., ROTHER C., DOLLÁR P.: Panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 9404–9413. 7

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics 42*, 4 (July 2023). URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/. 1, 3, 6, 10

[LC98] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 1998, pp. 347–353. 3, 9

[Lie03] LIEPA P.: Filling Holes in Meshes. In *Eurographics Symposium on Geometry Processing* (2003), Kobbelt L., Schroeder P., Hoppe H., (Eds.), The Eurographics Association. doi:/10.2312/SGP/SGP03/200-206. 6

[LLB23] LEE J. J., LI B., BENES B.: Latent l-systems: Transformer-based tree generator. *ACM Transactions on Graphics 43*, 1 (2023), 1–16. 2

[LOL*20] LU X., ONO E., LU S., ZHANG Y., TENG P., AONO M., SHIMIZU Y., HOSOI F., OMASA K.: Reconstruction method and optimum range of camera-shooting angle for 3d plant modeling using a multi-camera photography system. *Plant Methods 16*, 1 (2020), 118. 2

[LQN*25] LI J., QI X., NABAEI S. H., CHEN D., ZHANG X., LI Z.: A survey on 3d reconstruction techniques in plant phenotyping: from classical methods to neural radiance fields (nerf), 3d gaussian splatting (3dgs), and beyond. In *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping X* (2025), vol. 13475, SPIE, p. 134750B. 3

[LRX*24] LI Y., RAN X., XU L., LU T., YU M., WANG Z., XIANGLI Y., LIN D., DAI B.: Proc-gs: Procedural building generation for city assembly with 3d gaussians, 2024. URL: https://arxiv.org/abs/2412.07660, arXiv:2412.07660. 3

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. 41*, 4 (July 2022), 102:1–102:15. URL: https://doi.org/10.1145/3528223.3530127, doi:10.1145/3528223.3530127. 3

[MGSS24] MEYER L., GILSON A., SCHMIDT U., STAMMINGER M.: Fruitnerf: A unified neural radiance field based fruit counting framework. In *IROS* (2024). URL: https://meyerls.github.io/fruit_nerf. 3

[MŞS*24] MERTOĞLU K., ŞALK Y., SARIKAYA S. K., TURGUT K., EVRENESOĞLU Y., ÇEVIKALP H., GEREK Ö. N., DUTAĞACI H., ROUSSEAU D.: Planest-3d: A new annotated dataset for segmentation of 3d plant point clouds. *arXiv preprint arXiv:2407.21150* (2024). 7

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 3

[NST*16] NGUYEN T. T., SLAUGHTER D. C., TOWNSLEY B., CARRIEDO L., NN J., SINHA N.: Comparison of structure-from-motion and stereo vision techniques for full in-field 3d reconstruction and phenotyping of plants: An investigation in sunflower. In *2016 ASABE annual international meeting* (2016), American Society of Agricultural and Biological Engineers, p. 1. 2

[Oku22] OKURA F.: 3d modeling and reconstruction of plants and trees: A cross-cutting review across computer graphics, vision, and plant phenotyping. *Breeding Science 72*, 1 (2022), 31–47. 2

[OLMS24] OJO T., LA T., MORTON A., STAVNESS I.: Splanting: 3d plant capture with gaussian splatting. In *SIGGRAPH Asia 2024 Technical Communications*. 2024, pp. 1–4. 3

[Pau19] PAULUS S.: Measuring crops in 3d: using geometry for plant phenotyping. *Plant methods 15*, 1 (2019), 103. 2

[PBN*09] PRADAL C., BOUDON F., NOUGUIER C., CHOPARD J., GODIN C.: Plantgl: a python-based geometric library for 3d plant modelling at different scales. *Graphical models 71*, 1 (2009), 1–21. 2, 3

[Pea01] PEARSON K.: Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science 2*, 11 (1901), 559–572. 9, 10, 11

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 165–174. 3

[PHT*25] PANDEY K., HU A., TSANG C. F., PEREL O., SINGH K., SHUGRINA M.: Painting with 3d gaussian splat brushes. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (2025), pp. 1–10. 3

[PL12] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer Science & Business Media, 2012. 2

[PS25] PLAYCANVAS, SNAP INC.: Supersplat, 2025. [Computer software]. URL: https://github.com/playcanvas/supersplat. 3, 10

[PSB*12] PAPROKI A., SIRAULT X., BERRY S., FURBANK R., FRIPP J.: A novel mesh processing based technique for 3d plant analysis. *BMC plant biology 12*, 1 (2012), 63. 2

[SC*19] SHARP N., CRANE K., ET AL.: Geometrycentral: A modern c++ library of data structures and algorithms for geometry processing. 10

[SF16] SCHÖNBERGER J. L., FRAHM J.-M.: Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). 2

[SHT09] SEGAL A., HAEHNEL D., THRUN S.: Generalized-icp. In *Robotics: science and systems* (2009), vol. 2, Seattle, WA, p. 435. 3

[SJD*25] SHEN P., JING X., DENG W., JIA H., WU T.: Plantgaussian: exploring 3d gaussian splatting for cross-time, cross-scene, and realistic 3d plant visualization and beyond. *The Crop Journal 13*, 2 (2025), 607–618. 3

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *ACM SIGGRAPH 2006 Papers*. 2006, pp. 533–540. 3

[Son88] SONEIRA J.: Leonardo's rule, self-similarity and random walks in trees. *Physica A: Statistical Mechanics and its Applications 149*, 3-4 (1988), 641–652. doi:10.1016/0378-4371(88)90293-9. 7

[Spe17] SPEEDTREE: Speedtree, open research content archive (orca), July 2017. URL: http://developer.nvidia.com/orca/speedtree. 2, 3

[SPK*14] STAVA O., PIRK S., KRATT J., CHEN B., MĚCH R., DEUSSEN O., BENES B.: Inverse procedural modelling of trees. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 118–131. 2

[SWWG25] SONG H., WEN W., WU S., GUO X.: Comprehensive review on 3d point cloud segmentation in plants. *Artificial Intelligence in Agriculture 15*, 2 (2025), 296–315. URL: https://www.sciencedirect.com/science/article/pii/S2589721725000066, doi:https://doi.org/10.1016/j.aiia.2025.01.006. 2

[SZPF16] SCHÖNBERGER J. L., ZHENG E., POLLEFEYS M., FRAHM J.-M.: Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)* (2016). 2

[TLL*11] TALTON J. O., LOU Y., LESSER S., DUKE J., MECH R., KOLTUN V.: Metropolis procedural modeling. *ACM Trans. Graph. 30*, 2 (2011), 11–1. 2

[VMG*25] VIOLANTE N., MEULEMAN A., GAUTHIER A., DURAND F., GROUEIX T., DRETTAKIS G.: Splat and replace: 3d reconstruction with repetitive elements. *SIGGRAPH Conference Papers* (2025). 2, 3

[Wal07] WALD I.: On fast construction of sah-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing* (2007), IEEE, pp. 33–40. 10

[WFZ*24] WANG J., FANG J., ZHANG X., XIE L., TIAN Q.: Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2024), pp. 20902–20911. 3

[WHW*13] WANG Y., HAO W., WANG G., NING X., TANG J., SHI Z., WANG N., ZHAO M.: A method of realistic leaves modeling based on point cloud. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry* (2013), pp. 123–130. 2

[WJW*24] WU X., JIANG L., WANG P.-S., LIU Z., LIU X., QIAO Y., OUYANG W., HE T., ZHAO H.: Point transformer v3: Simpler, faster, stronger. In *CVPR* (2024). 7, 8, 9

[XABP24] XIE T., AIGERMAN N., BELILOVSKY E., POPA T.: Sketch-guided cage-based 3d gaussian splatting deformation. *arXiv preprint arXiv:2411.12168* (2024). 3

[XCH*25] XIAO H., CHEN Y., HUANG H., XIONG H., YANG J., PRASAD P., ZHAO Y.: Localized gaussian splatting editing with contextual awareness. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2025), IEEE, pp. 5207–5217. 3

[YDH*25] YANG Y., DONGNI M., HIROAKI S., YASUYUKI M., FUMIO O.: Neuraleaf: Neural parametric leaf modelswith shape and deformation disentanglement. *ICCV* (2025). 2

[YDYK24] YE M., DANELLJAN M., YU F., KE L.: Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision* (2024), Springer, pp. 162–179. 10

[YLS*24] YAN Z., LI L., SHAO Y., CHEN S., WU Z., HWANG J.-N., ZHAO H., REMONDINO F.: 3dsceneeditor: Controllable 3d scene editing with gaussian splatting. *arXiv preprint arXiv:2412.01583* (2024). 3

[YSG24] YU Z., SATTLER T., GEIGER A.: Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics* (2024). 2, 3, 9, 10

[ZG07] ZHU Y., GORTLER S.: 3d deformation using moving least squares. 3

[ZLB*25] ZHOU X., LI B., BENES B., HABIB A., FEI S., SHAO J., PIRK S.: Treestructor: Forest reconstruction with neural ranking. *IEEE Transactions on Geoscience and Remote Sensing* (2025). 2