Editable Parametric Dense Foliage from 3D Capture

Gaurav Chaurasia
Disney Research Zurich
gaurav.chaurasia@disneyresearch.com

Paul Beardsley
Disney Research Zurich
pab@disneyresearch.com

Abstract

We present an algorithm to compute parametric models of dense foliage. The guiding principles of our work are automatic reconstruction and compact artist friendly representation. We use Bézier patches to model leaf surface, which we compute from images and point clouds of dense foliage. We present an algorithm to segment individual leaves from colour and depth data. We then reconstruct the Bézier representation from segmented leaf points clouds using non-linear optimisation. Unlike previous work, we do not require laboratory scanned exemplars or user intervention. We also demonstrate intuitive manipulators to edit the reconstructed parametric models. We believe our work is a step towards making captured data more accessible to artists for foliage modelling.

1. Introduction

Foliage is a commonly occurring element in synthesised imagery. Production quality foliage is usually modelled using polygon meshes that are attached to procedurally generated branches [29]. This entails two main challenges: quantity and variety. Foliage requirements rapidly rise to hundreds of thousands of leaves, at which point polygon meshes become prohibitively expensive to create, store and render. Curve-based parametric models can alleviate this problem, but current models lack expressiveness. Foliage variety is limited because of the high cost of modelling individual leaves. Generating foliage from 3D capture is a possible solution, as shown by recent work on high quality 3D reconstruction of flowers [18, 53, 19, 55, 38]. For dense foliage however, 3D reconstruction algorithms lack robustness due to self-similarity and non-rigidity. They generate noisy unordered point clouds with no semantic structure which are hard to mesh and edit. These challenges have so far restricted virtual foliage to simple shapes created manually.

We address these challenges by capturing parametric models of foliage from imagery. We are guided by the principles of *automation* and *editability*. We use Bézier curves, the simplest non-linear parametric primitives. We identify salient components of a leaf, namely midrib, silhouettes, and cross-section, and model each with a Bézier curve. This semantically aware model can express a wide variety of single-lobed leaves including silhouette, symmetry and curvature variations. Reconstructing and manipulating the leaf amounts to estimating and editing the control points of 4 Bézier curves. It is more compact than polygon meshes and more expressive than 2D silhouette based leaf models. It also aids editability compared to meshes and generic parametric representations like NURBS. Our model has a canonical NURBS representation and inherits all existing NURBS editing techniques. Howsoever powerful, editing interfaces can be improved if the underlying model has semantics. We add semantics by modelling salient leaf components with dedicated Béziers. We demonstrate that semantics allow editing leaves with a very simple UI. These edits are also feasible on meshes or NURBS, but would require a more complex UI for vertex or segment selection. We choose Béziers for simplicity; other semantically aware parametric models may be equally effective.

We compute this model from colour and depth data of dense foliage, obtained using multi-view stereo or RGB-D sensors. We propose an automatic leaf segmentation algorithm that first extracts the most prominently visible leaves using colour and geometric heuristics, and then extracts partially occluded leaves in subsequent passes. This is a significant departure from semantic segmentation which focuses on large salient objects [4, 50]. Our segmentation is designed for small, repeating, partially visible elements. The Bézier model gives a closed-form expression for every point on the leaf surface. This underpins a non-linear optimisation to reconstruct the parametric model. We demonstrate robustness to occlusions, outliers and missing data.

In this work, we focus on a simple model for the simple yet dominant class of single-lobed leaves like beech, holly, cherry etc. We do not handle multi-lobed species like maple[1]. Single-lobed leaves are most common in nature: of the 203 species found in northeastern US and Canada, 99 were single-lobed, 59 multi-lobed, and 45 grasses or needle shaped which do not require modelling [25]. Thus,

[1] http://leafsnap.com/species/

we cover 99 of 158 species that need modelling, which is almost two-thirds. Simple shapes seem the most commonly modelled by artists. We therefore simplify a majority of modelling requirements. We also focus on dense foliage and ignore branches or twigs. Our work complements branch reconstruction techniques [32] for complete 3D reconstruction of vegetation.

Within the above scope, our main contributions are:

- a semantically aware parametric model for leaves using Bézier curves that is more compact than polygon meshes, and more robust and expressive than previous parametric models [51, 34],
- a segmentation algorithm to extract individual point clouds of small, repeating, partially visible elements like leaves without user intervention [44] or pre-scanned leaf exemplars [8],
- a non-linear optimisation for fitting the parametric model to point clouds, and
- intuitive handles for free form collective manipulation, instead of being restricted to axes of variation [8].

Overall, our work is a step towards usage of captured foliage in graphics applications.

2. Previous work

Image-based methods have investigated rendering [45, 11] and relighting [9] of trees. These are generally not used in visual effects because the lack of accurate geometry degrades rendering quality.

Interactive methods create 3D models of flora, either by interpreting freehand sketches [37, 3], or using sophisticated user interfaces [30, 12, 5]. These are drawing tools and not meant for large scale foliage synthesis.

Procedural methods like L-systems [29, 43, 42, 39] and statistical methods [14, 48] use fractals controlled by biological or aesthetic parameters to generate trees. These are used in commercial software [1]. These are well suited to artists' work flow because they allow control even though significant effort may be needed to model specific types of flora. Our parametric approach addresses this problem by allowing more realism at reduced modelling effort.

Tree reconstruction techniques create the skeletal branch structure using multi-view stereo [49] or laser scans [52, 32, 31]. Recent extensions include animated branch structures [27], branch growth simulations [40, 41], and guided procedural synthesis [47, 48]. These techniques create a branch structure and populate it with rudimentary leaves. They do not model foliage explicitly.

Foliage reconstruction has been attempted by meshing point clouds of leaves using multi-view stereo [44], structured light [36] or laser scans [10]. Non-planar or complex leaf surfaces have been reconstructed from point clouds using finite element methods [33, 22]. Yin *et al*. [54] propose a complete foliage and branch modelling method but require disassembling the entire plant in the process and scanning each component separately. This approach is neither practical, nor scalable to large plants. Biologically-inspired methods focus on the venation pattern [35, 16, 46, 10, 21]. These only handle a single leaf under ideal conditions. There is almost no scope for variation within or across species. They produce polygon meshes which can be challenging to store, manipulate and render for scenes with large amounts of foliage.

Most relevant to our work is Bradley *et al*. [8], who use a morphable model [7] and encode leaf variation within a given flora species. Their most important limitation is that they require leaf exemplars scanned in a controlled setting, thus introducing manual intervention. There are additional limitations from an artist's perspective. Firstly, there is no scope for explicit artistic manipulation outside the modes of variation of the morphable model, and these parameters are arbitrary from an artist's perspective. For example, there might not be a mode that corresponds to a physical manipulation like changing the curvature of the leaf. Secondly, the modes of variation may vary between foliage datasets introducing unwanted randomness into the artistic pipeline. The lack of consistent and meaningful editing methodology makes this approach less useful for artists.

Parametric methods We define *parametric* methods as those which describe complex shapes using simple geometric primitives such as planes, lines, curves etc. Such representations are compact, readily editable, and suitable for large scale modelling. B-splines have been used to represent silhouettes or veins [34, 51, 24] of 2D leaves. These are restricted to singleton leaves captured in laboratory conditions, making them unsuitable for dense foliage. Their B-spline models involve a large number of connected curves. This makes them unsuitable for graphics applications because they do not offer a concise model for an entire species and are also not artist-friendly.

In contrast, we propose a parametric model which can represent non-planar leaves. We reconstruct it directly from 3D data accounting for complex occlusions. Each leaf is fitted independently without any pre-scanned exemplars; we can therefore handle a wider variety of shape and size variations within the same dataset. The reconstructions can be edited easily and used to initialise foliage modelling.

3. Parametric model for foliage

Our choice of parametric representation is guided by two principles: compact representation and intuitive control. NURBS and subdivision surfaces are the standard parametric tools. They however do not offer any advantages over polygon meshes from an editing perspective because they are not specialised for leaves. B-splines involve a lot of

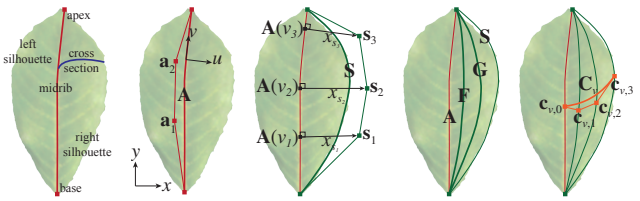


Figure 1. Parametric leaf model. (a) Semantically meaningful components of leaf geometry that we model as Bézier curves. (b) uv coordinate frame attached to the central axis Bézier $\mathbf{A}$ (red) along with its control points $\mathbf{a}_i$. (c) Control points $\mathbf{s}_i$ of silhouette Bézier (green), computed using displacement $\bar{v}_i$ along the axis and $\bar{x}_{s_i}$ perpendicular to the axis. (d) Longitudinal Béziers $\mathbf{F}$ and $\mathbf{G}$ at $\frac{1}{3}$ and $\frac{2}{3}$ of silhouette Bézier's control points. These afford control over leaf curvature. (e) Cubic Bézier to describe leaf surface, with control points $\mathbf{c}_{v,i}$ on longitudinal Béziers at v. All Béziers have independent displacements along z axis which (not shown here).

connected splines [51, 34] which complicate the parameter space and may not be artist friendly. In contrast, our insight is to separate leaf shape into salient components and model each with the simplest possible parametric representation.

We identify three defining features of leaf geometry: midrib, left/right silhouette and cross-section or the curve that connects points on the midrib to silhouette (Fig. 1(a)). We model each of these using non-linear curves. Separation into functional components allows fitting each component separately to 3D point cloud data, and editing the shape in semantically meaningful ways. We use the simplest non-linear representation: Bézier curves. We use Béziers for the midrib (Fig. 1(b)), hereafter referred as axis of the leaf, and the silhouettes (Fig. 1(c)). These curves share their first and last control points which coincide with the tips of the leaf. In order to model the cross-section, we create extra Béziers along the length of the leaf between the axis and silhouette curves (Fig. 1(d)). The cross-section is a Bézier whose control points lie on the axis, silhouette and the extra Béziers (Fig. 1(e)). Additionally, we model silhouette and longitudinal Bézier control points relative to the axis. The control points of Bézier in the left half are same as their counterparts in the right half. This symmetric axis-dependent model reduces the number of parameters, allows easy fitting to 3D data and intuitive manipulators for editing the shape, and somewhat restricts the model to plausible leaf shapes instead of diverging to arbitrary shapes. We model asymmetric leaves by changing the shape of the central axis (Fig. 7(b)).

Notation We denote any point on a Bézier curve $\mathbf{B}$ of degree n by $\mathbf{B}(u)$ for $u \in [0, 1]$.

$$\mathbf{B}(u) = \begin{bmatrix} 1 \\ \cdots \\ u^n \end{bmatrix}^T \begin{bmatrix} \\ \mathbf{Z} \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \cdots \\ \mathbf{b}_n \end{bmatrix} \tag{1}$$

where $\mathbf{b}_i$ are the control points of the curve and $\mathbf{Z}$ is the $(n+1)\times(n+1)$ matrix of Bernstein polynomial coefficients of n degree Bézier. The unit vector along the tangent to the curve is the normalised first derivative $\hat{\mathbf{B}}'(u)$ with respect to u. The unit vector $\hat{\mathbf{B}}_n(u)$ along the normal to the curve in the xy plane is obtained by rotating the tangent by $90°$ in the xy plane.

$$\hat{\mathbf{B}}_n(u) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \hat{\mathbf{B}}'(u) \tag{2}$$

Coordinate frames We model 3D leaves such that the length is aligned with Cartesian y axis, width with x axis and curvature with z axis. The world space position of the leaf is encoded by a rigid transformation that is kept separate from the parametric formulation. We use a normalised coordinate frame uv attached to the central axis of the leaf to measure displacements along and perpendicular to the axis (Fig. 1(b)) such that $u = 0$ corresponds to the central axis, $u = \pm 1$ correspond to the two silhouettes, v corresponds to longitudinal displacement along the axis from the base vertex of the leaf. Both xy and uv coordinate frames are in the same plane; uv is attached to the axis of the leaf and xy is attached to Cartesian coordinates. The coordinates for the central axis are in Cartesian space while those of all other components of the leaf are relative to the axis, i.e. uv coordinates. Displacement above or below the leaf surface is encoded by z coordinate which is shared by xy and uv systems. We fix the base and apex at $(0, 0, 0)$ and $(0, 1, 0)$ for simplicity.

We parameterise the surface of the leaf in the uv coordinate system such that $(u, v) \in [-1, 1] \times [0, 1]$. The parametric leaf model maps (u, v) coordinates to 3D vertices. We describe formulae for the right half of the leaf; formulae for the left half are analogous.

Bézier curves for leaves We model the central axis and silhouette as two Bézier curves $\mathbf{A}$ and $\mathbf{S}$ of degrees m and n. The base and apex are the first and last control points of both these curves. The $m-1$ intermediate control points of the axis are in Cartesian coordinates, and the $n-1$ control points of $\mathbf{S}$ are in the uv coordinates, computed using displacement along the axis $\bar{v}_i$, perpendicular to the axis $\bar{x}_{s_i}$ in the plane of the leaf and normal to the plane of the leaf $\bar{z}_{s_i}$ (Fig. 1(c)).

$$\mathbf{s}_i = \mathbf{A}(\bar{v}_i) + \bar{x}_{s_i}\hat{\mathbf{A}}_n(\bar{v}_i) + \bar{z}_{s_i}\hat{\mathbf{z}} \tag{3}$$

where $\hat{\mathbf{A}}_n$ is the unit vector along the normal to the central axis (Eq. 2) in the leaf plane and $\hat{\mathbf{z}}$ is the z axis.

In order to model surface curvature, we generate leaf vertices on a cross-section cubic Bézier $\mathbf{C}_v$ whose end points lie on the axis $\mathbf{A}(v)$ and silhouette $\mathbf{S}(v)$. The two intermediate control points of $\mathbf{C}_v$ lie on two additional longitudinal

Béziers $\mathbf{F}$ and $\mathbf{G}$ at $\frac{1}{3}$ and $\frac{2}{3}$ the width of the leaf (Fig. 1(d)). We reuse the parameters of the silhouette to compute their control points and introduce new displacements above the plane of the leaf $\bar{z}_{f_i}$ and $\bar{z}_{g_i}$. This reduces the total number of parameters and suffices because $\mathbf{F}$ and $\mathbf{G}$ are only useful for modelling displacements away from the leaf plane.

$$\mathbf{f}_i = \mathbf{A}(\bar{v}_i) + \frac{1}{3}\bar{x}_{s_i}\hat{\mathbf{A}}_n(\bar{v}_i) + \bar{z}_{f_i}\hat{\mathbf{z}} \tag{4a}$$

$$\mathbf{g}_i = \mathbf{A}(\bar{v}_i) + \frac{2}{3}\bar{x}_{s_i}\hat{\mathbf{A}}_n(\bar{v}_i) + \bar{z}_{g_i}\hat{\mathbf{z}} \tag{4b}$$

where $\hat{\mathbf{A}}_n$ is the unit vector along the normal to the central axis (Eq. 2) and $\hat{\mathbf{z}}$ is a unit vector along the z axis. Curves for the left half of the leaf use $-\bar{x}_{s_i}$ in the above formulae.

The surface of the leaf is described by a cross-section cubic Bézier $\mathbf{C}_v$ whose control points lie on the above four Béziers (Fig. 1(e)). The control points of this curve $\{\mathbf{c}_{v,0} \cdots \mathbf{c}_{v,3}\}$ are given by:

$$\begin{aligned} \mathbf{c}_{v,0} &= \mathbf{A}(v), \quad \mathbf{c}_{v,1} = \mathbf{F}(v) \\ \mathbf{c}_{v,2} &= \mathbf{G}(v), \quad \mathbf{c}_{v,3} = \mathbf{S}(v) \end{aligned} \tag{5}$$

The leaf model $\mathbb{L}_{\mathbf{\Omega}}$ computes 3D points by sampling the cross-section Bézier $\mathbf{C}_v$ at u using Eq. 1.

$$\mathbb{L}_{\mathbf{\Omega}}(u,v) = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}^T \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}}_{\text{Bernstein coefficients of cubic Bézier}} \begin{bmatrix} \mathbf{c}_{v,0} \\ \mathbf{c}_{v,1} \\ \mathbf{c}_{v,2} \\ \mathbf{c}_{v,3} \end{bmatrix} \tag{6}$$

To summarise, the set of leaf parameters $\mathbf{\Omega}$, given base and apex fixed at $(0,0,0)$ and $(0,1,0)$ respectively, consists of:

- control points $\{\mathbf{a}_1 \cdots \mathbf{a}_{m-1}\}$ of $\mathbf{A}$,
- $\{\bar{x}_{s_i}, \bar{v}_i, \bar{z}_{s_i}\}$ for control points of $\mathbf{S}$ (Eq. 3),
- $\{\bar{z}_{f_i}\}$ for control points of $\mathbf{F}$ (Eq. 4a), and
- $\{\bar{z}_{g_i}\}$ for control points of $\mathbf{G}$ (Eq. 4b).

We used 3rd degree Béziers for the central axis $(m = 4)$, and 4th degree curves for all other longitudinal Béziers $(n = 5)$, resulting in 21-D parameter space.

4. Automatic leaf segmentation

We describe computation of individual leaf point clouds from colour and depth data which are later used to compute the parametric model (Sec. 5). There is little research in automatic extraction of unordered fine-scale structures like leaves. Semantic segmentation from images [4], depth scans [50] and RGB-D superpixels [20] is suitable only for salient objects. As discussed in Sec. 2, previous tree reconstruction techniques did not handle foliage. Cheng *et al.* [13] extract repeated patterns using user intervention. Others can separate an image into foliage and background but cannot extract individual leaves [28, 17]. Our goal is to extract point clouds for each leaf separately. To this end, we segment the image into potential leaves, then assign a confidence metric to each leaf and retain the highest ranked leaves. We use these to refine the remaining leaves.

4.1. Initial segmentation of candidate leaves from each image independently

We start by oversegmenting the input images to create superpixels [2]. Superpixels provide fairly accurate colour segmentation (Fig. 2(b)). We collect the 3D points in each superpixel. We then fit a plane using RANSAC and user-provided threshold δ_{plane} (typically 2 cm), to the 3D points in each superpixel i and obtain a normal $\hat{\mathbf{n}}_i$. We also remove any 3D points that are far away from the mean of the 3D points in the superpixel after plane fitting. Superpixels that contain very few 3D points, or those that do not admit a valid plane fit are removed completely (Fig. 2(c)).

The remaining superpixels that contain 3D points and a valid plane fit are used as nodes of a superpixel graph. We add edges between superpixels that share a common boundary (Fig. 2(d)). We use planarity and 3D distance to assign an edge weight between two superpixels i and j. We measure planarity $d_{n_{ij}}$ by the cosine of angle between normals $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_j$ to the fitted planes. We measure 3D distance as the minimum 3D distance between points in one superpixel to the other as a fraction of the plane fitting threshold δ_{plane}.

$$d_{n_{ij}} = 1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j, \quad d_{p_{ij}} = \frac{\min\limits_{a,b} \|\mathbf{v}_i^{(a)} - \mathbf{v}_j^{(b)}\|}{\delta_{\text{plane}}} \tag{7}$$

where $\mathbf{v}_i^{(a)}$ is the a-th 3D point in i-th superpixel. We combine these terms into the affinity matrix $\mathbf{D}$ using a steep Gaussian kernel ($\sigma = 0.2$) to encourage isolated clusters.

$$\mathbf{D}_{ij} = e^{-\left(\frac{d_{n_{ij}}^2 + d_{p_{ij}}^2}{2\sigma^2}\right)} \tag{8}$$

We extract connected components from this graph after thresholding the affinity values at 0.85. We found affinity threshold more intuitive than the number or size of leaves required by most clustering techniques. We refer to the extracted segments as *candidate leaves* (Fig. 2(e)).

Leaf confidence Candidate leaves thus extracted often have spurious geometry from neighbouring regions in the image (Fig. 2(e), 4). We compute a leaf confidence metric to distinguish between well-segmented and conjoined leaves. We compute the minimum volume enclosing ellipse of candidate point clouds [23, 26] and use the percentage of enclosing ellipse occupied by the leaf mesh as confidence metric. Well-segmented meshes occupy most of the area while conjoined leaves have vacant space in the enclosing ellipse (Fig. 3).

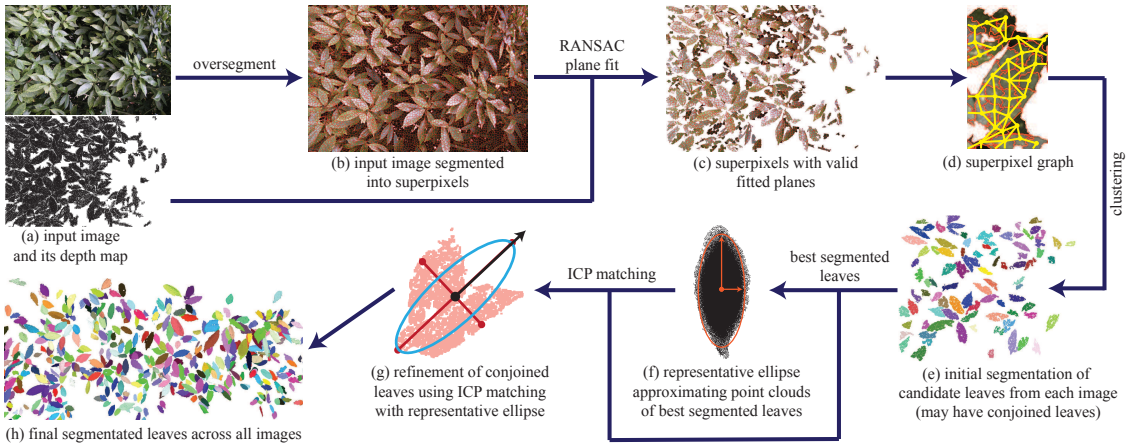


Figure 2. Leaf segmentation steps. Please refer to text for explanation of each step. Best appreciated on a computer screen.

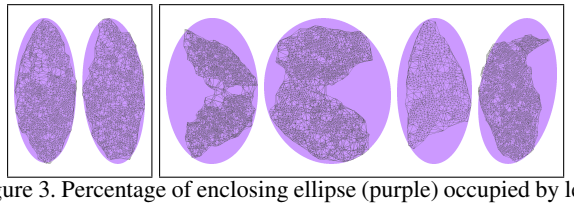


Figure 3. Percentage of enclosing ellipse (purple) occupied by leaf mesh as confidence. Well-segmented leaves have close to 100% confidence (left) while conjoined leaves, leaves with large missing or spurious chunks have low confidence score (right).

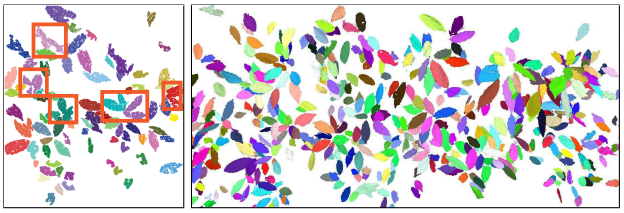


Figure 4. Leaf segmentation. Left: candidate leaves extracted from each image independently after initial segmentation may have conjoined leaves (annotated), and right: refining the segmentation across all images resolves duplicate and conjoined leaves.

4.2. Scale-invariant refinement across whole scene

We refine all candidate leaves by computing their similarity with a *scale-invariant template*. We retain 50 candidate leaves with highest confidence values across all images, scale them to unit length and compute an ellipse whose major-minor axis ratio is the same as the median length-width ratio of the high confidence leaves (Fig. 2(f)). This ellipse is the simplest scale-invariant representation of leaves in the dataset.

We then perform ICP matching [6] between the representative ellipse and candidate leaves, retaining matching subsets. We initialise ICP by aligning the template ellipse, shown in cyan in Fig. 2(g), along the major axis (shown in red) of the conjoined point cloud (shown in pink). ICP computes a rigid transform that best aligns the template to the candidate point cloud. We extract the set of points in the full scene within a δ_{plane} distance of the transformed template (δ_{plane} is the plane fitting threshold). This subset is free of spurious geometry (Fig. 4). While candidate leaves are extracted from each image independently, refinement is performed across the across points and leaves from all views, as highlighted in Fig. 4. As a result, this step also partially fills missing or occluded regions by extracting points visible in views other than the one from which the candidate point cloud was extracted. We use 300 iterations and δ_{plane} as the maximum correspondence distance for ICP. We repeat this process for each candidate leaf in decreasing order of confidence and size: we bin the confidence metric of all candidate leaves into 25 bins, and process the bins in decreasing order of confidence and leaves within each bin in decreasing order of size.

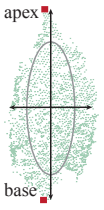
Bradley *et al*. [8] used non-rigid ICP to deform a prescanned exemplar leaf mesh to match the 3D scan. They had to prune a leaf and scan it in controlled conditions to obtain the exemplar leaf mesh. In contrast, our template ellipse is computed automatically. We use rigid ICP to match the template ellipse to the candidate point cloud and reject the non-matching points from the candidate. Our approach is automatic, simpler and extracts more leaves (Fig. 9).

5. Parametric model computation

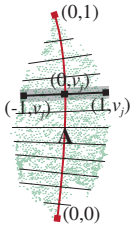
We now describe the computation of parametric leaves (Sec. 3) from segmented point clouds (Sec. 4). We estimate model parameters $\mathbf{\Omega}$ by matching the parametric leaf $\mathbb{L}_{\mathbf{\Omega}}$ to the segmented point cloud given its UV mapping $\mathbb{P}_{\mathbf{A}}$. The UV mapping $\mathbb{P}_{\mathbf{A}}$ is a latent variable estimated using the current estimate of parameters $\mathbf{\Omega}$. We alternate between computing $\mathbf{\Omega}$ and $\mathbb{P}_{\mathbf{A}}$ in an iterative optimisation.

Axis alignment and base/apex estimation We compute our leaf model in the xy plane such that z coor-

dinate represents displacement out of leaf plane (Sec 3). We first transform the leaf point cloud from world space coordinates to axis-aligned coordinates using SVD. We then identify the base and apex of the leaf as the extremities of the transformed point cloud along the y axis. We compute a rigid transformation that maps the base and apex to $(0,0,0)$ and $(0,1,0)$ respectively. The inverse of the combination of this transformation and the axis-alignment matrix give the transformation of the leaf model to world space. We initialise the central axis as a straight line joining the base and apex.



UV mapping In order to optimise for the leaf model $\mathbb{L}_{\mathbf{\Omega}}$, we compute a UV mapping $\mathbb{P}_{\mathbf{A}}$ that assigns (u,v) coordinates to each vertex in the point cloud with respect to the current estimate of axis $\mathbf{A}$, These (u,v) coordinates can be used to index into the point cloud and also leaf model $\mathbb{L}_{\mathbf{\Omega}}$ (Eq. 6). We draw scanlines (shown in black in the adjoining figure) at regular intervals $\{v_0, v_1 \cdots\} \in [0,1]$ perpendicular to the axis. For the scanline at v_j, we find all points that lie on or within δ distance (grey) of the the scanline, where δ is half the distance between successive scanlines. We use v_j as the v coordinate for each of these points, and normalised distance from axis $\mathbf{A}$ as u coordinate. The normalisation factor is the maximum distance from the axis across all scanlines. We use negative u coordinates for the left half.



Given the current estimate of $\mathbb{P}_{\mathbf{A}}$, we compute the ideal parameters $\hat{\mathbf{\Omega}}$ by minimising the distance between vertices with same (u,v) coordinates and the parametric model $\mathbb{L}_{\mathbf{\Omega}}$:

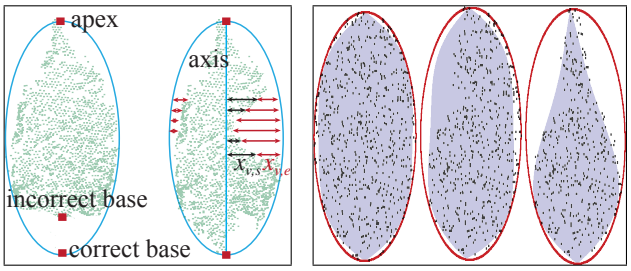
$$\hat{\mathbf{\Omega}} = \arg\min_{\mathbf{\Omega}} \sum_{\forall (u,v) \in \mathbb{P}_{\mathbf{A}}} \|\mathbb{L}_{\mathbf{\Omega}}(u,v) - \mathbb{P}_{\mathbf{A}}(u,v)\|^2 \quad (9)$$

5.1. Occlusion handling

Leaf point clouds may have missing sections because of occlusions or segmentation failing to capture the full leaf. Occluded extremities lead to incorrect base and apex, and occluded silhouettes confuse the UV mapping. These cause the optimisation in Eq. 9 to give incorrect results in the form of narrow or stunted leaves. We alleviate this by using the *minimum volume enclosing ellipse* [23, 26]. The enclosing ellipse extrapolates occluded sections of the point cloud towards the true silhouettes (Fig. 5(a)). We use the (x,y) coordinates of the major axis extremities of this ellipse as the (x,y) coordinates of the base and apex, since the extremities must lie on the enclosing ellipse.

We also use the ellipse to compute a confidence weight w_v for any scanline at v during UV mapping (Fig. 5(a)):

$$w_v = \frac{x_{v,s}}{x_{v,s} + x_{v,e}} \quad (10)$$



(a) UV mapping (b) Model fitting

Figure 5. Occlusion handling. (a) Point clouds with missing regions and their enclosing ellipse [23] (cyan). Missing top/bottom sections can cause incorrect base/apex estimation, while missing sides can lead to incorrect silhouettes. We use the extremities of the enclosing ellipse as base/apex and penalise scanlines whose left/right ends are far away from the ellipse (red arrows). (b) Model fitting with occlusions on synthetic test case. First shows ground truth fit with point cloud (grey), enclosing ellipse (red) and reconstructed leaf shape (purple), second shows that the enclosing ellipse guides the fit in regions with missing points and remains close to ground truth, and third shows that the result does not overfit to ellipse if the point cloud is from a narrow leaf.

where, $x_{v,s}$ is the distance of the farthest sample in the point cloud from the central axis on the same scanline as shown in Fig. 5(a), and $x_{v,e}$ is the distance of this point from the ellipse. This penalises samples on a scanline that fall significantly short of the enclosing ellipse. The visible sections of the leaf dominate the optimisation. For narrow leaves, both halves get equally penalised and the resulting shape follows the point cloud boundary. Therefore, the optimisation does not overfit to the enclosing ellipse for narrow leaves, and only intervenes for occluded sections (Fig. 5(b)). We incorporate the confidence weight w_v in Eq. 9.

$$\hat{\mathbf{\Omega}} = \arg\min_{\mathbf{\Omega}} \sum_{\forall (u,v) \in \mathbb{P}_{\mathbf{A}}} w_v \|\mathbb{L}_{\mathbf{\Omega}}(u,v) - \mathbb{P}_{\mathbf{A}}(u,v)\|^2 \quad (11)$$

We use iterative Levenberg-Marquardt optimisation to solve this non-linear energy function. Each iteration computes a new $\mathbf{\Omega}$, which we use to update the current UV mapping $\mathbb{P}_{\mathbf{A}}$. We initialise the axis to a straight line joining the base and apex, and $\mathbf{\Omega}$ to the enclosing ellipse. The optimisation converges within 4–15 iterations to produce a physically-based parametric leaf.

6. Artistic manipulation

We demonstrate semantically meaningful manipulation of leaves via intuitive handles. Morphable models [8] do not allow direct edits (Sec. 2), and NURBS or subdivision surfaces do not allow separation of editing tasks.

For all operations, we keep the base and apex fixed and manipulate Bézier control points. For the axis, we manipulate the control points directly. For other curves, we ma-

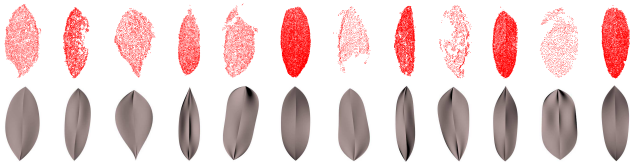


Figure 6. Leaf point clouds segmented from dense foliage and fitted parametric leaves. The fitting algorithm accounts for occlusions, and gracefully adapts to silhouettes and curvature. Best appreciated in the accompanying video.

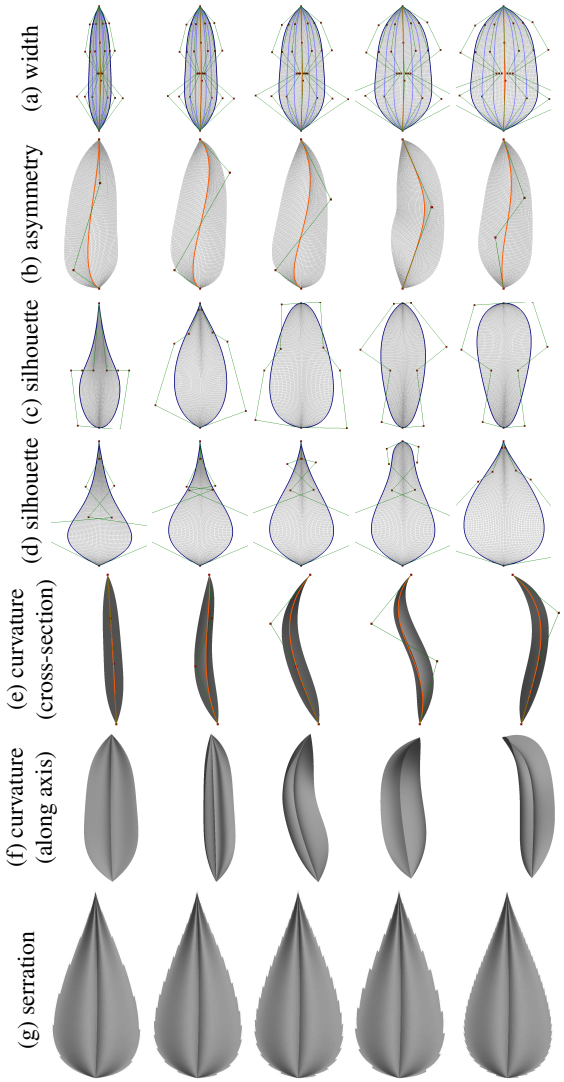


Figure 7. Different editing operations supported by our leaf model. Control polygon of the modified Béziers is also shown.

nipulate displacements with respect to axis. Our interface shows the average leaf of a dataset; these edits are applied identically to all leaves.

Width scaling We use a single slider to scale the lateral displacement of all control points of longitudinal Béziers to change the width uniformly (Fig. 7(a)).

Asymmetry The leaf model is parameterised relative to the central axis and symmetric if the axis is a straight line. We simulate asymmetric shapes by skewing the axis Bézier: we allow the artist to drag the control points of the axis in xy plane while keeping all other parameters fixed (Fig. 7(b)).

Silhouette shape We simulate a wide variety of leaf outlines by displacing control points relative to axis in the xy plane. This updates the all longitudinal Béziers in either half (Fig. 7(c),(d)).

Curvature Leaf curvature allows artists to express interesting variations and effects such as ageing. We use separate handles for editing curvature along the cross-section and the axis. Cross-section curvature refers to rolling the leaf about an axis parallel to the x axis. To this end, we displace axis control points outside the plane of the leaf. This deforms the whole surface of the leaf since it is parameterised relative to the axis (Fig. 7(e)). We can also roll each half of the leaf about an axis parallel to y axis by displacing control points of longitudinal Béziers outside the leaf plane. We provide three sliders: one for each of the three longitudinal Béziers (Fig. 7(f)). This gives sufficient control; providing a separate handle for each point did not add to the expressive ability and complicated the user interface.

Jittered silhouettes We can simulate serrated or noisy edges by jittering silhouette vertices (Fig. 7(g)). We store jitter values in a 1D texture or array. While evaluating leaf vertices from (u, v) using Eq. 6, we displace silhouette vertices ($u = \pm 1$) by looking up the amount of jitter using the v coordinate. The jitter is evaluated in OpenGL shaders during rendering; consequently there is not need to bake them as is the case with traditional representations like polygon meshes, NURBS etc. We can modify the density and degree of jitter in real-time, facilitating artistic usage.

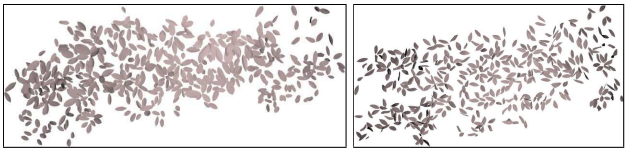
7. Results

We tested our C++ implementation on a 12-core 3.5 Ghz CPU with 32 GB RAM. Processing times vary from 18 to 52 minutes for datasets with 475 to 845 reconstructed leaves respectively. During rendering, we compute Bézier curves (Eq. 6) in an OpenGL vertex shader. This allows real-time rendering without storing tessellated meshes.

Parametric foliage We show reconstructed foliage for different datasets in Fig. 8. All datasets have 1–15 meters of foliage, captured by 30-40 images and point clouds extracted using multi-view stereo [15]. Foliage density makes segmentation ambiguous and frequent occlusions complicate model fitting. Our approach handles these challenges gracefully and produces visually plausible results.

Comparisons Most previous tree reconstruction techniques only extract branch structures [49, 52, 32, 31, 27, 40, 41], while others focus on singleton leaves [35, 16, 46, 10,

Figure 8. Parametric foliage reconstruction results. Please see accompanying video for more results.



(a) Ours (475 leaves) (b) Bradley *et al*. (418 leaves)

Figure 9. Comparison with Bradley *et al*. [8]: our result is denser and accounts for different leaf sizes.

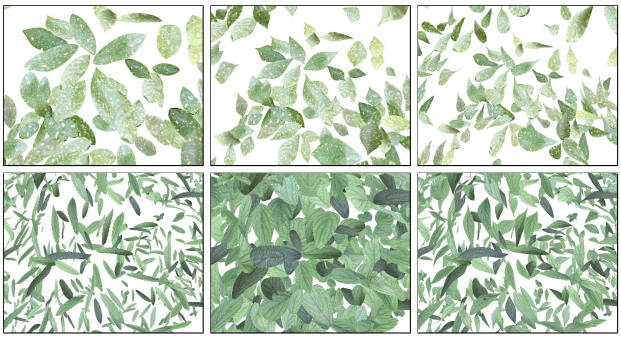


Figure 10. Artistic edits. Each row is a separate dataset showing edited versions of the same leaves. Please see accompanying video for real-time editing.

21, 51, 34, 24]. We compare our results to Bradley *et al*. [8], the only dense foliage reconstruction, on their dataset. Our results are noticeably denser as shown in Fig. 9. Our results are also more realistic because we account for different leaf sizes while they are restricted to the shape and size of a pre-scanned leaf exemplar leaf.

Artistic manipulation A key motivation for the work is to provide artistic control over the captured foliage. Fig. 10 shows a variety of artistic manipulations applied to the leaves viewed from the same viewpoint. The possibilities for styling and artistic control over curvature, silhouette and width, as demonstrated in the accompanying video, improves the ability to reach a desired appearance. These advantages facilitate adoption by artists.

8. Discussion

Our main limitation is that we model only single-lobed leaves. We consider multi-lobed leaves as future work. Compositing multiple single lobes is a promising next step which will also require extending the segmentation and model fitting algorithms. Nonetheless, our approach has the potential to generalise and is of significant value in its current form. Dense foliage has so many occlusions that almost half the leaves are not sufficiently visible and deemed unreliable during segmentation. This can be alleviated by densifying the reconstruction [8]. In the future, we will target quantitative evaluation using synthetically generated foliage, given that real ground truth foliage data is hard to acquire.

Conclusion Our goal is not just 3D reconstruction of the widest gamut of leaves. We seek to address artist requirements, which necessitate lightweight editable representation and automatic reconstruction. Within this context, we presented a parametric foliage model designed for easy rendering and artistic control. We proposed novel algorithms for segmenting leaves, and computing the model from colour and depth data. We envision real applications will use tree reconstruction [32] to compute branch structures and our work to provide controllable foliage. We addressed single-lobed leaves which constitute a majority of species found in nature and used in graphics applications. This work provides a robust platform for generalising to a broader class of vegetation in the future.

References

[1] Plant generation software packages. http://vterrain.org/Plants/plantsw.html. Last accessed: 2017-03-17. 2

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, Nov 2012. 4

[3] F. Anastacio, M. C. Sousa, F. Samavati, and J. A. Jorge. Modeling plant structures using concept sketches. In *NPAR*, pages 105–113, 2006. 2

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011. 1, 4

[5] B. Benes, N. Andrysco, and O. Stava. Interactive modeling of virtual ecosystems. In *Eurographics Workshop on Natural Phenomena*, pages 9–16, 2009. 2

[6] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, Feb 1992. 5

[7] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pages 187–194, 1999. 2

[8] D. Bradley, D. Nowrouzezahrai, and P. Beardsley. Image-based reconstruction and synthesis of dense foliage. *ACM Trans. Graph.*, 32(4):74:1–74:10, July 2013. 2, 5, 6, 8

[9] M. Cabral, N. Bonneel, S. Lefebvre, and G. Drettakis. Relighting photographs of tree canopies. *IEEE Trans. Vis. Comput. Graph.*, 17(10):1459–1474, Oct 2011. 2

[10] J. Chambelland, M. Dassot, B. Adam, N. Dons, P. Balandier, A. Marquier, M. Saudreau, G. Sonohat, and H. Sinoquet. A double-digitising method for building 3D virtual trees with non-planar leaves: application to the morphology and light-capture properties of young beech trees (Fagus sylvatica). *Functional Plant Biology*, 35(10):1059–1069, 2008. 2, 8

[11] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, July 2013. 2

[12] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using markov random field. *ACM Trans. Graph.*, 27(5):109:1–109:9, Dec. 2008. 2

[13] M.-M. Cheng, F.-L. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. RepFinder: Finding approximately repeated scene elements for image editing. *ACM Trans. Graph.*, 29(4):83:1–83:8, July 2010. 4

[14] O. Deussen and B. Lintermann. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer-Verlag, 2005. 2

[15] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, Aug 2010. 7

[16] S. M. Hong, B. Simpson, and G. V. G. Baranoski. Interactive venation-based leaf shape modeling. *Computer Animation and Virtual Worlds*, 16(3-4):415–427, 2005. 2, 8

[17] H. Huang, L. Zhang, and H.-C. Zhang. RepSnapping: Efficient image cutout for repeated scene elements. *Comput. Graph. Forum*, 30(7):2059–2066, 2011. 4

[18] T. Ijiri, S. Owada, M. Okabe, and T. Igarashi. Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3):720–726, July 2005. 1

[19] T. Ijiri, S. Yoshizawa, H. Yokota, and T. Igarashi. Flower modeling via X-ray computed tomography. *ACM Trans. Graph.*, 33(4):48:1–48:10, July 2014. 1

[20] I. Jebari and D. Filliat. Color and depth-based superpixels for background and object segmentation. *Procedia Engineering (International Symposium on Robotics and Intelligent Sensors)*, 41:1307–1315, 2012. 4

[21] S. Jeong, S.-H. Park, and C.-H. Kim. Simulation of morphology changes in drying leaves. *Comput. Graph. Forum*, 32(1):204–215, 2013. 2, 8

[22] D. M. Kempthorne, I. W. Turner, and J. A. Belward. A comparison of techniques for the reconstruction of leaf surfaces from scanned data. *SIAM Journal on Scientific Computing*, 36(6):B969–B988, 2014. 2

[23] L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Math. Oper. Res.*, 21(2):307–320, May 1996. 4, 6

[24] D. Kim and J. Kim. Procedural modeling and visualization of multiple leaves. *Multimedia Systems*, pages 1–15, 2016. 2, 8

[25] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and J. a. V. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *ECCV*, pages 502–516, 2012. 1

[26] P. Kumar and E. A. Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005. 4, 6

[27] C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall. Modeling and generating moving trees from video. *ACM Trans. Graph.*, 30(6):127:1–127:12, Dec. 2011. 2, 7

[28] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, Aug. 2004. 4

[29] A. Lindenmayer. Mathematical models for cellular interactions in development. *J. Theoretical Biology*, 18(3):280–315, 1968. 1, 2

[30] B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Comput. Graph. Appl.*, 19(1):56–65, Jan 1999. 2

[31] Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. Texture-lobes for tree modelling. *ACM Trans. Graph.*, 30(4):53:1–53:10, Aug. 2011. 2, 7

[32] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph.*, 29(6):151:1–151:8, Dec. 2010. 2, 7, 8

[33] B. I. Loch, J. A. Belward, and J. S. Hanan. Application of surface fitting techniques for the representation of leaf surfaces. In *MODSIM05: International Congress on Modelling and Simulation: Advances and Applications for Management and Decision Making*, pages 1272–1278, Dec. 2005. 2

[34] T. Miao, C. Zhao, X. Guo, and S. Lu. A framework for plant leaf modeling and shading. *Mathematical and Computer Modelling*, 58(34):710–718, 2013. 2, 3, 8

[35] L. Mundermann, P. MacMurchy, J. Pivovarov, and P. Prusinkiewicz. Modeling lobed leaves. In *Computer Graphics International*, pages 60–65, July 2003. 2, 8

[36] T. T. Nguyen, D. C. Slaughter, N. Max, J. N. Maloof, and N. Sinha. Structured light-based 3D reconstruction system for plants. *Sensors*, 15(8):18587–18612, 2015. 2

[37] M. Okabe, S. Owada, and T. Igarash. Interactive design of botanical trees using freehand sketches and example-based editing. *Comput. Graph. Forum*, 24(3):487–496, 2005. 2

[38] A. Owens, M. Cieslak, J. Hart, R. Classen-Bockhoff, and P. Prusinkiewicz. Modeling dense inflorescences. *ACM Trans. Graph.*, 35(4):136:1–136:14, July 2016. 1

[39] A. Peyrat, O. Terraz, S. Merillou, and E. Galin. Generating vast varieties of realistic leaves with parametric 2Gmap L-systems. *The Visual Computer*, 24(7-9):807–816, 2008. 2

[40] S. Pirk, T. Niese, O. Deussen, and B. Neubert. Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. Graph.*, 31(6):169:1–169:10, Nov. 2012. 2, 7

[41] S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Mch, B. Benes, and O. Deussen. Plastic trees: Interactive self-adapting botanical tree models. *ACM Trans. Graph.*, 31(4):50:1–50:10, July 2012. 2, 7

[42] P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. In *SIGGRAPH*, pages 351–358, 1994. 2

[43] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990. 2

[44] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based plant modeling. *ACM Trans. Graph.*, 25(3):599–604, July 2006. 2

[45] A. Reche-Martinez, I. Martin, and G. Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph.*, 23(3):720–727, Aug. 2004. 2

[46] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.*, 24(3):702–711, July 2005. 2, 8

[47] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen, and B. Benes. Inverse procedural modelling of trees. *Comput. Graph. Forum*, 2014. 2

[48] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11:1–11:14, Apr. 2011. 2

[49] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3), July 2007. 2, 7

[50] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. SemanticPaint: Interactive 3D labeling and learning at your fingertips. *ACM Trans. Graph.*, 34(5):154:1–154:17, Nov. 2015. 1, 4

[51] X. Wang, L. Li, and W. Chai. Geometric modeling of broad-leaf plants leaf based on B-spline. *Mathematical and Computer Modelling*, 58(34):564–572, 2013. 2, 3, 8

[52] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4), Oct. 2007. 2, 7

[53] F. Yan, M. Gong, D. Cohen-Or, O. Deussen, and B. Chen. Flower reconstruction from a single photo. *Comput. Graph. Forum*, 33(2):439–447, 2014. 1

[54] K. Yin, H. Huang, P. Long, A. Gaissinski, M. Gong, and A. Sharf. Full 3D plant reconstruction via intrusive acquisition. *Comput. Graph. Forum*, 35(1):272–284, 2016. 2

[55] Q. Zheng, X. Fan, M. Gong, A. Sharf, O. Deussen, and H. Huang. 4D reconstruction of blooming flowers. *Comput. Graph. Forum*, 2016. 1