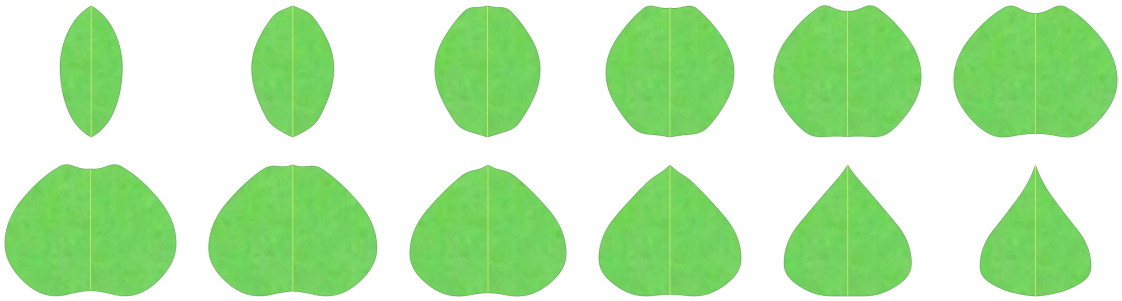
Leaf Modeling and Constrained Leaf Morphing in Leaf Space

Saurabh Garg [1], Leow Wee Kheng [1]

[1] School of Computing, National University of Singapore, Singapore
saurabhg@comp.nus.edu.sg, leowwk@comp.nus.edu.sg



Morphing from an elliptic leaf (first row, first image) to a deltoid leaf (second row, last image) with a constraint of a leaf with both basal and apical extension (second row, first image).

Abstract

Leaf modeling is a very important and challenging problem because of the wide variations in the shape, size, and structure of the leaves among different species of plants. The main drawback of existing methods for synthesizing leaves is that they are non-intuitive and tedious to use. With these methods, leaves of different shapes are either reconstructed from images individually or defined by different sets of complex rules. In this paper, we present a novel parametric leaf model based on botanical considerations for generating the geometric shape of a wide variety of leaves. The shape of the leaf is represented by a set of landmark points on the leaf boundary and tangents to the boundary at these points. The geometric shape of a leaf is generated by fitting quadratic B-spline curves to the landmark points and tangents. The proposed leaf model is intuitive and can be used to generate multiple instances of a leaf, each having the same overall shape but differs slightly in detail. In addition, a leaf morphing method is proposed for morphing leaf shapes in the parametric leaf space as defined by the leaf model. Reference leaf shapes can be easily specified by the user as soft constraints for leaf morphing. Given the source, target, and reference shapes, a NURBS curve is fitted over them in the leaf space to generate a smooth morphing path, which is then used to synthesize the specific leaf shapes along the path. This method can produce smooth morphing of leaf shapes for simulating leaf growth and for computer animation applications.

1 Introduction

Leaf modeling is a very important problem in botany, horticulture, agriculture, forestry, and ecology [26, 3, 30]. Leaf model can be used to simulate various plant functions and interaction of plants with the environment. These simulations can help to increase the production of plants by optimizing the use of available resources and produce healthy plants with more effective pest management [26, 3, 30]. The simulation of plant functions also enables botanists to conduct virtual experiments that are impractical otherwise, for example calculating the percentage of available sunlight intercepted by plants [3, 30], and understanding how diseases are spread by rainfall splash [27].

Building a realistic leaf model suitable for simulating its interaction with the environment consists of two steps: (1) modeling leaf shape in 2D plane, and (2) modeling leaf deformation in 3D space. This paper presents an algorithm for procedurally generating a wide variety of leaf shapes in 2D plane.

Modeling leaf shape is a difficult and challenging problem due to the wide variations in the shape, size, and structure of the leaves among different species of plants (Figure 4, 5, 9). Even in the same plant, no two leaves are identical. The challenge is to design a compact leaf model that can intuitively represent and synthesize a wide variety of shapes.

This paper presents a novel parametric approach for modeling, generating, and morphing leaf shapes. It models leaf shapes using landmark points and tangents along the leaf boundary. It can model and generate a wide variety of leaf shapes, as well as synthesize various instances of a given leaf, each having the same overall shape but differs in shape details. It is numerically stable so that a small change in the parameter values produces a small change in the leaf shape. As an application of the parametric leaf model, this paper presents a leaf morphing method. The leaf morphing is performed in the parametric leaf space, which is defined by the leaf model. Reference leaf shapes can be easily specified by the user as soft constraints for leaf morphing. This method produces smooth morphing of leaf shapes that can be used for simulating leaf growth and for computer animation applications.

2 Related work

Tree modeling: Many methods have been developed for modeling plants and trees: interactive [2, 6], rule-based [23, 11, 1], image-based [20], biology-based [28], and machine learning [4]. However, these methods model only the branching structure or the crown of trees and plants. The leaves, in these methods, are modeled using simple geometric shapes such as quadrilateral, triangle, ellipse, or disk textured mapped with a leaf image.

Leaf deformation modeling: Many methods have been proposed for modeling deformation of a leaf in 3D space [19, 9, 15, 13, 14]. All these methods use a scanned image of a leaf to construct leaf shape in 2D. Using the leaf shape, these methods then use free-form deformation

[19], skeleton-based deformation [9, 13, 14], and ad-hoc techniques [15] to model the leaf deformation.

The technique presented in this paper compliments existing tree modeling and leaf deformation methods by providing a procedural algorithm for generating a wide variety of leaves.

Leaf shape modeling: The existing methods for generating the geometric shapes of leaves can be divided into two categories: image-based and rule-based. The image-based methods attempt to reconstruct the surface of leaves from 2D images [24, 29, 16]. These methods work well for digitizing an existing plant for visualization in architectural walk-through or virtual reality. However, they are not suitable for synthesizing leaves of a wide variety of shapes. It is too tedious and time-consuming to capture and process data from real plants of all possible leaf shapes.

On the other hand, rule-based methods based on the L-system [12] define a set of rules for generating leaf shapes [7, 21, 25]. By including relevant rules, these methods can potentially generate a wide variety of leaf shapes. Unfortunately, these methods model leaves using complex rules that contain conditional and recursive statements. They are not intuitive to use as it is very difficult to imagine what the shape looks like by reading the rules. Moreover, there is no standard procedure to follow for creating the rules required for a given leaf. It can be very tedious and time-consuming to specify the rules. Due to the drawback of existing methods, it is tedious to perform leaf morphing based on these methods.

This paper introduces a novel method for modeling and generating leaf shapes. The use of landmark points and tangents on the boundary of a leaf makes leaf modeling simple and intuitive while at the same time general enough to model a wide variety of leaf shapes. The proposed method can use a reference image, like image-based methods, to digitize an existing leaf, or it can procedurally generate, like rule-based methods, large number of leaves having same overall shape but differ in details.

3 Overview of Leaf Modeling

There are two main types of leaves: *narrow leaves* and *broad leaves*. Narrow leaves look like needles, awl, or scales, and are found in plants that have adapted to conversing water. Broad leaves can be classified as *simple leaves* and *compound leaves* [5]. Each leaflet of a compound leaf can be considered as a simple leaf in our model. This paper focuses on modeling broad leaves because about 85% of the plant species on the Earth have broad leaves.

For ease of computational modeling and application, this paper categorizes (simple) broad leaves into two computational categories: *unilobed* and *multilobed*. Unilobed leaves have a single lobe, whereas multilobed leaves have multiple lobes. The proposed leaf model is based on unilobed leaves (Section 3.1). Multilobed leaves are modeled as a combination of unilobed leaves (Section 3.2).

3.1 Unilobed Leaves

Botanists categorize the shape of a unilobed leaf by the shapes at the *base*, the *apex*, and the *waist*, which is the widest part of the leaf [5]. The base shapes are categorized into six types: straight, concave, convex, concavo-convex, complex, and cordate. Both concavo-convex and complex bases have multiple points of inflections along the leaf margins (the boundaries). Leaves with cordate base have extensions below the base called *basal extensions* (Figure 5l).

The apex shapes are categorized into four types: straight, convex, acuminate (i.e., concave or concavo-convex), and emarginate. Emarginate apex have extensions above the apex called *apical extensions* (Figure 5m).

Table 1: Common shape types of unilobed leaves. C: concave, S: straight, V: convex, X: with extension.

Waist Shape	Base Shape	Apex Shape	Number
Elliptic	C, S, V, X	C, S, V, X	16
Ovate	C, S, V, X	C, S, V, X	16
Obovate	C, S, V, X	C, S, V, X	16
Oblong	V	V	1
Linear	S	S	1
Total			50

The waist shapes are categorized into five types, namely elliptic, ovate, obovate, oblong, and linear, depending on the location and extent of the waist. For elliptic, ovate, and obovate shapes (Figure 5a, h, i), the widest part of the leaf is, respectively, in the middle, near the base, and near the apex of the leaf. In oblong shape (Figure 5k), opposite sides of the leaf in the middle portion are parallel. In linear shape (Figure 5l), the widest part of the leaf is very small (less then one tenth) compared to the length of the leaf.

Considering all possible combinations of these shapes, there are $5 \times 6 \times 4 = 120$ possible types of leaf shape. However, not all of them occur naturally in real leaves. For example, oblong leaves (Figure 5k) always have convex base and convex apex. Leaves with linear shape (Figure 5f) are very thin compared to their lengths. So, there is only one oblong shape and one linear shape. It is estimated, from the real leaf samples that we have collected, that about 26 types of unilobed leaf shapes occur commonly in nature.

In order not to induce extraneous complexity into the proposed leaf model, the following shape features are omitted:

- Teeth along the leaf margin (Figure 7) are omitted as they do not contribute significantly to the overall shape of the leaf. They can be added to the margin using methods such as curve analogies [31].
- Leaves with complex base shape are omitted. The number of leaf types with complex base shape is very small. So, they can be omitted.

In summary, 50 types of unilobed leaf shapes are modeled by the proposed model (Table 1, Figure 4, 5). Of these 50 shapes, 26 occur naturally. The remaining 24 shapes may occur in nature but we are unable to find real leaf examples of them. Note that each type of leaf shape admits many variations depending on the aspect ratio, and the amount of concavity, convexity, and extension of the base and the apex. Moreover, the divisions between shape types can be fuzzy. For example, straight base is a transitional shape between concave and convex bases. There is no strict rule as to how straight a base needs to be before it is classified as straight as opposed to concave or convex. Nevertheless, these shape types serve as a useful method for botanists and the general users to intuitively describe the shape of a leaf.

Some leaves have asymmetric shapes (Figure 6). These leaf shapes can be modeled either with different shape types for the left and the right side, or with the same shape but different parameter values. Some leaves have long slender tips called *drip tips* (Figure 7). The apex shapes of these leaves are initially concave and then convex.

3.2 Multilobed Leaves

There are three basic types of multilobed leaves: palmately lobed, pinnately lobed, and bilobed. In a palmately lobed leaf (Figure 9b–f), the lobes originate at the base of the leaf. These leaves have an odd number of lobes. In a pinnately lobed leaf (Figure 9g), the lobes originate along the primary vein of the leaf. These leaves typically have many lobes, and the number of lobes can be even or odd. Leaves with an odd number of lobes have a lobe at their apexes. A bilobed leaf (Figure 9e) has two lobes that originate at the base. Unlike a palmately lobed leaf, there is no lobe at the apex.

Multilobed leaves can be symmetric or asymmetric, i.e., the lobes on the left and right sides of the leaves can have the same or slightly different shapes. Each lobe in a multilobed leaf can be symmetric or asymmetric. Multilobed leaves can also have teeth along the margin. As for unilobed leaves, teeth are omitted because they do not contribute significantly to the overall shape of a leaf.

4 Modeling of Unilobed Leaves

This section presents a parametric leaf model of unilobed leaf shapes. A leaf shape is represented by a set of landmark points on its margin and tangents to the margin at these points. These parameters can be specified intuitively by the user using a GUI and a reference image (Figure 1). The parameters of the leaf model are used by the leaf shape generation algorithm to generate the curves for the leaf margin.

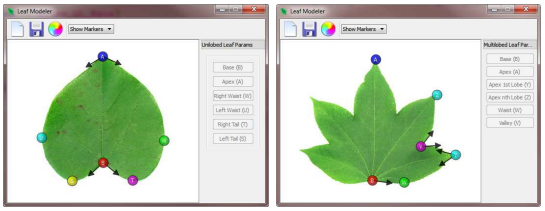
4.1 Model Parameters

The parametric leaf model is defined on a local coordinate system placed on the leaf with the origin at the base and the y-axis pointing towards the apex of the leaf. The x-y plane is set as the plane of the leaf surface. The primary vein is assumed to be straight. It is defined to be aligned with the y-axis and have a unit length (Figure 5.1) All parameters are defined relative to the primary vein. In this way, a leaf instance can be placed in some global coordinate system by appropriate scaling, rotation, and translation. The surface of a leaf is divided by the primary vein into the left side and the right side. Parameters for the two sides are defined separately so that asymmetric leaf shapes can be modeled.

For a unilobed leaf with basal extensions, one side of its margin is defined by four landmark points and the corresponding unit tangents at these points (Figure 1b). Landmark point $\mathbf{p}_b$ is the base, which is fixed at $(0, 0)$. $\mathbf{p}_t$ is called the *tail*, at which the basal extension is maximum. $\mathbf{p}_w$ is the waist, at which the width is maximum. $\mathbf{p}_a$ is the apex, which is fixed at $(0, 1)$. The tangent $\mathbf{t}_b$ can vary clockwise from $(1, 0)$ to $(0, -1)$ and $\mathbf{t}_a$ can vary counter-clockwise from $(0, 1)$ to $(-1, 0)$. By definition of the tail and waist, respectively, $\mathbf{t}_t$ is parallel to the x-axis and $\mathbf{t}_w$ is parallel to the y-axis. Thus, there are only 6 free parameters for defining one side of the margin: θ_b, θ_a, $\mathbf{p}_t = (x_t, y_t)$, and $\mathbf{p}_w = (x_w, y_w)$. The landmark points and tangents are related to these parameters as follows:

$$\begin{array}{ll} \mathbf{p}_b = (0, 0), & \mathbf{t}_b = (\sin\theta_b, \cos\theta_b), \\ \mathbf{p}_t = (x_t, y_t), & \mathbf{t}_t = (1, 0), \\ \mathbf{p}_w = (x_w, y_w), & \mathbf{t}_w = (0, 1), \\ \mathbf{p}_a = (0, 1), & \mathbf{t}_a = (-\sin\theta_a, \cos\theta_a). \end{array} \tag{1}$$

Other leaf types are defined in a similar manner. In particular, a leaf without extension has 4 free parameters, a leaf with apical extensions has 6, and a leaf with both basal and apical extensions has 8 free parameters.



(a) (b)

Figure 1: GUI for specifying landmark points and tangents. (a) Unilobed leaf. (b) Multilobed leaf.

4.2 Leaf Shape Generation Algorithm

The margin of a leaf is generated by fitting a pair of quadratic B-spline curves to the landmark points and tangents, one for each side of the leaf. Each B-spline curve passes through the points $\mathbf{p}_i$ and the unit tangents to the B-spline curve at the points $\mathbf{p}_i$ are $\mathbf{t}_i$. B-spline curves are used because they allow for intuitive control over the leaf shape.

A degree-d B-spline is a piecewise polynomial curve defined as follows [22]:

$$\mathbf{p}(u) = (x(u), y(u)) = \sum_{k=0}^{n} B_{k,d}(u)\mathbf{q}_k \tag{2}$$

where $\mathbf{p}(u)$ are points on the B-spline curve parameterized by u that lies in the range $[0, 1]$. The points $\mathbf{q}_k, k = 0, \ldots, n$, are the control points. The functions $B_{k,d}(u)$ are the B-spline basis functions. Degree-2 B-spline curves are chosen because degree-1 B-spline curves are just piecewise linear segments passing through the control points, and degree-3 or higher-degree B-spline curves can produce overly complex shapes that may have self intersections (Figure 2).

We apply the algorithm in [22] to generate B-spline curves to fit the landmark points of the leaf model. The algorithm finds the parameters of a quadratic B-spline curve that passes through a set of landmark points $\mathbf{p}_i, i = 1, \ldots, N$, such that the unit tangents to the curve at points $\mathbf{p}_i$ are $\mathbf{t}_i$. For each point $\mathbf{p}_i$ with tangent $\mathbf{t}_i$, a parameter value u_i can be determined for the B-spline curve that passes through it:

$$\mathbf{p}_i = \mathbf{p}(u_i) = \sum_{k=0}^{n} B_{k,2}(u_i)\mathbf{q}_k \tag{3}$$

$$\alpha_i \mathbf{t}_i = \mathbf{p}'(u_i) = \sum_{k=0}^{n} B'_{k,2}(u_i)\mathbf{q}_k, \tag{4}$$

where $\mathbf{q}_k$, $0 \le k \le n$, are the control points, and α_i is a scalar that scales the unit tangent $\mathbf{t}_i$ to match the first derivative of the curve at u_i.

Parameters u_i, and α_i are estimated by approximating the B-spline curve by a polyline formed by connecting the points $\mathbf{p}_i$. Then, the position of the control points $\mathbf{q}_k$ are computed by solving Eq. 3 and 4. To obtain a unique set of control points $\mathbf{q}_k$, the number of control points must be equal to the number of equations. Since there are $2N$ equations, the number of control points $(n + 1)$ must be equal to $2N$.

The ideal choice for the parameter values u_i are the normalized arc lengths of the B-spline curve. Since the curve is not yet known, the arc lengths are approximated using chord lengths between points $\mathbf{p}_i$ [22]. Let D denote

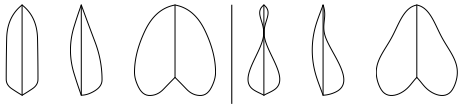


Figure 2: B-spline fitting results. (Left) Degree-2 B-splines produce desirable shapes. (Right) Degree-3 B-splines produce overly complex shapes that can have self intersections.

the total chord length:

$$D = \sum_{i=2}^{N} ||\mathbf{p}_i - \mathbf{p}_{i-1}||. \tag{5}$$

Then, the B-spline parameters are defined as

$$\begin{aligned} &u_1 = 0,\ u_N = 1, \\ &u_i = u_{i-1} + \frac{||\mathbf{p}_i - \mathbf{p}_{i-1}||}{D},\ \text{for } i = 2, \ldots, N-1. \end{aligned} \tag{6}$$

For leaf shapes without basal and apical extension, $N = 3$. So, the linear equations that relate the landmark points $\mathbf{p}_i$ and control points $\mathbf{q}_k$ (Eq. 3, 4) are defined by a single parameter u_2 together with the positions and tangents of the landmark points. For leaf shapes with either basal extension or apical extension (but not both), $N = 4$, and Eq. 3 and 4 are defined by two parameters u_2 and u_3 together with the landmark points. Leaf shapes with both basal and apical extensions have $N = 5$ and three free parameters.

It is important that the leaf shape generation algorithm is numerically stable so that small change in parameter values produces small change in the generated shape. The algorithm is stable if the coefficient matrix of Eq. 3 has a small condition number. Figure 3 plots the condition number with respect to the defining u_i for the case of $N = 3$ and $N = 4$. It is clear that the condition number is small except when the defining u_i is very close to zero or one. These exceptions occur when the waist is very close to the base or the apex, or, in the case of $N = 4$, the basal extension is very close to the base or the apical extension is very close to the apex.

When the defining u_i is equal to zero or one, the coefficient matrix is singular. Then, the coefficient matrix is not invertible and the control points of B-spline curve cannot be estimated by matrix inversion. In this case, closed form solutions of Eq. 3 and 4 can be obtained to estimate the control points. In the implementation, closed form solutions are used to avoid the singular matrix problem.

4.3 Leaf Shape Generation Examples

Our leaf model can generate 50 types of unilobed leaf shapes as categorized in Section 3.1 (Table 1). Figure 4 illustrates 48 of them with different combinations of waist, base, and apex shapes. The remaining two shape types, oblong and linear, are illustrated in Figure 5(k, f). Among them, 26 shape types have real leaf examples. The others may also occur in nature but we are unable to find real leaf examples for them.

Figs. 5 illustrates leaf shapes that have been given specific names by botanists [10]. Note that some of these leaf shapes belong to the same type. For example elliptic, oval, and orbicular leaves in Figure 5 belong to the same type with elliptic (mid) waist, convex base, and convex apex. They differ by their aspect ratios and the roundedness of their shapes. These figures show that the leaf shapes generated by our model match those of the real leaves very well.

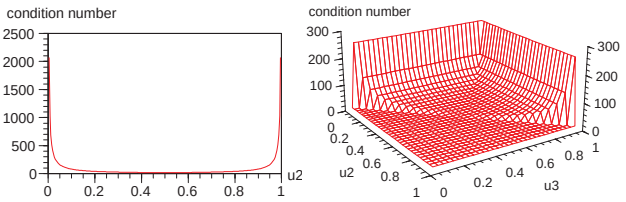


Figure 3: Condition number of system equation. (a) $N = 3$. (b) $N = 4$. Condition number is small everywhere except when u_2 or u_3 is very close to 0 or 1.

Figure 6 shows that our model can generate asymmetric leaf shapes. Figure 7 illustrates more complex leaf shapes. Top row shows that our model can generate leaf shapes with drip tips. For a leaf with a very long and slender drip tip, the match between the generated tip and the real tip is not perfect (Figure 7, third case) due to the small number of landmark points used to generate the B-spline curves.

Technically, the match can be made perfect by including an additional landmark point. Bottom row of Figure 7 shows that the generated margins fit the overall shapes of the leaves with teeth. As discussed in Section 3.1, teeth are omitted in our model and can be added using methods such as curve analogies [31].

Figure 8 shows that different instances of a leaf shape can be generated by perturbing the parameter values. The generated instances have the same overall shapes as the reference shape but differ in shape details. This property allows the user to easily generate realistic leaf instances that differ in detailed shapes for practical applications.

Timing for generating leaf instances was measured on a laptop with Intel Quad Core processor. The proposed method can generate about 550 leaf instances per second ($\sim$ 220 triangles). This shows that the laminar shape generation algorithm is fast and can be used for quickly generating large number of leaves instances.

5 Modeling of Multilobed Leaves

The shape of a multilobed leaf is modeled as a combination of unilobed leaves. Each lobe is modeled as for a unilobed leaf. To combine the lobes, they are first arranged and placed in space. The placement of lobes is defined by the primary veins of multilobed leaves. For a palmately lobed leaf (Figure 9b–f) and a bilobed leaf (Figure 9a), the lobes are placed at the base of the primary vein. For a pinnately lobed leaf (Figure 9g), the lobes are placed along the primary vein.

The positions and orientations of each lobe can be specified by the user (Figure 1b), if detailed specification is desired. Otherwise, our model places them according to a linear relationship as follows. Let r_i, θ_i, l_i denote the position (y-coordinate), orientation, and length of lobe i. Then,

$$r_i = ir_1 + \frac{i(i-1)}{2}\Delta s, \tag{7}$$

$$\theta_i = \theta_1 + (i-1)\frac{\theta_n - \theta_1}{n-1}, \tag{8}$$

$$l_i = l_1 + (i-1)\frac{l_n - l_1}{n-1}, \tag{9}$$

where Δs is a constant rate of change of spacing among the lobes, and n is half the number of lobes. In addition, valley points and valley tangents may be specified.

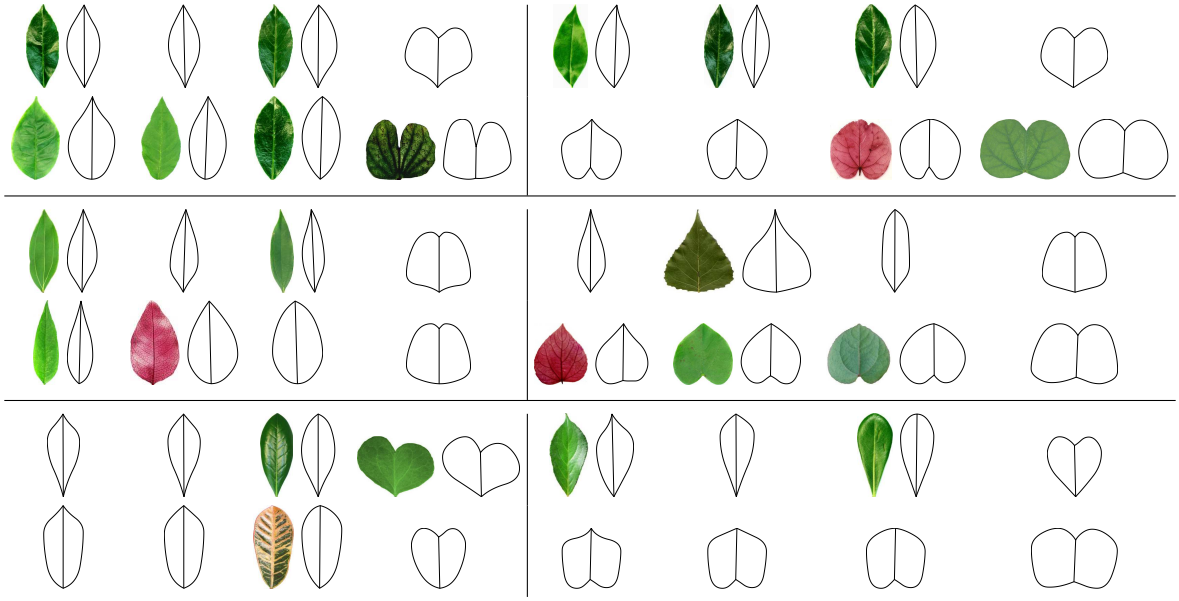


Figure 4: Unilobed leaf shapes. (Rows 1, 2) Elliptic (mid) waist, (Rows 3, 4) ovate (low) waist, (Rows 5, 6) obovate (high) waist. (Odd row, left column) Concave base, (odd row, right column) straight base, (even row, left column) convex base, (even row, right column) basal extension. (Each half row, from left to right) Concave, straight, convex apex, and apical extension. Real leaf examples are shown for available cases.

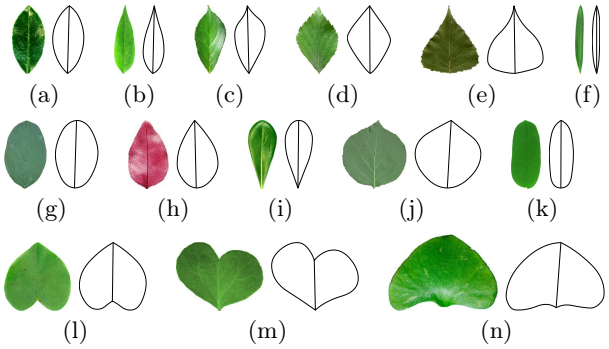


Figure 5: Unilobed leaf shapes with specific names. (a) Elliptic, (b) lanceolate, (c) oblanceolate, (d) rhomboidal, (e) deltoid, (f) linear, (g) oval, (h) ovate, (i) obovate, (j) orbicular, (k) oblong, (l) cordate, (m) obcordate, (n) reniform.

In generating the margin of a multilobed leaf, the lobes are first scaled by their lobe lengths and placed at the required positions and orientations. Next, adjacent lobes are combined by fitting additional B-splines curves at the valley points. In the case that the valley points are not specified, the intersections of adjacent margins form the valleys. On the other hand, if adjacent margins do not intersect, then additional B-spline curves are fitted at default valley points placed at a predefined distance from the primary vein. Figure 9 illustrates generation of multilobed leaves. The generated shapes match the shapes of real leaves well. Time taken for generating multilobed leaves depends on the number of lobes. About 370 leaf instances per second can be generated for a leaf with 3 lobes ($\sim$ 700 triangles) and 180 leaf instances per second for a leaf with 7 lobes ($\sim$ 1800 triangles).

6 Constrained Leaf Morphing

6.1 Overview of Leaf Morphing

Leaves generally change shapes as they grow [17, 18]. In some species, the leaf shapes can change significantly from one type to the other [18]. Leaf morphing can be used to simulate leaf growth for biological studies, as well as to generate morphing sequences for computer animation. To produce the correct morphing sequence for a particular species of leaves, shape change has to be constrained. The constraints can be most easily provided by the user as an ordered list of intermediate reference shapes. They should be soft constraints so that the user can determine how much each reference shape influences the morphing sequence to produce the desired shape change.

A straightforward method of leaf morphing is to use the reference shapes as key frames and perform linear morph between the key frames. This method has several drawbacks. First, key frames are hard constraints, which is not desirable. Second, shape change may be abrupt. To generate a smooth nonlinear morph across the key frames, it is necessary to measure shape change, which is very difficult to accomplished in the physical space of the leaf shape.

In contrast, our method models leaf shapes by shape parameters. So, shape change can be easily measured in the parameter space of leaf shapes. Leaf morphing can then be cast as a problem of obtaining a smooth morphing path in the parameter space under soft constraints of reference shapes.

This paper presents only the method of morphing unilobed leaf shapes. The method for morphing between unilobed and multilobed leaves is discussed elsewhere. However, an example of morphing from unilobed to multilobed leaf is presented. Without loss of generality, we shall focus our discussion on morphing symmetric unilobed leaf shapes. The same method can be applied to morphing asymmetric unilobed leaf shapes with a doubling of the dimensionality of the leaf space.

Figure 6: Asymmetric leaf shapes.

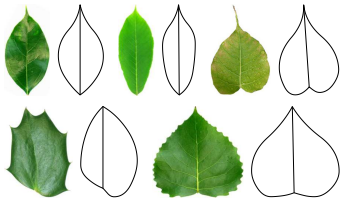


Figure 7: Leaves with more complex shapes. (Top) Leaves with drip tips. (Bottom) Leaves with teeth.

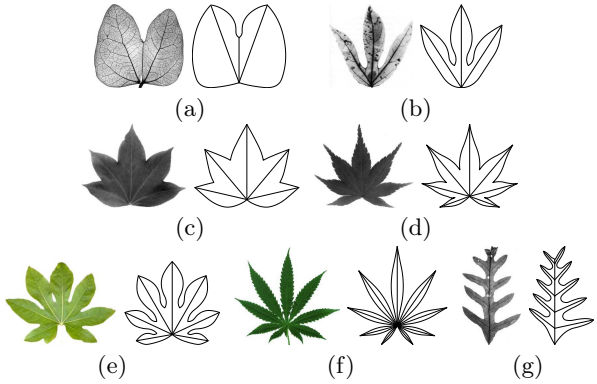


Figure 9: Multilobed leaves. (a) Bilobed, (b–f) palmately lobed, (g) pinnately lobed.

6.2 Unified Leaf Space

As discussed in Section 4, depending on whether a leaf shape has basal and apical extensions, the number of free parameters for a (symmetric) leaf is either 4, 6, or 8. So, when the source, target, and reference shapes have different degrees of freedom, they have to mapped into a *unified leaf space* before morphing can proceed.

Let S and S' denote two consecutive shapes in the morphing sequence. Without loss of generality, suppose S has no basal extension and S' has basal extension. Then, S and S' are mapped into a unified leaf space as follows. All the landmark points $\mathbf{p}_i$ and tangents $\mathbf{t}_i$ of S are mapped to the corresponding landmark points $\mathbf{p}'_j$ and tangents $\mathbf{t}'_j$ of S', i.e., base to base, waist to waist, and apex to apex. The tail $\mathbf{p}'_t$ of S' is mapped to a point $\mathbf{p}$ on S such that $\mathbf{p}$ has the same arc-length ratio from the base and the waist as does $\mathbf{p}'_t$ in S':

$$\frac{L(\mathbf{p}, \mathbf{p}_w)}{L(\mathbf{p}, \mathbf{p}_b)} = \frac{L(\mathbf{p}'_t, \mathbf{p}'_w)}{L(\mathbf{p}'_t, \mathbf{p}'_b)} \tag{10}$$

where L is the arc length measured along the leaf margin.

The point $\mathbf{p}$ in S is dependent on the B-spline curve that fits the margin of S. So, its tangent can be computed from Eq. 4. As the point $\mathbf{p}$ moves towards $\mathbf{p}'_t$, its tangent also changes from the computed value to the (default) value of $\mathbf{t}'_t$. So, the unified leaf space requires one additional dimension compared to the degree of freedom of the leaf shape with basal extensions so as to include the tangent angle of $\mathbf{p}$. In this case, the number of dimensions is 7. Now, S and S' can be mapped to two shape points in the 7-D leaf space.

Analogous mapping can be applied for S' with apical extensions. When S' has both basal and apical extensions, two additional landmark points are added to S, and the unified leaf space has two additional dimensions compared to the degree of freedom of the leaf shape with both basal and apical extensions. In this case, the number of dimensions is 10. When S and S' have different types of extensions, then an additional landmark point is added to both S and S' to map their extension points to the others. This process also raises the dimensionality of the unified leaf space to 10.

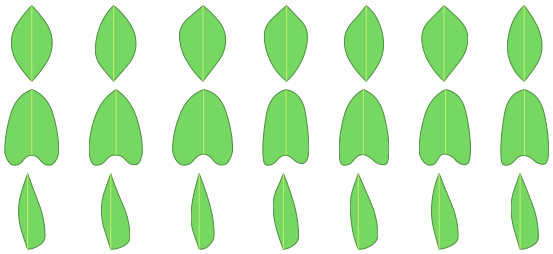


Figure 8: Generation of leaf instances. Various instances of the leaf shapes in the first column are generated by perturbing the parameter values randomly by up to 20%.

6.3 Generation of Morphing Path

Given the source shape $S = R_0$, the target shape $T = R_{n+1}$, and an ordered list of reference shapes R_k, $1 \leq k \leq n$, the lowest-dimensional leaf space $\mathcal{S}$ that unifies these shapes is determined. Next, the shapes are mapped to $\mathcal{S}$ by first mapping the tails, if they exist, using the following algorithm:

1. Initialize list L to contain all R_k.
2. Repeat until L is empty:
   (a) Identify shapes R_i in L that contain tails.
   (b) For each immediate neighbor R_j of R_i that does not contain a tail, map the tail of R_i to R_j and insert a tail into R_j. Then, remove R_i from L.

Mapping of shoulders (the point where apical extension is maximum), if they exist, is performed in a similar manner. After mapping, all the shapes have the same degree of freedom and they correspond to shape vectors in $\mathcal{S}$.

Next, a morphing path $\mathcal{M}$ is computed by fitting a NURBS curve to the shape points in $\mathcal{S}$ such that it passes through S and T, and approximates R_k:

$$\mathbf{s}(u) = \sum_{k=0}^{n+1} w_k B_{k,2}(u) \mathbf{R}_k \left(\sum_{k=0}^{n+1} w_k B_{k,2}(u) \right)^{-1} \tag{11}$$

where $\mathbf{s}(u)$ is a point on $\mathcal{M}$ and w_k is the weight of shape vector $\mathbf{R}_k$ in the unified leaf space. The weights w_0 and w_{n+1} of $R_0 = S$ and $R_{n+1} = T$ are set to 1.

After obtaining the morphing path $\mathcal{M}$, shape vectors are sampled at regular interval along $\mathcal{M}$, and intermediate shapes are generated from the shape vectors using the method discussed in Section 4.2. The larger the weights w_k, the more $\mathcal{M}$ is pulled towards the reference shapes. So, the user can determine the amount of influence imposed by each reference shape on the morphing sequence.

Figure 10 compares linear morphing with our proposed nonlinear morphing. The results show that shape change is more smooth with nonlinear morphing.

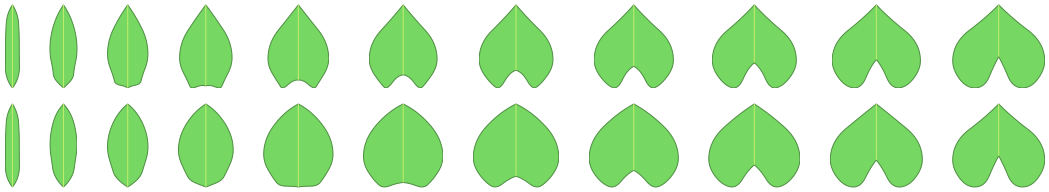


Figure 10: (Row 2) Nonlinear morphing produces smoother shape change than (Row 1) linear morphing.

Figure 11 illustrates examples of constrained leaf morphing. The first shape was morphed to the last shape, constrained by the shape in the middle of row 3. Rows 1, 2, and 3 show the morphing sequences with, respectively, zero, small, and large weight. With zero weight, no constraint was imposed and the first shape was morphed to the last shape by first losing basal extensions followed by growing apical extensions. With non-zero weight, the apical extensions grew out before the basal extensions were lost. The larger weight imposed more influence by the reference shape. Row 4 shows an example of morphing with two reference shapes.

Figure 12 illustrates examples of constrained leaf morphing for simulating the growth of real leaf [18]. This morphing sequence also scaled the generated shapes according to the actual sizes of the real leaves. Row 1 shows the real leaves at various stages of development. In the unconstrained morphing sequence (row 2), the aspect ratio of the leaf shapes remain roughly unchanged. When constrained by just the third real leaf, the leaf shapes remained elongated longer (row 3), and basal extensions developed earlier. With both constraints, the leaf shapes remained elongated longer (row 4), and basal extensions developed later.

Figure 13 illustrates an example of morphing from a unilobed to a multilobed leaf.

7 Conclusions

This paper presented a novel parametric leaf model that can generate a wide variety of leaf shapes. The model is intuitive to use and it generates leaf shapes that match those of real leaves very well. It can also generate multiple instances of a leaf, each having the same overall shape but differs in shape details. A morphing algorithm is proposed that performs constrained leaf shape morphing in a unified leaf space. It can generate smooth morphing sequences under the soft constraints of reference leaf shapes. It can be used to simulate leaf growth for biological studies, as well as to generate morphing sequences for computer animation. This constrained leaf morphing method will be extended to multilobed leaves in the future.

One possible extension of the existing method is to develop a parametric leaf deformation algorithm similar to the leaf generation algorithm. For simulating interaction of plant with environment, it is necessary to model physics-based deformation. One way to extend proposed leaf model is to use Cosserat tree model [8] on the venation pattern of leaf. Another minor improvement is to automatically estimate the leaf shape parameters from the input reference image.

References

[1] F. Anastacio, P. Prusinkiewicz, and M. C. Sousa. Sketch-based parameterization of L-systems using illustration-inspired construction lines and depth modulation. *Computers & Graphics*, 33:440–451, 2009.

[2] F. Anastacio, M. C. Sousa, F. Samavati, and J. A. Jorge. Modeling plant structures using concept sketches. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 105–113, 2006.

[3] C. Birch, B. Andrieu, C. Fournier, J. Vos, and P. Room. Modelling kinetics of plant canopy architecture - concepts and applications. *European Journal of Agronomy*, 19(4):519–533, 2003.

[4] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using markov random field. In *ACM SIGGRAPH Asia 2008*, pages 109:1–109:9, 2008.

[5] B. Ellis, D. C. Daly, L. J. Hickey, K. R. Johnson, J. D. Mitchell, P. Wilf, and S. Wing. *Manual of Leaf Architecture*. Cornell University Press, 2009.

[6] B. Ganster and R. Klein. 1-2-tree: Semantic modeling and editing of trees. In O. Deussen, D. Keim, and D. Saupe, editors, *Vision, Modeling, and Visualization 2008*, October 2008.

[7] M. S. Hammel, P. Prusinkiewicz, and B. Wyvill. Modelling compound leaves using implicit contours. In *Proc. Int. Conf. of Computer Graphics Society on Visual Computing: Integrating Computer Graphics with Computer Vision*, pages 199–212, 1992.

[8] L. Hao. *Predictive Surgical Simulation for Preoperative Planning of Complex Cardiac Surgeries*. PhD thesis, School of Computing, National University of Singapore, 2010.

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

[10] A. Huxley, M. Griffiths, and M. Levy, editors. *The New RHS Dictionary of Gardening*, volume 1. Macmillan and Stockton Press, 1992.

[11] T. Ijiri, S. Owada, and T. Igarashi. The sketch L-system: Global control of tree modeling using free-form strokes. In *Smart Graphics*, pages 138–146, 2006.

[12] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. In *Journal of Theoretical Biology*, volume 18, pages 280–315, 1968.

[13] S. Lu, X. Guo, C. Zhao, and C. Li. Model and animate plant leaf wilting. In *Proc. Int. Conf. on Technologies for E-Learning and Digital Entertainment*, pages 728–735, 2008.

[14] S. Lu, C. Zhao, and X. Guo. Venation skeleton-based modeling plant leaf wilting. *Int. Journal of Computer Games Technology*, 2009:1–8, 2009.

[15] S. Lu, C. Zhao, X. Guo, and C. Li. Modeling curled leaves. In *Proc. Int. Conf. on Image and Graphics*, pages 909–914, 2007.

[16] W. Ma, H. Zha, J. Liu, X. Zhang, and B. Xiang. Image-based plant modeling by knowing leaves from their apexes. In *Proc. Int. Conf. on Pattern Recog.*, pages 1–4, 2008.

[17] R. Maksymowych. *Analysis of Leaf Development*. Cambridge University Press, 1973.

[18] E. Milne-Redhead. Variation in leaf-shape within a species: Some examples from the gold coast. *Kew Bulletin*, 5(2):261–264, 1950.

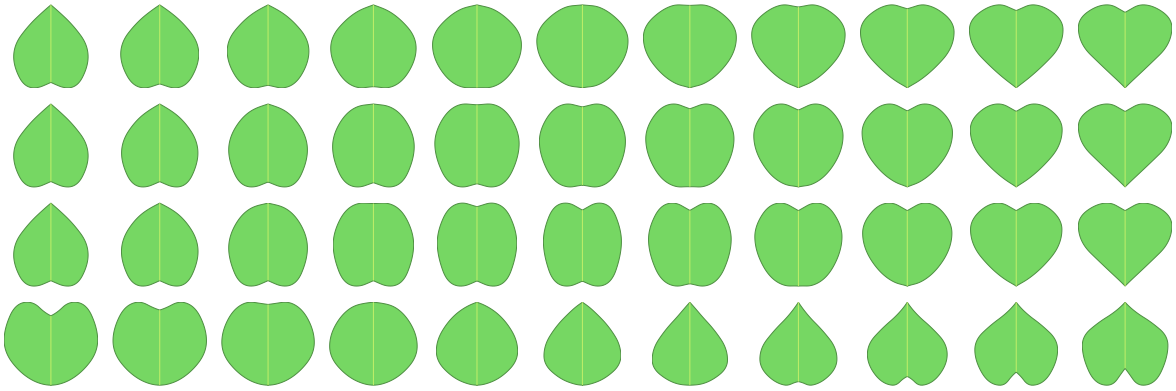


Figure 11: Constrained leaf morphing. Examples of morphing the first shape to the last shape. (Row 1) No constraint. (Row 2) Small constraint on shape 6. (Row 3) Large constraint on shape 6. (Row 4) Constrained by shapes 4 and 7.

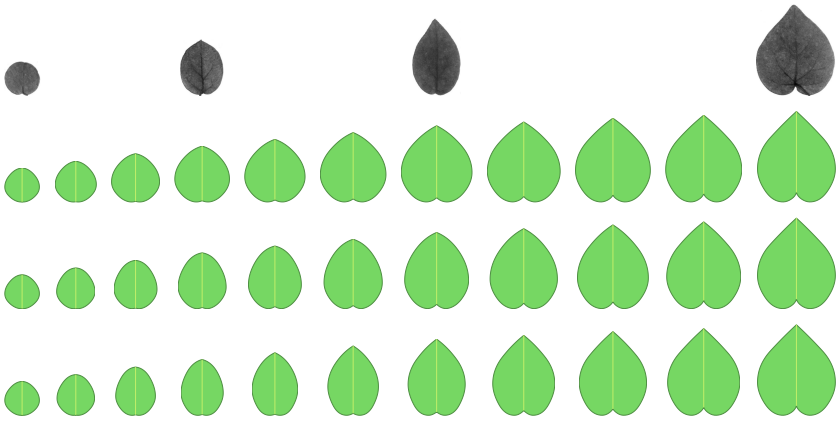


Figure 12: Leaf morphing for real leaf. (Row 1) Real leaves. (Rows 2–4) Morphing with 0, 1 (third real leaf), and 2 constraints.

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

[20] B. Neubert, T. Franken, and O. Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):88:1–8, 2007.

[21] 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):807–816, 2008.

[22] L. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag New York, Inc., 1997.

[23] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300, 2001.

[24] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based plant modeling. In *ACM SIGGRAPH*, pages 599–604, 2006.

[25] Y. Rodkaew, C. Lursinsap, T. Fujimoto, and S. Siripant. Modeling leaf shapes using L-systems and genetic algorithms. In *Proc. Int. Conf. NICOGRAPH*, pages 73–78, 2002.

[26] P. Room, J. Hanan, and P. Prusinkiewicz. Virtual plants: new perspectives for ecologists, pathologists and agricultural scientists. *Trends in Plant Science*, 1(1):33–38, 1996.

[27] S. Saint-Jean, M. Chelle, and L. Huber. Modelling water transfer by rain-splash in a 3D canopy using monte carlo integration. In *Agricultural and Forest Meteorology*, pages 183–196, 2004.

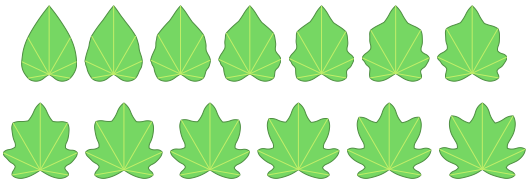


Figure 13: Morphing from a unilobed to a multilobed leaf.

[28] L. Streit, P. Federl, and M. C. Sousa. Modelling plant variation through growth. *Computer Graphics Forum*, 24(3):497–506, 2005.

[29] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Transactions on Graphics*, 26(3), 2007.

[30] J. Vos, J. Evers, G. Buck-Sorlin, B. Andrieu, M. Chelle, and P. de Visser. Functional-structural plant modelling: a new versatile tool in crop science. In *Journal of Experimental Botany*, pages 2101–2115, 2009.

[31] S. Zelinka and M. Garland. Mesh modelling with curve analogies. In *Proc. Pacific Conf. on Computer Graphics and Applications*, pages 94–98, 2004.