PROCEDURAL MODELING AND CONSTRAINED MORPHING OF LEAVES

SAURABH GARG
(B.Tech. (Hons.), Banaras Hindu University, India, 2002)

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

2011

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Saurabh Garg
May 2011

Acknowledgements

Finishing the PhD thesis has been a long and hard journey. I have been fortunate to met many people who have encouraged and helped me along the way. Here, I would like to express my gratitude to the people who have helped made this thesis a reality.

Working with my thesis advisor, Dr. Leow Wee Kheng, has been very enriching experience. He taught me not only how to do research but also, more importantly, how to think independently. He also taught me, through endless number of drafts, how to write well. Though, I still have a lot to learn. I thank him for being so patient, supportive, encouraging, and inspiring throughout the PhD.

I thank my thesis committee: Dr. Terence Sim, Dr. Low Kok Lim, and Dr. Alan Cheng for reading this thesis and providing insightful comments to improve the thesis. For the large part this thesis was supported by the research scholarship by NUS, I thank them for the opportunity.

In the beginning, I had a chance to work with Dr. Ng Teck Khim. I thank him for being so supportive and allowing me to explore interesting problems. It was always fun and informative to listen to his stories on life, management, and research.

I am thankful to have so many great labmates. Wang Rui Xuan was very supportive and advised me in most difficult times. Li Hao was always available for discussions and I learned a lot from him. Harish Katti supported me through the last year of the thesis. Ehsan Rehman, Hanna Kurniawati, Piyush Kanti Bhunre, and Raj were wonderful lunch companions and we had lots of interesting discussions. Zhang Sheng, Lu HaiYun, Jean-Romain Dalle, Pradeep Kumar Atre, and Ding Fong made computer vision lab a fun and stimulating place to work in.

I am grateful to have excellent friends who made my life easy and fun. Hemendra Singh Negi helped me settle down when I first came to Singapore. Satish Kumar Verma was fun to be with and I had blast watching all those movies with him. Amit Bansal was very easy going and we had lots of interesting discussions late

into night. Ankit Goel has been a great friend and I thank him for all the help and support he has provided over years and most importantly for introducing me to my wife. Navendu Singh became like a brother to me and was a great mentor. He was very patient and unconditionally supported me through the most difficult time in his life. I wish he was here to see me finish.

Most importantly, I thank my wife for being so encouraging and supportive in last couple of years. Without her unconditional love, support, and sacrifices, I would not have been able to finish this thesis. I also thank my parents for teaching me good values, making me independent, and being always there for me. Lastly, my three year old son made me happy when I was most stressed and taught me how to enjoy little things again.

Saurabh Garg
May 2011

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. 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.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Motivation

Leaf modeling is a very important problem in botany, horticulture, agriculture, forestry, and ecology [RHP96, BAF+03, VEBS+09]. Leaf model can be used to simulate various plant functions and interaction of plants with the environment. These simulations can help us increase the production of plants by optimizing the use of available resources and produce healthy plants with more effective pest management. The simulation of plant functions also enables us to conduct virtual experiments that are impractical otherwise, for example calculating the percentage of available sunlight intercepted by plants.

One of the important applications of leaf modeling is to simulate the interception of sunlight by leaves [BAF+03, Loc04, VEBS+09]. The total amount of sunlight available determines how many plants can grow at the same time within a given area of land. If there are too few plants, then sunlight is not fully utilized. If there are too many plants, then there is not enough sunlight for all the plants. The simulation of light interception by leaves can be used to determine the optimal population and position of plants for maximizing the use of available sunlight. This can help to increase plant production for a given area of land.

Another important application of leaf modeling is in pest management [RHP96, BAF+03, VEBS+09]. A popular method of protecting plants from weeds, insects and other pathogens is by spraying a liquid pesticide over plants. The pesticide cover the surface of leaves and a leaf is protected as long as it is covered with the pesticide. With current spraying techniques, majority of the pesticide fails to land on the leaves [RHNB00, HRY03]. Simulating the interaction of spraying of pesticide with leaf canopy can help us understand and develop better spraying techniques. It can also help us understand if pesticides can penetrate the plant canopy and reach the inner parts of the plant. Additionally, simulating the motion of pesticide droplets on the surface of a leaf can help us determine the optimal amount of pesticide to be sprayed. If too much pesticide is sprayed, it will drip from a leaf to lower leaves and finally to the ground. This method not only wastes pesticide but also pollutes the environment.

Leaf modeling can be used to develop preventive cures for some diseases spread by rainfall splash. When a rain drop hits the surface of a leaf, it splashes and spreads pathogens present on the leaves or in the rain drops. Simulation of rainfall splash on the surfaces of leaves can help us understand how such diseases spread and develop preventive measures for them [SML04].

The level of details of leaf shapes required for simulating the interaction of plants with the environment depends on the scale of the experiment. Statistical models such as turbid medium [MML+95, KKM+98] which model leaf area per unit area of soil (leaf area index) are sufficient for simulating at the canopy level. However, at the organ level (leaf, fruits, flowers etc.), statistical models are not accurate enough and surface model of leaves are required [CCS+07, BLEG+11]. Existing methods in botanical literature for simulating interaction of plants with the environment uses simple geometric primitives such as right angled triangle [Ski04] or a small number of polygons to approximate leaf shape [CEC+07].

Leaf modeling is a very difficult and challenging problem because of the wide variations in the shape, size, and structure of the leaves among different species of plants (Figure 1.1). Even in the same plant, no two leaves are identical. The challenge is to design a model of leaf that can intuitively represent a wide variety of leaves using as few parameters as possible. For some applications, it is necessary to model the deformation of leaf surface due to interaction with the environment. Thus, it is important that the leaf model can include physical properties for physically accurate deformation.

Few methods have been developed for generating the geometric shapes of leaves. Among them, image-based methods attempt to reconstruct the surface of leaves from 2D images [QTZ+06, MZL+08]. These methods work well for digitizing an existing plant for visualization in architectural walk-through or virtual reality. However, they are not suitable for biological simulations because it is too tedious and time-consuming to capture and process data from real plants of all possible shapes and sizes. On the other hand, rule-based methods define a set of rules for generating leaf shapes [HPW92, RLFS02, PTMG08]. By including the relevant rules, these methods can potentially generate a wide variety of leaf shapes. Unfortunately, existing methods model leaves using either implicit functions [HPW92] or complex rules containing conditional and recursive statements [RLFS02, PTMG08] (Figures 3.1 and 3.2). So, 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 that generate the required shape of a given leaf. It can be very tedious and time-consuming to specify the rules.

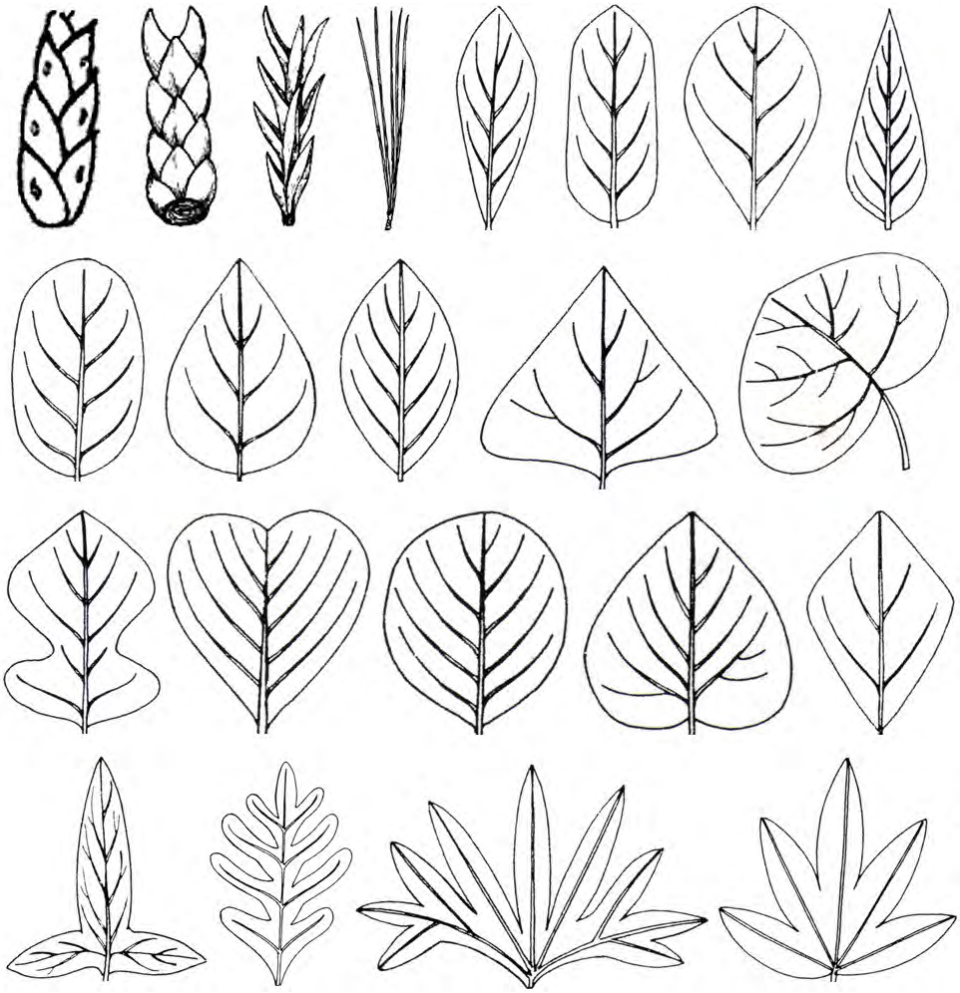


Figure 1.1: Diversity in the shapes of leaves (Sources: 1, 4: [Bro10]; 2, 3: [Cum10]; 5–22: [HGL92]).

1.2 Research Objectives

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.

The first goal of this research is to develop a leaf model for generating the geometric shape of a wide variety of leaves. Theoretically, a leaf model which can generate very detailed leaf shapes would be most accurate and can be used for simulations at very fine scale. However, due to the complex nature of simulations, such a leaf model would be computationally infeasible. Therefore, the proposed leaf model generates the overall shape of leaves but ignore ornamentations such as teeth (jagged edges along the leaf boundary), drip-tips (very sharp

(a) (b) (c) (d)

Figure 1.2: Various details in the leaf shapes. (a) 2D leaf without any details, (b) leaf with jagged edges (teeth), (c) leaf with very sharp tip (drip-tip), and (d) leaf warped in 3D space.

leaf tips) and warping of leaf shape in 3D (Figure 1.2). As a leaf ages, it changes color and starts to wrinkle slowly due to decreasing moisture content. This wrinkling effect can be considered as a type of leaf warping. In order to keep the leaf model simple, aging is not considered in this thesis.

For the leaf model to be useful in a wide variety of applications, it should have the following properties:

- **General:** The leaf model should be able to generate the geometric shape of a wide variety of leaves. There are two main types of leaves: narrow leaves and broad leaves [Bre10]. In this thesis, narrow leaves are not modeled because in comparison to broad leaves, in which leaf surface have negligible thickness, narrow leaves have 3D structure. Thus, narrow leaves would need a different kind of model. Since about 85% of the plant species on the Earth have broad leaves, modeling them will cover majority of leaves.
- **Intuitive:** The leaf model should be intuitive so that it is possible to specify exactly which kind of leaf shape will be generated. This is an important property because each plant has a specific kind of leaves. For plant simulation, it is necessary to generate the correct leaf shapes of a particular plant.
- **Concise:** The leaf model should be able to represent the shapes of a wide variety of leaves using as few parameters as possible. This is to ensure that the leaf model is simple and as general as possible while still being intuitive.
- **Generative:** Since no two leaves are identical even in a single plant, the leaf model should be able to generate many instances of the same kind of leaf. The instances of a leaf have the same overall shape but differ in size and shape details.
- **Numerically Stable:** The leaf shape generation algorithm should be numerically stable. A small change in the parameters should produce a small change in the leaf

shape. This ensures that modifying the parameter values produce predictable change in the generated shape.

The second goal of this thesis is to develop a morphing method for leaf shapes. Leaf morphing is useful for modeling leaf growth for plants in which young and adult leaves are of different shapes and sizes, and for computer animation applications. For leaf morphing to be useful in these applications, it should have the following properties:

- **General:** It should be possible to morph between any two leaf shapes modeled by their leaf models.
- **Automatic:** Leaf morphing should be automatic and should not rely on the user to establish correspondence between the two leaf shapes.
- **Soft constraints:** To produce the correct morphing sequence for modeling growth of a particular species of leaves, shape change has to be constrained. Computer animation applications also require control over the intermediate leaf shapes. Thus, leaf morphing should be constrained by reference leaf shapes that are provided as soft constraints.

The contributions of this thesis are as follows:

- Design of a leaf model for intuitively specifying the geometric shapes of a wide variety of leaves. A leaf shape is represented by a set of parameters specifying important geometric features of the leaf shape.
- Development of an efficient algorithm for creating instances of various kinds of leaves. The algorithm is numerically stable so that small change in parameter values produce predictable change in the generated shape.
- Development of an algorithm for unifying leaf spaces of different kinds of leaves. This is to allow for morphing between any two leaf shapes modeled by the leaf models.
- Development of an algorithm for constrained morphing of leaf shapes in the unified parametric leaf space. The constraints are reference leaf shapes specified by the user.

1.3 Thesis Organization

This thesis presents a novel parametric approach for modeling, generating, and morphing leaf shapes. In order to understand the variations in the shapes of natural leaves, it is necessary to first discuss the botanical classification of the types of leaves and their characteristics (Chapter 2). Next, existing leaf modeling methods are reviewed in Chapter 3. For ease of computational modeling and application, the botanical characteristics discussed in Chapter 2 are used to enumerate and characterize the leaf shapes that are modeled by the leaf model in

Chapter 4. This thesis categorizes leaves into two broad categories: unilobed and multilobed leaves. The proposed leaf model is based on unilobed leaves. Chapter 5 presents the leaf model and the leaf shape generation algorithm for unilobed leaves. Then, in Chapter 6 the leaf model for unilobed leaves is extended to multilobed leaves. Constrained leaf morphing using soft constraints is discussed in Chapter 7. The limitations of the leaf model and morphing and possible future work are discussed in Chapter 8. Finally, Chapter 9 concludes this thesis.

Chapter 2

Botanical Background

The overall goal of this research is to develop a leaf model for generating a wide variety of leaves. To achieve this goal, the types of leaves that can be modeled must be characterized (Section 2.1) and the structure of these leaves should be understood (Section 2.2).

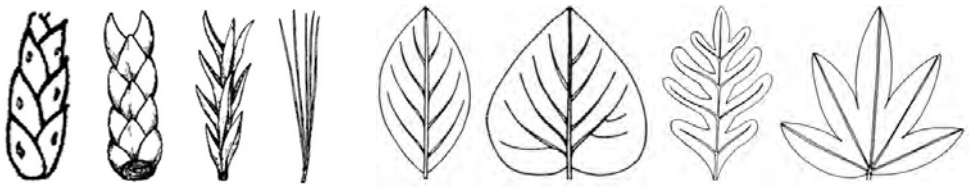
2.1 Types of Leaves

There are over 300,000 species of plants on the Earth consisting of leaves having huge variations in the shape, size, and structure. Plants can be broadly classified into four divisions [Arm10, cma10]: bryophytes (mosses, liverworts, and hornworts), pteridophytes (club-moss, horsetails, and ferns), gymnosperms (conifers, cycads, and gingko), and angiosperms (monocots, and dicots). Of these, bryophytes, pteridophytes, and gymnosperms typically have narrow leaves. Leaves from these plants have adapted to conserve water and are needle-like, awl-like or scale-like (Figure 2.1a). On the other hand, angiosperms (flowering plants) have broad leaves with negligible thickness compared to the leaf surface area (Figure 2.1b). This thesis considers only broad leaves because narrow leaves have 3D structure and cannot be modeled using same method as broad leaves.

2.2 Structure of Broad Leaves

A typical broad leaf is made up of two parts [EDH+09]: (1) petiole, where the leaf is attached to the branch and (2) blade or lamina, which is the main part of the leaf. A leaf with a single continuous blade is called a simple leaf (Figure 2.2a) and a leaf with a blade that is divided into a number of smaller parts (leaflets) is called a compound leaf. If the leaflets are attached to the apex of the petiole, it is called palmately compound leaf (Figure 2.2b). If the leaflets are arranged along the rachis which is an extension of petiole, it is called pinnately compound leaf (Figures 2.2c to 2.2e).

The lamina of a simple leaf has four main parts: base, apex, margin, and veins (Figure 2.3).



(a) (b)

Figure 2.1: Types of leaves. (a) Narrow leaves found in bryophytes, pteridophytes, and gymnosperms are slender (Sources: 1, 4: [Bro10]; 2, 3: [Cum10]). (b) Broad leaves found in angiosperms are wide and their thickness is negligible compared to the surface area (Source: [HGL92]).

The base is the proximal (closest to the petiole) 25% of the lamina. The apex is the distal (farthest from the petiole) 25% of the lamina. The margin forms the boundary of a leaf. The veins are embedded in the lamina and are used for transporting water and other nutrients.

Based on curvature of the margin, the shape of the base is categorized into six types [EDH+09]: straight, concave, convex, concavo-convex, complex, and cordate (Figure 2.4). In a cordate base, the lamina extends below the base to form the basal extension. Similarly, based on curvature of the margin, the shape of the apex is categorized into four types [EDH+09]: straight, convex, acuminate (concave or concavo-convex), and emarginate (Figure 2.5). In an emarginate apex, the lamina extends above the apex to form the apical extension.

A simple leaf can be categorized into five types based on the position and the extent of the maximum width of the lamina [EDH+09]: elliptic, obovate, ovate, oblong, and linear (Figure 2.6). In elliptic, obovate and ovate leaves, the widest part of the lamina is in the middle one fifth, distal two fifth and proximal two fifth, respectively. In oblong leaves, the opposite sides of the lamina are parallel for at least the middle one third. The leaves with linear shape are very thin. Their widths are less then one tenth of the lengths of the lamina. Some of the common leaf shapes have been given specific names by botanists [HGL92]. These leaf shapes are illustrated in Figure 2.7.

A simple leaf can be categorized into three types based on the marginal projections[1] [EDH+09] (Figure 2.8). If the margin of a leaf is smooth (no projection), it is called entire. If it has small teeth-like projections, it is called toothed. If it has large projections, resulting in distinguishable lobes, it is called lobed. Leaves in which the lobes start radially from the base are called palmately-lobed and leaves in which the lobes start along the primary vein

[1] In botany, marginal projection is the term used for protrusions of the lamina along the leaf margin.

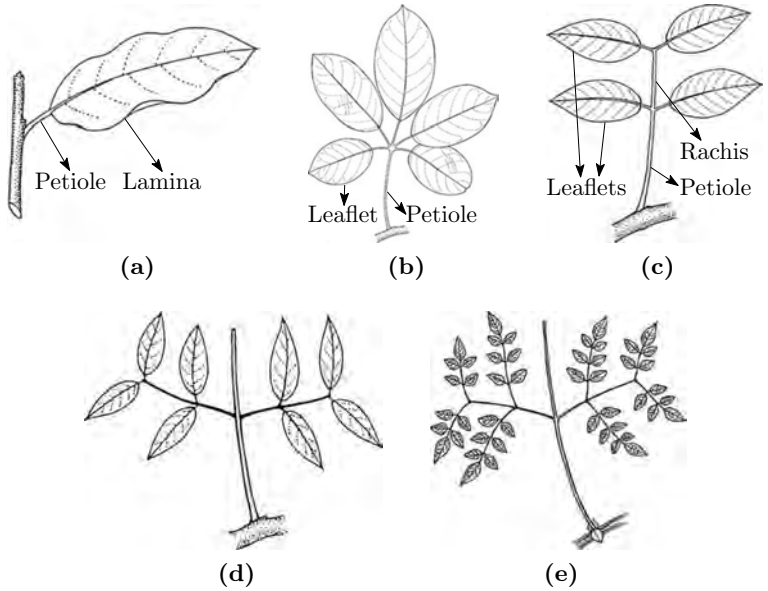


Figure 2.2: Organization of broad leaves. (a) A simple leaf has a single lamina. (b–e) A compound leaf has a lamina that splits into a number of small leaflets. (b) In a palmately compound leaf, the leaflets are attached to the apex of the petiole. (c–e) In a pinnately compound leaf, the leaflets are arranged along the rachis. A pinnately compound leaf can be (c) once pinnate, (d) bipinnate, or (e) tripinnate (Source: [EDH+09]).

are called pinnately-lobed. Lobed leaves can also have toothed margins.

The shape of the lamina on the two sides of the primary vein can be asymmetric. There are two types of asymmetries in simple leaves: asymmetric maximum width and asymmetric base. Asymmetric maximum width occurs when the positions and the extents of the maximum width of the lamina are different on the two sides of the primary vein (Figure 2.9a). Asymmetric base is further divided into two types: (1) the shape of the basal extensions are different on the two side of the primary vein (Figure 2.9b) and (2) only one side of the leaf has basal extension (Figure 2.9c).

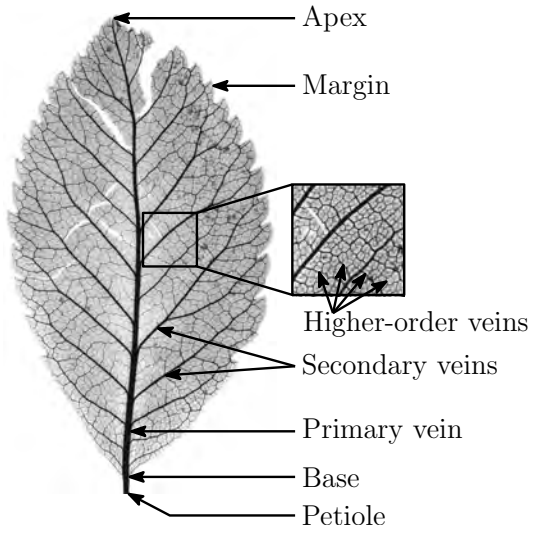


Figure 2.3: Parts of a simple Leaf. A simple leaf has a single blade (lamina) consisting of base, apex, margin and veins. The veins are categorized into primary veins, secondary veins, and higher-order veins depending on their course and thickness (Source: [EDH+09]).

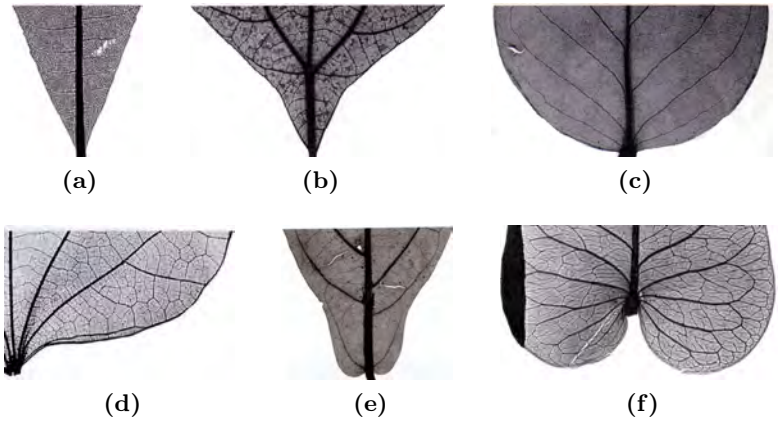


Figure 2.4: The types of base shapes. (a) Straight: the margin is straight. (b) Concave: the margin curves towards the primary vein. (c) Convex: the margin curves away from the primary vein. (d) Concavo-convex: the margin is concave proximally and convex distally. (e) Complex: the margin has more then one point of inflection. (f) Cordate: the margin extends below the base (Source: [EDH+09]).

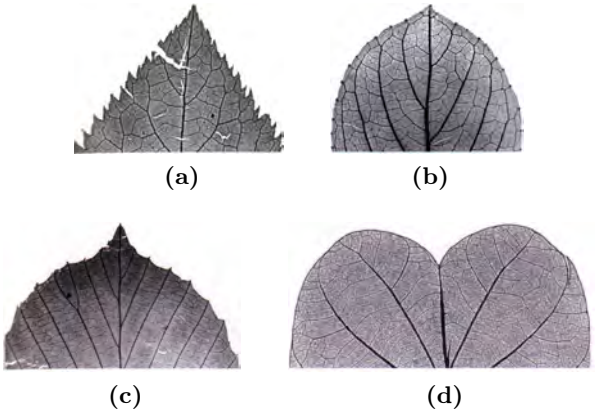


Figure 2.5: The types of apex shapes. (a) Straight: the margin is straight. (b) Convex: the margin curves away from the primary vein. (c) Acuminate: the margin is concave proximally and convex distally or concave only. (d) Emarginate: the margin extends above the apex (Source: [EDH+09]).

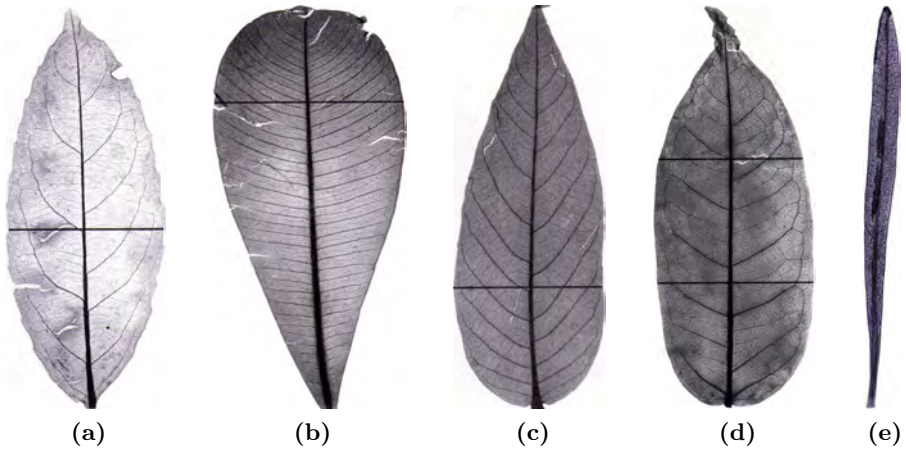


Figure 2.6: The types of leaves based on the position and the extent of the maximum width of the lamina. (a) Elliptic: the widest part of the lamina is in the middle one fifth of the leaf. (b) Obovate: the widest part of the lamina is in the distal two fifth. (c) Ovate: the widest part of the lamina is in the proximal two fifth. (d) Oblong: the opposite sides of lamina are parallel for at least the middle one third. (e) Linear: the widest part of the leaf is very small (less then one tenth) compared to the length of the leaf (Source: [EDH+09]).

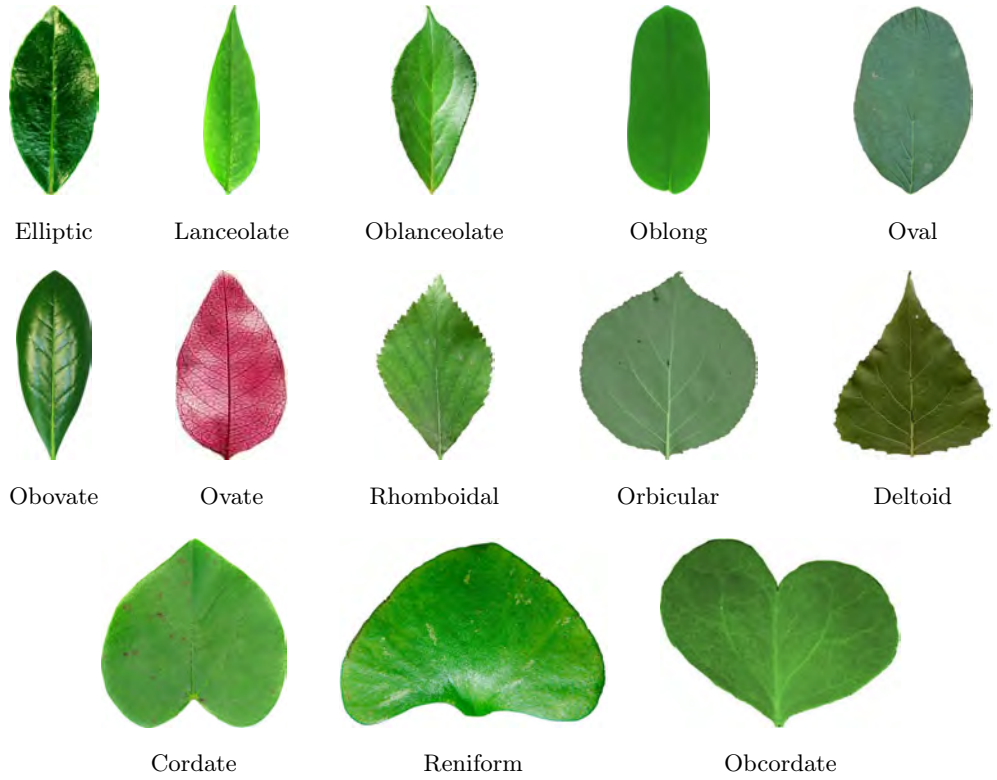


Figure 2.7: Leaf shapes commonly discussed in botanical literature.

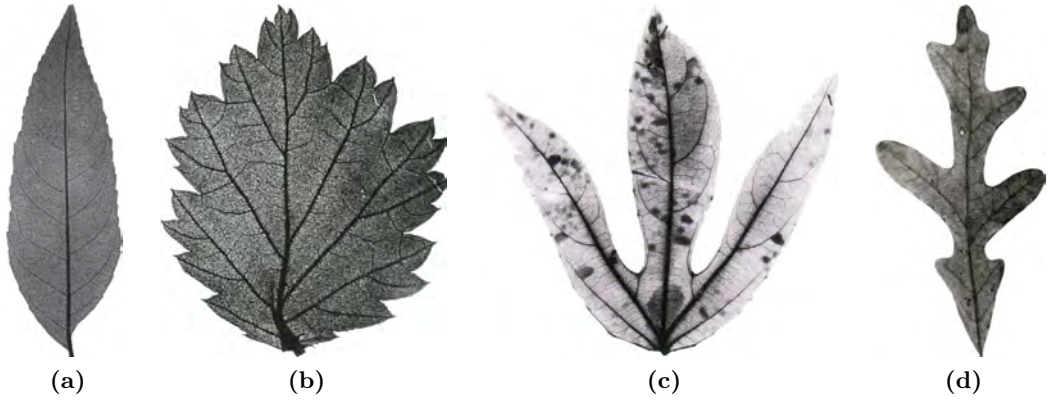
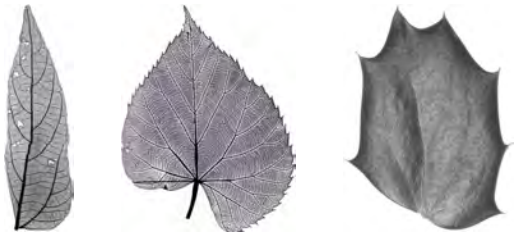


Figure 2.8: The types of leaves based on marginal projections. (a) Entire margin has no projection, (b) toothed margin has small projections. (c) Palmately lobed leaf has large projections originating from the base, and (d) pinnately lobed leaf has large projections originating along the main vein (Source: [EDH+09]).



(a) (b) (c)

Figure 2.9: The types of laminar asymmetries. (a) A simple leaf with asymmetric maximum width. (b) A simple leaf with asymmetric cordate base. (c) A simple leaf with cordate base on one side and no extension on the other (Source: 1, 2: [EDH+09]).

2.3 Venation Patterns in Broad Leaves

The veins of a broad leaf are categorized according to their thickness and course into primary veins, secondary veins, and higher-order veins (Figure 2.3). The arrangement of veins in the lamina is called venation pattern. There are many venation patterns and there is no fixed rule as to which type of leaves can have which venation pattern. This section illustrates common veins and venation patterns [EDH+09].

2.3.1 Primary Veins

The main or primary or first-order vein is the thickest vein and it goes from the base to the apex of a leaf. In some leaves, there is more than one thick vein. If the thickness of these veins is at least 75% of the thickest vein, they are considered as primary veins. In some leaves there is more than one vein originating from the base and their course is similar to that of the thickest vein. These veins are considered as primary veins if their thickness is 25–75% of the thickest vein. If a leaf has more than one primary vein, they are collectively called primaries.

A leaf with only a single primary vein is said to have pinnate venation pattern (Figure 2.10a), whereas a leaf with three or more primaries is said have palmate venation pattern (Figures 2.10b to 2.10g). Palmate venation pattern is further divided into categories based on the number of primaries and the thickness and course of primaries. There can be either a small number (3–10) of thick primaries or a large number (> 10) of thin primaries. These primaries can either diverge from the base or converge towards the apex (Table 2.1).

Table 2.1: Palmate venation patterns of the primary veins. Palmate venation patterns are defined based on the number of primaries and the thickness and course of primaries. D: Primaries diverge from the base. C: Primaries converge towards the apex. Thk: Primaries are thick. Thn: Primaries are thin. B: Primaries branch into other veins.

Venation Pattern	D	C	Thk	Thn	B	Remarks
Actinodromous	✓		✓			3 or more primaries (Figure 2.10b)
Palinactinodromous	✓		✓		✓	3 or more primaries (Figure 2.10c)
Acrodromous		✓	✓			3 or more primaries (Figure 2.10d)
Flabellate	✓			✓	✓	Many primaries (Figure 2.10e)
Parallelodromous		✓		✓		Many primaries (Figure 2.10f)
Campylodromous		✓		✓		Many primaries, strongly curved (Figure 2.10g)

2.3.2 Secondary Veins

The secondary or second-order veins are thinner than the primary veins. These veins vary substantially in both thickness and course. Secondary veins can be further categorized into the following types:

1. **Major Secondaries**
   These are the rib-forming veins that originate from the primary vein and run towards the margin. Venation patterns are defined based on whether the major secondaries reach the margin, or branch into other veins before reaching the margin, or form loops with other veins (Table 2.2).

2. **Minor Secondaries**
   The minor secondary veins branch from lateral primaries or major secondaries and run towards the margin. If the minor secondaries terminate at the margin, it is called craspedodromous pattern (Figure 2.12a). If they branch near the margin and one of the branches terminate at the margin and the others join adjacent minor secondaries, it is called semicraspedodromous pattern (Figure 2.12b). If they join together to form loops, it is called simple brochidodromous pattern (Figure 2.12c).

3. **Inter-secondaries**
   The inter-secondary veins have a course similar to major secondaries but they do not reach the margin. Their thickness is between those of major secondaries and higher-order veins (Figure 2.13a).

Table 2.2: Venation patterns of the major secondaries. M: Major secondaries reach the margin. B: Major secondaries branch into other veins. L: Major secondaries form loops with other veins.

Venation Pattern	M	B	L	Remarks
Craspedodromous	✓			Figure 2.11a
Semicraspedodromous	✓	✓	✓	Single loop (Figure 2.11b)
Festooned semicraspedodromous	✓	✓	✓	Several loops (Figure 2.11c)
Eucamptodromous			✓	Several loops via tertiary veins (Figure 2.11d)
Reticulodromous		✓	✓	Form network of higher-order loops (Figure 2.11e)
Cladodromous		✓		Branch freely (Figure 2.11f)
Brochidodromous			✓	Several loops (Figure 2.11g)

4. **Interior Secondaries**
   The interior secondary veins join two primaries or a primary vein with a marginal or intramarginal vein (Figure 2.13b).

5. **Intramarginal Secondaries**
   The intramarginal secondary veins run parallel to the margin (Figure 2.13c).

6. **Marginal Secondaries**
   The marginal secondary veins run along the margin (Figure 2.13d).

2.3.3 Higher-order Veins

Higher-order veins are thinner than secondary veins. They link various primaries and secondaries forming the "fabric" of the lamina. When they join adjacent secondaries, it is called percurrent pattern (Figure 2.14a). When they form a network of veins, it is called reticulate pattern (Figure 2.14b). When they form tree-like structures, it is called ramified pattern (Figure 2.14c).

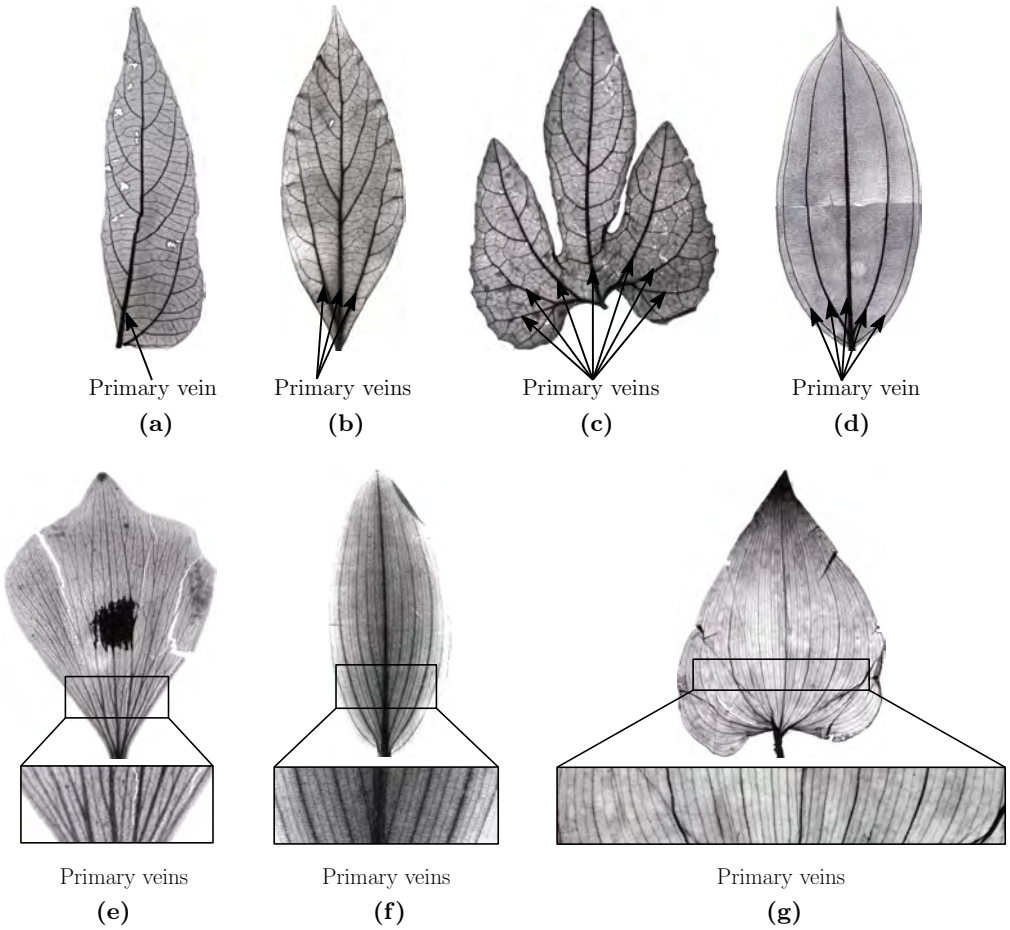


Figure 2.10: Primary vein venation patterns. (a) Pinnate pattern has a single primary. (b) In actinodromous pattern, three or more primaries diverge radially. (c) In palinactinodromous pattern, three or more primaries diverge in a series of branches. (d) In acrodromous pattern, three or more primaries run in convergent arches towards the apex. (e) In flabellate pattern, many thin primaries diverge radially and branch towards the apex. (f) In parallelodromous pattern, many thin primaries converge towards the apex. (g) In campylodromous pattern, many thin primaries run in strongly recurved arches that converge towards the apex (Source: [EDH+09]).

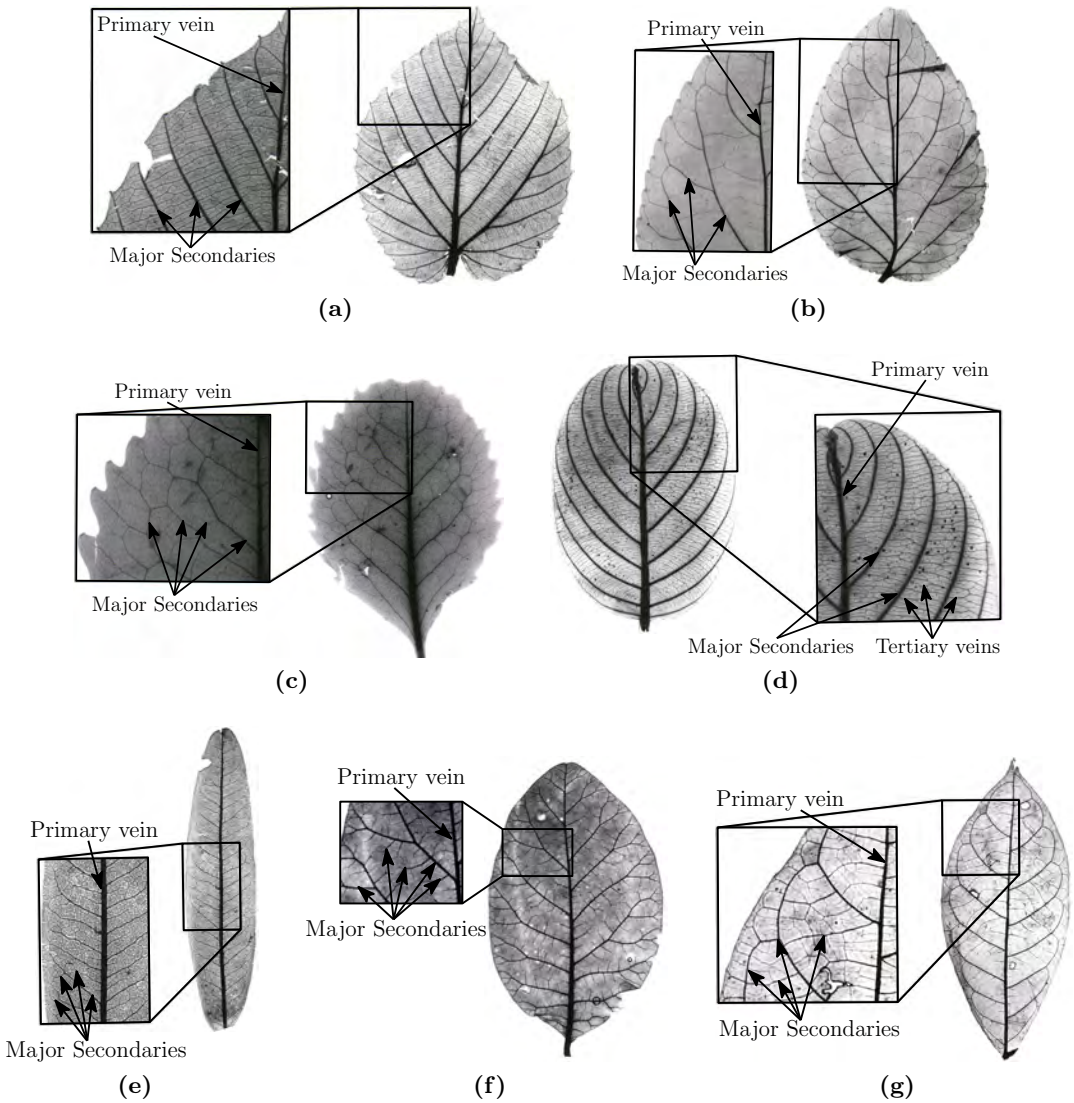


Figure 2.11: Major secondary venation patterns. (a) In craspedodromous pattern, major secondaries reach the margin. (b) In semicraspedodromous pattern, major secondaries branch near the margin, one of the branches reaches the margin and the others form loops with adjacent major secondaries. (c) Festooned semicraspedodromous pattern is similar to semicraspedodromous except that adjacent major secondaries form several loops. (d) In eucamptodromous pattern, major secondaries form loops via tertiary veins. (e) In reticulodromous pattern, major secondaries form a network of higher-order veins. (f) In cladodromous pattern, major secondaries branch freely forming tree-like structures. (g) In simple brochidodromous pattern, adjacent major secondaries form several loops (Source: [EDH+09]).

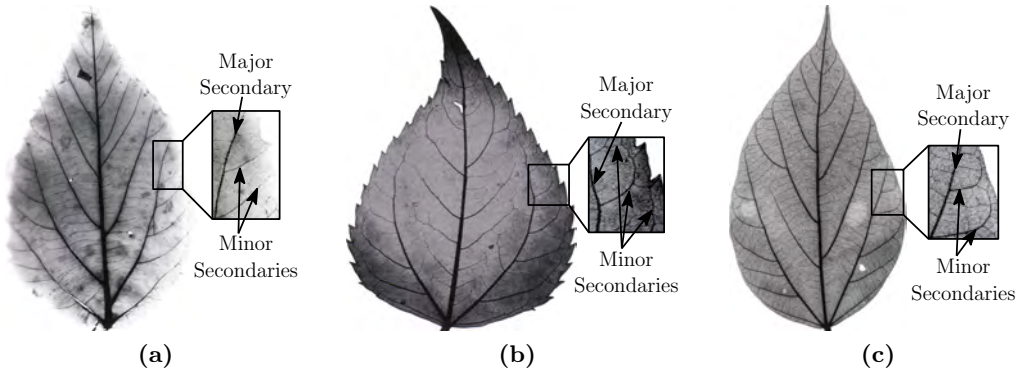


Figure 2.12: Minor secondary venation patterns. (a) In craspedodromous pattern, minor secondaries reach the margin. (b) In semicraspedodromous pattern, minor secondaries branch near the margin. (c) In simple brochidodromous pattern, minor secondaries form loops (Source: [EDH+09]).

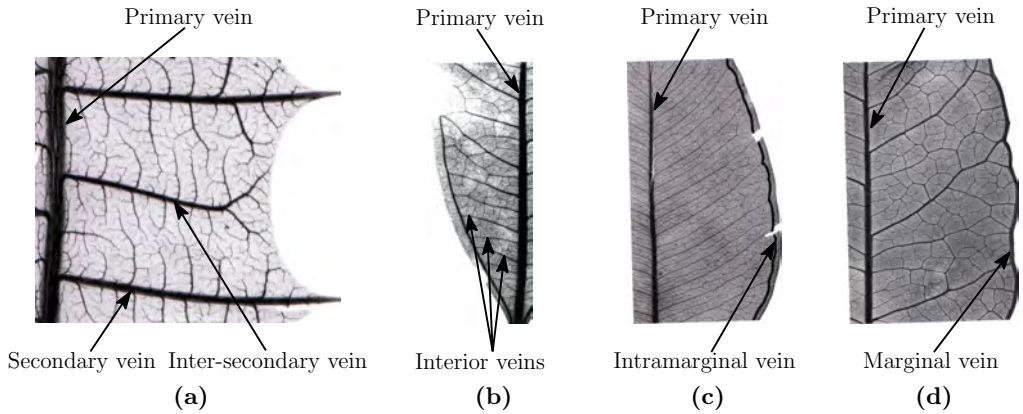


Figure 2.13: Miscellaneous secondary venation patterns. (a) Inter-secondary veins run parallel to major secondaries and do no reach the margin. (b) Interior veins join two primaries. (c) Intramarginal veins run parallel to the margin. (d) Marginal veins run along the margin (Source: [EDH+09]).

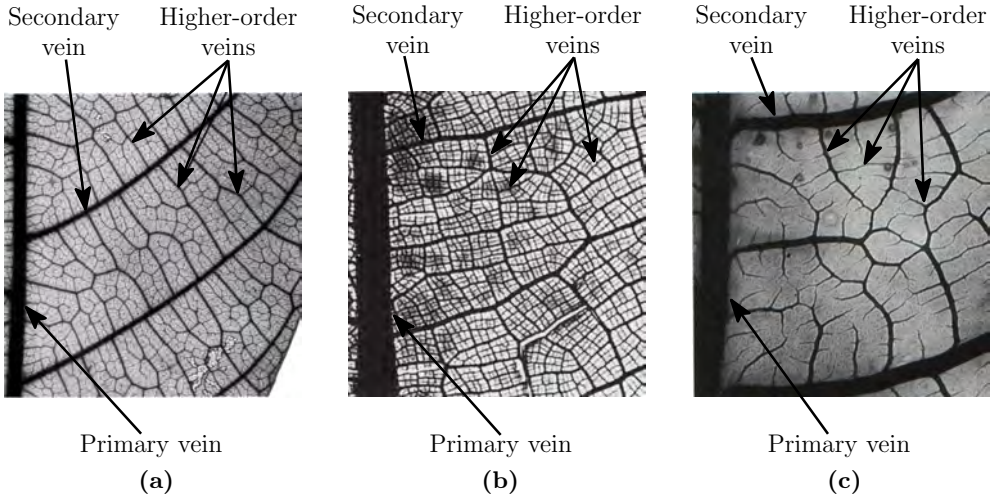


Figure 2.14: Higher-order venation patterns. (a) In percurrent pattern, higher-order veins join two secondaries. (b) In reticulated pattern, higher-order veins form networks. (c) In ramified pattern, higher-order veins form tree-like structures (Source: [EDH+09]).

Chapter 3

Literature Review

Over the last decades many methods have been developed for modeling plants and trees: interactive [LD99, BPF+03, OOI05, ASSJ06, SLCS06, WZW+06, GK08, WZW09], rule-based [PMKL01, IOI06, VK06, Han07, APS09], image-based [NFD07, TZW+07, ZTZ+08], 3D data-based [PH02, GP04a, GP04b, PGW04, XGC05, XGC07, ZZHJ08, YWM+09], biology-based [SFS05], machine learning [CNX+08], and ad hoc [RLP07]. Several softwares [Luf, wdi, KHT+, Sch, KL, Per, Vis, Bon, Cre, XFr] are also available for modeling plants and trees [RACJ09]. 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.

This chapter discusses the current state of the art in leaf modeling. Based on the technique used, existing methods are categorized into image-based methods (Section 3.1) and rule-based (Section 3.2) methods.

3.1 Image-based Methods

Image-based methods [QTZ+06, MZL+08] attempt to reconstruct the 3D geometry of leaves from a set of images. These methods use computer vision techniques to recover 3D information from the images. The 3D information is then used to segment the leaves as well as estimate their position and orientation. Then, a generic leaf model of the given plant is placed at the estimated positions. The generic leaf model is built manually using the image of a leaf from the input images. Finally, instances of the generic leaf model are deformed to match the leaves in the images. Various methods differ in the technique used for each step.

Quan et al. [QTZ+06] used structure from motion to estimate a sparse set of 3D points from the input images. The 3D points and the images are used in an interactive graph-based leaf segmentation algorithm. The segmentation is done on both input images and estimated 3D points. The user interaction is minimal and the user have to only click to confirm

segmentation, draw to split and refine segments, and click to merge two adjacent segments. At the end, the generic flat leaves are scaled and warped using 3D points and leaf boundaries extracted from the image.

Since leaves heavily occlude each other in images, Ma et al. [MZL+08] proposed an approach to model leaves by detecting apexes of leaves from the volumetric data. The volumetric data is estimated from the images by voxel coloring with zero-mean normalized cross-correlation photo consistency constraints. The idea is to first automatically segment only the apexes, and then use the orientations of the apexes to automatically segment the leaves. In order to segment the apexes reliably, the method assumes that leaves are large enough in input images. The apexes are detected in the volumetric data using a sharp feature detection algorithm.

In addition to model leaves, image-based methods also reconstruct the branches and build the geometric model of the entire plant. These approaches work very well and are able to produce realistic reconstruction of a variety of plants. However, since a single instance of an existing plant is reconstructed, another method is required for generating plant instances which have same overall shape but differ slightly in detail. In addition, these methods do not have a unifying model for representing different kind of leaves. Thus, data from real plants must be captured and processed for generating instances of many different leaves, which is too tedious and time-consuming.

3.2 Rule-based Methods

Rule-based methods describe the shape of a leaf by a set of rules. These rules are then interpreted by an algorithm to generate the geometric shape of the leaf. The most popular rule-based method is the L-systems. It was originally introduced by Lindenmayer [Lin68] to formalize the development of multicellular organisms and subsequently expanded by Prusinkiewicz and Lindenmayer to model branching structure and plants [PL90]. L-systems consists of an axiom or initial state and a set of production rules. They starts with the axiom and recursively expands the axiom using the production rules. Thus, L-systems are particularly suitable for modeling self-similar objects such as branching structure of a plant or tree.

Since the geometric shapes of leaves do not exhibit self-similarity, they cannot be directly modeled by the L-systems. However, several methods have been proposed to extend L-systems to the modeling of geometric shapes of leaves. Rodkaew et al. [RLFS02] used genetic algorithm with a parametric L-system to reconstruct the shape of a leaf from reference image.

The genetic algorithm is used to estimate the parameters of the L-system by minimizing the Euclidean distance between the silhouette of a leaf in the reference image and the silhouette generated by the L-system.

Peyrat et al. [PTMG08] proposed a method that combines 2D generalized map (2Gmap) with L-systems for modeling leaves. 2Gmap is the topological model that represents the topology of any 2D subdivision. The idea is to define operations using a 2Gmap for growing, glueing, and splitting 2D faces. These operations are used in the production rules of an L-system for generating the venation pattern of the leaf. The leaf shape is then generated by iteratively adding faces to the veins using 2Gmap topology. The production rules of the L-system are also extended for generating texture and modeling aging of leaves.

Hammel et al. [HPW92] proposed a method for modeling lobed leaves using L-systems and implicit contours. L-systems is used to generate the venation pattern of a lobed leaf and implicit contours are used to generate the margin of the leaf. For each vein in the venation pattern, an implicit function is defined by the length of the vein, radius of influence at each end of the vein, and a method to interpolate influence between two end-points. The margin of the leaf is defined as a level set of the summation of implicit functions of all the veins.

Rule-based methods are quite powerful and they can be used to generate realistic looking plants and trees. However, they are difficult to use for non-experts because there is no standard method for writing the production rules for a given leaf. Moreover, the rules tend to be complex, requiring constants, variables, and conditional statements even for simple leaf shapes. Figures 3.1 and 3.2 illustrate instance of a unilobed and multilobed leaf generated by the 2Gmap L-system [PTMG08]. Tables 3.1 and 3.2 illustrate the set of production rules used to generate these shapes, respectively. It is not immediately obvious what kind of leaf shapes will be produced by the production rules and what are the effects on the leaf shapes if the rules are modified. It is also difficult to provide theoretical guarantee on the numerical stability of the system. A small change in the production rules might produce large changes in the leaf shape.

3.3 Summary

Image-based methods are very easy to use as they only require a set of images which can be captured using a hand-held camera. Thus, with some user interaction they be used to quickly create an instance of a real plant. Their main drawback is that generating many different kind of leaves is too tedious and time-consuming as data for many plants must be captured and processed. Rules-based methods are very powerful for generating the self-similar objects

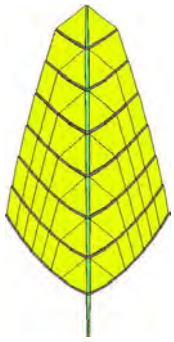


Figure 3.1: Instance of a unilobed leaf generated from 2Gmap L-system [PTMG08].

Table 3.1: The production rules of the 2Gmap L-system used to generate Figure 3.1

```
#define A(4,1,0,0,-90,1,0.1)
#define B(4,2,0,0,0,2,0.1)
#define D(4,1,0,0,0,0.2,0.1)
#define E(4,1,0,0,0,1,0.1)
#define F(4,1,0,0,0,1,0.1)

@variables@
#define anglesecondaire = 55
#define anglecourbure = 3
#define anglesecondairecurling = 2
#define curling = 2

#axiome : A
p00 A -> A[B(,,,,,,)]_{E} // Face B is created and glued on A_E
p00 B -> B[D(,,,<curling>,,,)]_{E}
p00 D -> D[B(,,,<curling>,,,)]_{E}
p01 D -> D[E(,,,,<anglesecondaire>,,)]_{C1}
p01 D -> D[F(,,,,<-anglesecondaire>,,)]_{C2}
p02 E -> E[E(,,,<anglesecondairecurling>,<-anglecourbure>,,)]_{E}
p02 F -> F[F(,,,<anglesecondairecurling>,<anglecourbure>,,)]_{E}
```

such as branching structure of plants. However, since the rules in the existing methods tend to be complex, it is difficult for non-experts to use these methods. In addition, it is not clear how to write the rules for a given leaf.

In contrast, the leaf model proposed in this thesis is simple and intuitive. The user can easily specify the desired shape of a leaf using a simple GUI. Moreover, as will be analyzed in Section 5.4, the algorithm that generates the leaf instances is numerically stable. Table 3.3 compares the existing methods with respect to the desirable properties of the leaf model listed in Section 1.2.

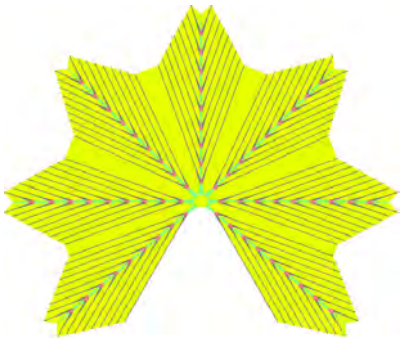


Figure 3.2: Instance of a multilobed leaf generated from 2Gmap L-system [PTMG08].

Table 3.2: The production rules of the 2Gmap L-system used to generate Figure 3.2

```
#define A(8,1,0,0,-90,1,1)
#define B(4,1.5,0,0,0,0.2,0.1)
#define C(4,1,0,0,0,2,0.1)
#define D(4,1.5,0,0,0,0.1,0.1)
#define E(4,2,0,0,0,1,0.1)
#define F(4,1.5,0,0,0,0.1,0.1)
#define G(4,2,0,0,0,1,0.1)
#define H(4,1,0,0,0,0.1,0.1)
#define I(4,1,0,0,0,1,0.1)
#define J(4,1,0,0,0,1,0.1)

@variables@
#define angle = 1/etapeFinale
#define ecartementnervure = 20

#axiome : A
p00 A -> A[B(,,,,130,,)]_{C1}
p00 A -> A[B(,,,,90,,)]_{C2}
p00 A -> A[B(,,,,45,,)]_{C3}
p00 A -> A[B(,,,,,,)]_{E}
p00 A -> A[B(,,,,-45,,)]_{C4}
p00 A -> A[B(,,,,-90,,)]_{C5}
p00 A -> A[B(,,,,-130,,)]_{C6}
p01 B -> B[D(,,,<angle>,,,)]_{E}
p01 D -> D[B(,,,<angle>,,,)]_{E}
p01 B {} {etape  == etapeFinale } {} -> B[B(,<etapeFinale/5.0>,,<angle>,,,)]_{E}
p01 C -> C[C(,,,<angle>,,,)]_{E}
p02 D -> D[E(,,,,<ecartementnervure>,,)]_{C1}
p02 D -> D[E(,,,,<-ecartementnervure>,,)]_{C2}
p03 E -> E[E(,,,<etapeFinale/10>,,,)]_{E}
```

Table 3.3: Comparison of leaf generation methods.

Property	Image-based	Rule-based	Proposed model
General	✓	✓	✓
Intuitive	✓	✗	✓
Concise	—	✗	✓
Generative	✗	✓	✓
Numerically stable	—	?	✓

Chapter 4

Overview of Computational Leaf Modeling

The goal of this research is to develop a method for generating the geometric shape of a variety of leaves. To accomplish this goal, it is important to first enumerate and characterize the laminar shapes that are modeled. For ease of computational modeling and application, this thesis categorizes leaves into two broad categories: unilobed and multilobed. The proposed leaf model is based on unilobed leaves, which are characterized in Section 4.1. Multilobed leaves are modeled as a combination of unilobed leaves, and they are characterized in Section 4.2. Given the model of a leaf shape, multiple instances of the leaf can be generated, each having the same overall shape but differ slightly in detail. The algorithms for generating leaf instances and new laminar shapes are presented in Chapters 5, 6 and 7.

4.1 Types of Unilobed Leaves Modeled

As discussed in Section 2.2, the shape of unilobed leaves can be characterized by the shapes at the base and the apex, and the location and the extent of the widest part of the lamina. In this thesis, the widest part of the lamina is called the waist. Based on the leaf shapes discussed in Section 2.2, the base shapes are categorized into six types: straight, concave, convex, concavo-convex, complex, and cordate (Figure 2.4), the apex shapes are categorized into four types: straight, convex, acuminate (concave or concavo-convex), and emarginate (Figure 2.5), and the waist shapes are categorized into five types: elliptic, obovate, ovate, oblong, and linear (Figure 2.6). Table 4.1 summarizes the shape characteristics of unilobed leaves.

Considering all possible combinations of these shapes, there are $5 \times 6 \times 4 = 120$ possible types of leaf shapes. However, not all combinations occur in nature. Oblong leaves (Figure 2.6d) have convex base and convex apex. So, there is only one shape for oblong leaves. Leaves with linear shape (Figure 2.6e) are very thin compared to the length of the lamina and the shape of the base and the apex are similar for all leaves with linear shape.

Table 4.1: Characteristics of unilobed leaves. The shape of unilobed leaves can be characterized by the location and the extent of the widest part of the lamina (waist shape) and the shapes of the margin at the base (base shape) and the apex (apex shape).

Waist Shape	Base Shape	Apex Shape
Elliptic	Straight	Straight
Obovate	Concave	
Ovate	Convex	Convex
Oblong	Concavo-Convex	Acuminate
Linear	Complex	
	Cordate	Emarginate

(a) **(b)** **(c)**

Figure 4.1: Laminar shapes omitted by the leaf model. (a) A leaf with teeth. (b) Leaves with drip-tips. (c) A leaf with complex base (Sources: 1,4: [HGL92]).

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

- Teeth along the leaf margin (Figure 4.1a) 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 [ZG04].
- Some leaves have long slender tips called *drip-tips* (Figure 4.1b). The apex shapes of these leaves are initially concave and then convex. Drip-tips are not modeled as they do not contribute significantly to the overall shape of the leaf.
- Leaves with complex base shape (Figure 4.1c) are omitted. The number of leaf types with complex base shape is very small. So, they can be omitted.

Table 4.2: Laminar shapes modeled by the leaf model. C: Convave, S: Straight, V: Convex. X: with extension. #: The number of combination of the base and the apex shapes.

Waist Shape	Base Shape	Apex Shape	#	Figure
Elliptic	C, S, V, X	C, S, V, X	16	4.2
Obovate	C, S, V, X	C, S, V, X	16	4.3
Ovate	C, S, V, X	C, S, V, X	16	4.4
Oblong	V	V	1	4.5a
Linear	S	S	1	4.5b
Total			50	

In summary, 50 types of unilobed leaf shapes are modeled by the proposed model (Table 4.2). Of these 50 shapes, 26 occur naturally. The remaining 24 shapes may occur in nature but the author is unable to find real leaf examples of them. Shapes with naturally occurring leaves are illustrated with real leaf examples in Figures 4.2 to 4.5. For other leaves computer generated shapes are inserted into the figures for the purpose of illustration. 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.

As discussed in Section 2.2, the leaf shape can be asymmetric on the two sides of the primary vein (Figure 2.9). Asymmetric leaves can be modeled either with different shapes for the left and the right side, or with the same shape but with different parameter values.

Some of the common leaf shapes have been given specific names by botanists (Figure 2.7). These leaf shapes are included in Figures 4.2 to 4.5. Table 4.3 characterizes leaf shapes in Figure 2.7 according to the shapes of the base, the waist and the apex.

4.2 Types of Multilobed Leaves Modeled

There are three basic types of multilobed leaves: palmately lobed, pinnately lobed, and bilobed. In a palmately lobed leaf (Figures 4.6a to 4.6c), the lobes originate at the base of

Table 4.3: Shape characteristics of common leaf shapes illustrated in Figure 2.7.

Leaf Name	Waist Shape	Base Shape	Apex Shape	Figure
Elliptic	Elliptic	Convex	Convex	4.2 (c3)
Lanceolate	Ovate	Convex	Straight	4.4 (c2)
Oblanceolate	Obovate	Straight	Convex	4.3 (b3)
Oblong	Oblong	Convex	Convex	4.5a (c3)
Oval	Elliptic	Convex	Convex	4.2 (c3)
Obovate	Obovate	Straight	Convex	4.3 (b3)
Ovate	Ovate	Convex	Straight	4.4 (c2)
Orbicular	Elliptic	Convex	Convex	4.2 (c3)
Rhomboidal	Elliptic	Straight	Straight	4.2 (b2)
Cordate	Ovate	Cordate	Straight	4.4 (d2)
Deltoid	Ovate	Concave	Straight	4.4 (a2)
Reniform	Ovate	Cordate	Straight	4.4 (d2)

the leaf. These leaves have an odd (typically three, five, or seven) number of lobes. In a pinnately lobed leaf (Figure 4.6d), 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 4.6e) 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.

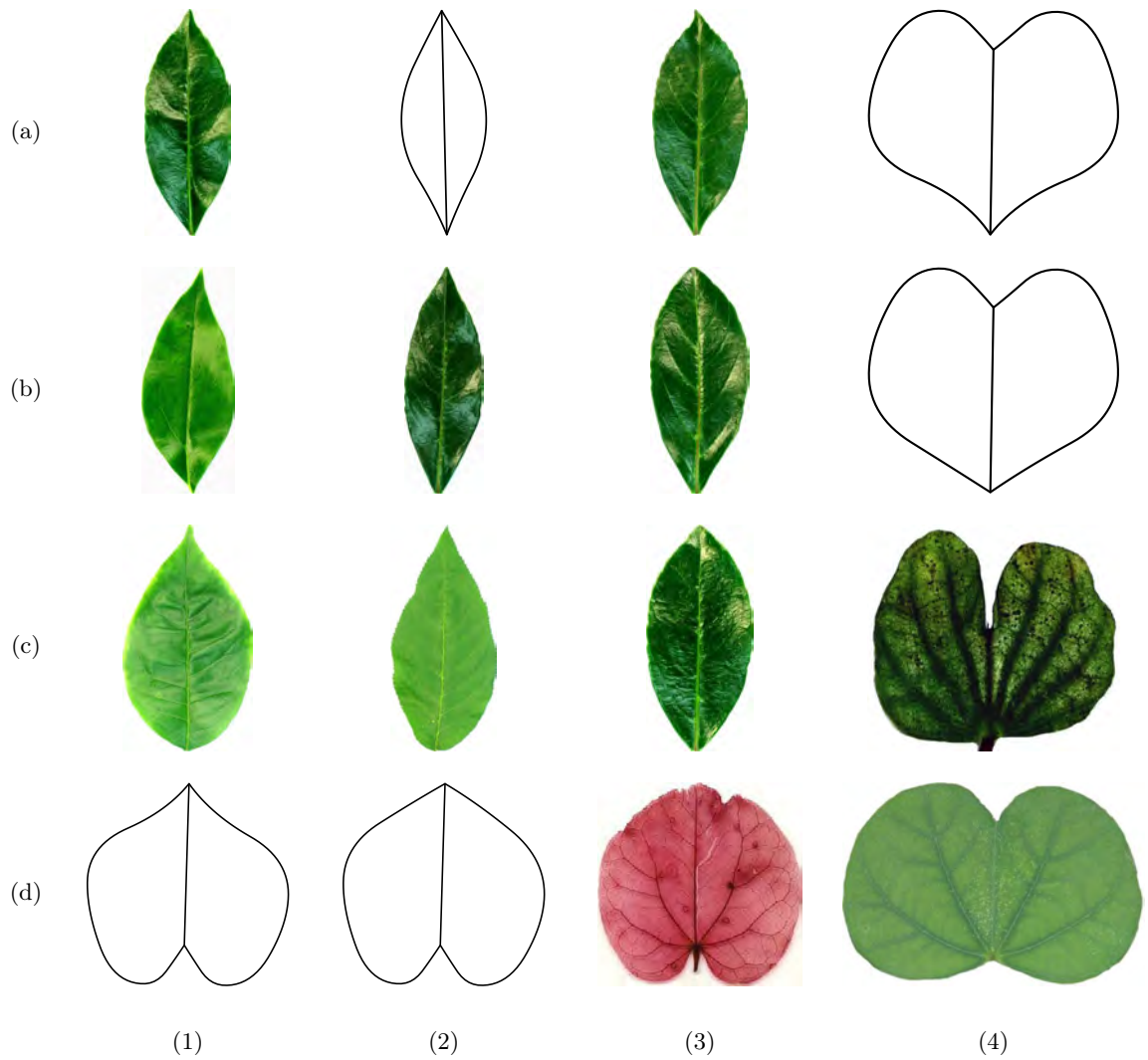


Figure 4.2: Examples of various shapes of leaves with elliptic waist. Rows correspond to leaves with same base shapes. (a) Concave. (b) Straight. (c) Convex. (d) Base with extension. Columns correspond to leaves with same apex shapes. (1) Concave. (2) Straight. (3) Convex. (4) Apex with extension.

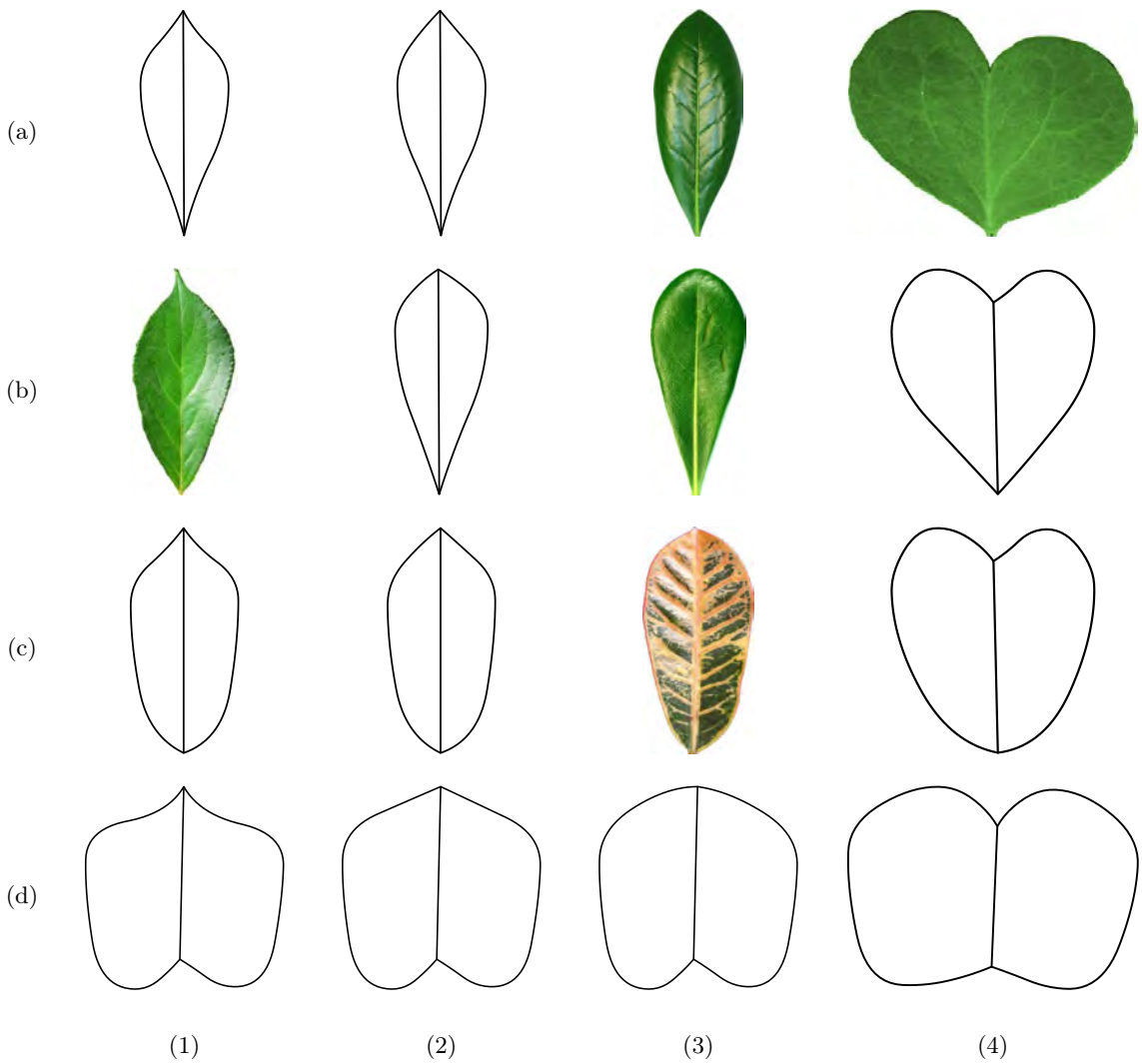


Figure 4.3: Examples of various shapes of leaves with Obovate waist. Rows correspond to leaves with same base shapes. (a) Concave. (b) Straight. (c) Convex. (d) Base with extension. Columns correspond to leaves with same apex shapes. (1) Concave. (2) Straight. (3) Convex. (4) Apex with extension.

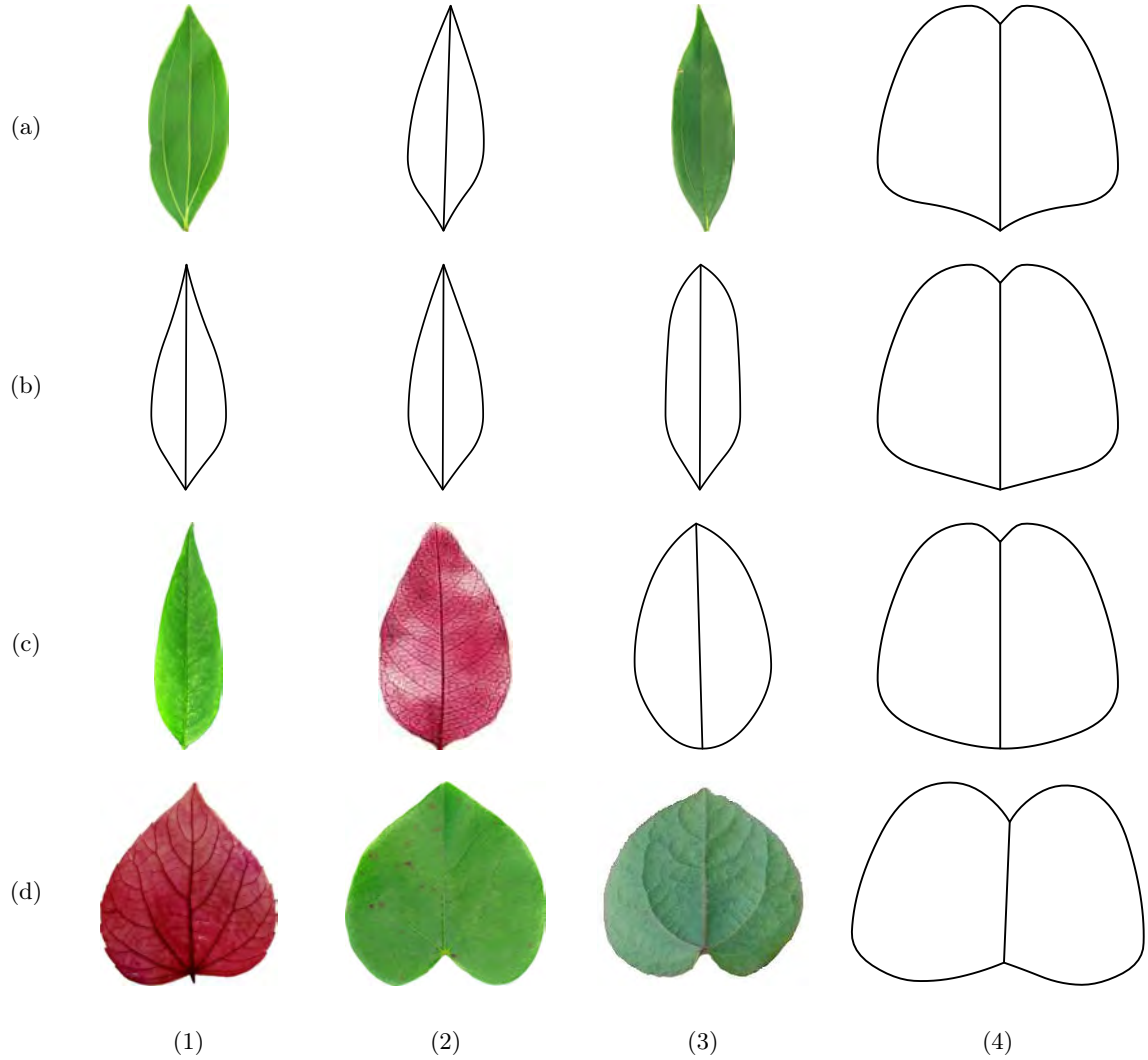


Figure 4.4: Examples of various shapes of leaves with Ovate waist. Rows correspond to leaves with same base shapes. (a) Concave. (b) Straight. (c) Convex. (d) Base with extension. Columns correspond to leaves with same apex shapes. (1) Concave. (2) Straight. (3) Convex. (4) Apex with extension.

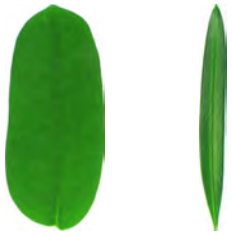


Figure 4.5: Example of leaves with linear and oblong waist. (a) A leaf with oblong waist. (b) A leaf with linear waist.

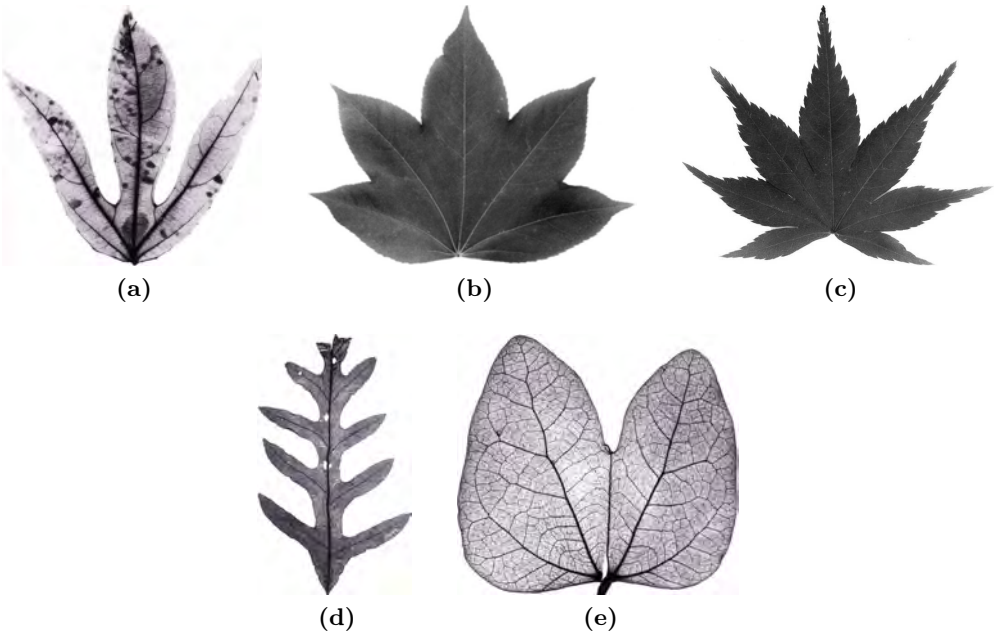


Figure 4.6: Examples of various shapes of multilobed leaves. (a–c) Palmately lobed leaves with three, five, and seven lobes. (d) A pinnately lobed leaf with ten lobes. (e) A bilobed leaf.

Chapter 5

Modeling of Unilobed Leaves

This chapter presents a parametric leaf model of the geometric shapes of unilobed leaves discussed in Section 4.1. The shape of the leaf is represented by a set of landmark points on the margin and tangents to the margin at these points. These parameters can be specified intuitively using a GUI and a reference image (Section 5.2). The parameters of the leaf model are used by the laminar shape generation algorithm to generate the laminar surface (Section 5.3). Performance of the algorithm is discussed in Sections 5.4 and 5.5.

5.1 Parametric Leaf Model

The parametric leaf model is defined on a local right-handed 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 laminar plane of the leaf. The primary vein is defined to 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 without basal extension, one side of its margin is defined by three landmark points $\mathbf{p}_i, i = 1, 2, 3$, and the corresponding unit tangents $\mathbf{t}_i$ at these points (Figure 5.2a). Landmark point $\mathbf{p}_1$ is the base, which is fixed at $(0, 0)$. $\mathbf{p}_2$ is the waist, the point at which the laminar width is maximum. $\mathbf{p}_3$ is the apex, which is fixed at $(0, 1)$. Tangent $\mathbf{t}_1$ can vary clockwise from $(0, 1)$ to $(1, 0)$ and $\mathbf{t}_3$ can vary counter-clockwise from $(0, 1)$ to $(-1, 0)$. By definition of the waist, $\mathbf{t}_2$ is parallel to the y-axis. Thus, there are only 4 free parameters for defining one side of the leaf margin, namely θ_a, θ_b, and $\mathbf{p}_2 = (x_2, y_2)$, where $0 \leqslant \theta_a, \theta_b \leqslant \frac{\pi}{2}$, $x_2 > 0$, and $0 < y_2 < 1$. The landmark points and tangents are related

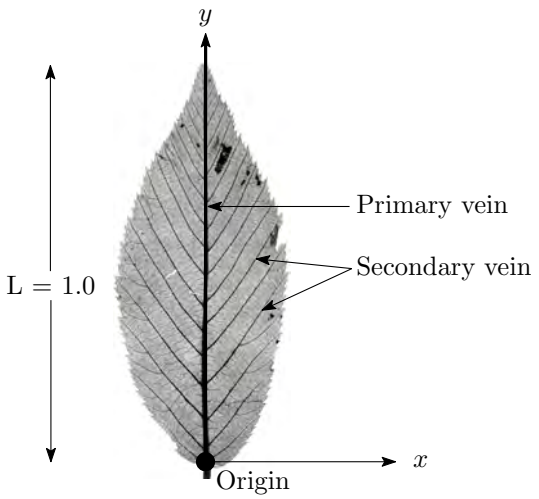


Figure 5.1: Coordinate system of the leaf model. The parameters of the leaf model are expressed in a right-handed coordinate system with origin at the base and the y-axis pointing towards the apex of the leaf. The primary vein is defined to have a unit length.

to these free parameters as follows:

$$\begin{aligned}
\mathbf{p}_1 &= (0,0),\\
\mathbf{p}_2 &= (x_2, y_2),\\
\mathbf{p}_3 &= (0,1),\\
\mathbf{t}_1 &= (\sin\theta_b, \cos\theta_b),\\
\mathbf{t}_2 &= (0,1),\\
\mathbf{t}_3 &= (-\sin\theta_a, \cos\theta_a).
\end{aligned}$$

For a unilobed leaf with basal extension, one side of its margin is defined by four landmark points $\mathbf{p}_i$, $i = 1, \ldots, 4$, and the corresponding unit tangents $\mathbf{t}_i$ at these points (Figure 5.2b). Landmark point $\mathbf{p}_1$ is the base, which is fixed at $(0,0)$. $\mathbf{p}_2$ is the tail, at which the basal extension is maximum. $\mathbf{p}_3$ is the waist, at which the laminar width is maximum. $\mathbf{p}_4$ is the apex, which is fixed at $(0,1)$. The tangent $\mathbf{t}_1$ can vary clockwise from $(1,0)$ to $(0,-1)$ and $\mathbf{t}_4$ can vary counter-clockwise from $(0,1)$ to $(-1,0)$. By definition of the tail and waist, respectively, $\mathbf{t}_2$ is parallel to the x-axis and $\mathbf{t}_3$ is parallel to the y-axis. Thus, there are only 6 free parameters for defining one side of the margin: θ_a, θ_b, $\mathbf{p_2} = (x_2, y_2)$, and $\mathbf{p_3} = (x_3, y_3)$, where $0 \leqslant \theta_a \leqslant \frac{\pi}{2}$, $\frac{\pi}{2} \leqslant \theta_b \leqslant \pi$, $x_2 > 0$, $y_2 < 0$, $x_3 > 0$, and $0 < y_3 < 1$. The landmark

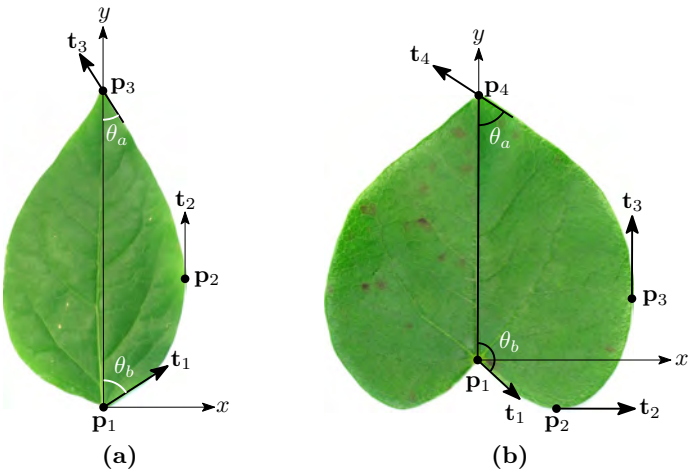


Figure 5.2: Parameters of the leaf model for unilobed leaves. (a) Leaf without basal extension. (b) Leaf with basal extension.

points and tangents are related to these free parameters as follows:

$$
\begin{aligned}
\mathbf{p}_1 &= (0, 0), \\
\mathbf{p}_2 &= (x_2, y_2), \\
\mathbf{p}_3 &= (x_3, y_3), \\
\mathbf{p}_4 &= (0, 1), \\
\mathbf{t}_1 &= (\sin\theta_b, \cos\theta_b), \\
\mathbf{t}_2 &= (1, 0), \\
\mathbf{t}_2 &= (0, 1), \\
\mathbf{t}_3 &= (-\sin\theta_a, \cos\theta_a).
\end{aligned}
$$

A unilobed leaf with apical extension is defined by analogous landmark points and tangents, with 6 free parameters for one side of its margin.

5.2 User Interface

GUI provides an interactive means for a user to specify the shape of a unilobed leaf intuitively. Figure 5.3 illustrates an example of specifying a unilobed leaf without basal extension. After loading a reference image of a leaf, the user places a point with an arrow at the base of the reference leaf. The point represents the position and the arrow represents the tangent to the margin at the point. The user then rotates the arrow about the point such that

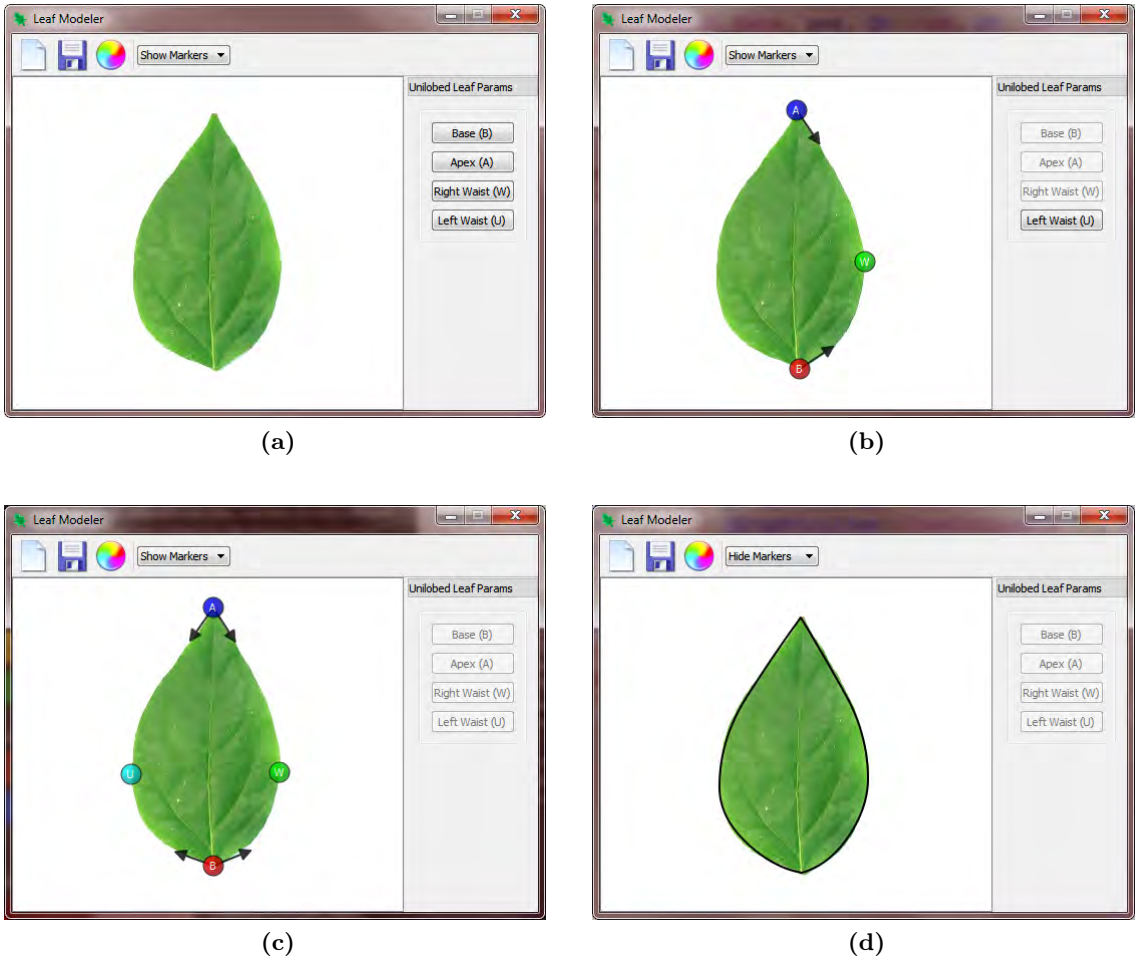
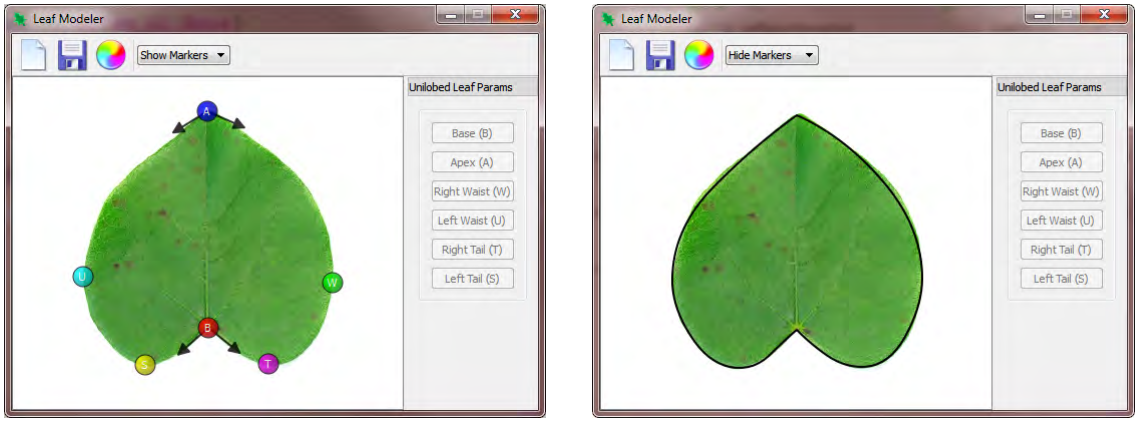


Figure 5.3: User specification of laminar shape of unilobed leaves without basal extension. (a) First, the user loads a reference image of a leaf. Then, the user specifies the landmark points and tangents for the (b) right side and the (c) left side. (d) The margin of the leaf generated by the system.

the arrow is tangent to the margin at the base. Similarly, the user specifies the apex and the tangent to the margin at the apex. For ease of use, the user specifies the tangent $-\mathbf{t}_3$ at the apex. Finally, the user places a point on the waist of the leaf, at which the leaf has the maximum width. The tangent at the waist is by definition parallel to the y-axis. So, the user does not need to specify this tangent. The system then generates the margin for one side of the leaf. The margin for the other side of the leaf is generated similarly.

To define a leaf with basal extension, a similar method is used. In addition, the user places a point on the tail of the leaf, at which the basal extension is maximum (Figure 5.4).



(a) **(b)**

Figure 5.4: User specification of laminar shape of unilobed leaves with basal extension. (a) The user specifies the landmark points and tangents for the right side and the left side. (b) The margin of the leaf generated by the system.

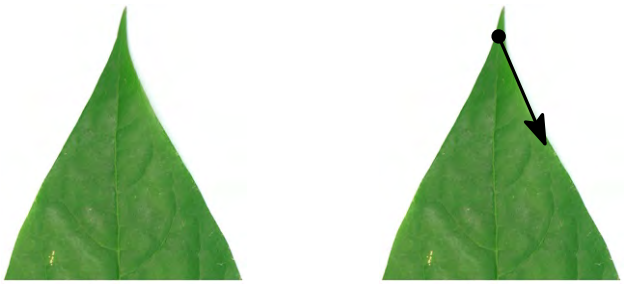


Figure 5.5: Specifying the position of the apex for a leaf with drip tip. For this leaf, the apex should be specified at a point such that tangents to the margin are just beyond the concave part of the margin.

Some unilobed leaves have a drip-tip or a drip-base. As discussed in Section 4.1, these leaves are not modeled directly by the proposed leaf model. For leaves with drip-tips, the apex should be specified at a point such that the tangents to the margin at apex start just beyond the concave part of the drip-tip (Figure 5.5). The base for leaves with drip-base should be specified similarly.

5.3 Laminar 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 have many nice properties with geometric significance [PT97]. The

degree of B-spline curves is independent of the number of control points. B-spline curves have local control, i.e., modifying a control point only changes the shape of a part of the curve. A part of a degree-d B-spline curve is contained in the convex hull of d consecutive control points. Furthermore, no straight line intersects a B-spline curve more times than it intersects the curve's control polygon (variation diminishing property). These properties are important for intuitive control over the laminar shape. They ensure that the estimated margin is within the user's expectation.

A degree-d B-spline is a piecewise polynomial curve defined as follows [PT97]

$$\mathbf{p}(u) = (x(u), y(u)) = \sum_{i=0}^{n} B_{i,d}(u)\mathbf{q}_i, \tag{5.1}$$

where $\mathbf{p}(u)$ are points on the B-spline curve parametrized by u, which lies in the range $[0, 1]$. The points $\mathbf{q}_i,\ i = 0, \ldots, n$ are the $n+1$ control points. The functions $B_{i,d}(u)$ are the B-spline basis functions. They are defined by a sequence of non-decreasing real numbers called knots $v_0, \ldots, v_m$ and $v_i \leqslant v_{i+1}$. Intuitively, they are the points in the parameter space at which the curve changes from one polynomial to another. The ith B-spline basis function of degree-d, $B_{i,d}(u)$, is defined as

$$B_{i,0}(u) = \begin{cases} 1 & \text{if } v_i \leqslant u < v_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

$$B_{i,d}(u) = \frac{u - v_i}{v_{i+d} - v_i} B_{i,d-1}(u) + \frac{v_{i+d+1} - u}{v_{i+d+1} - v_{i+1}} B_{i+1,d-1}(u). \tag{5.3}$$

The degree d, the number of control points $n+1$, and the number of knots $m+1$ are related by $m = n + d + 1$.

For a given u, the derivative of the B-spline curve can be obtained by computing the derivative of its basis functions:

$$\mathbf{p}'(u) = \sum_{i=0}^{n} B'_{i,d}(u)\mathbf{q}_i. \tag{5.4}$$

The derivative of the ith B-spline basis function of degree-d is given by

$$B'_{i,d} = \frac{d}{v_{i+p} - v_i} B_{i,d-1}(u) - \frac{d}{v_{i+p+1} - v_{i+1}} B_{i+1,d-1}(u). \tag{5.5}$$

For the proposed leaf model, degree-two B-spline curves are chosen. Degree-one B-spline curves are just a set of straight line segments passing through the control points. Degree-three or higher-degree B-spline curves sometimes create unnecessary points of inflection, resulting in overly complex leaf shapes that may have self-intersections (Figure 5.6). Thus, degree-2,

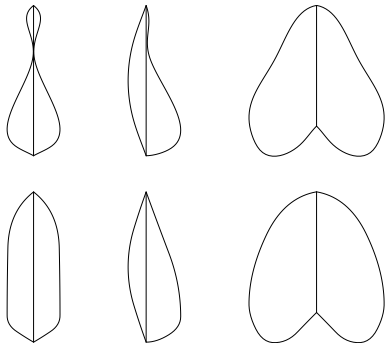


Figure 5.6: Effect of varying the degree of B-spline curves on leaf shapes. The leaf shapes in the first row were generated using degree-3 B-spline curves. The second row shows the corresponding leaf shapes generated using degree-2 B-spline curves.

i.e., quadratic, B-spline curves are the most appropriate.

5.3.1 B-spline Fitting

Many methods have been proposed in the literature for fitting a B-spline curve to a set of points with known first derivatives. In this thesis, the algorithm given in Section 9.2.4 of [PT97] is used. The B-spline fitting algorithm finds the parameters of a quadratic B-spline curve that passes through a set of 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 $\mathbf{p}_i$, a parameter value u_i can be determined such that the B-spline curve passes through it:

$$\mathbf{p}_i = \mathbf{p}(u_i) = \sum_{k=0}^{n} B_{k,2}(u_i)\mathbf{q}_k. \tag{5.6}$$

The first derivatives of the B-spline curves are given by

$$\alpha_i \mathbf{t}_i = \mathbf{p}'(u_i) = \sum_{k=0}^{n} B'_{k,2}(u_i)\mathbf{q}_k, \tag{5.7}$$

where α_i is a scalar used to scale the unit tangent $\mathbf{t}_i$ to match the first derivative of the curve at u_i.

The unknowns in Equations 5.6 and 5.7 are the control points $\mathbf{q}_k$, the knots v_i, the parameter values u_i, and the scalars α_i. The value of v_i, u_i, and α_i are estimated by approximating the B-spline curve by a polyline formed by connecting the points $\mathbf{p}_i$. The position of the control points $\mathbf{q}_k$ are computed by solving Equations 5.6 and 5.7. To obtain a unique set of control points $\mathbf{q}_k$, the number of control points must be equal to the number of equa-

tions. 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 $\mathbf{p}_i$ [PT97]. Let D be the total chord length

$$D = \sum_{i=2}^{N} ||\mathbf{p}_i - \mathbf{p}_{i-1}||. \tag{5.8}$$

Then the parameters are defined as

$$\begin{aligned} u_1 &= 0 \\ u_i &= u_{i-1} + \frac{||\mathbf{p}_i - \mathbf{p}_{i-1}||}{D}, \quad i = 2, \ldots, N. \end{aligned} \tag{5.9}$$

To ensure that the B-spline curve passes through the first and the last control point, the first three knots are set to 0 and the last three knots are set to 1 [PT97]. The remaining knots are estimated by averaging the parameter values u_i:

$$\begin{aligned} v_0 = v_1 = v_2 &= 0, \\ v_{m-2} = v_{m-1} = v_m &= 1, \\ v_{j+2} &= \frac{1}{2} \sum_{i=j}^{j+1} u_i \quad j = 1, \ldots, n-2. \end{aligned} \tag{5.10}$$

There are two cases for modeling one side of the leaf margin with a B-spline curve: leaves without basal extension ($N = 3$) and leaves with basal extension ($N = 4$). These cases are discussed separately.

Case 1: N = 3

For leaves with no basal extension, $N = 3$ and the inputs to the B-spline fitting algorithm are $\mathbf{p}_1, \mathbf{t}_1, \mathbf{p}_2, \mathbf{t}_2, \mathbf{p}_3, \mathbf{t}_3$. The parameter values u_i are estimated using Equation 5.9 as

$$u_1 = 0, \; u_2 = \frac{||\mathbf{p}_2 - \mathbf{p}_1||}{D}, \; u_3 = 1. \tag{5.11}$$

The number of control points $(n + 1)$ must be equal to the number of equations $(2N)$. Thus, n is equal to 5. The number of knots $m + 1$ is related to the number of control points $(n + 1)$ and the degree d by $m = n + d + 1$ [PT97], which implies that m is equal to 8. The knots v_i are estimated using Equation 5.10 to yield

$$v_i = 0, 0, 0, \frac{u_2}{2}, u_2, \frac{u_2 + 1}{2}, 1, 1, 1. \tag{5.12}$$

Using Equations 5.11 and 5.12 and setting all $\alpha_i = D$, Equations 5.6 and 5.7 can be expanded as

$$\begin{aligned}
\mathbf{p}_1 &= \mathbf{q}_0, \\
D\mathbf{t}_1 &= -\frac{4}{u_2}\mathbf{q}_0 + \frac{4}{u_2}\mathbf{q}_1, \\
\mathbf{p}_2 &= (1-u_2)\mathbf{q}_2 + u_2\mathbf{q}_3, \\
D\mathbf{t}_2 &= -4\mathbf{q}_2 + 4\mathbf{q}_3, \\
\mathbf{p}_3 &= \mathbf{q}_5, \\
D\mathbf{t}_3 &= -\frac{4}{1-u_2}\mathbf{q}_4 + \frac{4}{1-u_2}\mathbf{q}_5.
\end{aligned} \tag{5.13}$$

In matrix form, the linear system is given by

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-\frac{4}{u_2} & \frac{4}{u_2} & 0 & 0 & 0 & 0 \\
0 & 0 & 1-u_2 & u_2 & 0 & 0 \\
0 & 0 & -4 & 4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & -\frac{4}{1-u_2} & \frac{4}{1-u_2}
\end{bmatrix}
\begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \mathbf{q}_4 \\ \mathbf{q}_5 \end{bmatrix}
=
\begin{bmatrix} \mathbf{p}_1 \\ D\mathbf{t}_1 \\ \mathbf{p}_2 \\ D\mathbf{t}_2 \\ \mathbf{p}_3 \\ D\mathbf{t}_3 \end{bmatrix}. \tag{5.14}$$

This linear system can be solved analytically for q_i:

$$\begin{aligned}
\mathbf{q}_0 &= \mathbf{p}_1, \\
\mathbf{q}_1 &= \mathbf{p}_1 + D\mathbf{t}_1\left(\frac{u_2}{4}\right), \\
\mathbf{q}_2 &= \mathbf{p}_2 - D\mathbf{t}_2\left(\frac{u_2}{4}\right), \\
\mathbf{q}_3 &= \mathbf{p}_2 + D\mathbf{t}_2\left(\frac{1-u_2}{4}\right), \\
\mathbf{q}_4 &= \mathbf{p}_3 - D\mathbf{t}_3\left(\frac{1-u_2}{4}\right), \\
\mathbf{q}_5 &= \mathbf{p}_3.
\end{aligned} \tag{5.15}$$

Case 2: N = 4

For leaves with basal extensions, $N = 4$ and the inputs to the B-spline fitting algorithm are $\mathbf{p}_1, \mathbf{t}_1, \mathbf{p}_2, \mathbf{t}_2, \mathbf{p}_3, \mathbf{t}_3, \mathbf{p}_4, \mathbf{t}_4$. The parameter values u_i are estimated using Equation 5.9 as

$$u_1 = 0,\ u_2 = \frac{d1}{D},\ u_3 = \frac{d2}{D},\ u_4 = 1, \tag{5.16}$$

where $d_1 = ||\mathbf{p}_2 - \mathbf{p}_1||$, and $d_2 = ||\mathbf{p}_3 - \mathbf{p}_2||$. The number of control points $(n+1)$ is equal to number of equations (2N). Thus, n is equal to 7. The number of knots $m+1$ is equal to $n+d+1$, which implies that $m = 11$. The knots v_i are estimated using Equation 5.10 as

$$v_i = 0, 0, 0, \frac{u_2}{2}, u_2, \frac{u_2+u_3}{2}, u_3, \frac{u_3+1}{2}, 1, 1, 1. \tag{5.17}$$

Using Equations 5.16 and 5.17 and setting all $\alpha_i = D$, Equations 5.6 and 5.7 can be expanded as

$$\begin{aligned}
\mathbf{p}_1 &= \mathbf{q}_0, \\
D\mathbf{t}_1 &= -\frac{4}{u_2}\mathbf{q}_0 + \frac{4}{u_2}\mathbf{q}_1, \\
\mathbf{p}_2 &= \frac{u_3-u_2}{u_3}\mathbf{q}_2 + \frac{u_2}{u_3}\mathbf{q}_3, \\
D\mathbf{t}_2 &= -\frac{4}{u_3}\mathbf{q}_2 + \frac{4}{u_3}\mathbf{q}_3, \\
\mathbf{p}_3 &= \frac{1-u_3}{1-u_2}\mathbf{q}_4 + \frac{u_3-u_2}{1-u_2}\mathbf{q}_5, \\
D\mathbf{t}_3 &= -\frac{4}{1-u_2}\mathbf{q}_4 + \frac{4}{1-u_2}\mathbf{q}_5, \\
\mathbf{p}_4 &= \mathbf{q}_7, \\
D\mathbf{t}_4 &= -\frac{4}{1-u_3}\mathbf{q}_6 + \frac{4}{1-u_3}\mathbf{q}_7.
\end{aligned} \tag{5.18}$$

In matrix form, the linear system is given by

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{4}{u_2} & \frac{4}{u_2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{u_3-u_2}{u_3} & \frac{u_2}{u_3} & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{4}{u_3} & \frac{4}{u_3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-u_3}{1-u_2} & \frac{u_3-u_2}{1-u_2} & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{4}{1-u_2} & \frac{4}{1-u_2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{4}{1-u_3} & \frac{4}{1-u_3}
\end{bmatrix}
\begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \mathbf{q}_4 \\ \mathbf{q}_5 \\ \mathbf{q}_6 \\ \mathbf{q}_7 \end{bmatrix}
=
\begin{bmatrix} \mathbf{p}_1 \\ D\mathbf{t}_1 \\ \mathbf{p}_2 \\ D\mathbf{t}_2 \\ \mathbf{p}_3 \\ D\mathbf{t}_3 \\ \mathbf{p}_4 \\ D\mathbf{t}_4 \end{bmatrix}. \tag{5.19}$$

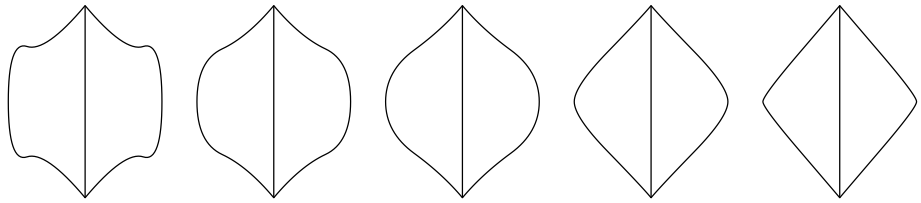


Figure 5.7: Effect of changing the value of α_i on the leaf shape.

This linear system can be solved in closed form for q_i as

$$\begin{aligned}
\mathbf{q}_0 &= \mathbf{p}_1, \\
\mathbf{q}_1 &= \mathbf{p}_1 + D\mathbf{t}_1 \left(\frac{u_2}{4}\right), \\
\mathbf{q}_2 &= \mathbf{p}_2 - D\mathbf{t}_2 \left(\frac{u_2}{4}\right), \\
\mathbf{q}_3 &= \mathbf{p}_2 + D\mathbf{t}_2 \left(\frac{u_3 - u_2}{4}\right), \\
\mathbf{q}_4 &= \mathbf{p}_3 - D\mathbf{t}_3 \left(\frac{u_3 - u_2}{4}\right), \\
\mathbf{q}_5 &= \mathbf{p}_3 + D\mathbf{t}_3 \left(\frac{1 - u_3}{4}\right), \\
\mathbf{q}_6 &= \mathbf{p}_4 - D\mathbf{t}_4 \left(\frac{1 - u_3}{4}\right), \\
\mathbf{q}_7 &= \mathbf{p}_4.
\end{aligned} \tag{5.20}$$

For visualization, the B-spline curve on each side of the primary vein is discretized into polylines with m sample points. In implementation, m is set to 64. Thus, each unilobed leaf is represented by 128 points.

5.4 Analysis of Laminar Shape Generation Algorithm

The parameter α_i in Equation 5.7 controls the tension of the B-spline curve. The smaller the value of α_i, the tighter is the B-spline curve. The effect of varying the value of α_i on the leaf shape is illustrated in Figure 5.7. For rhomboidal and deltoid leaf shapes in Figure 5.19, a value of $D/2$ was used for α_i, where D is the total chord length of the landmark points.

Figure 5.8 illustrates the effect of varying the parameter values for leaves without basal extension. The first row shows that as the tangent angle θ_b is increased from 0° to 90°, the shape of the base changes from concave (Figure 5.8a) to straight (Figure 5.8b,c) to convex

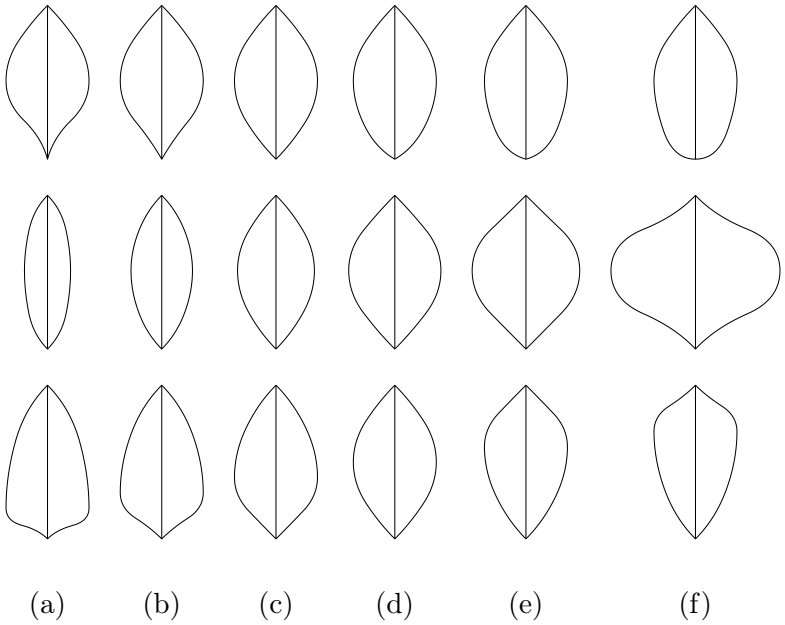


Figure 5.8: Effect of varying the parameter values in leaves without basal extension. The first row shows the effect of varying the tangent angle at the base. The second and the third rows show the effect of varying the x and y-coordinates of the waist, respectively.

(Figure 5.8d–f). The second row shows that as the x-coordinate of the waist is increased, the lamina becomes wider. Since, the tangents at the base and the apex are constant, the shapes of the base and apex change from convex (Figure 5.8a–c) to straight (Figure 5.8d,e) to concave (Figure 5.8f). This illustrates that the shape of the base and apex depends not only on the tangent angles θ_b and θ_a, respectively, but also on the x-coordinate of the waist. Let ϕ be the angle between the y-axis and the line-segment between the base and the waist. Then, the base shape is concave if $\theta_b < \phi - \epsilon$, straight if $\phi - \epsilon \leqslant \theta_b \leqslant \phi + \epsilon$, and convex if $\theta_b > \phi + \epsilon$, where ϵ is a small constant. The apex shape is defined similarly. The third row shows that as y-coordinate of the waist is increased, the shape of the waist changes from ovate (Figure 5.8a,b) to elliptic (Figure 5.8c,d) to obovate (Figure 5.8e,f).

Figure 5.9 illustrates the effect of varying the parameter values for the shape of the tail in leaves with basal extension. The first row shows that as the tangent angle θ_b is increased from 90° to 150°, the tail becomes flatter. The second row shows that as the x-coordinate of the tail is increased, the point of maximum basal extension moves away from the primary vein. The third row shows that as the y-coordinate of the tail is increased, the tail become longer.

The laminar shape generation algorithm should be numerically stable so that a small change

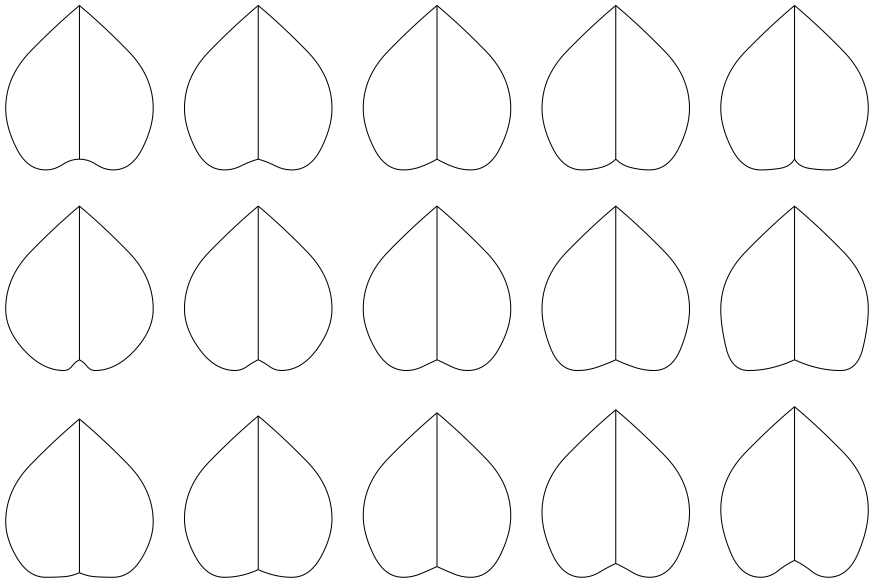


Figure 5.9: Effect of varying the parameter values in leaves with basal extension. The first row shows the effect of varying the tangent angle at the base. The second and the third rows show the effect of varying the x and y-coordinates of the tail, respectively.

in parameter values produce a small change in the leaf shape. The stability of the algorithm for leaves without basal extension is estimated from the linear system of Equation 5.14. A linear system of equations is unstable if the coefficient matrix has a large condition number. In Equation 5.14, the coefficient matrix is defined by a single parameter u_2. The condition number of the coefficient matrix is illustrated in Figure 5.10. It is clear that the condition number is small except when u_2 is close to zero or one. Thus, the laminar shape generation algorithm is stable everywhere expect when the waist is close to the base or the apex.

Similar analysis can be performed for the laminar shape generation algorithm for leaves with basal extension. The coefficient matrix in Equation 5.19 is defined by two parameters u_2 and u_3 with $u_2 < u_3$. The condition number of the coefficient matrix is illustrated in Figure 5.11. The laminar shapes generation algorithm is stable everywhere except when the waist is close to the apex or the tail, or when the tail is close to the base.

In the Equation 5.14, when u_2 is equal to zero or one, the coefficient matrix has a zero determinant. Thus, the coefficient matrix is not invertible and the control points of B-spline curve cannot be computed. In this case, closed form solution (Equation 5.15) can be used to compute the control points of the B-spline curve. In this case, the tangent at the base or the apex has no effect on the shape of the curve. In the implementation, Equations 5.15 and 5.20 are used.

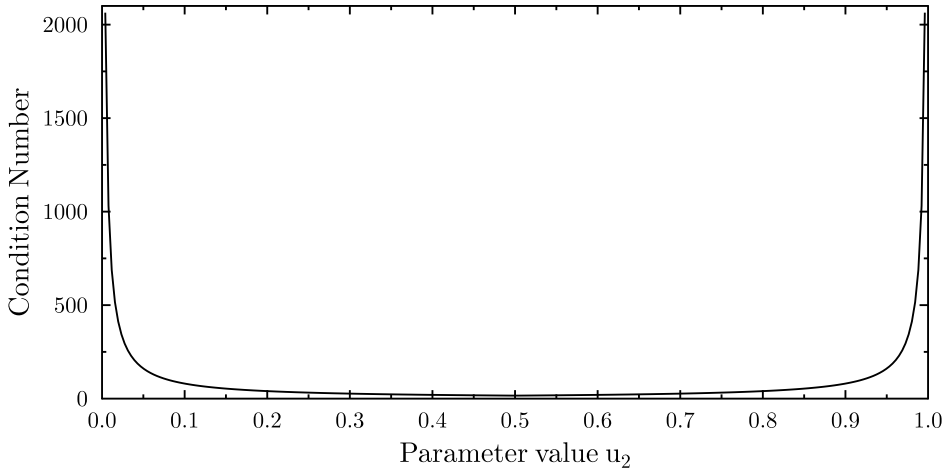


Figure 5.10: Numerical stability of the laminar shape generation algorithm for leaves without basal extension. The condition number is small for all values of the parameter u_2 except when it is close to zero or one.

The laminar shape generation algorithm is both time and space efficient. The time for fitting a B-spline curve to a set of N landmark points is $O(N^3)$, which is the time required to solve the linear system in Equations 5.14 or 5.19. For the unilobed leaves, the number of landmark points is 2, 3, or 4. Hence, the time complexity of laminar shape generation algorithm is $O(1)$. The space required for fitting a B-spline curve to a set of N landmark points is $O(N^2)$, which is the space required to store the matrices in Equations 5.14 or 5.19. Again, since the number of landmark points is small, the laminar shape generation algorithm needs $O(1)$ space. Thus, the laminar shape generation algorithm is constant time and space for all unilobed leaves.

5.4.1 Accuracy of Generated Leaf Shapes

One of the main goals of the leaf model is to generate many instances of a leaf for simulating interaction of plants with the environment. For the simulation to be realistic, it is important that the generated instances should match the real leaf shapes. 34 real leaves (Figure 5.12) were selected from examples given in Chapters 2 and 4 to evaluate the accuracy of the laminar shape generation algorithm.

The differences between the generated laminar shapes and the laminar shapes of real leaves were computed as follows. First, for each real leaf in Figure 5.12, the laminar shape was extracted from the image manually aid by image processing tool. In addition, the base and the apex points were manually marked on the images. Then, for each real leaf, the

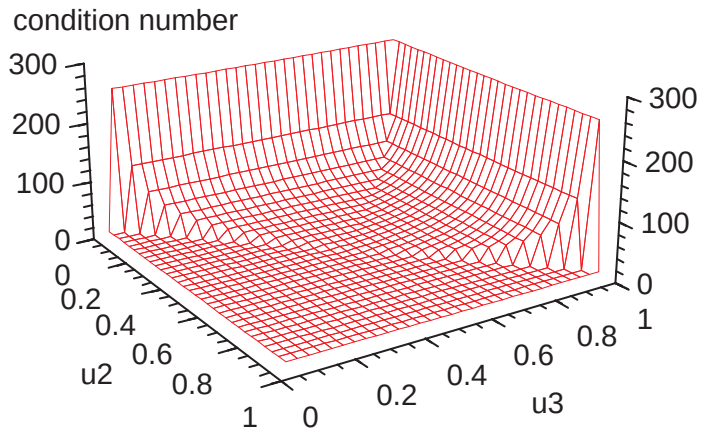


Figure 5.11: Numerical stability of the laminar shape generation algorithm for leaves with basal extension. The condition number is small for all values of the parameters u_2 and u_3 except when u_2 is close to zero, or u_3 is close to u_2 or one.

laminar shape was generated using the graphical user interface described in Section 5.2. Both laminar shapes, the real and the generated, were represented as polygons and the positions of the base and the apex were also saved. Since the real leaf images were of different sizes, the laminar shapes should be normalized for comparison. To normalize the laminar shapes, bases were translated to the origin, and apexes were transformed so that they laid at (0, 1). Thus, the primary vein for all laminar shapes had unit length and laid on the y-axis.

After normalizing the laminar shapes, the difference between the generated and the real laminar shapes was measured by the Euclidean distance between their corresponding points. For a point $\mathbf{p}_i$ on the generated laminar shape, its corresponding point on the real laminar shape was computed by intersecting the line along the normal to the generated laminar shape at $\mathbf{p}_i$ with the real laminar shape. In general, the line along the normal at $\mathbf{p}_i$ can have more than one intersection with the real laminar shape. In this case, the point of intersection on the real laminar shape closest to $\mathbf{p}_i$ was selected as the corresponding point. This algorithm was used because it worked well even when real laminar shape had teeth and drip-tips.

Figure 5.13 illustrates the boxplots of the Euclidean distance between the corresponding points in the real and the generated laminar shapes. Boxplot is a convenient way to summarize a sample using five statistics: minimum (lower end of the vertical line), 25th percentile (lower edge of rectangle), median (horizontal line in the box), 75th percentile (upper edge of rectangle), and maximum (upper end of the vertical line). It can be seen that most of the generated laminar shapes have maximum error smaller than 0.03 (3% of the length of the primary vein). Eight generated laminar shapes have maximum error larger than 0.03. The

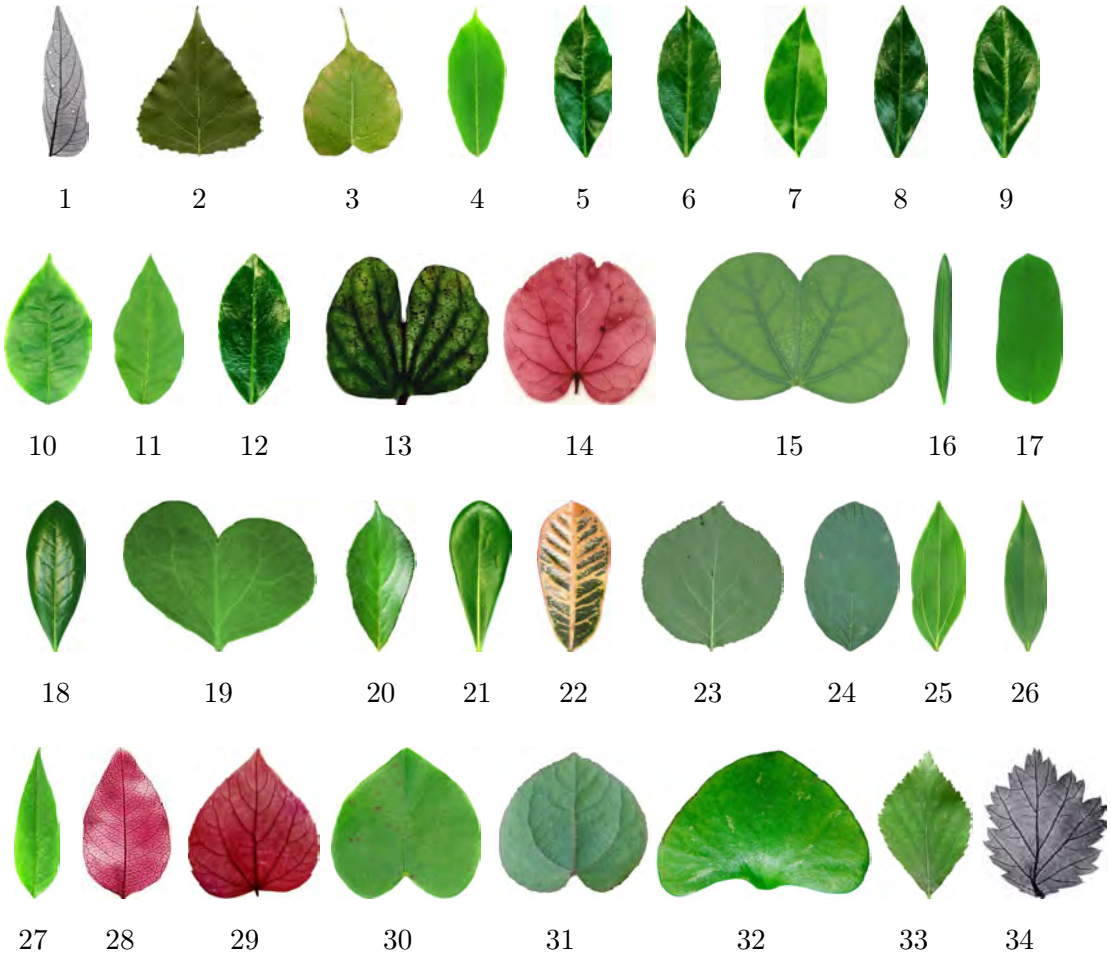


Figure 5.12: Real leaves used for evaluating the accuracy of laminar shape generation algorithm. These leaves were selected from examples given in Chapters 2 and 4.

real and the generated laminar shapes of these leaves are illustrated in Figure 5.14. Leaf number 3 has a drip-tip and leaf number 23 and 24 have teeth which are not modeled. Leaf number 13, 14, and 31 have rough margins which are not captured by the leaf model. The proposed algorithm only generates smooth laminar shapes. Leaf number 15 and 32 are not precisely captured by the leaf model but the generated shapes are still very close to the real shapes.

In conclusion, using a sample of leaves with various shapes, it is shown that the leaf model can generate leaf shapes that match the real leaf shapes well. Extensive verification, however, is not practical because of the huge variations in leaf shapes. Even from the same plant, leaf shapes can vary a lot in minute details.

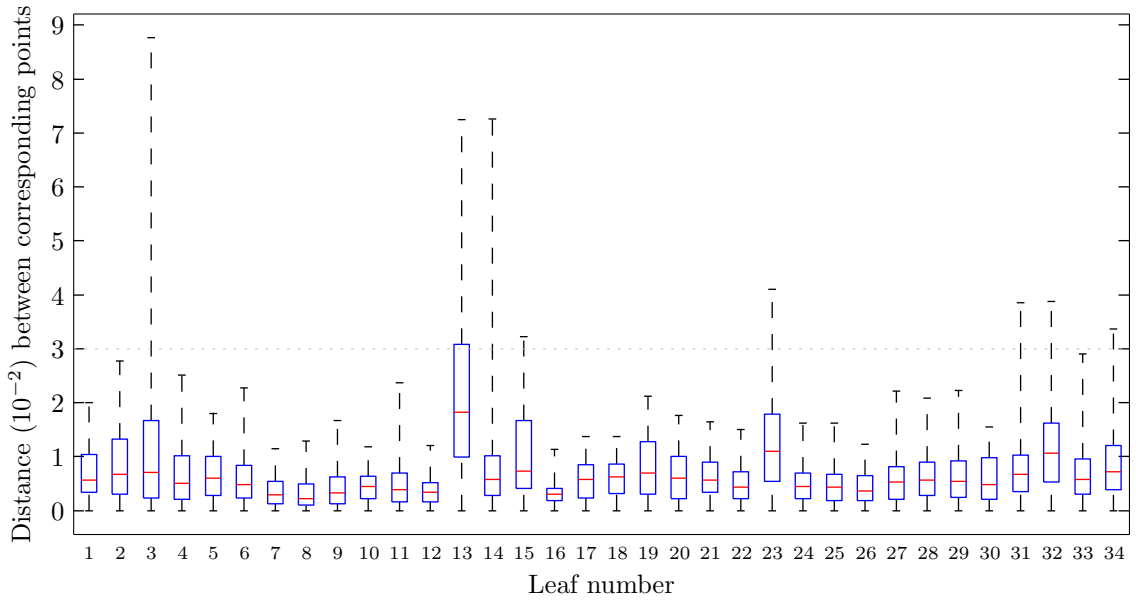


Figure 5.13: Boxplots of the Euclidean distance between the corresponding points in real and generated laminar shapes. Each boxplot describes the distribution of distances of all corresponding points between a real and the generated laminar shape. The numbers along the x-axis indicate the real leaves illustrated in Figure 5.12.

5.5 Leaf Shape Generation Examples

The proposed leaf model in this thesis can generate 50 types of unilobed leaf shapes as categorized in Section 4.1 (Table 4.2). Figures 5.15 to 5.17 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 5.18. Among them, 26 shape types have real leaf examples. The others may also occur in nature but the author is unable to find real leaf examples for them.

Figure 5.19 illustrates leaf shapes that have been given specific names by botanists [HGL92]. Note that some of these leaf shapes belong to the same type. For example elliptic, oval, and orbicular leaves in Figure 5.19 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 the proposed model match those of the real leaves very well.

Figure 5.20 shows that our model can generate asymmetric leaf shapes. Figure 5.21 illustrates more complex leaf shapes. Top row shows that laminar shape generation algorithm can generate leaf shapes with drip tips. For a leaf with a very long and slender drip tip, the

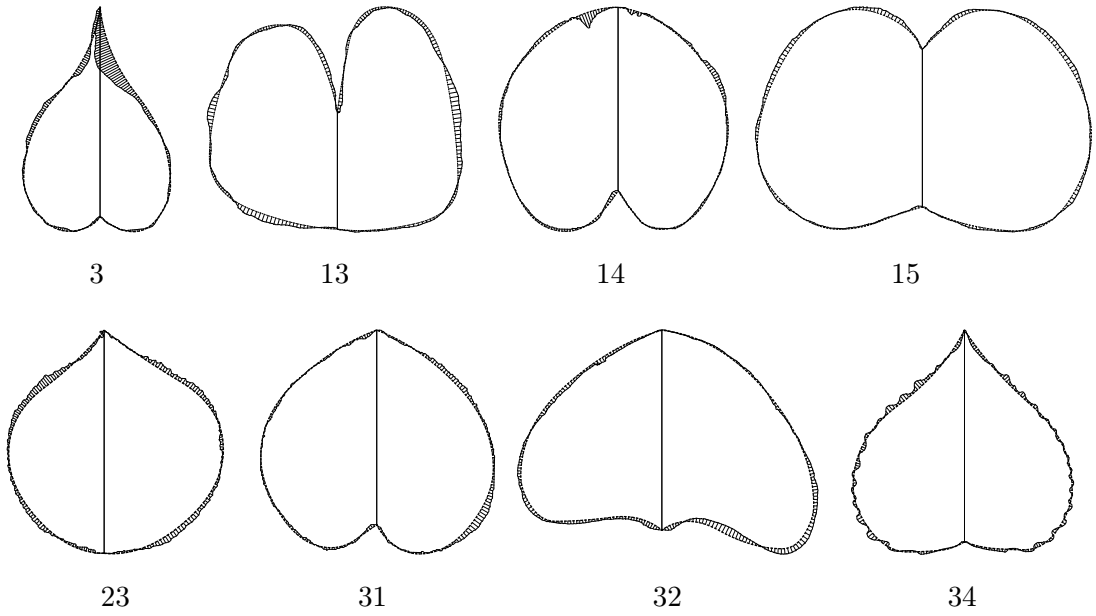


Figure 5.14: Comparison of the generated and the real laminar shapes with maximum error greater than 0.03 in Figure 5.13. The solid lines represent real laminar shapes and dotted lines represent the generated laminar shapes. The lines connecting them are the corresponding points. The numbers below the laminar shapes refer to the real leaves in Figure 5.12.

match between the generated tip and the real tip is not perfect (Figure 5.21, 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 5.21 shows that the generated margins fit the overall shapes of the leaves with teeth. As discussed in Section 4.1, teeth are omitted in our model and can be added using methods such as curve analogies [ZG04].

Figure 5.22 illustrates an example of lotus leaf in which the base is at the center of the lamina. The proposed laminar shape generation algorithm can generate instance of such leaves, however, the base will be at the margin of the leaf, instead of the center. Some unilobed leaves have naturally curved primary vein. For simplicity, the proposed leaf model assumes that the primary vein is straight but the laminar shape generation algorithm is general and can generate curved unilobed leaves. Leaves with curved veins are generated by specifying additional parameters to define a curved vein and then the landmark points are expressed relative to the curved veins (Figure 5.23).

For many applications, many leaf instances for a given kind of leaf should be generated. The

laminar shape generation algorithm can generate multiple leaf instances by perturbing the parameter values. The maximum amount of perturbation is specified by the user and is expressed as a percentage of the parameter value. The algorithm adds a random percentage within the maximum perturbation to each parameter value. Figure 5.24 illustrates five instances each for elliptic, cordate and asymmetric leaves generated by adding 20% perturbations to the parameter values. The effect of perturbation is illustrated in Figure 5.25. The instances in the first, second, and third rows are generated by adding 20%, 30%, and 40% perturbations to the parameter values, respectively.

Note that some combinations of extreme parameter values can produce shapes that do not look like leaves. Figure 5.26 illustrate some of these non-leaf shapes. Such non-leaf shapes can be used to populate virtual worlds.

On a laptop with Intel Core i7 2.0 GHz processor, on an average 2500 leaves per second can be generated. This shows that the laminar shape generation algorithm is fast and can be used for quickly generating large number of leaves instances.

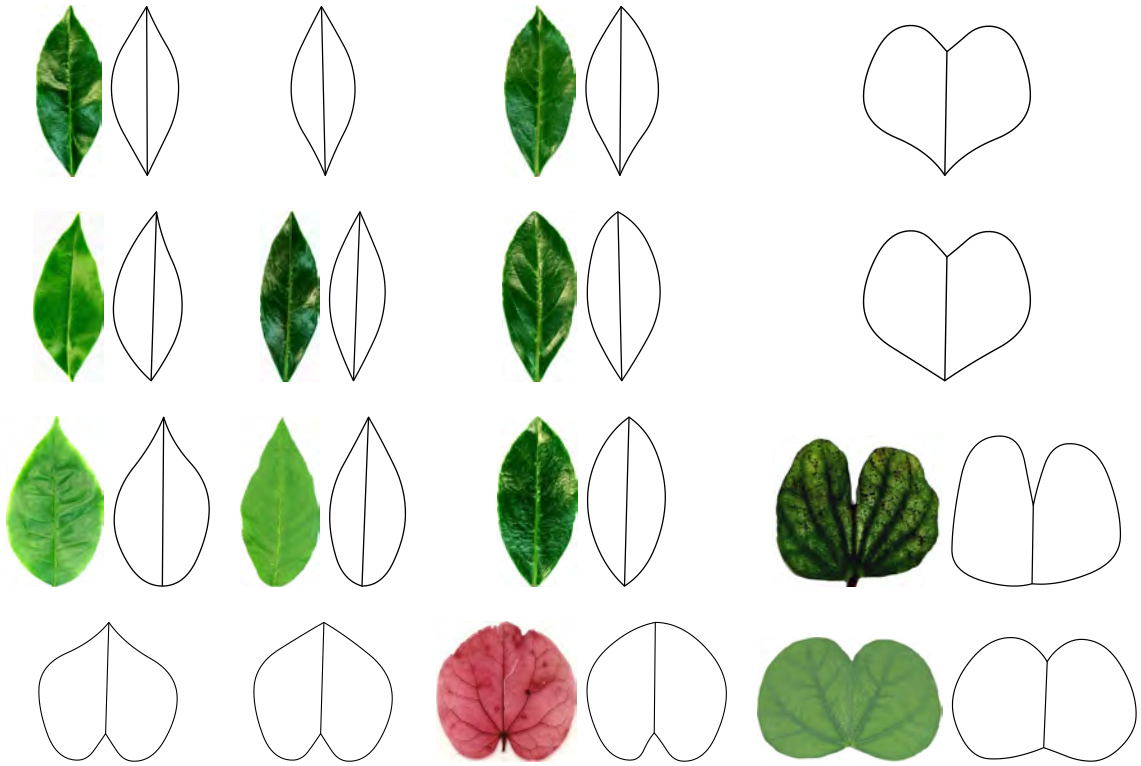


Figure 5.15: Laminar shapes generated for leaves with elliptic waist illustrated in Figure 4.2. Rows correspond to leaves with same base shapes. (Row 1) Concave. (Row 2) Straight. (Row 3) Convex. (Row 4) Base with extension. Columns correspond to leaves with same apex shapes. (Column 1) Concave. (Column 2) Straight. (Column 3) Convex. (Column 4) Apex with extension.

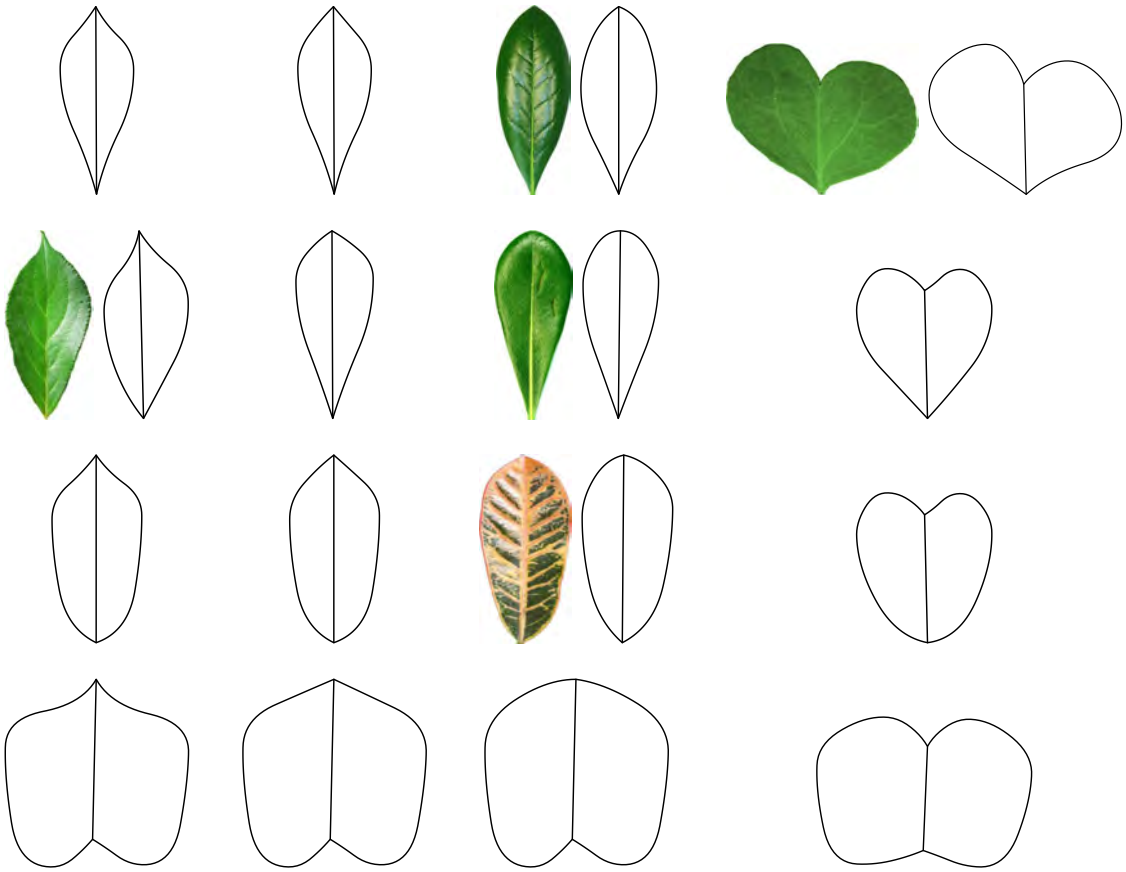


Figure 5.16: Laminar shapes generated for leaves with Obovate waist illustrated in Figure 4.3. Rows correspond to leaves with same base shapes. (Row 1) Concave. (Row 2) Straight. (Row 3) Convex. (Row 4) Base with extension. Columns correspond to leaves with same apex shapes. (Column 1) Concave. (Column 2) Straight. (Column 3) Convex. (Column 4) Apex with extension.

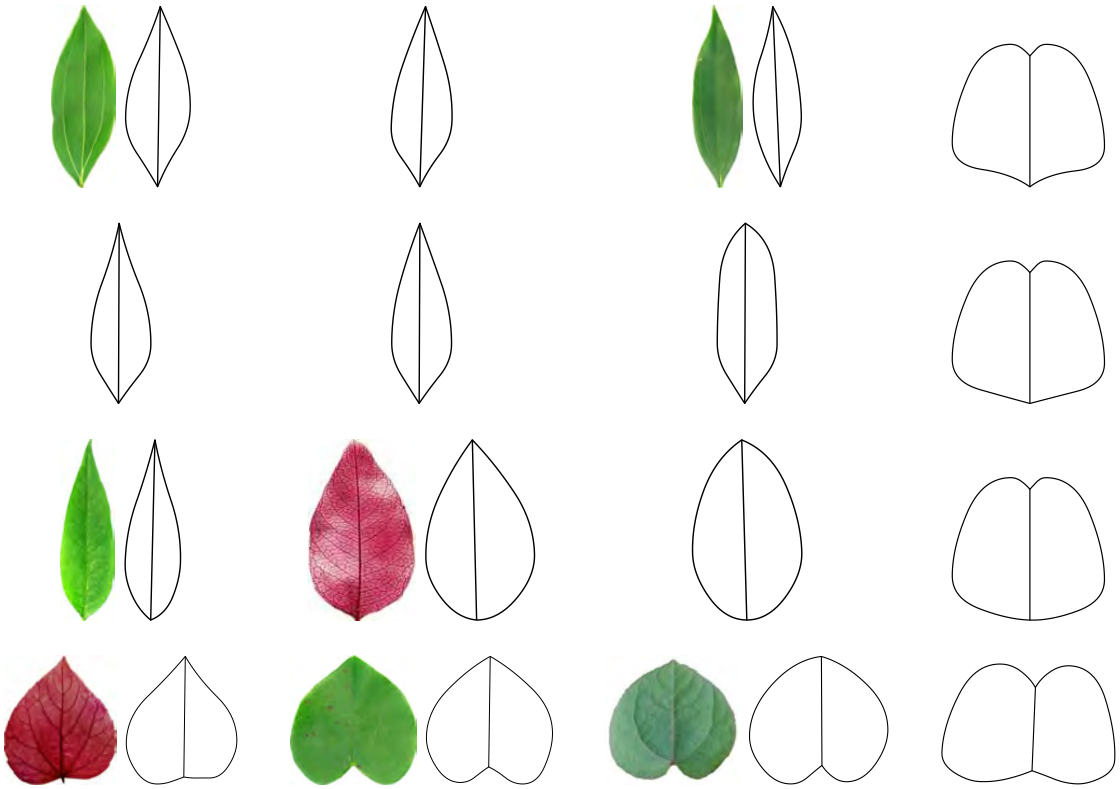


Figure 5.17: Laminar shapes generated for leaves with Ovate waist illustrated in Figure 4.4. Rows correspond to leaves with same base shapes. (Row 1) Concave. (Row 2) Straight. (Row 3) Convex. (Row 4) Base with extension. Columns correspond to leaves with same apex shapes. (Column 1) Concave. (Column 2) Straight. (Column 3) Convex. (Column 4) Apex with extension.

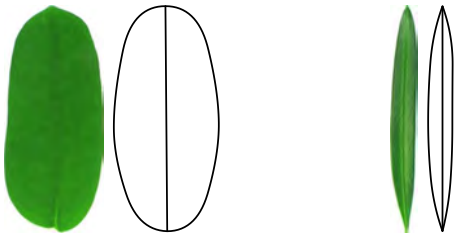


Figure 5.18: Generated instances for oblong and linear leaves illustrated in Figure 4.5. (a) oblong leaf, and (b) linear leaf

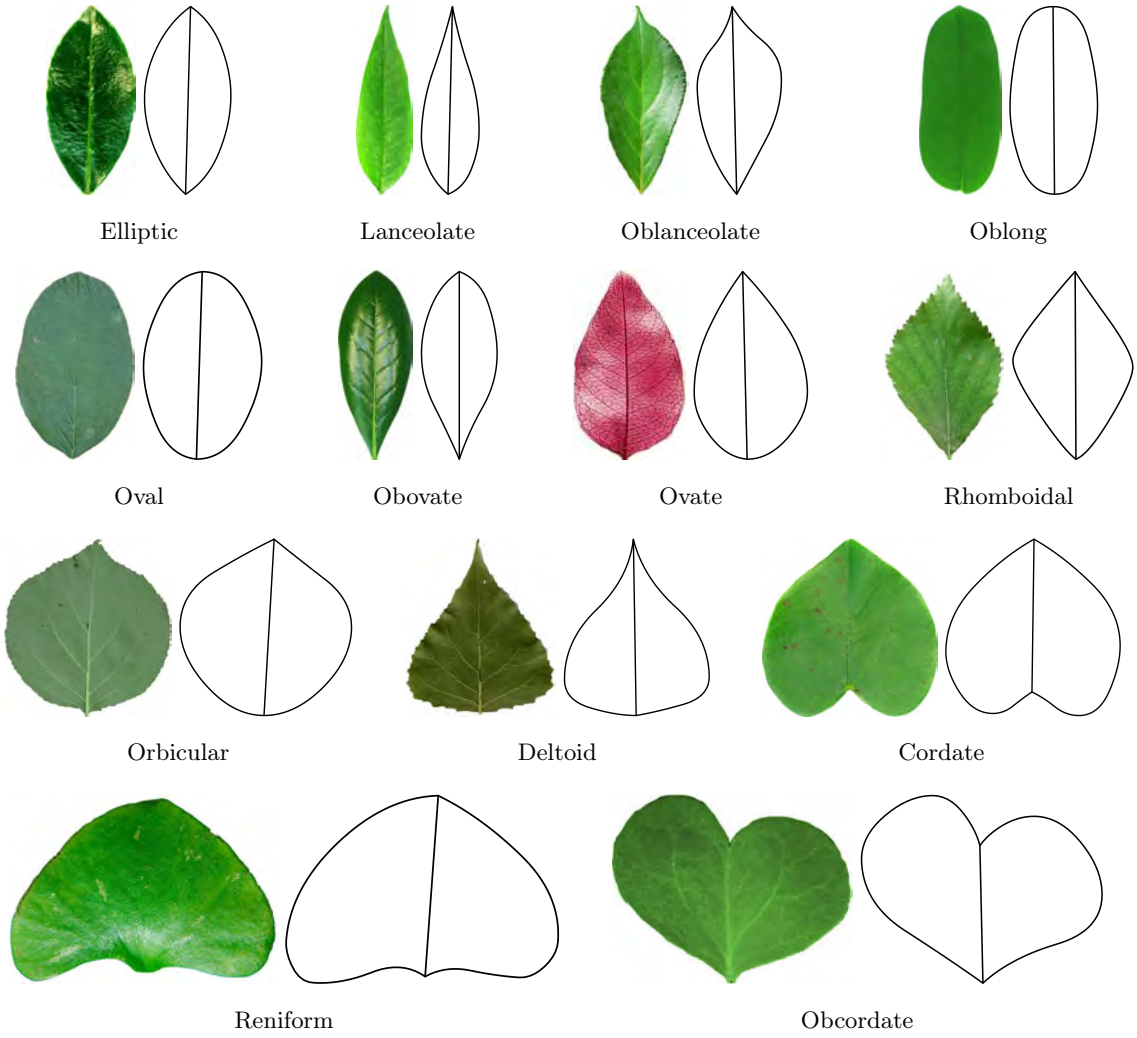


Figure 5.19: Examples of leaf shapes commonly discussed in botanical literature illustrated in Figure 2.7

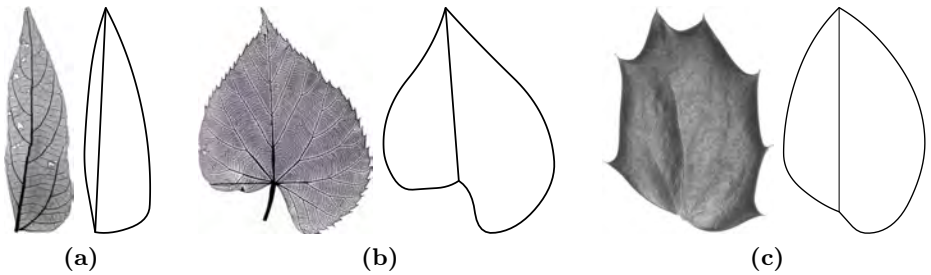


Figure 5.20: Generated instances of asymmetric leaves shapes illustrated in Figure 2.9. (a) A leaf with asymmetric waist. (b) A leaf with asymmetric basal extension. (c) A simple leaf with basal extension on the one side and no extension on the other.

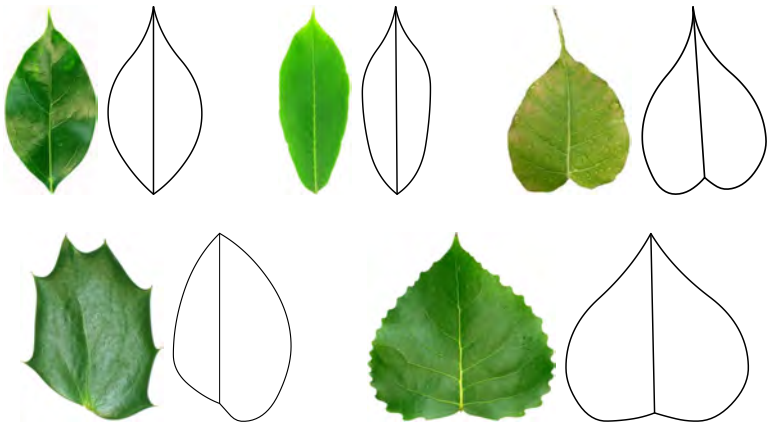


Figure 5.21: Laminar shapes generated for complex shapes, (Top) Leaves with drip-tips. (Bottom) Leaves with teeth.

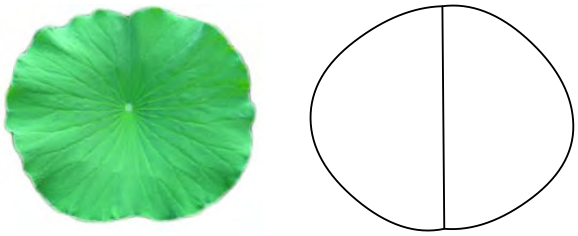


Figure 5.22: Generated instance of a lotus leaf.

Figure 5.23: Generated instance of a leaf with curved primary vein.

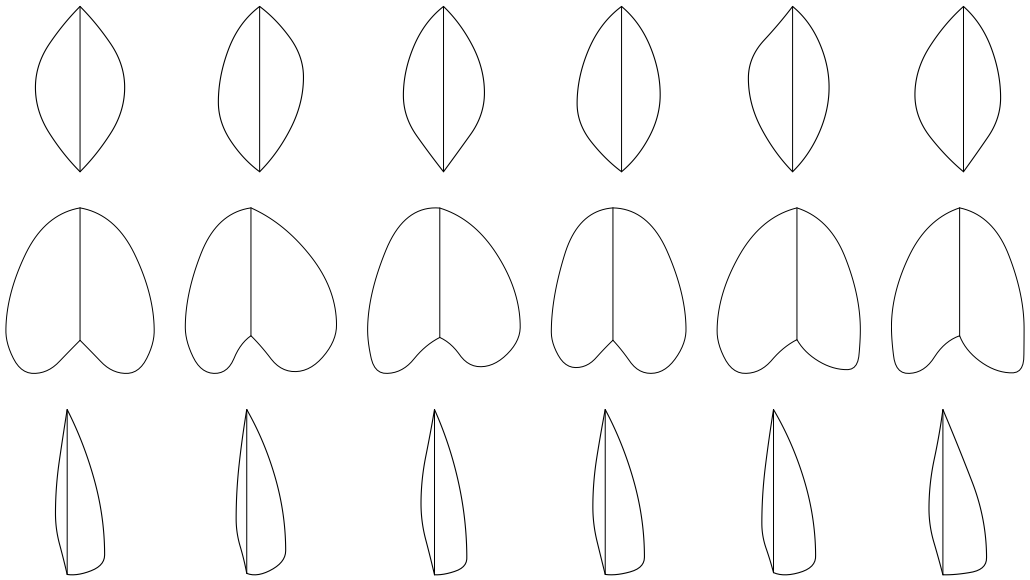


Figure 5.24: Leaf instances generated for elliptic, cordate, and asymmetric leaves. The instances in the first column are generated without perturbing the parameter values. Instances in the remaining columns are generated by adding 20% perturbations to the parameter values.

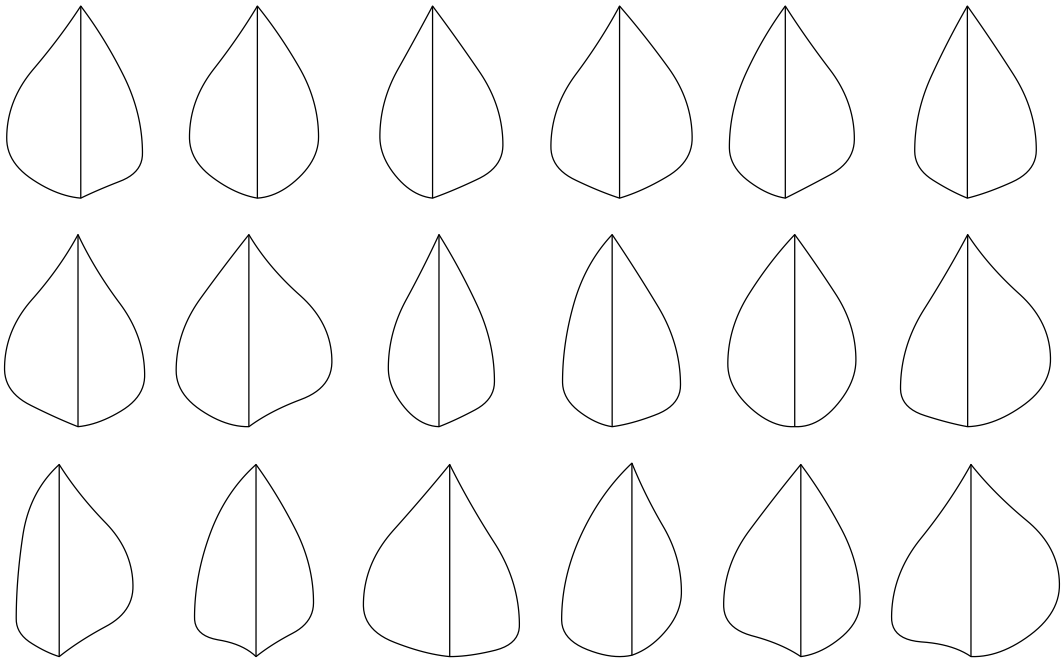


Figure 5.25: Effect of perturbation in leaf shapes. The leaf instances in the first, second, and third rows are generated by adding 20%, 30%, and 40% perturbations to the parameter values, respectively.

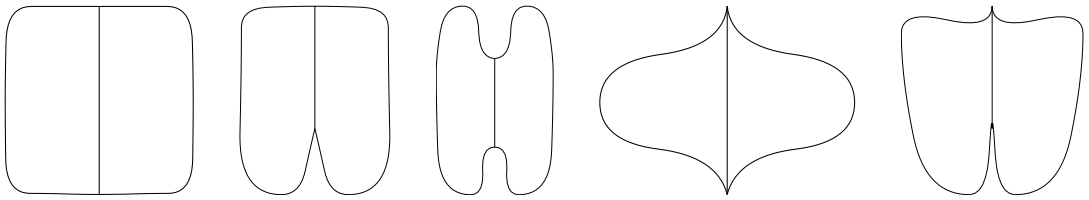


Figure 5.26: Examples of non-leaf shapes.

Chapter 6

Modeling of Multilobed Leaves

This chapter presents a parametric model of the geometric shapes of multilobed leaves discussed in Section 6.1. The shape of a multilobed leaf is represented by a combination of unilobed leaves, one for each lobe. A GUI is used to interactively specify the parameters using a reference image of a leaf (Section 6.2). The laminar shape generation algorithm generates the laminar surface using the parameters of the leaf model (Section 6.3). Performance of the algorithm is discussed in Section 6.5.

6.1 Parametric Leaf Model

The margin of a multilobed leaf is represented by a combination of unilobed leaves. One unilobed leaf is used for each lobe of the multilobed leaf. To combine the unilobed leaves, first they must be placed and arranged in space. The placement and arrangement of unilobed leaves is defined by the venation pattern of multilobed leaves. For a palmately lobed leaf (Figure 2.8c) whose lobes originate at the base, primary veins (Figure 2.10c) are used for placement. For a pinnately lobed leaf (Figure 2.8d) whose lobes originate along the primary vein, the primary vein and major secondary veins (Figure 2.11a) are used for placement. To simplify notation, this thesis categorizes veins in a multilobed leaf into two types: α_0-vein and α-veins. α_0-vein is the primary vein which runs from the base to the apex of the leaf. α-veins are the remaining primary veins in palmately lobed and major secondary veins in pinnately lobed leaves. α-veins originate from the α_0-vein and run to the apex of the lobes. They are ordered from the base to the apex of a leaf (Figure 6.1).

In general, the placement and arrangement of the α-veins may not follow a strict linear relationship. For simplicity of user specification, linear relationship is modeled. In this way, the user has to specify the placement of only the first and the last lobes. The placement of the other lobes can be computed automatically. However, the laminar shape generation algorithm is independent of the function and, in general, any function can be used. It is also possible to specify the start and end-point of each α-vein.

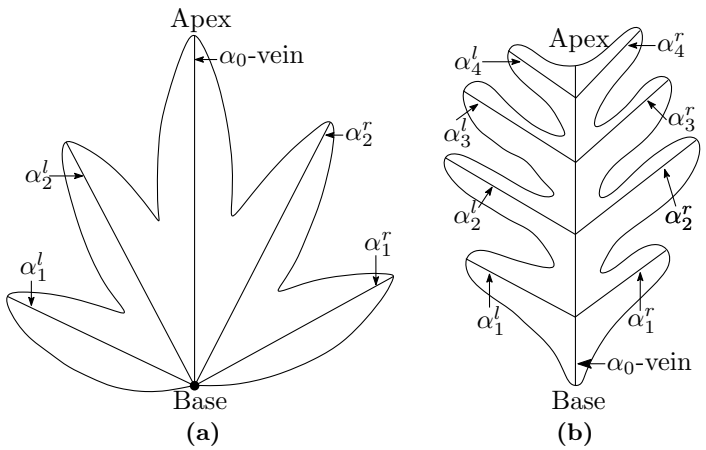


Figure 6.1: Venation model for multilobed leaves. Multilobed leaves have two types of veins: the primary vein α_0, which runs from the base to the apex and the α-veins, which originate from α_0-vein and runs to the apex of the lobes.

α_0-vein is defined by the base $\mathbf{p}_b$ and the apex $\mathbf{p}_a$. $\mathbf{p}_b$ is fixed at $(0, 0)$ and $\mathbf{p}_a$ is fixed at $(0, 1)$. α-veins on one side of the leaf are defined by their endpoints $\mathbf{r}_i$ and $\mathbf{e}_i$, $i = 1, \ldots, n$, where n is the number of lobes on one side of the leaf. Let N be the total number of lobes, then n is equal to $\lfloor N/2 \rfloor$. The position of the base of the ith α-vein α_i on the right side of the α_0-vein is given by:

$$\mathbf{r}_i = (0, r_i), \quad i = 1, \ldots, n. \tag{6.1}$$

The y-coordinate r_i of the α-vein α_i is defined in terms of the y-coordinate r_{i-1} of the α-vein α_{i-1} and the spacing s_{i-1} between them:

$$r_i = r_{i-1} + s_{i-1}. \tag{6.2}$$

The spacing between two α-veins α_i and α_{i-1} is a linear function:

$$s_i = s_0 + i\Delta s, \tag{6.3}$$

where Δs is a constant rate of change of spacing and $s_0 = r_1$ is the distance of the first α-vein α_1 on the right side of the α_0-vein from the origin (Figure 6.2). The position of the base of the ith α-vein α_i is computed recursively by expanding Equation 6.2 and then

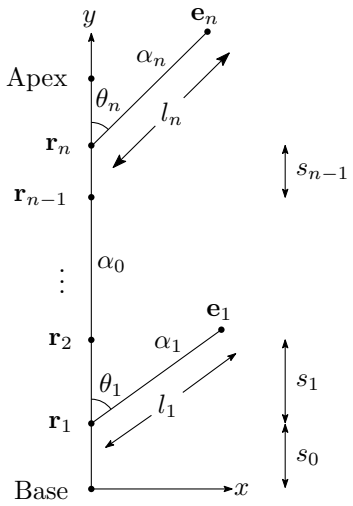


Figure 6.2: Parameters of the venation pattern for multilobed leaves.

substituting s_i with s_0 and Δs from Equation 6.3:

$$\begin{aligned} r_i &= r_{i-1} + s_{i-1}, \\ &= r_{i-2} + s_{i-2} + s_{i-1}, \\ &= r_{i-3} + s_{i-3} + s_{i-2} + s_{i-1}, \\ &\vdots \\ &= r_1 + s_1 + s_2 + \cdots + s_{i-1} \\ &= s_0 + (\Delta s + s_0) + (2\Delta s + s_0) + \cdots + ((i-1)\Delta s + s_0). \end{aligned}$$

Therefore,

$$r_i = is_0 + \frac{i(i-1)}{2}\Delta s. \tag{6.4}$$

In general, initial spacings s_0^l and s_0^r of the left and right α-veins can be different. s_0 and Δs are zero for palmately lobed leaves because all lobes originate at the base of the leaf. Since the apex is fixed at $(0, 1)$, the y-coordinate r_n of the last α-vein α_i must be less then one:

$$r_n = ns_0 + \frac{n(n-1)}{2}\Delta s < 1. \tag{6.5}$$

Therefore, for a given value of s_0, Δs has a upper bound:

$$\Delta s < \frac{2(1 - ns_0)}{n(n-1)}. \tag{6.6}$$

The angle of the α-vein α_i relative to the α_0-vein is given by:

$$\theta_i = \theta_1 + (i-1)\left(\frac{\theta_n - \theta_1}{n-1}\right), \tag{6.7}$$

where θ_1 and θ_n are the angles of the first and the last α-vein, respectively (Figure 6.2). θ_i varies from 0 to π. Similarly, the length of α-vein α_i relative to the α_0-vein is given by:

$$l_i = l_1 + (i-1)\left(\frac{l_n - l_1}{n-1}\right), \tag{6.8}$$

where l_1 and l_n are the lengths of the first and the last α-vein, respectively (Figure 6.2). l_i is expressed relative to the length of the primary vein and lies between 0 and 1.

In asymmetric leaves, the angles and lengths of α-veins can be different for two sides of the leaf. Similar to spacing, the linear function is used to define orientation, and length of α-veins because of its simplicity. In general, any function can be used and in particular, the user can specify the angle and length of each α-vein. For simplicity, the leaf model assumes that α-veins are straight, instead of curved. However, this is not inherent limitation of the laminar shape generation algorithm. As illustrated in Figure 5.23, curved veins can be specified using additional parameters and then the landmark points can be expressed with respect to curved veins.

The position $\mathbf{e}_i$ of the apex of the ith α-vein α_i is defined by the base $\mathbf{r}_i$, orientation θ_i and the length l_i of the α-vein as:

$$\mathbf{e}_i = \mathbf{r}_i + l_i \left[\sin\theta_i, \cos\theta_i\right]^T. \tag{6.9}$$

The shapes of lobes are defined by the model of unilobed leaf without basal extension. As discussed in Section 5.1, the margin of one side of the unilobed leaf without basal extension is defined by four parameters: θ_b, θ_a, and $\mathbf{W}$. θ_b and θ_a define the tangents to the margin at the base and the apex, respectively, and $\mathbf{W}$ is the waist of the unilobed leaf.

The positions of the valleys are defined by two parameters ϕ and m (Figure 6.3). ϕ is the orientation of the valleys relative to the α-veins. m is a scalar which define the distances of the valley positions from the origins $\mathbf{r_i}$ relative to the lengths l_i of the α-veins. The position $\mathbf{h}_i$ of the valley between lobes at α-veins α_i and α_{i+1} is given by:

$$\mathbf{h_i} = \mathbf{r_i} + m \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} (\mathbf{e_i} - \mathbf{r_i}) \tag{6.10}$$

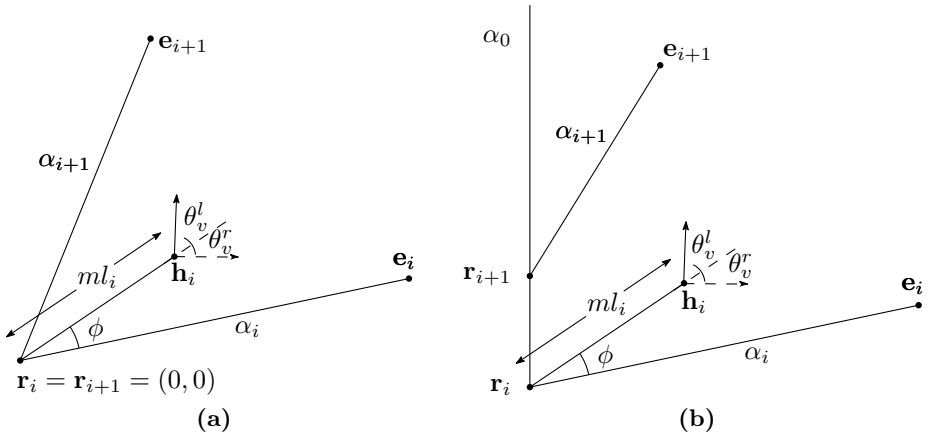


Figure 6.3: The parameters for specifying the valley position and shape in multilobed leaves. (a) Parameters for a palmately lobed leaf. (b) Parameters for a pinnately lobed leaf.

The shapes of the valleys between lobes at α-veins α_i and α_{i+1} are defined by the tangents $\mathbf{t}_i^l$ and $\mathbf{t}_i^r$ to the margin at the valley. $\mathbf{t}_i^l$ and $\mathbf{t}_i^r$ are defined by the angles θ^l and θ^r, respectively, relative to $(\mathbf{h}_i - \mathbf{r_i})$. In pinnately lobed leaves with even number of lobes, there is a valley at the apex of the leaf. The shape of the valley is specified by the angle ψ the tangent to margin makes with the α_0-vein.

In summary, the leaf model for the multilobed leaves consists of 26 parameters:

- The number of lobes: N.
- Parameters defining spacing between α-veins: s_0^l, s_0^r, and Δs.
- Parameters defining the orientations of α-veins: θ_1^l, θ_n^l, θ_1^r, and θ_n^r.
- Parameters defining the lengths of α-veins: l_1^l, l_n^l, l_1^r, and l_n^r.
- Parameters of the 1st lobe: θ_b^l, θ_b^r, $\mathbf{W^l}$, $\mathbf{W^r}$, θ_a^l, and θ_a^r.
- Parameters defining position and shape of valleys: m, ϕ, θ_v^r, θ_v^l, and ψ.

6.2 User Interface

A user intuitively specifies the shape of a multilobed leaf using an interactive GUI. Figure 6.4 illustrates an example of specifying the parameters of a multilobed leaf. The user first loads a reference image and specifies the number of lobes in the leaf (Figure 6.4a). Then, the user specifies the α_0-vein by placing a point at the base and the apex of the leaf (Figure 6.4b). For a palmately lobed leaf (Figure 6.4c), the α_i-veins originate from

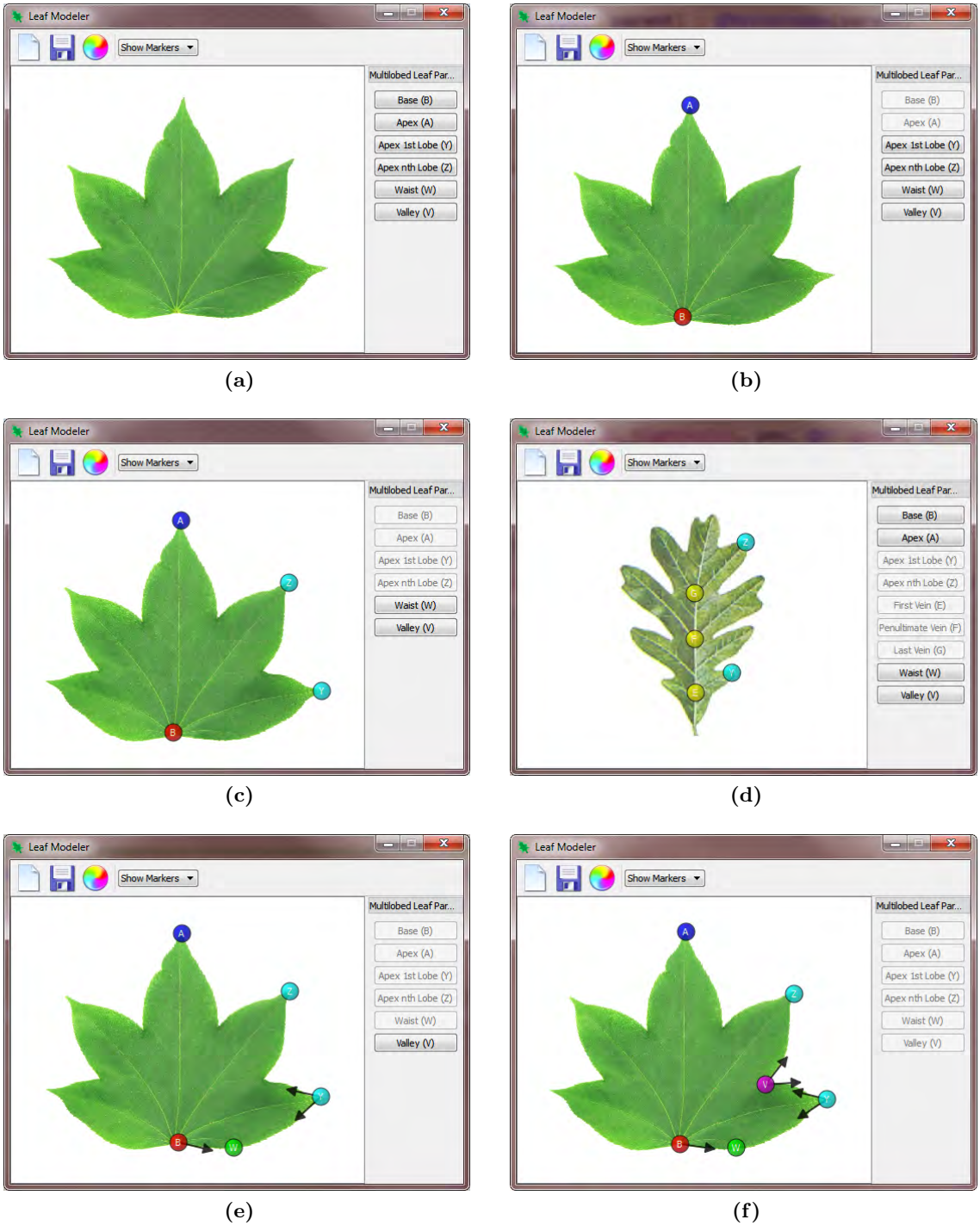


Figure 6.4: Specifying the parameters of a multilobed leaf using interactive GUI. (a) Reference image. (b) Specifying α_0-vein. (c),(d) Specifying α_i-veins. (e) Specifying the shape of first lobe. (f) Specifying the position and shape of valleys.

the base. So, the user specifies the orientation and the lengths of the α_i-veins by first

placing a point each at the apex of the first and the last lobe on the right side of the leaf. The orientation and lengths of the other lobes will be derived by the algorithm. The user specify the shape of the first lobe by placing tangent arrows at its base and apex (Figure 6.4e). If the leaf is asymmetric, the user can also specify the lobes on the left side of the leaf. Finally, the user specifies the position and shape of the valley by placing a point with two arrows at the valley between the first and the second lobe on the right side of the leaf.

For a pinnately lobed leaf, the user can specify the parameters in the same way. In addition the user has to specify the spacing of the α_i-veins by placing a point each at the base of the first, penultimate, and last lobe.

In general, the user may choose to specify the position, orientation, and length of each α_i-vein, the shape of each lobe and the position and shape of each valley. This is possible without effecting the laminar shape generation algorithm, though it would be tedious. In this case, the parameters are all specified by the user instead of being derived by Equations 6.4, 6.7 and 6.8.

6.3 Laminar Shape Generation Algorithm

The margin of a multilobed leaf is generated in three main steps (Figure 6.5): (1) generation of venation pattern, (2) generation of lobes, and (3) combination of lobes.

In the first step, the venation pattern is generated. The end-points of the α_0-vein are fixed at $(0, 0)$ and $(0, 1)$. The end-points $\mathbf{r}_i$ and $\mathbf{e}_i$ of α-veins are computed from $s_0, \Delta s, \theta_1, \theta_n, l_1$, and l_n using Equations 6.4 and 6.9 (Figure 6.5a).

In the second step, a lobe L is generated from θ_b, $\mathbf{W}$, and θ_a using the algorithm discussed in Section 5.3. A scaled copy of L is placed along each α-vein such that the base and the apex of the copy of L is at the base and apex of the α-vein. In palmately lobed leaves and pinnately lobed leaves with odd number of lobes, a scaled copy of L is also placed at the α_0-vein (Figure 6.5b).

Finally, the adjacent lobes are combined to form the margin. The algorithm first computes the positions of the valleys as the point of intersection of adjacent lobes (Figure 6.5c, left). Each lobe is discretized into a set of connected line segments. The point of intersection of two adjacent lobes is the point of intersection of a pair of line segments, one from each lobe. If two adjacent lobes do not intersect, then the point of intersection of lobes with the y-axis

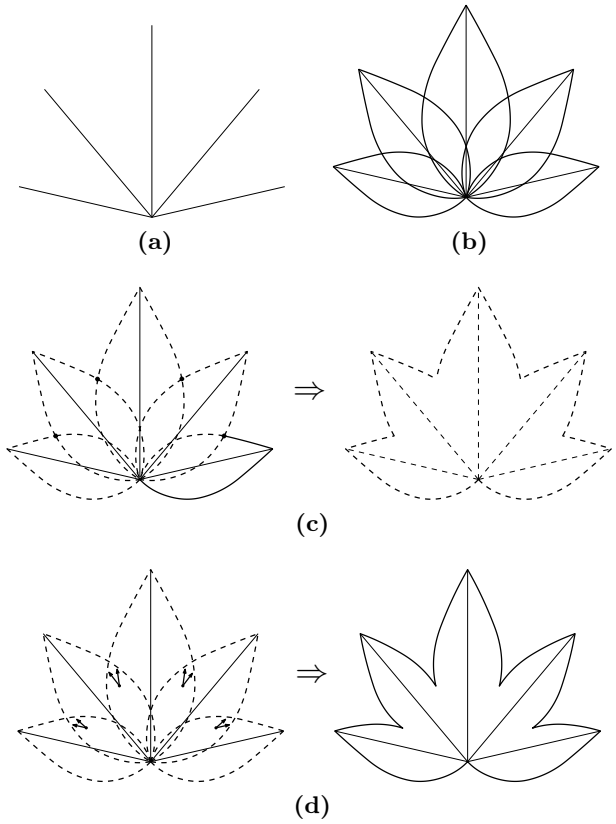


Figure 6.5: Laminar shape generation algorithm. The margin of a multilobed leaf is generated in three steps: (a) First, the venation pattern is generated. (b) Second, lobes are created along α-veins. Finally, adjacent lobes are combined to generate the margin. (c) To combine adjacent lobes, their points of intersection are computed and the margin is generated. (d) If the valley position or shape is to be modified according to user inputs, then the margin is recomputed by fitting B-spline curves.

is computed. Then, the algorithm removes the parts of the lobes that lie completely in the intersection (dashed lines in Figure 6.5c, left) to generate the leaf margin (Figure 6.5c, right).

If the parameters for adjusting the valley positions (m and ϕ) or the shape (θ) are defined by the user, then the margin must be recomputed. The algorithm first removes all lobes except the first half of the first lobe and the second half of the last lobe (dashed lines in Figure 6.5d, left). Then, the algorithm fits a B-spline curve for each half-lobe for all lobes using the method described in Section 5.3 to generate the leaf margin (Figure 6.5d, right). If the waist of a lobe lies between the valley and the apex, then a B-spline curve is fitted to three landmark points: valley, waist, and apex. Otherwise, a B-spline curve is fitted to only two landmark points: valley and apex. To visualize the curves in GUI, each B-spline is discretized into polylines with m sample points. In implementation, m is set to 64. Thus,

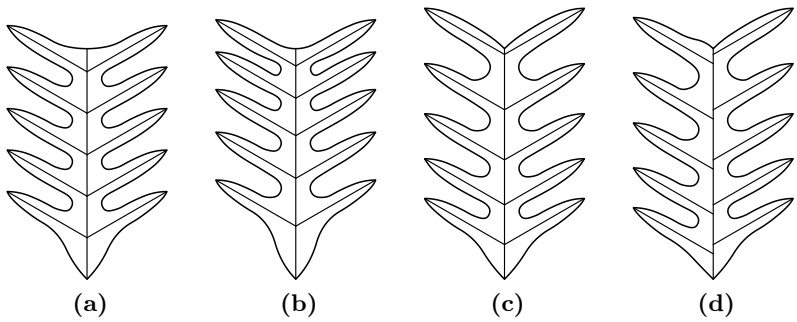


Figure 6.6: Effect of varying the initial spacing s_0 and the rate of change of spacing Δs in multilobed leaves. (a) Δs is constant, lobes are equally spaced. (b) Δs is less then zero, spacing between lobes decreases towards the apex. (c) Δs is greater then zero, spacing between lobes increases towards the apex. (d) Initial spacings s_0^l and s_0^r are different for the two side of the leaf.

each lobe of a multilobed leaf is represented by 128 points.

6.4 Analysis of Laminar Shape Generation Algorithm

Figure 6.6 illustrate the effect of varying the initial spacing s_0 and the rate of change of spacing Δs. In pinnately lobed leaves Δs controls the distance between lobes. When Δs is zero then the lobes are equally spaced (Figure 6.6a). Negative Δs decreases the distance between lobes towards the apex (Figure 6.6b) and positive Δs increases the distance (Figure 6.6c). If the initial spacing is different for the left side and the right side, then the lobes on the left side and the right side start at different points along the α_0-vein (Figure 6.6d).

Figures 6.7 and 6.8 illustrates the effect of varying the parameters of the first lobe in a multilobed leaf. As discussed in Section 5.5, the tangent angle θ_b at the base controls the shape of the base. Increasing θ_b increases the width of the base. In multilobed leaves, the effect of θ_b depends on the width of the lobes. If the lobes are wide enough to hide the base in the intersection of adjacent lobes, then θ_b has no significant effect (Figure 6.7, first row). However, if the lobes are narrow and the base is visible, then as θ_b is increased, the base becomes wider and the depths of the valleys decrease (Figure 6.7, second row). Analogously, as the x-coordinate of the waist of the first lobe increase, the lobes become broader and the valleys become less deep (Figure 6.8, first row). Increasing the y-coordinates of the waist moves the widest parts of the lobes towards the apex (Figure 6.8, second row).

Figure 6.9 illustrates the effect of varying the tangent angle θ_v at the valley to the margin.

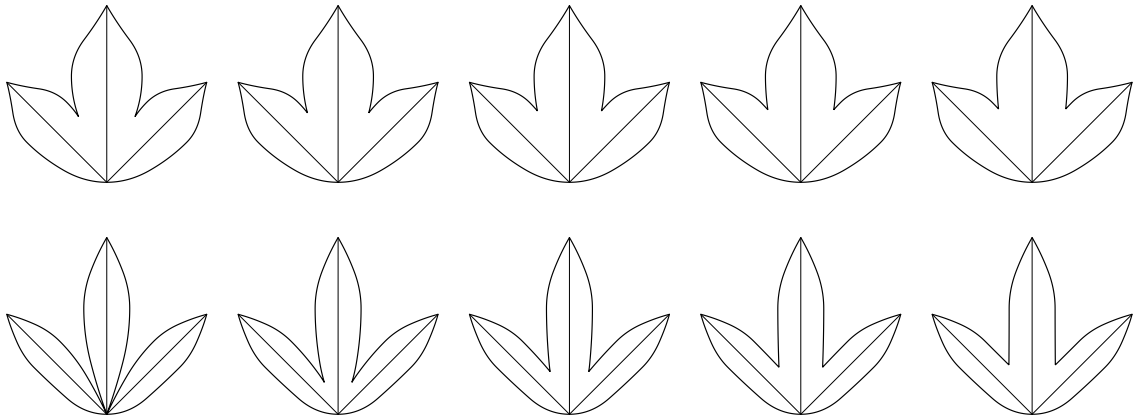


Figure 6.7: Effect of varying the tangent angle θ_b at the base to the margin of the first lobe in a multilobed leaf. (Top row) If the lobes are wider, then θ_b has no significant effect. (Bottom row) If the lobes are narrow, then as θ_b is increased the lobes become wider and depths of valleys decrease.

Increasing θ_v changes the valley shape from sharp to smooth. Figure 6.10 illustrates the effect of varying the valley orientation ϕ. As ϕ is increased the valley positions moves from the first neighboring lobe to the second neighboring lobe of the valley. Figure 6.11 illustrates the effect of varying the valley distance m. As m is increased, the valleys become less shallow.

It can be shown that the laminar shape generation algorithm is numerically stable by analyzing each step of the algorithm presented in Section 6.3. The first step generates the venation pattern using linear function, which is numerically stable. The second step generates the lobes along each α-veins using unilobed leaf shape generation algorithm, which was shown to be numerically stable except at a few isolated points (Section 5.3). Finally, the margin is generated by computing the points of intersection between adjacent lobes, which is also numerically stable. If the valley position and shape needs to be modified then a pair of B-spline curves are computed for each valley, which is also numerically stable (Section 5.3). Hence, the laminar shape generation algorithm is numerically stable.

The laminar shape generation algorithm is both time and space efficient. The time for generating the margin of a multilobed leaf with n lobes is either $O(nm^2)$ (default valley position) or $O(n)$ (valley position is specified by the user). m is the number of points used to discretize B-spline curves. With default valley positions, the algorithm needs $O(nm^2)$ time to compute the points of intersection of all pair of adjacent lobes. When the valley positions is specified by the user, the algorithm needs $O(n)$ time to fit a B-spline curves for each lobe. Since, the number of lobes in multilobed leaves is usually small ($n < 10$), the time complexity of laminar shape generation algorithm is either $O(m^2)$ or $O(1)$. The laminar

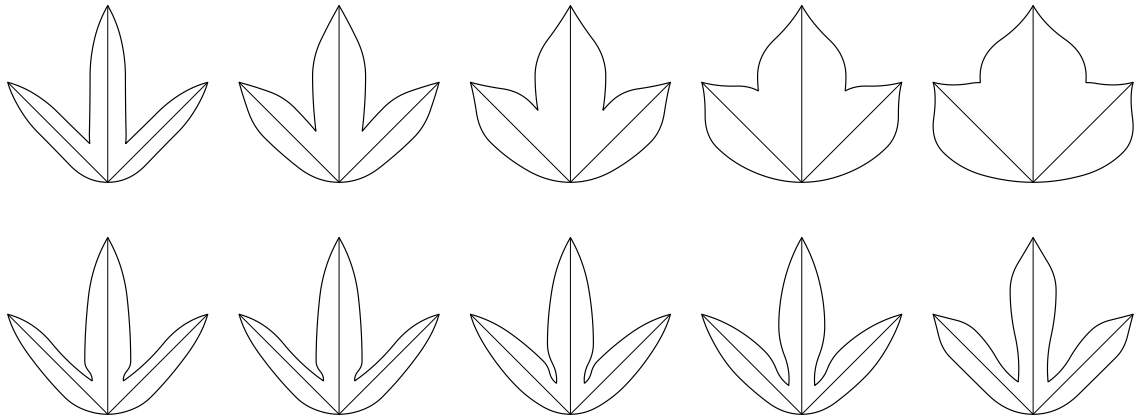


Figure 6.8: Effect of varying the waist of the lobes in a multilobed leaf. (Top row) As the x-coordinate of the waist is increased the lobes become wider. (Bottom row) As the y-coordinate of the waist is increased the widest part of lobes move towards the apex.

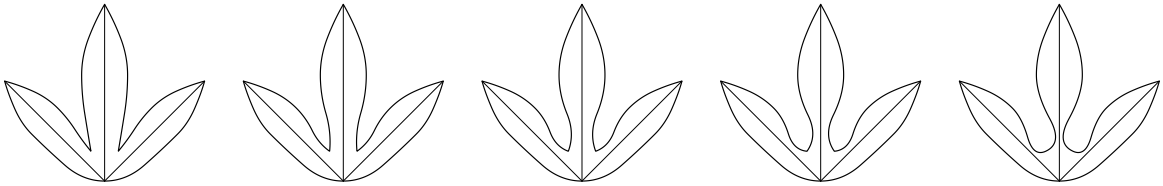


Figure 6.9: Effect of varying the tangent angle θ_v at the valley to the margin in multilobed leaves. As θ_v is increased the valley shape changes from sharp to smooth.

shape generation algorithm takes $O(nm)$ space for generating margin, which is the space required to store m points for n lobes.

6.5 Leaf Shape Generation Examples

This section illustrates sample leaf shapes generated by the algorithm. Figure 6.12 illustrates leaf shapes corresponding to those enumerated in Section 4.2.

For many applications, many leaf instances for a given kind of leaf should be generated. Laminar shape generation algorithm generates leaf instances by perturbing parameter values. The amount of perturbation to add is specified by the user and is expressed as a percentage of the parameter value. Figures 6.13 and 6.14 illustrates instances of palmately lobed and pinnately lobed leaf, respectively, generated by adding 15% perturbation to the parameter values.

On a laptop with Intel Core i7 2.0 GHz processor, the laminar shape generation algorithm

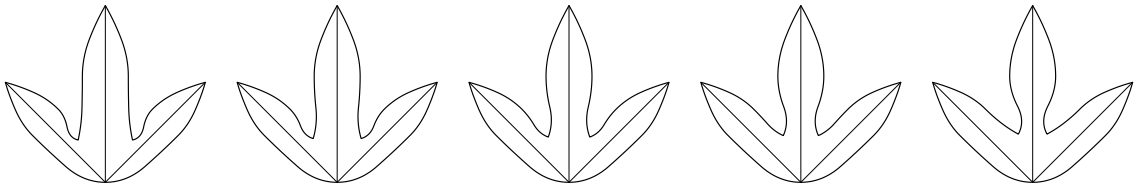


Figure 6.10: Effect of varying the valley orientation ϕ in multilobed leaves. As ϕ is increased the valley positions moves from the first neighboring lobe to the second neighboring lobe of the valley.

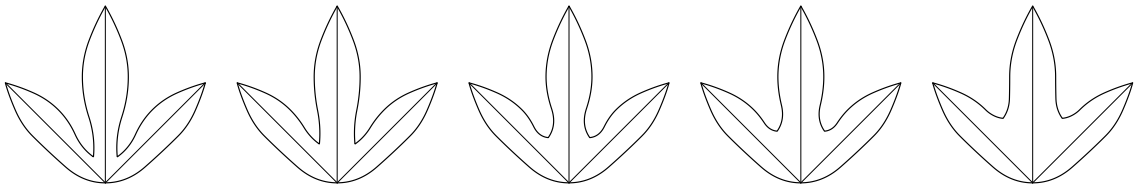


Figure 6.11: Effect of varying the valley distance m in multilobed leaves. As m is increased, the valley become shallow.

can generate on an average 400 multilobed leaves with three lobes per second and 200 multilobed leaves with seven lobes per second. This shows that the laminar shape generation algorithm is fast and can be used for quickly generating large number of leaves instances.

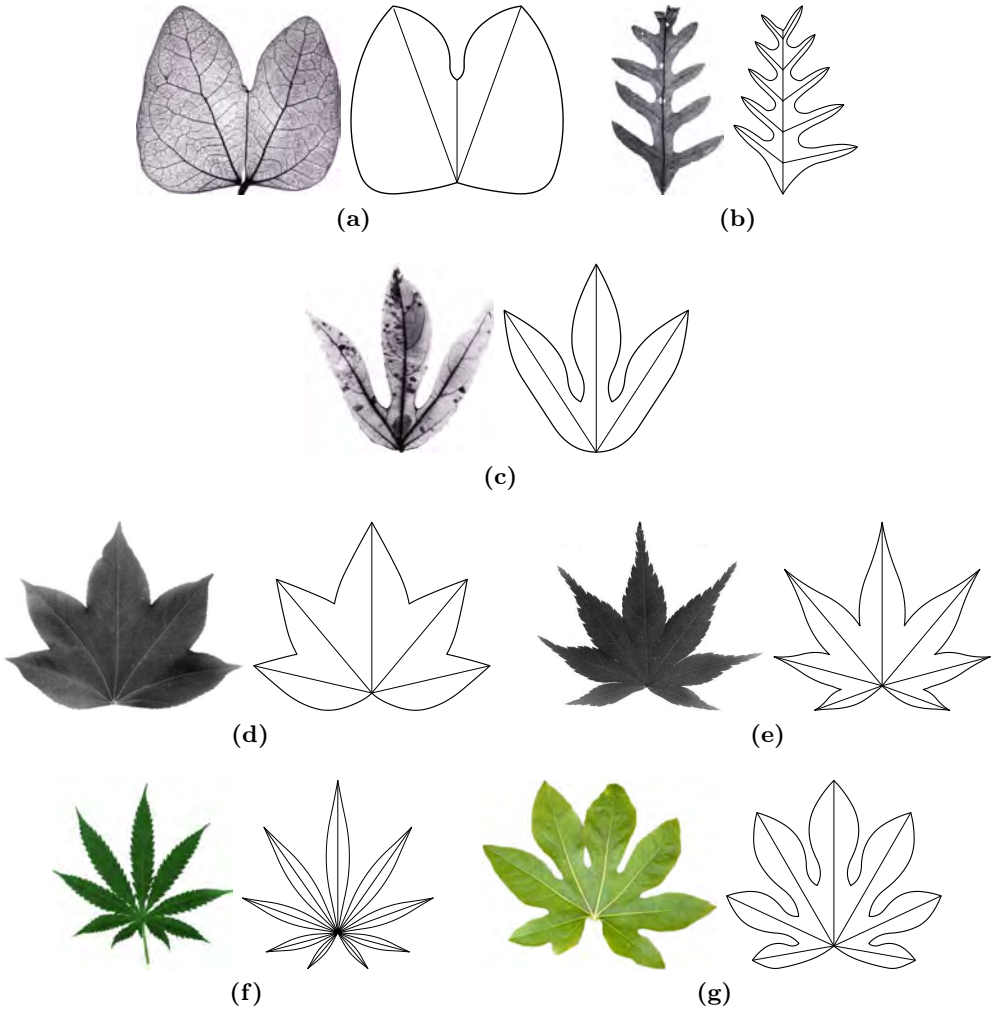


Figure 6.12: Laminar shapes generated for various multilobed leaves. (a) A leaf with two lobes. (b) A pinnately lobed leaf with 10 lobes. (c)–(g) Palmately lobed leaves.

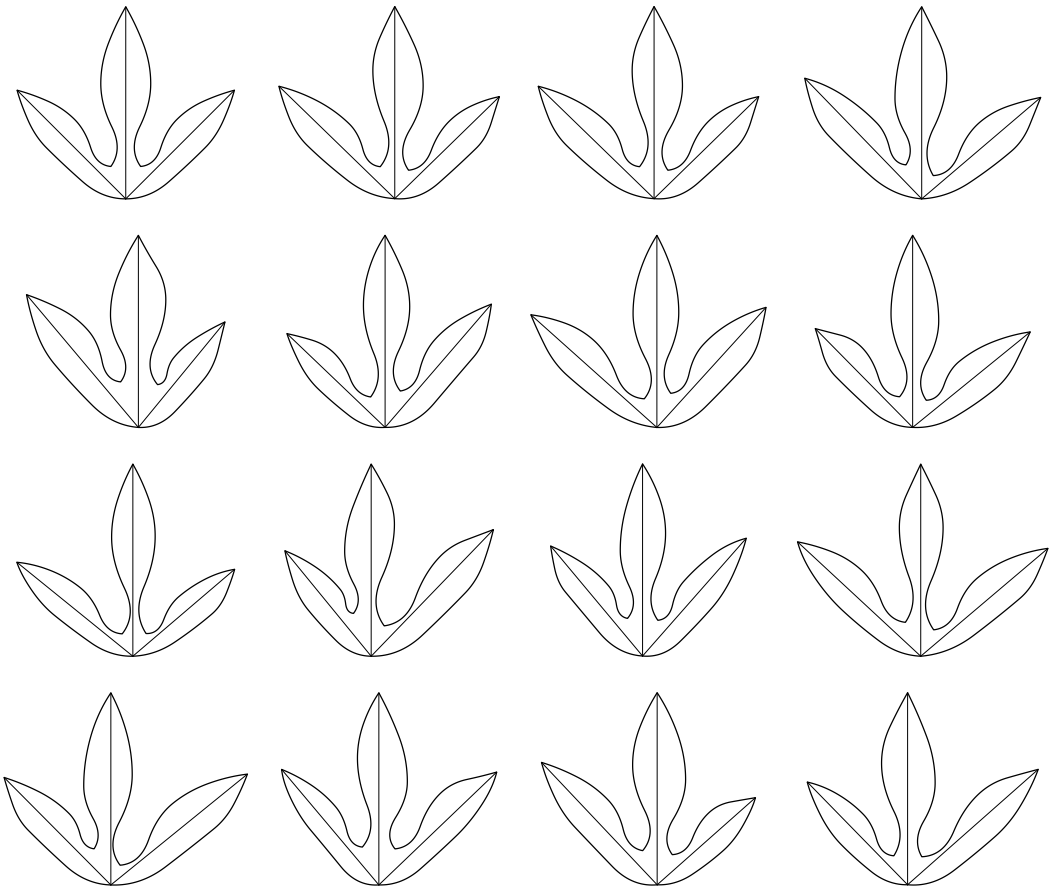


Figure 6.13: Leaf instances generated for a palmately lobed leaf. The instances are generated by adding 15% perturbations to the parameters values.

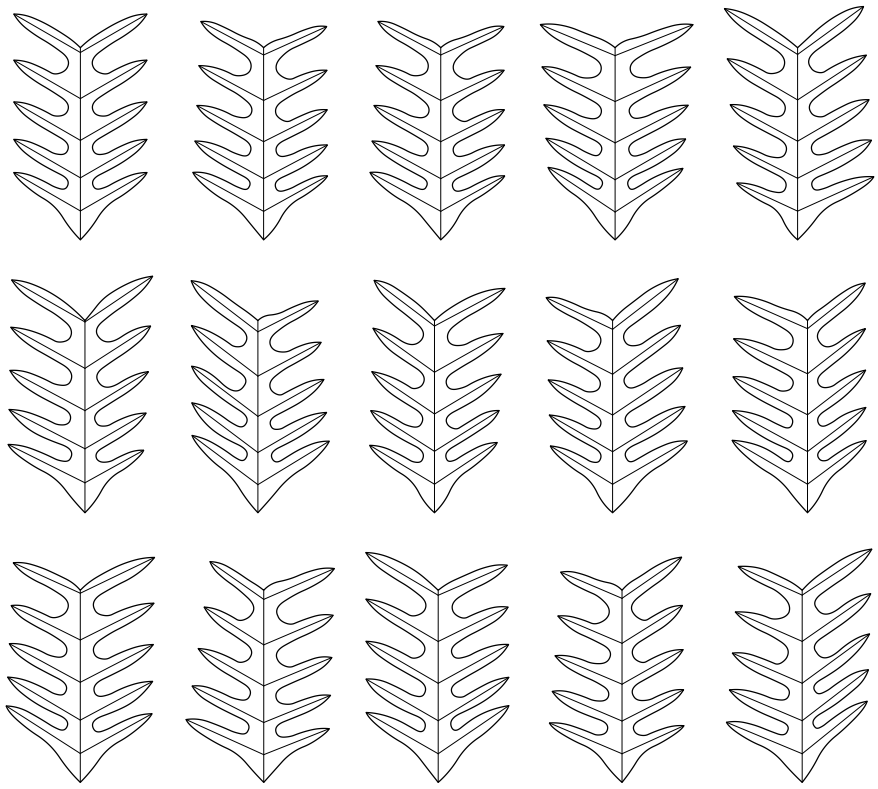


Figure 6.14: Leaf instances generated for a pinnately lobed leaf. The instances are generated by adding 15% perturbations to the parameter values.

Chapter 7

Constrained Leaf Morphing

This chapter presents a leaf morphing method for morphing leaf shapes in the parameter space of leaf shapes (leaf space). Reference leaf shapes can be easily specified by the user as soft constraints for morphing from the source to the target leaf shape (Section 7.1). Since, the number of free parameters vary for different kind of leaves, the leaf space for these leaves must be united for morphing (Section 7.2). The constrained morphing path is computed in the unified leaf space by fitting a NURBS curve over the source, the target and reference shapes (Section 7.3). Leaf space plays the central role in morphing and understanding how leaf shapes are distributed in the leaf space can be used to find interesting morphing paths (Section 7.4). Section 7.5 illustrates examples of constrained leaf morphing and leaf growth.

7.1 Overview of Leaf Morphing

Leaves generally change shapes as they grow [Mak73, MR50]. In some species, the leaf shapes can change significantly from one type to the other [MR50]. 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, the proposed method models leaf shapes by shape parameters. So, shape change can be easily measured in the parameter space of leaf shapes, called leaf space. Each point in

the leaf space represents a leaf shape. 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.

Without loss of generality, morphing of symmetric leaf shapes is discussed. The same method can be applied to morphing asymmetric leaf shapes with a doubling of the dimensionality of the leaf space.

7.2 Unification of Leaf Spaces

As discussed in Sections 5.1 and 6.1, the number of free parameters vary for different types of leaves. So, when the source, target, and reference shapes have different degrees of freedom, they have to be mapped into a *unified leaf space* before morphing can proceed. The mapping among various types of unilobed leaves is discussed in Section 7.2.1. The mapping between unilobed and multilobed leaves is discussed in Section 7.2.2.

7.2.1 Unilobed Leaves

As discussed in Section 5.1, depending on whether a leaf shape has basal and apical extensions, the number of free parameters for a (symmetric) unilobed leaf is either 4 (no extension), 6 (either basal or apical extension), or 8 (both extensions). Thus, when morphing from one unilobed leaf type to another unilobed leaf type they should be mapped to a unified leaf space.

Let S and D denote two consecutive shapes in the morphing sequence. Without loss of generality, suppose S has no basal extension and D has basal extension. Then, S and D 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 D, i.e., base to base, waist to waist, and apex to apex. The tail $\mathbf{p}'_t$ of D 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 D:

$$\frac{L(\mathbf{p}, \mathbf{p}_w)}{L(\mathbf{p}, \mathbf{p}_b)} = \frac{L(\mathbf{p}'_t, \mathbf{p}'_w)}{L(\mathbf{p}'_t, \mathbf{p}'_b)} \tag{7.1}$$

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 Equation 5.7. 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 D can be mapped to two shape points in the 7-D leaf space.

Analogous mapping can be applied for D with apical extensions. When D 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 D have different types of extensions, then an additional landmark point is added to both S and D to map their extension points to the others. This process also raises the dimensionality of the unified leaf space to 10.

7.2.2 Multilobed Leaves

Let S and D denote two consecutive shapes in the morphing sequence. Without loss of generality, suppose S is a unilobed leaf shape and D is a multilobed leaf shape. Then, S and D are mapped into a unified leaf space by estimating the parameters of a multilobed leaf S', such that the difference in the leaf shapes S and S' is minimized. One way to estimate the parameters is to use non-linear optimization, which is computationally expensive and might get stuck in local minimum. However, since the shape and structure of multilobed leaves is well defined and known a priori, the parameters of S' can be estimated directly from S and D (Figure 7.1).

The parameters of multilobed leaves are divided into three groups: parameters for specifying α-veins, parameters for specifying the position and shape of valleys, and parameters for specifying lobe shapes. The number of lobes N, initial spacing s_0, rate of change of spacing Δs, and angles θ_1 and θ_n of S' are set to those of D. The lengths l_i of α-veins are estimated by intersecting rays c_i with the margin of S. The origin of the ray c_i is set to the origin $\mathbf{r}_i$ of the α-vein α_i in D and the ray is parallel to $\mathbf{e}_i - \mathbf{r}_i$, where $\mathbf{e}_i$ is the end-point of the α-vein α_i in D (Figure 7.1b). The points of intersection of c_i with the margin of S are the apexes $\mathbf{A}_i$ of lobes.

The position $\mathbf{V}_i$ of valleys in S' are set to the mid-point, along the margin of S, between two adjacent apexes $\mathbf{A}_i$ and $\mathbf{A}_{i+1}$. The angles θ_v at valleys are set to the angles between tangent vectors to the margin of S at $\mathbf{V}_i$ and the vectors $\mathbf{h}_i$. $\mathbf{h}_i$ is the vector from the average of the origin of adjacent α-veins to the position of the valley $\mathbf{V}_i$ (Figure 7.1b).

The shape of lobes in multilobed leaves are defined by three parameters: tangent angle at the base θ_b, tangent angle at the apex θ_a, and the waist $\mathbf{W}$. As discussed in Section 6.3, when the valley parameters are specified then the θ_b is not used. So, θ_a and $\mathbf{W}$ are the only two free parameters for lobe shapes. θ_a is set to the angle between tangent to the margin of

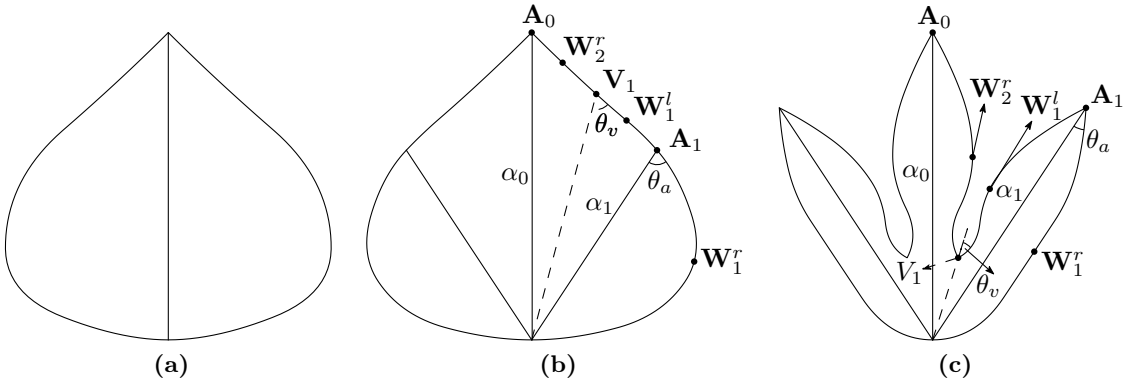


Figure 7.1: Mapping a unilobed leaf to a multilobed leaf. (a) The source unilobed leaf. (b) The mapping of the source unilobed leaf to the destination multilobed leaf. (c) The destination multilobed leaf.

S at the apex $\mathbf{A}_i$ and the α-vein α_i. The waist $\mathbf{W}$ for lobes is set to the mid-point, along the margin of S, between the adjacent valley $\mathbf{V}_i$ and the apex $\mathbf{A}_i$.

7.3 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, $1 \leqslant k \leqslant n$, 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:

$$\mathbf{s}(u) = \frac{\sum_{k=0}^{n+1} w_k B_{k,2}(u)\mathbf{R}_k}{\sum_{k=0}^{n+1} w_k B_{k,2}(u)} \tag{7.2}$$

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 5.3. 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.

7.4 Visualizing Leaf Space

Leaf space is essential for morphing as it allows leaf shapes with different number of parameters to be morphed to each other by mapping them into a unified leaf space. Thus, it is important to visualize the leaf space to understand the distribution of leaf shapes in the leaf space. This can be used to find interesting morphing paths. For example, to model leaf growth it is necessary that all intermediate leaf shapes generated by morphing are real and exist in nature. However, as discussed in Section 5.5, not all sets of parameter values generate real leaf shapes which exist in nature. Thus, the proposed leaf morphing algorithm cannot guarantee that all intermediate leaf shapes are real due to the presence of non-real leaf shapes in the leaf space. It is possible for the morphing path to go through the non-real leaf shapes.

As discussed in the last section, different kinds of leaves are defined in different leaf spaces. For ease of explanation, this section discusses methods for visualizing unilobed leaves with no extension as they have the least number of parameters. Leaf spaces for other leaf shapes can be analyzed similarly. The leaf space for unilobed leaves with no extension has four parameters: tangent angle at the base and the apex, and x and y coordinates of the waist.

To visualize the leaf space, ten points were uniformly sampled for each parameter dimension of the leaf space. Thus, 10,000 points were sampled from the entire leaf space. Then, for

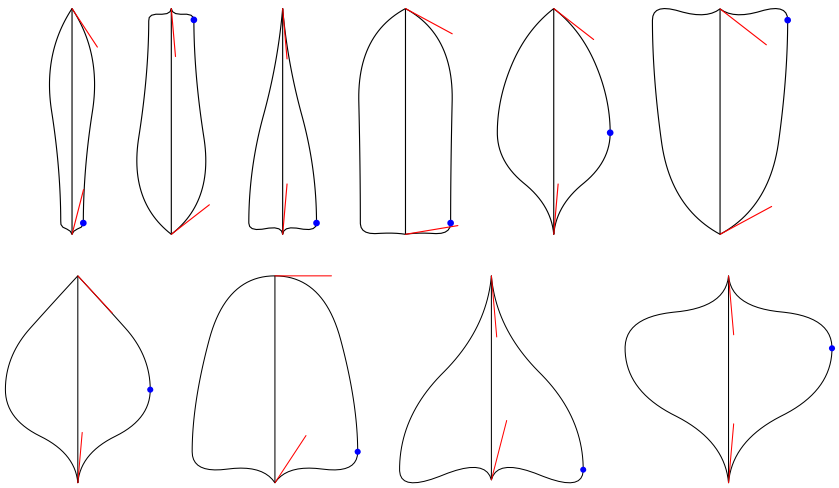


Figure 7.2: Examples of non-real leaf shapes generated for visualizing leaf space. The lines (tangents at base and apex) and the circle (waist) indicate the parameters used to generate the leaf shape.

each sampled point in the leaf space, an instance of unilobed leaf was generated and saved as an image. Finally, each image was manually marked as either real or non-real leaf shape. It was found that out of 10,000 leaf shapes, 5,008 were real and 4,992 were non-real leaf shapes. Figure 7.2 illustrates some examples of the generated non-real leaf shapes.

It is not possible to directly visualize the 4D leaf space to determine how leaf shapes are distributed in the leaf space. In this thesis, the 4D leaf space is visualized by using 3D subspaces of the leaf space. Since, each parameter of the leaf shape is sampled with ten points, there is a total of 40 3D subspaces. Each 3D subspace consists of 1,000 points and each point is labeled with -1 (non-real leaf shape) or $+1$ (real leaf shape). Thus, a 3D subspace forms an implicit function and can be visualized as a level set surface. At the level of 0, the level set surface defines the boundary between real and non-real leaf shapes. With this boundary, the leaf morphing algorithm can ensure that all intermediate leaf shapes are real by finding a path in the leaf space that does not cross any boundary level set surface between real and non-real leaf shapes. Figure 7.3 illustrates 10 3D subspaces of the leaf space at a particular tangent angle at the base. The surface indicates the boundary between the real and non-real leaf shapes. A sharp boundary is illustrated only for visualization. In principle, the boundary is fuzzy as there is a smooth shape transition from real to non-real leaf shapes (Figure 7.4).

7.5 Leaf Morphing Examples

Figure 7.5 compares linear morphing with the proposed nonlinear morphing. The results show that shape change is more smooth with nonlinear morphing.

Figure 7.6 illustrates examples of constrained leaf morphing. The first shape was morphed to the last shape, constrained by the underlined shapes. 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.

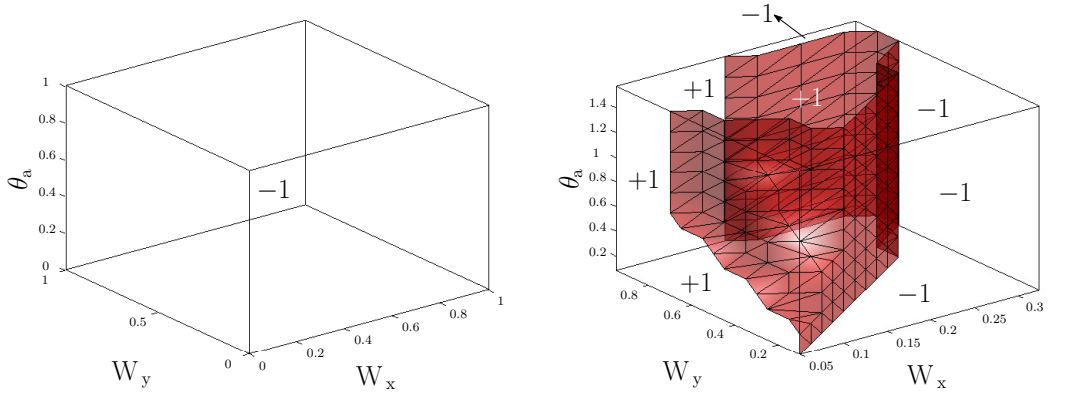
Unification of parameter space of different kind of leaf shapes allows to morph not only between unilobed leaf shapes but also from unilobed to multilobed leaf shapes and from multilobed to multilobed leaf shapes. Figures 7.7a and 7.7b illustrate examples of morphing from unilobed leaf shapes to multilobed leaf shapes. The first and last leaf shapes are the source and target shapes. The source unilobed leaf shape is first mapped to the parameter space of the target multilobed leaf shape (t=0). Then, the mapped source shape is morphed linearly to the target shape in the space of the target multilobed leaf shape.

Figure 7.8 illustrates an example of constrained morphing from a multilobed leaf shape with three lobes to a multilobed leaf shape with seven lobes, constrained by a unilobed leaf shape with basal extension. The source multilobed leaf shape and the constraint unilobed leaf shape are first mapped to the parameter space of the target multilobed leaf. Then, the mapped source shape, the mapped constraint shape and the target shape are nonlinearly morphed in the space of multilobed leaves of the target shape. Figure 7.9 illustrates an example of linear morphing from the source to the target shapes from the previous example without any constraint. The ability to morph any two kinds of leaf shapes by mapping them to a common parameter space is possible because of the simplicity of the leaf model. With a complex leaf model, it might not be easy to morph arbitrary leaf shapes.

For simplicity, Section 7.2 discussed the unification of leaf spaces for symmetric leaves. However, by doubling the dimensionality of the leaf space, asymmetric leaves can be morphed. Figure 7.10 illustrates the linear morphing of a unilobed leaf shape with basal extension on the left to a unilobed leaf shape with basal extension on the right.

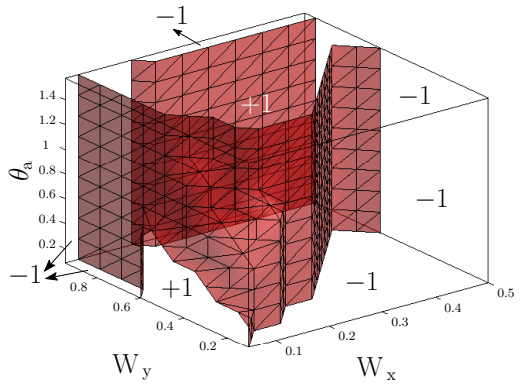
Figure 7.11 illustrates examples of constrained leaf morphing for simulating the growth of

real leaf [MR50]. 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.

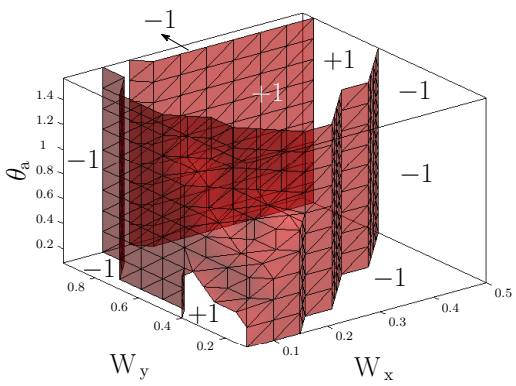


$\theta_b = 0.087$

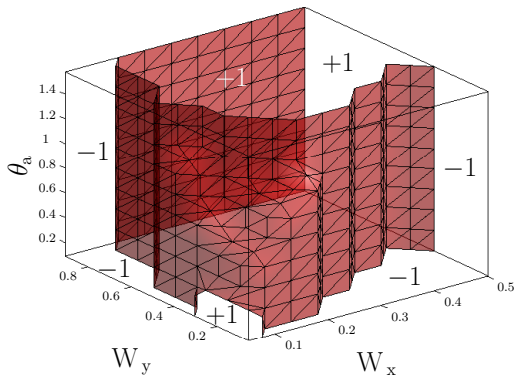
$\theta_b = 0.252$



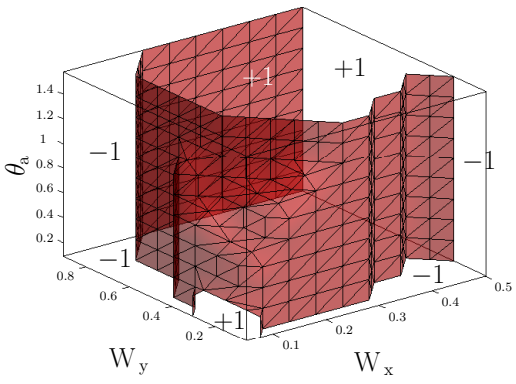
$\theta_b = 0.417$



$\theta_b = 0.582$



$\theta_b = 0.747$



$\theta_b = 0.911$

Figure 7.3: 3D subspaces of the leaf space with constant tangent angle at the base. Each subspace is visualized by the boundary between real and non-real leaves (red surface). The first subspace contains only non-real leaf shapes.

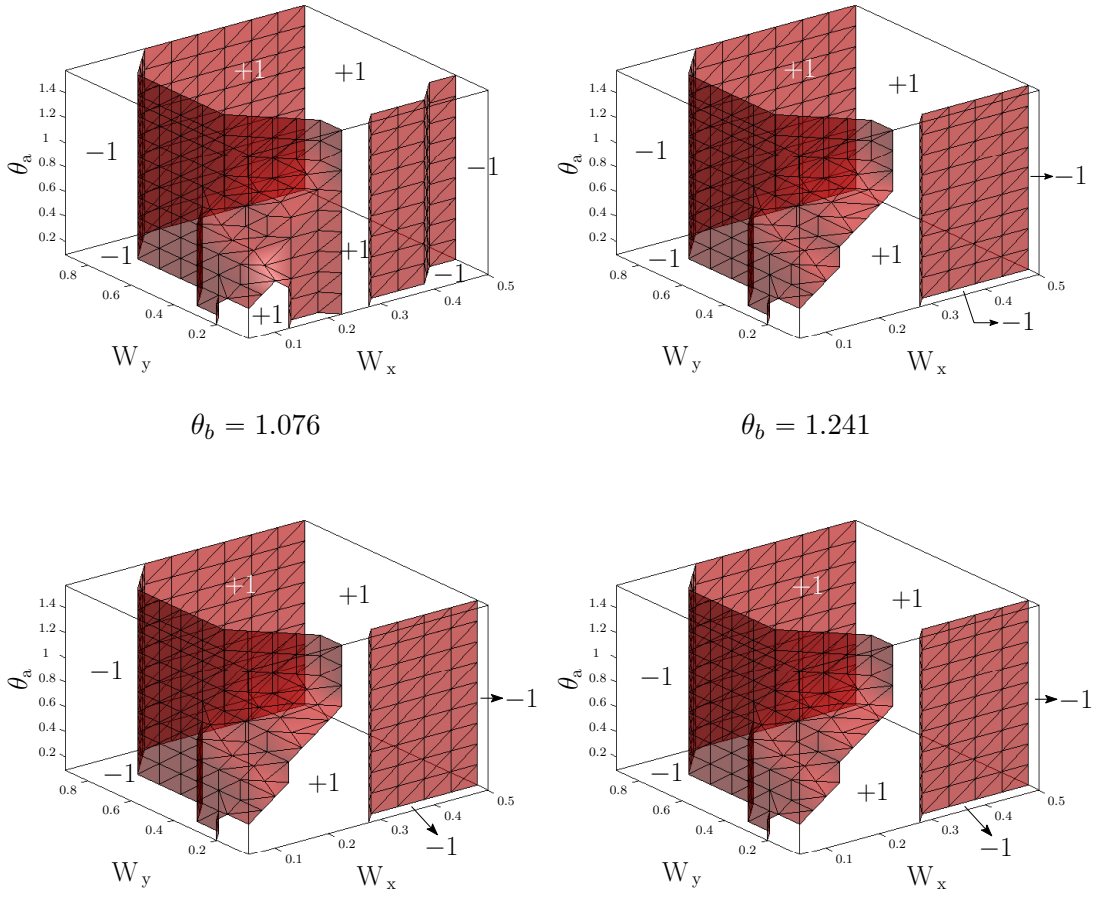


Figure 7.3 (continued): 3D subspaces of the leaf space with constant tangent angle at the base. Each subspace is visualized by the boundary between real and non-real leaves (red surface). The first subspace contains only non-real leaf shapes.

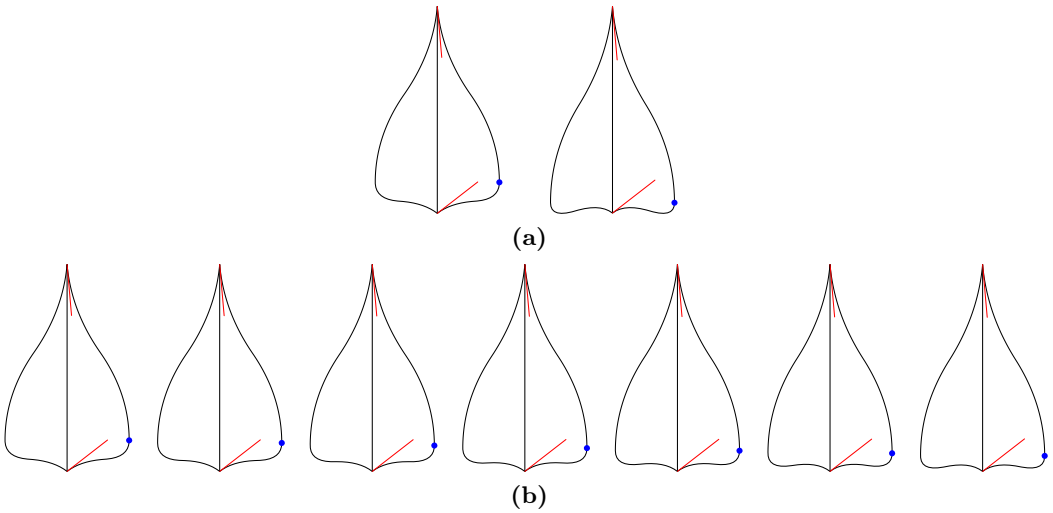


Figure 7.4: Fuzzy boundary between real and non-real leaf shapes. (a) A sample real leaf shape (left) and non-real leaf shape (right) adjacent to it in the sampled leaf space. (b) Leaf shapes generated between the real and non-real leaf shapes. Boundary between real and non-real leaf shapes is fuzzy as the leaf shapes changes smoothly from real to non-real.

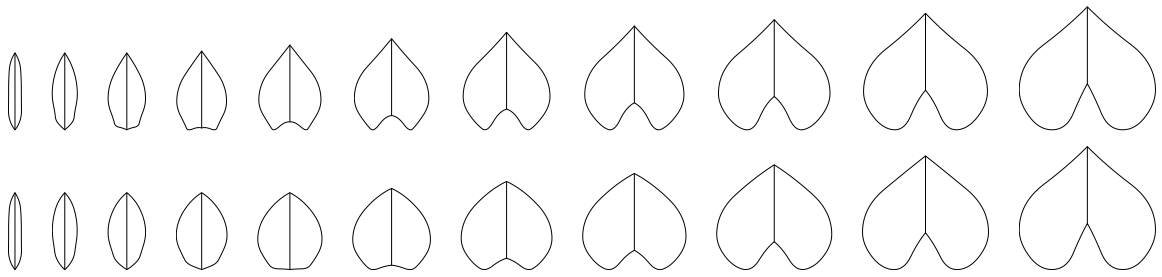
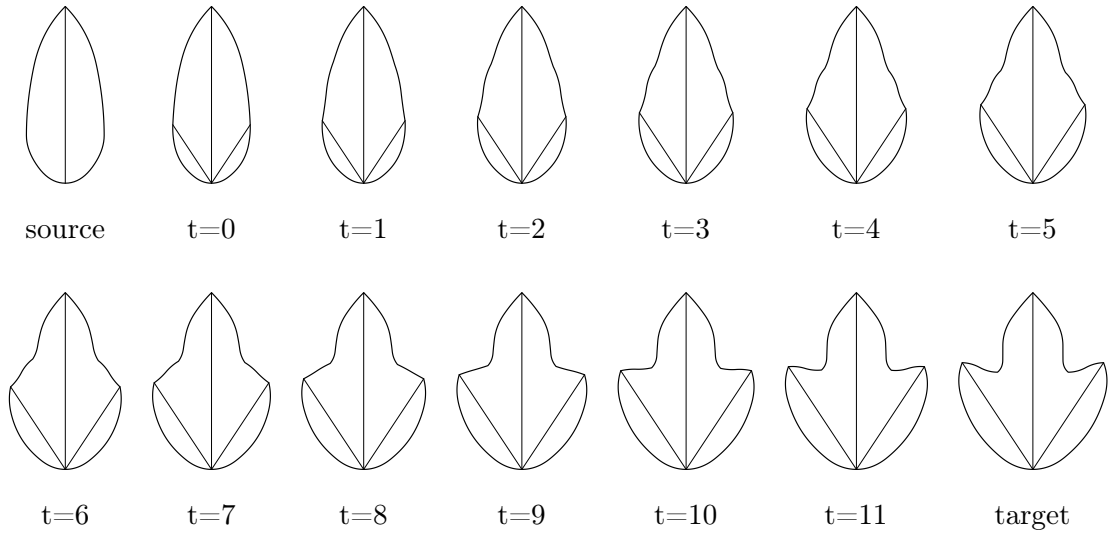
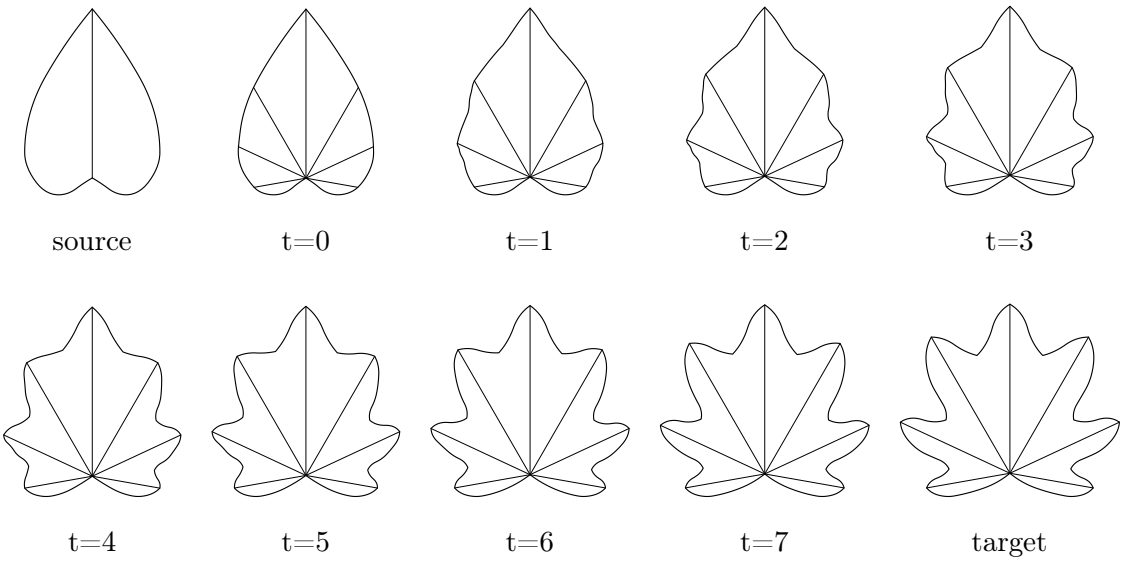


Figure 7.5: Comparison of linear morphing with proposed nonlinear morphing. Nonlinear morphing (Row 2) produces smoother shape change than linear morphing (Row 1).

Figure 7.6: 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.



(a)



(b)

Figure 7.7: Leaf morphing from unilobed leaf shapes to a multilobed leaf shapes. At t=0, the source leaf shape is mapped to the parameter space of the target leaf shape. Then, the mapped source shape is linearly morphed to the target shape in the parameter space of the target shape. (a) Morphing from unilobed leaf shape without any extension to a multilobed leaf shape with three lobes. (b). Morphing from unilobed leaf shape with basal extension to a multilobed leaf shape with seven lobes

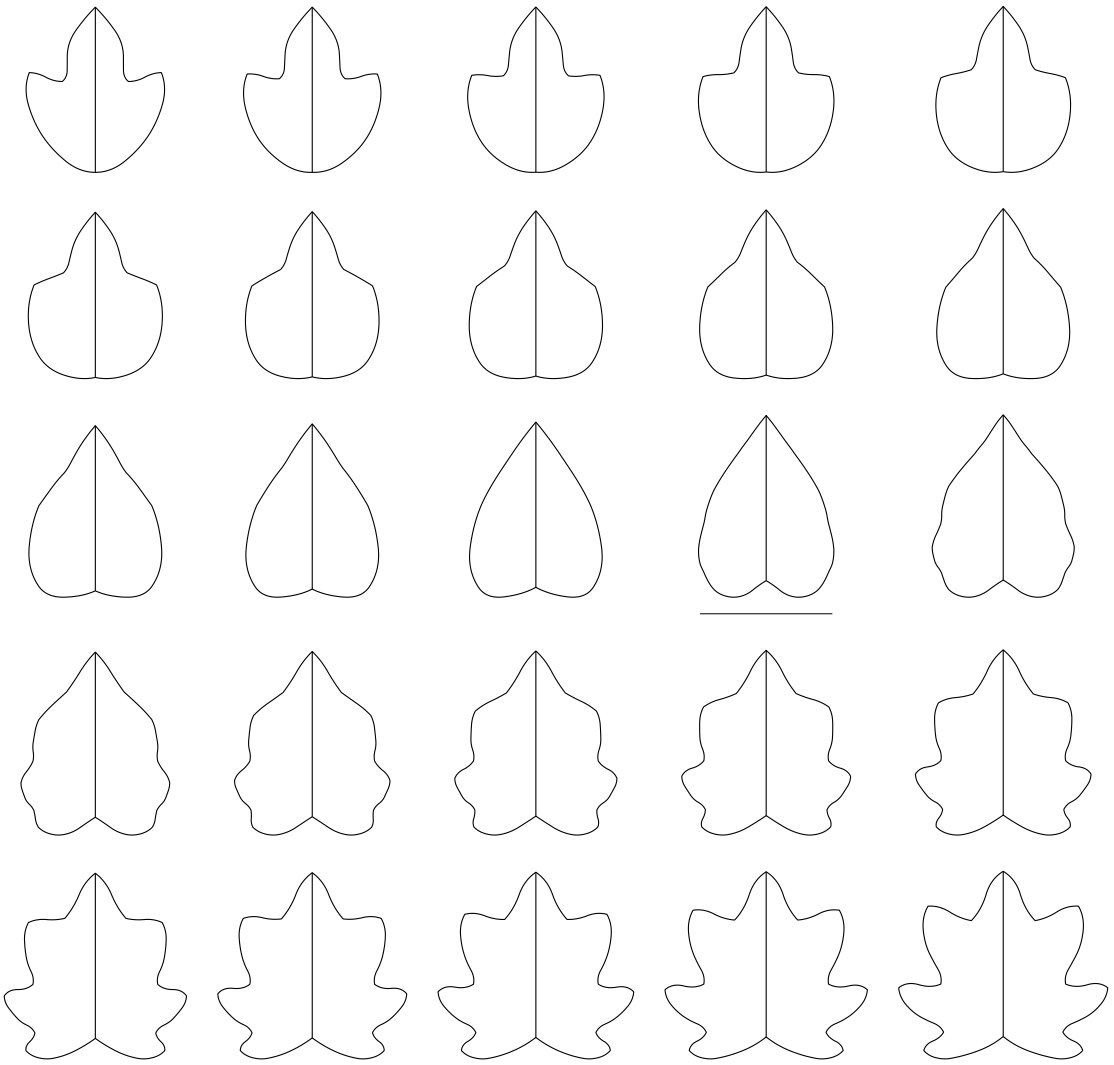


Figure 7.8: Constrained leaf morphing from a multilobed leaf with three lobes to a multilobed leaf with seven lobes. The source multilobed leaf (first row, first leaf) and the constraint (underlined) are mapped in the space of target multilobed leaf.

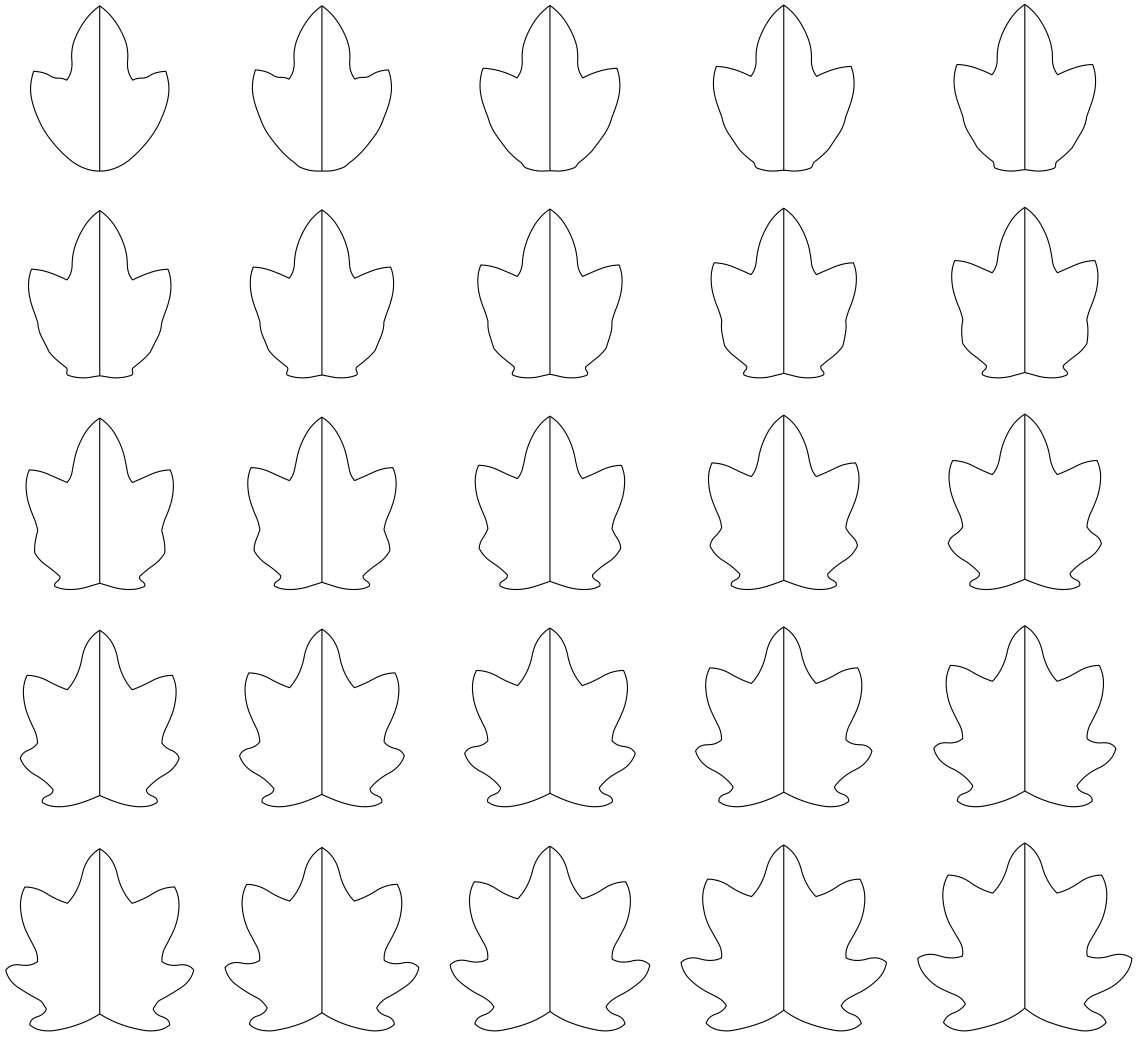


Figure 7.9: Leaf morphing without constraint from a multilobed leaf with three lobes to a multilobed leaf with seven lobes. The source multilobed leaf (first row, first leaf) is mapped in the space of target multilobed leaf.

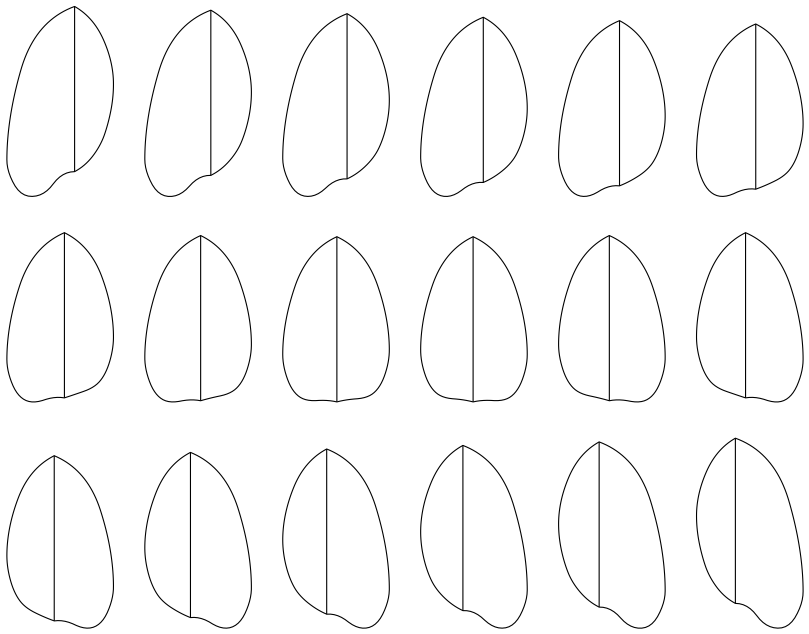


Figure 7.10: Morphing asymmetric leaves. Morphing a unilobed leaf shape with basal extension on the left to a unilobed leaf shape with basal extension on the right.

Figure 7.11: Modeling leaf growth using constrained leaf morphing. (Row 1) Real leaves. (Rows 2–4) Morphing with 0, 1, and 2 constraints (underlined).

Chapter 8

Future Work

The leaf model presented in this thesis can be extended in several ways.

8.1 Automatic Estimation of Model Parameters

This thesis illustrates a GUI for the user to interactively specify the parameters of the leaf model. This step can be automated so that generating instances of a known leaf is as simple as taking a photograph of the leaf and running a software program. One way to solve this problem is to use a computer vision algorithm for first extracting the margin of the leaf from the image. Then, the margin can be analyzed to automatically detect the landmark points and their tangents. To detect teeth, the algorithm can analyze the frequency of change of gradient along the margin. High-frequency changes correspond to the present of teeth and can be filtered out to yield smooth envelope of the margin without teeth.

8.2 Modeling Laminar Warping and Aging

The laminar shape generation algorithm presented in this thesis generates flat leaf shapes. However, leaves naturally warp in 3D because of differential growth of the lamina. Laminar warping can be accomplished in two steps. First, the veins can be warped according to user specified constraints. Then, the laminar surface can be warped so that the warped veins lay in the warped laminar surface. The laminar surface can be warped using free-form deformations [MMPP03], harmonic interpolation [HSB05], skeleton-based deformations [LGZL08, LZG09], and physics-based deformations [GHDS03, BWH+06, GGWZ07].

As a leaf ages, it changes color and in general starts to wrinkle with decreasing moisture content. The wrinkles in an aging leaf can be modeled as laminar warping.

8.3 Ornamentation

Teeth, drip-tips, and venation pattern are necessary for realistic visualization of leaves. In this thesis, teeth and drip-tips are not modeled, as they don't contribute significantly to the overall shape of the leaves. Teeth can be added to the leaf margin by using methods such as curve analogies [HOCS02]. Drip-tips can be modeled by adding a landmark point at the point of inflection on the leaf margin caused by the drip-tip.

The proposed leaf model for multilobed leaves uses α-veins for defining the placement, orientations and lengths of the lobes. The same idea can be extended to model the secondary and higher-order veins. The secondary veins can be defined by their placement and orientation relative to the α-veins using Equations 6.4 and 6.7. The lengths of the secondary veins can be computed by intersecting them with the margin. Higher-order veins can be defined similarly, by placing them along the secondary veins. This algorithm produces venation pattern with straight veins. Such a venation pattern is, in general, sufficient for modeling laminar warping and aging (Section 8.2). Realistic venation pattern for visualization can be generated using existing methods such as [RFL+05, JGZ09].

8.4 Compound and Narrow Leaves

In compound leaves, the lamina is divided into a number of smaller parts called leaflets. Compound leaves can be generated by modeling each leaflet as a unilobed leaf and then placing them along the petiole.

Based on the shape of lamina, there are two kinds of leaves: broad leaves and narrow leaves. This thesis presented a procedural model for generating broad leaves. Narrow leaves have 3D structure and cannot be modeled using the same method as broad leaves. There are two types of narrow leaves: needles-like leaves and scale-like leaves. Needle-like narrow leaves have long cylindrical tubes emanating from a common base. These leaves can be modeled by generalized cylinders. Scale-like narrow leaves have small leaflets wrapped around tree-like structures. These leaves can be generated by modeling each leaflet as unilobed leaves and then placing them appropriately.

8.5 Modeling Laminar Deformation

One of the important application of the leaf model is to simulate the interaction of plants with the environment, for example when a rain drop hits the surface of a leaf. Thus, it is necessary to extend the leaf model to include physical properties for physically accurate

simulation of deformation. There are many existing methods for simulating the interaction of a deformable object with other objects such as mass spring models [EWS96, BW98, EGS03], finite element methods [CC91, JP99], and thin shell models [GHDS03, BWH+06, GGWZ07].

These methods are very general and tend to be computationally expensive. The Cosserat tree model proposed by Li Hao [Hao10] seems more appropriate because it is physically correct and computationally efficient. Leaves generally deform, curl, and twist relative to its primary and major secondary veins. Thus, one can use a hybrid model [HLC10] to bind a Cosserat tree to the laminar surface. The Cosserat tree models the veins of the leaf and surface mesh models the surface details of the laminar surface.

Chapter 9

Conclusions

In this thesis, a novel procedural leaf model for generating a wide variety of leaves was developed. Leaf modeling is a difficult and challenging problem because there is a huge variation in shape and structure of leaves. Thus, one of the important tasks in developing a general leaf model is to find the geometric features common to all leaves. Based on botanical literature, all possible leaf shapes were characterized and enumerated. Then, based on the computational requirements, leaves were divided into two broad categories: unilobed, and multilobed leaves. The proposed leaf model is based on the unilobed leaves. The shapes of unilobed leaves is represented by a set of landmark points on the margin and tangents to the margin at these points. The shapes of multilobed leaves is represented by a combination of unilobed leaves, one leaf for each lobe.

For the leaf model to be useful in a wide variety of applications, it should have the following properties: general, intuitive, concise, generative, and numerically stable. It was shown that the leaf model can generate all possible enumerated leaf shapes. It was also shown that the generated leaf shapes match those of real leaves very well. Thus, the proposed leaf model is general. Since, the parameters of the model are based on the landmark points, they have geometric meaning and the user can easily visualize the leaf shape that will be generated from the parameter values. Thus, the leaf model is intuitive to use. The leaf model is concise as the parameters are independent of each other. It was also shown that the leaf model can generate multiple instances of a leaf, each having the same overall shape but differs in details, by perturbing the parameter values. Finally, the leaf model was shown to be numerically stable, which ensures that a small change in parameter values produces a small change in the generated leaf shape.

This thesis also developed a morphing algorithm that performs constrained leaf morphing in a unified leaf space. The proposed morphing algorithm 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. Morphing in unified leaf shape has two advantages: First, it is possible to

morph between any two leaf shapes. Thus, the morphing algorithm is general. The ability to morph between any two leaf shapes by mapping them to a common parameter space is possible because of the simplicity of the leaf model. Second, in the unified leaf space, the correspondence between two leaf shapes can be computed automatically by matching their corresponding landmarks. Thus, there is no need for the user to manually establish correspondence between two leaf shapes.

In conclusion, this thesis has made the following contributions:

- Design of a leaf model for intuitively specifying the geometric shapes of a wide variety of leaves.
- Development of an efficient algorithm for creating instances of various kinds of leaves.
- Development of an algorithm for constrained morphing of leaf shapes in the unified parametric leaf space.

References

[APS09] Fabricio Anastacio, Przemyslaw Prusinkiewicz, and Mario Costa Sousa. Sketch-based parameterization of L-systems using illustration-inspired construction lines and depth modulation. *Computers & Graphics*, 33:440–451, 2009.

[Arm10] W. P. Armstrong. Major divisions of life. http://waynesword.palomar.edu/trmar99.htm, September 2010.

[ASSJ06] Fabricio Anastacio, Mario Costa Sousa, Faramarz Samavati, and Joaquim 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.

[BAF+03] C.J. 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.

[BLEG+11] R Barillot, G Louarn, AJ Escobar-Gutiérrez, P Huynh, and D Combes. How good is the turbid medium-based approach for accounting for light partitioning in contrasted grass–legume intercropping systems? *Annals of Botany*, 108(6):1013–1024, 2011.

[Bon] Nicolas Bonneel. Treegenerator. http://www.treegenerator.com.

[BPF+03] Frederic Boudon, Przemyslaw Prusinkiewicz, Pavol Federl, Christophe Godin, and Radoslaw Karwowski. Interactive design of bonsai tree models. *Computer Graphics Forum*, 22(3):591–599, 2003.

[Bre10] Pat Breen. Plant identification: examining leaves. http://oregonstate.edu/dept/ldplants/Plant%20ID-Leaves.htm, September 2010.

[Bro10] C. Frank Brockman. Broadleaved trees of yosemite national park. http://www.yosemite.ca.us/library/broadleaved_trees/field_key.html, September 2010.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[BWH+06] Miklós Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. A quadratic bending model for inextensible surfaces. In *Fourth Eurographics Symposium on Geometry Processing*, pages 227–230, 2006.

[CC91] Laurent D. Cohen and Isaac Cohen. Finite element methods for active contour models and balloons for 2D and 3D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1991.

[CCS+07] Didier Combes, Michaël Chelle, Hervé Sinoquet, Abraham Escobar-Gutiérrez, and Claude Varlet-Grancher. Evaluation of a turbid medium model to simulate light interception by plant canopies at three spatial scales. In Przemyslaw Prusinkiewicz, Jim Hanan, and Brendan Lane, editors, *Proceedings of the 5th International Workshop on Functions-Structural Plant Models*, 2007.

[CEC+07] Michaël Chelle, Jochem B Evers, Didier Combes, Claude Varlet-Grancher, Jan Vos, and Bruno Andrieu. Simulation of the three-dimensional distribution of the red:far-red ratio within crop canopies. *New Phytologist*, 176(1):223–234, 2007.

[cma10] cmassengale. Introduction to the plant kingdom. http://biologyjunction.com/introduction%20to%20plants.ppt, September 2010.

[CNX+08] Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based tree modeling using markov random field. In *ACM SIGGRAPH Asia 2008*, pages 109:1–109:9, 2008.

[Cre] The Game Creators. Treemagik G3. http://www.thegamecreators.com/?m=view_product&id=2087.

[Cum10] Candace Cummings. Terminology - leaf, twig, and fruit characteristics used in tree identification. http://www.clemson.edu/extension/natural_resources/landowner/youth_environ_education/terminology.html, September 2010.

[EDH+09] Beth Ellis, Douglas C. Daly, Leo J. Hickey, Kirk R. Johnson, John D. Mitchell, Peter Wilf, and Scott Wing. *Manual of leaf architecture*. Cornell University Press, 2009.

[EGS03] Olaf Etzmuss, Joachim Gross, and Wolfgang Strasser. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):538–550, 2003.

[EWS96] B. Eberhardt, Andreas Weber, and W. Strasser. A fast, flexible particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–59, 1996.

[GGWZ07] Akash Garg, Eitan Grinspun, Max Wardetzky, and Denis Zorin. Cubic shells. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 91–98, 2007.

[GHDS03] Eitan Grinspun, Anil Hirani, Mathieu Desbrun, and Peter Schröder. Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 62–67, 2003.

[GK08] Björn Ganster and Reinhard 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.

[GP04a] Ben Gorte and Norbert Pfeifer. 3D image processing to reconstruct trees from laser scans. In *Proceedings of the 10th annual conference of the Advanced School for Computing and Imaging*, 2004.

[GP04b] Ben Gorte and Norbert Pfeifer. Structuring laser-scanned trees using 3D mathematical morphology. In *In Proceedings of International Archives of Photogrammetry and Remote Sensing*, pages 929–933, 2004.

[Han07] Jinshu Han. Plant simulation based on fusion of L-system and IFS. In *Proceedings of the 7th international conference on Computational Science, Part II*, pages 1091–1098, 2007.

[Hao10] Li Hao. *Predictive Surgical Simulation for Preoperative Planning of Complex Cardiac Surgeries*. PhD thesis, School of Computing, National University of Singapore, 2010.

[HGL92] Anthony Huxley, Mark Griffiths, and Margot Levy, editors. *The new RHS dictionary of gardening*, volume 1. Macmillan and Stockton Press, 1992.

[HLC10] Li Hao, Wee Kheng Leow, and Ing-Sh Chiu. Elastic tubes: Modeling elastic deformation of hollow tubes. *Computer Graphics Forum*, 29(6):1770–1782, 2010.

[HOCS02] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 233–246, 2002.

[HPW92] Mark S. Hammel, Przemyslaw Prusinkiewicz, and Brian Wyvill. Modelling compound leaves using implicit contours. In *International Conference of the Computer Graphics Society on Visual Computing: Integrating Computer Graphics with Computer Vision*, pages 199–212, 1992.

[HRY03] Jim Hanan, Michael Renton, and Emily Yorston. Simulating and visualising spray deposition on plant canopies. In *Computer graphics and interactive techniques in Australasia and South East Asia*, pages 259–260, 2003.

[HSB05] Sung Min Hong, Bruce Simpson, and Gladimir V.G. Baranoski. Interactive venation-based leaf shape modeling. *Journal of Visualization and Computer Animation*, 16(3–4):415–427, 2005.

[IOI06] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. The sketch L-system: Global control of tree modeling using free-form strokes. In *Smart Graphics*, pages 138–146, 2006.

[JGZ09] Wenbiao Jin, Wenzhe Gu, and Zhifeng Zhang. An improved method for modeling of leaf venation patterns. In *Image and Signal Processing*, pages 1–5, 2009.

[JP99] Doug L. James and Dinesh K. Pai. ArtDefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 65–72, 1999.

[KHT+] Vladlen Koltun, Pat Hanrahan, Jerry Talton, Daniel Gibson, and Chris Platz. Dryad. http://dryad.stanford.edu.

[KKM+98] Yuri Knyazikhin, Jörn Kranigk, Ranga B. Myneni, Oleg Panfyorov, and Gode Gravenhorst. Influence of small-scale structure on radiative transfer and photosynthesis in vegetation canopies. *Journal of Geophysical Research*, 103(D6):6133–6144, 1998.

[KL] Radoslaw Karwowski and Brendan Lane. L-studio. http://algorithmicbotany.org/lstudio.

[LD99] Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, January 1999.

[LGZL08] Shenglian Lu, Xinyu Guo, Chunjiang Zhao, and Chengfeng Li. Model and animate plant leaf wilting. In *International Conference on Technologies for E-Learning and Digital Entertainment*, pages 728–735, 2008.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. In *Journal of Theoretical Biology*, volume 18, pages 280–315, 1968.

[Loc04] Birgit Ilka Loch. *Surface fitting for the modelling of plant leaves*. PhD thesis, School of Physical Sciences, University of Queensland, 2004.

[Luf] Thomas Luft. An ivy generator. http://graphics.uni-konstanz.de/~luft/ivy_generator.

[LZG09] Shenglian Lu, Chunjiang Zhao, and Xinyu Guo. Venation skeleton-based modeling plant leaf wilting. *International Journal of Computer Games Technology*, 2009:1–8, 2009.

[Mak73] Roman Maksymowych. *Analysis of Leaf Development*. Cambridge University Press, 1973.

[MML+95] R.B. Myneni, S. Maggion, J. Laquinta, J.L. Privette, N. Gobron, B. Pinty, D.S. Kimes, M.M. Verstraete, and D.L. Williams. Optical remote sensing of vegetation: Modeling, caveats, and algorithms. *Remote Sensing of Environment*, 51(1):169–188, 1995.

[MMPP03] Lars Mundermann, Peter MacMurchy, Juraj Pivovarov, and Przemyslaw Prusinkiewicz. Modeling lobed leaves. In *Computer Graphics International Conference*, pages 60–65, 2003.

[MR50] E. Milne-Redhead. Variation in leaf-shape within a species: Some examples from the gold coast. *Kew Bulletin*, 5(2):261–264, 1950.

[MZL+08] Wei Ma, Hongbin Zha, Jia Liu, Xiaopeng Zhang, and Bo Xiang. Image-based plant modeling by knowing leaves from their apexes. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[NFD07] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):88:1–8, 2007.

[OOI05] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum*, 24(3):487–496, 2005.

[Per] Timothy C. Perz. L-system 4. http://www.reocities.com/tperz/L4Home.htm.

[PGW04] Norbert Pfeifer, Ben Gorte, and Daniel Winterhalder. Automatic reconstruction of single trees from terrestrial laser scanner data. In *International Society for Photogrammetry and Remote Sensing*, pages 114–119, 2004.

[PH02] Ulla Pyysalo and Hannu Hyypp. Reconstructing tree crowns from laser scanner data for feature extraction. In *In International Society for Photogrammetry and Remote Sensing Commission III, Symposium*, 2002.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag, 1990.

[PMKL01] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan 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.

[PT97] Les Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag New York, Inc., 1997.

[PTMG08] Alexandre Peyrat, Olivier Terraz, Stephane Merillou, and Eric Galin. Generating vast varieties of realistic leaves with parametric 2Gmap L-systems. *The Visual Computer*, 24(7):807–816, 2008.

[QTZ+06] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. In *ACM SIGGRAPH*, pages 599–604, 2006.

[RACJ09] Armando Re, Francisco Abad, Emilio Camahort, and M. C. Juan. Tools for procedural generation of plants in virtual scenes. In *Proceedings of the 9th International Conference on Computational Science*, pages 801–810, 2009.

[RFL+05] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transaction on Graphics*, 24(3):702–711, July 2005.

[RHNB00] P. Room, J. Hanan, B. Nolan, and R. Battaglia. Pesticide targeting: Measuring and simulating effects of plant architecture on pesticide deposition. In *Insect Pest Management in Sweet Corn, (Workshop No. 3), Queensland Department of Primary Industries, Bowen Research Station*, pages 20–25, 2000.

[RHP96] Peter Room, Jim Hanan, and Przemyslaw Prusinkiewicz. Virtual plants: New perspectives for ecologists, pathologists and agricultural scientists. *Trends in Plant Science*, 1(1):33–38, January 1996.

[RLFS02] Yodthong Rodkaew, Chidchanok Lursinsap, Tadahiro Fujimoto, and Suchada Siripant. Modeling leaf shapes using L-systems and genetic algorithms. In *International Conference NICOGRAPH*, pages 73–78, 2002.

[RLP07] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Eurographics Workshop on Natural Phenomena*, 2007.

[Sch] Michael Schernau. Fractree. http://archives.math.utk.edu/software/msdos/fractals/fractree.

[SFS05] Lisa Streit, Pavol Federl, and Mario Costa Sousa. Modelling plant variation through growth. *Computer Graphics Forum*, 24(3):497–506, 2005.

[Ski04] David J Skirvin. Virtual plant models of predatory mite movement in complex plant canopies. *Ecological Modelling*, 171:301–313, 2004.

[SLCS06] Lisa Streit, Paul Lapides, and Ehud Costa Sousa, Mario amd Sharlin. Modeling plant variations through 3D interactive sketches. In *3rd Eurographics Workshop on Sketch-based Interfaces and Modeling*, pages 99–106, 2006.

[SML04] Saint-Jean S., Chelle M., and Huber L. Modelling water transfer by rain-splash in a 3D canopy using monte carlo integration. In *Agricultural and Forest Meteorology*, pages 183–196, 2004.

[TZW+07] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics*, 26(3), 2007.

[VEBS+09] J. Vos, J.B. Evers, G.H. Buck-Sorlin, B. Andrieu, M. Chelle, and P.H.B. de Visser. Functional-structural plant modelling: A new versatile tool in crop science. In *Journal of Experimental Botany*, pages 2101–2115, 2009.

[Vis] Interactive Data Visualization. Speedtree. http://www.speedtree.com.

[VK06] Rawin Viruchpinta and Noppadon Khiripet. Real-time 3D plant structure modeling by L-system with actual measurement parameters. In *Biological ESTEEM Collection*, 2006.

[wdi] wdiestel. Arbaro - tree generation for povray. http://arbaro.sourceforge.net.

[WZW+06] Zhongke Wu, Mingquan Zhou, Xingce Wang, Xuefeng Ao, and Rongqing Song. An interactive system of modeling 3D trees with ball b-spline curves. In *Proceedings of the 2006 International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 259–265. IEEE Computer Society, 2006.

[WZW09] Zhongke Wu, Mingquan Zhou, and Xingce Wang. Interactive modeling of 3D tree with ball B-spline curves. *The International Journal of Virtual Reality*, 8(2):101–107, June 2009.

[XFr] XFrog. http://www.xfrog.com.

[XGC05] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge-based modeling of laser-scanned trees. In *ACM SIGGRAPH 2005 Sketches*, 2005.

[XGC07] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics*, 26(4), October 2007.

[YWM+09] Dong-Ming Yan, Julien Wintz, Bernard Mourrain, Wenping Wang, Frederic Boudon, and Christophe Godin. Efficient and robust tree model reconstruction from laser scanned data points. In *11th IEEE International conference on Computer-Aided Design and Computer Graphics*, pages 572–575, 2009.

[ZG04] Steve Zelinka and Michael Garland. Mesh modelling with curve analogies. In *Pacific Conference on Computer Graphics and Applications*, pages 94–98, 2004.

[ZTZ+08] Tonglin Zhu, Feng Tian, Yan Zhou, Hock Soon Seah, and Xiaolong Yan. Plant modeling based on 3D reconstruction and its application in digital museum. *Internaltion Journal of Virtual Reality*, 7(1):81–88, 2008.

[ZZHJ08] Chao Zhu, Xiaopeng Zhang, Baogang Hu, and Marc Jaeger. Reconstruction of tree crown shape from scanned data. In *Proceedings of the 3rd international conference on Technologies for E-learning and Digital Entertainment*, Edutainment '08, pages 745–756, 2008.