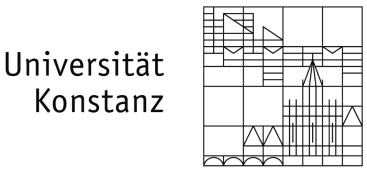
Efficient Processing of Plant Life in Computer Graphics

Dissertation

zur Erlangung des akademischen Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
an der Universität Konstanz im Fachbereich
Informatik und Informationswissenschaft

vorgelegt von
Sören Pirk



Konstanz 2013

Tag der mündlichen Prüfung: 10. Oktober 2013
Referent: Prof. Dr. Oliver Deussen
Referentin: Prof. Dr. Dorit Merhof

Advisor
Prof. Dr. Oliver Deussen, University of Konstanz, Germany

Reviewers
Prof. Dr. Oliver Deussen, University of Konstanz, Germany
Prof. Dr. Dorit Merhof, RWTH Aachen, Germany

Date of Submission
July 1st, 2013

Date of Defense
October 10th, 2013

Abstract

The role plants play in our daily lives gives synthetically modeled trees and plants particular significance. Vegetation is part of almost all virtual scenes and also in various application domains. The inhomogenous structure of these natural objects yields an enormous geometric complexity, which continues to pose challenges to computer graphics researchers to this day.
This thesis presents new techniques, methods and paradigms that contribute to the field of processing vegetation in computer graphics. Each of the following three approaches helps to migrate the so far mostly static tree models to dynamically adapting and developing ones.
A novel tree modeling paradigm is presented, which abstracts a given tree model into a light-weight intermediate representation. This data structure is easy to store and transmit and allows faithful reconstruction of a wide range of different models. Existing tree models as well as laser-scanned point sets can be converted into this representation. This either enables compression of the data necessary to describe a model, or to reconstruct exemplars from point sets.
When defining virtual sceneries, artists, designers or architects often need to adjust tree models according to the constraints of a scene. This is a requirement that is not addressed by today's modeling approaches. A novel interaction technique is presented, which allows trees to react dynamically to changing environmental conditions. The main branching structure of a given model is deformed based on transformations applied to the underlying skeletal graph. The new technique maintains a biologically plausible structure and is efficient enough to be used in interactive and real-time applications.
Another new method is introduced, which allows the developmental stages of static input tree models to be computed. Besides creating a large variety of exemplars from a single input model, the interpolation of these stages enables the tree's natural growth to be replicated. As an input tree model does not contain all the branches the tree will have developed during its entire lifetime, a new approach is presented, which allows additional branches to be generated for the younger developmental stages. This process utilizes the structural similarity of the input and is controlled by several measures known from botanical research.
Finally, techniques are explored, which enable vegetation to be rendered in real-time. This includes approaches for generating the mesh of branching structures as well as means for maintaining a photo-realistic appearance.

Zusammenfassung

Das häufige Vorkommen von Vegetation in unserem täglichen Leben weist synthetisch modellierten Bäumen und Pflanzen eine wichtige Rolle zu. Vegetation kommt in fast allen virtuellen Szenen und in verschiedenen Anwendungsbereichen vor. Die inhomogene Struktur dieser natürlichen Objekte führt zu einer enormen geometrischen Komplexität, die Forscher in der Computer Grafik bis heute vor Herausforderungen stellt.
Diese Arbeit präsentiert neue Techniken, Methoden und Paradigmen, die zu dem Feld der Verarbeitung von Vegetation in der Computer Grafik beitragen. Jeder der drei folgenden Ansätze hilft, die bisher meist statischen Baummodelle in dynamisch anpassbare und sich entwickelnde Modelle zu überführen.
Ein neues Paradigma zum Modellieren von Bäumen wird präsentiert, das ein gegebenes Baummodell in eine speicherarme Zwischen-Repräsentation abstrahiert. Die Datenstruktur ist leicht zu speichern, einfach zu übertragen und erlaubt eine glaubhafte Rekonstruktion einer Vielzahl verschiedener Modelle. Sowohl existierende Baummodelle, als auch gescannte Punktwolken können in diese Repräsentation konvertiert werden. Das erlaubt sowohl eine Kompression der zur Beschreibung notwendigen Daten, als auch die Rekonstruktion verschiedener Modelle.
Beim Definieren von virtuellen Szenen ist es oft erforderlich, dass Künstler, Designer oder Architekten Bäume den gegebenen Einschränkungen nach anpassen müssen. Eine Bedingung, die nicht durch die bisherigen Mehoden erfüllt werden kann. Eine neue Interaktions-Technik für Bäume wird präsentiert, die es Modellen erlaubt, dynamisch auf sich verändernde Einflüsse in ihrer Umgebung zu reagieren. Die Hauptaststruktur eines gegebenen Modells wird über Transformationen deformiert, die auf den unterliegenden Graphen angewendet werden. Diese neue Technik erlaubt das Aufrechterhalten einer biologisch plausiblen Struktur und ist effizient genug, um in interaktiven und auch in Echtzeit-Anwendungen verwendet zu werden.
Eine weitere neue Methode wird vorgestellt, die das Berechnen von Entwicklungszuständen aus statischen Baummodellen erlaubt. Neben dem Erzeugen einer Vielzahl an Exemplaren aus einem einzigen Eingabemodell erlaubt die Interpolation dieser Zustände das Nachbilden des natürlichen Wachstums eines Baums. Da die Eingabemodelle nicht alle Äste enthalten, die der Baum während seiner gesamten Lebenszeit jemals besaß, wird ein neuer Ansatz vorgestellt, der das Erzeugen von Ästen der jüngeren Entwicklungszustände erlaubt. Dieser Prozess nutzt die strukturelle Ähnlichkeit des Eingabemodells und wird durch mehrere Maße kontrolliert, die aus der botanischen Forschung bekannt sind.
Schlussendlich werden verschiedene Techniken vorgestellt, die das Rendern von Vegetation in Echtzeit erlauben. Dies beinhaltet sowohl das Erzeugen des Gitternetzes der Aststrukturen, als auch verschiedene Mittel zum Aufrechterhalten einer photorealistischen Erscheinung.

Acknowledgements

This dissertation was done by the kind help of many collaborators, colleagues and friends who directly and indirectly helped and inspired me.
First of all, I thank my advisor, Prof. Dr. Oliver Deussen for the unique opportunity to work in his group, for the guidance and encouragement that helped me to pursue this dissertation and for giving me the freedom working on my own ideas. I also thank Prof. Dr. Dorit Merhof for her helpful comments and for reviewing this thesis.
Many thanks to my colleagues and friends from the computer graphics group at the University of Konstanz. I especially thank Julian Kratt, Till Niese and Boris Neubert for the intense and successful work on our projects and papers that became an integral part of this thesis.
I am also thankful to Michael Balzer, Hendrik Strobelt, Daniel Heck, Thomas Schlömer, Joachim Böttger, Thomas Lindemeier, Ferdinand Eisenkeil, Marc Spicker, Jens Metzner and Torsten Hädrich for many profound, inspiring and motivating discussions, the nice time we spent outside the university, and, of course, the occasional gaming sessions during our lunch breaks. Special thanks to Ingrid Baiker, who always guided me smoothly through all the administrative tasks at the university and perfectly planned all of my oversea visits and journeys.
Xièxiè to Baoquan Chen from the Shenzhen Institute of Advanced Technology for inviting me to his lab in Shenzhen as well as to Daniel Cohen-Or, Yotam Livny, Zhanglin Cheng and Feilong Yan for the especially interesting time in China.
I want to thank Bedrich Benes, Ondrej Stava, Michel Abdul Massih and Alejandro Guayaquil from the Purdue University as well as Radomir Měch from Adobe Inc. for the amazingly collaborative and extremely interesting time while working on different projects in various time zones.
One of the outstanding stages during my PhD was the visit at Microsoft Research in Redmond, USA. Many thanks to Johannes Kopf, Michael Cohen and Matt Uyttendaele for supporting me during this time and for opening my mind to the areas of computer vision and computational photography.
I am deeply grateful for the support of my family that they provided me with throughout my entire life and in particular the love, patience and encouragement of Marieke Eggert.
Many thanks to all my friends that shared their time and life with me and thereby unbeknowingly supported this thesis and made this period of my life an enjoyable experience.
I appreciate that this work was partially supported by the DFG Graduiertenkolleg 1042 “Explorative Analysis and Visualization of Large Information Spaces”.

Contents

1

Introduction

The science of synthesizing images, more commonly known as the field of computer graphics, developed with an astonishing speed over the past decades. Starting with the early attempts of Ivan Sutherland in the 1960s, computer graphics has evolved into a ubiquitous domain with a wide range of applications. Today, this spans from small resolution graphics on smartphones and tablets to large-scale cinema productions, from simple user interfaces on everyday devices to precise visualization in medical software, from computationally expensive simulations to high-speed real-time graphics in games.

Since the early beginnings computer graphics researchers wanted to synthesize images as closely as possible real-world scenes. This demand for photo-realism exists to this day. Techniques have since been developed, which enable virtual scenes to be created at an astonishing level of visual fidelity. Today it is even possible to seamlessly embed computer generated objects to real-world content, as we experience in a wide range of movies.

Another distinct area within the field of computer graphics involves the processing of content in real-time. The fact that a virtual world can respond directly to actions caused by a user holds a strong fascination, and, together with poto-realistic rendering, has long been a driving force within the computer graphics domain. The ability to immediately adjust one's virtual surroundings is subject to a hard set of constraints, expressed by the need to process a single frame at around 20 milliseconds. This limited amount of time poses a whole different set of problems and requires sophisticated algorithms and techniques.

Today's personal computers and game consoles still host powerful graphics processors. The computational power of these devices increases every year, allowing more and more complex algorithms to be processed on increasingly complex scenes, even on consumer hardware.

However, the extensive development of the Internet and lower costs for developing on dedicated systems will enable centralized servers to either provide content or process computer graphics directly on demand. These changes in the infrastructure of creating computer graphics call on new methods to generate renderings on the fly or to distribute content efficiently.

1.1 Motivation

Synthesizing natural scenes with vegetation as richly as in the real world has always been a strong motivation of computer graphics research. As frequent objects in our daily life, vegetation is required in almost all virtual sceneries, ranging from single plants and trees to wide outdoor landscapes. The visual appearance and the inhomogeneous structure of trees and plants still pose many challenges to computer graphics researchers. The obvious main reason is that - compared to most man-made objects - even small trees and plants are extremely complex, yielding large amounts of geometry when represented as surface meshes. Figure 1.1 illustrates this complexity. Even small scenes like this already consist of millions of surface triangles.

Figure 1.1: A glade rendered at interactive rates (Figure 13 from [112]). This scene consists of about thirty different trees rendered at 25 fps with a total geometry of over 40M triangles.

Although rich and detailed virtual environments become more popular in applications like games, movies or urban visualization, storing and transmitting as well as modeling or rendering such objects with full detail is beyond the capabilities even of modern hardware and render systems.
While early attempts tried to model this complexity by finding mathematical models [121, 133] or procedural descriptions [146, 42, 49, 135], today's efforts also concentrate on reconstructing models from images [130, 150], from laser-scanned point sets [113, 198], or guided by user-defined sketches [132, 28, 114].
The animation of vegetation and the dynamic interaction of trees and plants with their environment are not thoroughly covered. While it is possible to utilize simple biological or physical models in today's applications, more sophisticated approaches are necessary to approximate the complex behavior observable in nature. The visualization of biological simulations of plant behavior and development - either at microscopic or macroscopic level - just starts to be of interest to the computer graphics community.
The visual appearance predominantly depends on the underlying geometry and cor-

responding material properties of a model. Targeting photo-realism requires complex models and algorithms for approximating the transport of light. Shading and shadowing, both by themselves evolved areas of interest within the field of computer graphics, need to be developed or adopted when rendering trees and plants.
Often, vegetation is only a small, however, important detail when defining virtual scenarios. Content creators like artists, designers or architects are in the need to define these details without spending a majority of their time. Algorithms and techniques for editing and modeling trees are not yet satisfying these requirements. Complex procedural formalisms (i.e. L-Systems) or time-demanding manual modeling by artists are the current approaches for more natural reality in applications.

1.2 Problem Statement and Challenges

Synthesizing images at a high level of visual fidelity usually requires a sequence of stages, which carefully defines the geometric representations of real world objects and its visual characteristics. Sufficiently defined surface meshes are the foundation for the realistic appearance of objects in most of today's applications. Algorithms and techniques are necessary to model these increasingly complex geometries either automatically or to support artists when working on them.
However, today's applications require not only static models just by themselves, but also need to animate them in a realistic manner. In many situations models of living objects need to evolve continuously over time, react to their environment, or adapt according to user input. While editing objects is a special need of content creators when defining scenes, the interaction of objects with the environment or according to user input has broad application and is commonly used in interactive settings. Animation, i.e. creating the illusion of movement, is one of the most important aspects when defining living or natural objects in virtual scenes. Especially motions triggered by physical or biological descriptions create characteristic behavior, which dramatically increases the plausibility of a scene. However, it also goes hand in hand with having to update the geometrical description of the virtual representatives frequently.
Besides creating plausible movements, careful crafting of the material characteristics of an object is also important. The transport of light and the interaction with all sorts of materials is one of the major areas in the field of computer graphics and has been addressed by the community since the very beginning. The interplay between individual material properties of an object and changing lighting conditions is especially important for creating photo-realistic images.
Trees and plants, with their enormous geometric complexity, pose special requirements to the afore-mentioned areas and especially stress production and render pipelines. Although vegetation is frequently found in today's applications, trees and plants often play only a subsidiary role in target sceneries. Consequently their visual appearance suffers from carefully made trade-offs between memory consumption and

render speed, resulting in strong approximations and severe simplifications. Common data structures for accelerating the rendering procedure cannot be used directly for processing vegetation. Ranging from single grass blades to entire forests, vegetation spans a huge parameter space and it is not easy to find general approaches. The goal of this thesis was to find paradigms, techniques and algorithms that lie within the following problem domains:

Modeling. A large variety of different means exists to create convincing branching structures for trees and plants. These methods can be grouped into four categories: procedural, data-driven, sketch-based, and biological-motivated. The question, however, to what extent a human observer is able to recognize this detail has not been given much attention. Though highly efficient, the human perceptual system is not able to process arbitrarily complex geometries. A tree viewed from a short distance offers more detail than a human can recognize in standard situations. Utilizing this fact can yield more efficient processing of tree models, however, it requires different modeling paradigms. Similarly, storing and transmitting digital models of trees efficiently while maintaining visual complexity requires new approaches and techniques.

Interaction. The interaction of a plant with its environment is a complex biological process with many decisive factors. Reproducing this behavior for digital models requires a deep understanding of the underlying biology.
Manually adjusting a plant mesh to a given environment is a time demanding task, which can easily require several hours. Methods have been proposed to algorithmically model the behavior of a plant in accordance with its environment. The latest techniques are *Open L-Systems* [128] and *Space Colonization* [159]. While Open L-Systems are an extension of the well known L-Systems [146], Space Colonization is a method that models a tree's competition for light and space. Both approaches yield appealing results, however, due to their runtime performance they cannot be used in real-time applications.

Animation. In order to increase the perceived realism of vegetation in real-time scenarios and interactive applications it is important to animate the movement of plants. This ranges from subtle changes caused by different weather conditions to extreme physical interactions with solid objects.
The dynamic development of different plant organs, the influence of tropisms and the animation of growth are also of interest and even more difficult to handle. Due to the natural complexity, processing such effects is a challenging task. Animating the surface mesh of a tree requires continuous updates, placing a heavy burden on the render system. While the animation of trees according to wind fields has already received considerable attention [74, 134, 168], more complex and biologically-driven animations still need to be found.

Rendering. Photo-realistically rendering the overall appearance of trees and plants is a complicated task. The inhomogeneous structure of bark and the diverse appearance of leaf tissue require a broad set of specialized algorithms. In interactive

applications, thousands of leaves and branches need to be processed in a couple of milliseconds.
In addition, it is important to model the interplay of light and shadow to achieve the same visual fidelity as can be observed in nature. While shadowing requires rendering the tree's geometry multiple times [82] to either generate a shadow map or to define shadow volumes, realistic shading can only be achieved when several layers of textures [73] are combined with advanced light computations.
Although today's hardware is generally well suited to managing these operations, artifacts may occur when detailed close-ups and wide angle views of vegetation are wanted at the same time. New approaches are necessary to better utilize the geometric characteristics of vegetation when evaluating the reflectance function.

Level of Detail. When processing scenes with large quantities of vegetation, it is necessary to lower the geometric complexity of each individual instance. While acceleration structures like *Kd-* or *Binary-Space-Partitioning-Trees* are well researched and can generally be used to speed up the rendering of different scenarios, they cannot be used directly for vegetation. The surface of a forest consists of millions of triangles, each of which is potentially moving and also individually interacting with light. Thus, it is not feasible to treat botanical objects in a way similar to man-made or static objects.
A number of approaches address this problem by maintaining the aggregate structure of trees and plants [48, 35] when reducing the geometric detail. However, dynamically generating details as needed while maintaining the overall visual appearance remains a challenge in this domain.

1.3 Contributions

This thesis presents new approaches, paradigms and techniques to improve the processing of vegetation in the afore-mentioned problem domains. The contributions in particular are: [1]

- A modeling paradigm for digital tree models, which allows the faithful reconstruction of models from laser-scanned point sets. The representation is based on the observation that the tree's foliage details can be abstracted into canonical geometry structures. Thus a tree is sufficiently expressed as a skeletal graph, representing the main branching structure, and a set of clusters, approximating the foliage shape. The encoded tree serves as a light-weight intermediate representation, which facilitates efficient storage and transmission of massive amounts of trees.
  The author of this thesis was involved in developing the main idea, in creating

[1] Each of them can also be found in the Bibliography at the end of the thesis.

concepts for modeling and rendering approaches as well as in programming a prototype and writing the paper. The achieved results were published in

Livny, Y., Pirk, S., Cheng, Z., Yan, F., Deussen, O., Cohen-Or, D., and Chen, B. **Texture-lobes for Tree Modelling**. In ACM SIGGRAPH 2011 Papers (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 53:1-53:10.

- A new method is introduced, which allows the skeletal graph of a tree model to be transformed efficiently while maintaining its biological plausibility. As transformations only occur when required, the new method can be much faster in comparison with other state of the art approaches, such as *Open L-System* [128] or *Space Colonization* [159]. The presented approach enables content creators to model entire ecosystems interactively, without the knowledge of the underlying biological process or the usually required procedural formalisms.
  The author of this thesis developed the main idea, created concepts for the new modeling paradigm, wrote code for the prototype, supervised the other members of the team and wrote text for the paper. The findings of this work were published in

  Pirk, S., Stava, O., Kratt, J., Said, M. A. M., Neubert, B., Měch, R., Benes, B., and Deussen, O. **Plastic Trees: Interactive Self-adapting Botanical Tree Models**. ACM Trans. Graph. 31, 4 (July 2012), 50:1-50:10.

- An algorithm, which allows the automatic computation of the developmental stages of a model, representing the tree's natural growth and development. Based on structural similarity branches are added where pruning has been applied or branches have died over time. Processing the animation in real-time makes exploration and editing of the parameter space of trees easier, enabling users to create rich scenes from a small number of exemplars.
  The author of this thesis was involved in developing the idea for this work, led the other members of the team, wrote text for the paper and supervised the programming of a prototype. The results of this research were published in

  Pirk, S., Niese, T., Deussen, O., and Neubert, B. **Capturing and Animating the Morphogenesis of Polygonal Tree Models**. ACM Trans. Graph. 31, 6 (Nov. 2012), 169:1-169:10.

- A set of render techniques that allows the generation of photo-realistic representations of tree models in real-time. The GPU-based shader pipelines for branches and leaves generate and adapt meshes on a frame to frame basis and hence, support dynamic and interactive modeling even of complex tree exemplars. Techniques commonly known as *Alpha to Coverage* and *Deferred Shading* further enhance the visual fidelity of tree renderings while maintaining interactive rates. Finally, different shadowing techniques are explored.

1.4 Other Publications

The following list shows other publications of the author of this thesis, which are related to processing vegetation, but also other topics within the field of computer graphics in chronological order of their appearance. The results of this work are not further discussed in this thesis:

- Neubert, B., Pirk, S., Deussen, O., and Dachsbacher, C. **Improved model- and view-dependent pruning of large botanical scenes**. Computer Graphics Forum 30, 6 (2011), 1708-1718.
- Deussen, O., Lindemeier, T., Pirk, S., and Tautzenberger, M. **Feedback-guided stroke placement for a painting machine**. In Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging (Aire-la-Ville, Switzerland, Switzerland, 2012), CAe'12, Eurographics Association, pp. 25-33.
- Pirk, S., Cohen, M. F., Deussen, O., Uyttendaele, M., and Kopf, J. **Video enhanced gigapixel panoramas**. In SIGGRAPH Asia 2012 Technical Briefs (New York, NY, USA, 2012), SA '12, ACM, pp. 7:1-7:4.
- Lindemeier, T., Pirk, S., and Deussen, O. **Image stylization with a painting machine using semantic hints**. Computers & Graphics 37, 5 (2013), 293-301.

1.5 Outline

This thesis is organized in the following way:
Chapter 2 reviews related work and state of the art techniques in the field of processing vegetation in computer graphics. This comprises early approaches, commonly known techniques for modeling, rendering and animating vegetation and the latest advances in the field. However, the content of this chapter is given without claiming to provide a complete overview.
In Chapter 3, we present an efficient modeling paradigm for tree models. The introduced cluster-based representation allows approximation of a given model and enables exemplars to be reconstructed with a high level of visual fidelity. Results are shown for models reconstructed from laser-scanned points sets as well as from already existing tree models.
The algorithm presented in Chapter 4 allows tree models to react dynamically to their environment. The proposed approach yields visually similar results to those produced by other modeling paradigms but allows interactive rates to be maintained, even for large numbers of tree models.
In Chapter 5 we introduce a framework that enables the developmental stages of tree models to be computed, which approximate the tree's natural growth. We analyze

a given input model and introduce a new technique to generate additional branches for arbitrary developmental stages.
Various techniques for rendering trees in real-time are reviewed and discussed in Chapter 6. This includes the organization of data structures as well as techniques to improve the visual quality of tree renderings without losing real-time performance.
Chapter 7 concludes the thesis at hand and proposes a number of topics as future work.

2
State of the Art

Due to its frequent existence in the real world, vegetation is of particular importance in computer generated imagery. Today, a broad range of applications require the visual complexity of photo-realistically rendered natural scenes.
Starting with the early approaches of Ulam and Lindenmayer, processing trees and plants has enjoyed considerable research attention since. While Ulam introduced the first mathematical descriptions of trees and plants [184], Lindenmayer, later, proposed a more sophisticated formalism [110], which allows rules to be defined and executed repetitively to construct complex structures.
These *Lindenmayer-Systems* were originally proposed as a mathematical model for cell development, and later received a lot of attention for modeling branching structures. Today commonly known as *L-Systems*, they have long been the state of the art of tree-like structures and have been extended in a variety of ways [144, 145, 140, 147]. The most significant adaptation was introduced as *Open L-Systems* [128]. Here the construction process is extended by means which consider environmental effects, allowing more convincing results to be generated. The competition for resources was further developed by the use of *Space Colonization* algorithms [159], which control the growth, mostly by distributing resources in the environment. Recently, realistic rule-based models of trees have been created by techniques in which the competition for resources plays an even more important role [135, 86]. Most of these techniques rely on growth models or procedural systems, which are difficult to handle.

Today a plethora of different modeling approaches allows to define extremely detailed tree exemplars to various needs. Although many of these approaches are quite diverse, digital models of trees and plants are still represented either implicitly as a procedural system or explicitly as a skeletal graph or surface mesh. Both approaches have strong advantages and severe limitations. While a single procedural representation allows a multitude of different tree models to be encoded by changing the parameters of the corresponding procedural model, it is difficult to directly influence the modeling process. The explicit representation enables the appearance of trees to be designed directly by reconstructing the structure of branches based on photographs [130, 150], on laser-scans [113, 198] or by directly sketching [132, 28, 114] them. However, once the model is created, it is static and thus difficult to adjust to changing environmental conditions or according to the application's needs.

Figure 2.1: Various trees and plants modeled with the Xfrog modeling system (Deussen and Litermann [50], Figure 6.15).

The methods presented in this thesis contribute to these problem domains by proposing new paradigms and techniques that ease the modeling process and allow the algorithmic adjustment of explicitly defined models. This minimizes the gap between traditional modeling approaches and provides content creators with tools to work more efficiently on existing models.
The following sections provide an overview of the most common techniques and approaches in the areas of modeling and rendering trees. A thorough overview of processing vegetation can be found in [50] as well as in more recent survey papers [166, 202].

2.1 Procedural Modeling

Based on the early attempts of Ulam and Lindenmayer, many other procedural modeling techniques have been presented since then. Similar to *L-Systems*, most of them are based on the idea of replicating growth by repeatedly applying a small set of rules to an initial structure in order to yield complex results. Originally, these rules only captured the internal properties of trees, such as branching angles and internode length [85, 96, 2, 172, 133]. Later, more geometrical aspects such as textures and detailed branches were added [17].
Although procedural modeling is probably the most sophisticated way to represent trees and plants, what nearly all of the current approaches have in common is that the appearance of the resulting model can only be controlled by indirectly adjusting the parameters of the procedural model. Thus, generating models to specific needs can easily become a cumbersome task for unexperienced users.

2.1.1 User-Assisted-Modeling

A subclass of methods incorporates user-defined inputs to guide the modeling process. One of the first of these approaches was the work of Weber and Penn [195], who created convincing tree models by using a complex parametric model with a large set of user-defined parameters.
Although it is possible to generate fairly realistic models using such a system, the sheer number of parameters represents one of the major drawbacks. Even simple structures require fairly large sets of input, which easily increase when certain species characteristics or more advanced systems need to be replicated. Hence, only experienced users are able to anticipate the influence of specific parameters to the modeling process.
Lintermann and Deussen [111] developed the *Xfrog* modeling technique, which combines rule-based and procedural modeling and thereby eases the modeling process. Despite the fact that it is very popular, modeling trees with *Xfrog* still requires many different parameters and also sophisticated knowledge about the biology of a tree. Figure 2.1 shows various trees and plants generated with the *Xfrog* modeling system. To help the parameter-based modeling process further, Boudon *et al.* [19] introduced decomposition graphs as multi-scale representations of plant structures, also enabling novice designers to generate reasonable results.
However, even when these methods allow trees and plants to be modeled more efficiently, a user still has to find the correct values for a large set of parameters, which is a tedious process, or needs to have profound knowledge about the biology of a tree to achieve visually convincing results.

2.1.2 Environment-aware Modeling

It has been recognized that the environment plays an important role in the development of trees [162]. Modifications of the environment itself can be used as a way of controlling the procedural model. Arvo and Kirk [3], Greene [71] and later Beneš and Millán [14] simulated climbing plants that grow on support structures and are influenced by the light density in sub-volumes of the scene.
Various methods for computing light within trees have been proposed, such as the fast, but simplified, technique proposed by Rudnick *et al.* [157], where the light is estimated from the crown shape, or a more advanced method based on radiant energy transfer [173].
While Hart *et al.* [80] simulated the production of additional wood when a tree receives mechanical stress to its parts, Lam and King [103] went even further and proposed a method that models tree growth based on the interaction of the tree's internal attributes, also considering water distribution and chemical flow. A GPU-oriented approach for modeling trees under the influence of the environment was introduced by Beneš *et al.* [13]. The expressive power of these approaches is limited to a few deciduous tree species.

Figure 2.2: A sequence of images of the self-organizing trees presented in Palubicki et al. [135] (Figure 11). The snapshots illustrate the influence of apical control. The excurrent form of the young tree develops into a decurrent form. The trees also adapt according to their environment.

Modeling the competition for resources, predominantly light and space, has been further developed over the years. While Haevre *et al.* [75] focused on simulating the growth of a plant according to the light available in its environment, *Space Colonization* algorithms strive to model a tree by filling a predefined space. Originally developed to model and visualize venation patterns of tree, plant and even petal leaves [158], Ruions *et al.* [159] later utilized the same approach for modeling complex branching structures. *Space Colonization* algorithms generate tree-like structures by distributing attraction points in the environment that are successively approached over time.
Recently, realistic rule-based models of trees have been created by techniques in which the competition for resources plays an important role [135, 86]. A result of such self-organizing tree models can be seen in Figure 2.2. The proposed approaches allow plausible results to be generated with a high degree of visual fidelity, however, most of these techniques rely on growth models, which makes them impractical for interactive design or real-time modeling.

2.1.3 Animation-based Modeling

Only a few methods, such as [144, 111, 164] allow the creation of animated models. Animation systems for tree models require the ability to let models grow easily in a natural way on the one hand, while on the other they also need to be editable in order to achieve the desired effects: both requirements are typically in conflict. The *Xfrog* modeling system [111] enables key-frames to be defined that describe the developmental stages of tree models, which are subsequently interpolated in a later step, yielding growth animations of trees. Although this creates interesting results of developing models, it is not possible for models to dynamically react to their environment. Unfortunately, various combinations of parameters can lead to implausible

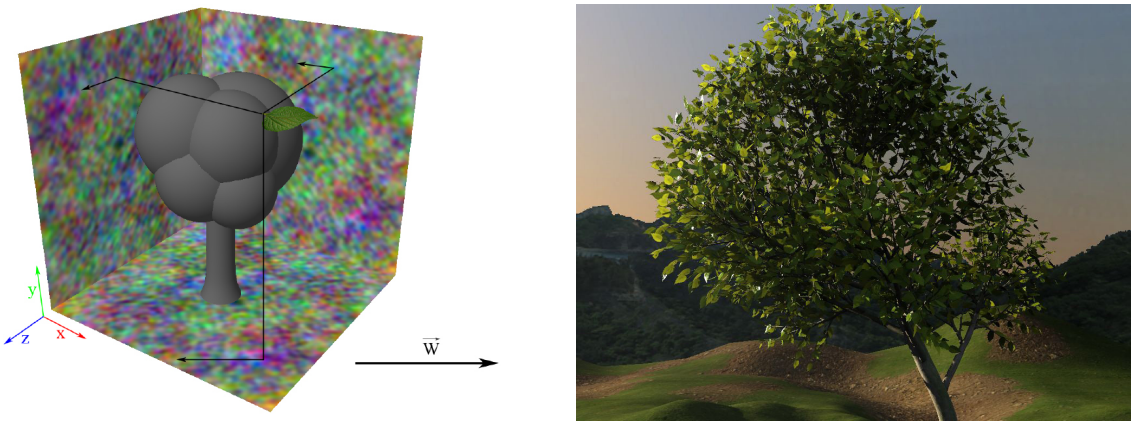


Figure 2.3: Left: 3D turbulence field used to animate the interaction of a tree with wind (Habel et al. [74], Figure 10). Right: detailed animated tree (Habel et al. [72], Figure 4.1).

animations when tree-specific constraints are missing. Animated L-Systems [143] also allow plant development to be modeled and yet are very hard to control.
Other methods focus on applying physical effects on tree and plant models [76] or on animating biological behavior, like the growth and development of plants [157]. While the animation of trees according to wind has already received considerable attention [134, 168], it is still difficult to continuously animate the development of plant organs, the influence of tropisms, or even the growth of entire groups of trees at interactive rates.
Most of these approaches utilize *Noise* functions to replicate the random nature of these effects. One of the latest methods was proposed by Habel *et al.* [72, 74] who use a 3D turbulence field to animate the effect of wind on tree models (see Figure 2.3). This introduces a new level of realism in animated scenes, however the motion of real objects often follows more complex rules.
Although today's graphics hardware enables to directly manipulate the vertices of surface meshes, some methods still require the careful preparation of the models in the initial modeling of a plant - a time-consuming undertaking, which prohibits automatic processing.

2.1.4 Inverse Procedural Modeling

The prevailing problems with the controllability of procedural modeling has motivated work which focuses on the problem of inverse procedural modeling. *Inverse Procedural Modeling* approaches try to find the parameters for a corresponding procedural model from a given set of inputs. Ijiri *et al.* [87] propose a system, which constructs a procedural model from an arrangement of 2D elements. Stava *et al.* [176] use transformation spaces to build procedural rules, which describe an arbitrary vector image. An extension of this approach to 3D is presented in [18]. In both of these works, convincing procedural models could be obtained only for highly regular

and symmetric inputs, but not for stochastic data. Finding a set of parameters that would generate a given input of a procedural model is a complex task. The parameters can be estimated using optimization techniques such as *Monte-Carlo Markov Chain* sampling [123]. This approach was used for finding the procedural representation of facades [154] or for directing the rewriting process in procedural modeling [179]. Both methods rely on a known procedural model with predefined parameter values, which are used to compute the model that best fits the input data. Vanegas *et al.* [185] proposed a method, which estimates the input parameters of a procedural model. However, their approach aims at matching stochastic models and at interactive modeling of regular virtual cities in order to provide high-level control.
Moreover, Cournède *et al.* [39] presented a method, which automatically detects parameters of stochastic procedural plant models. However, it relies on precise measurements of biomass distribution in real trees, which would be infeasible in real-time applications. By allowing a complex surface mesh to be converted into parameters of a corresponding developmental model, *Inverse Procedural Modeling* bridges the gap between explicit and implicit defined models.

2.2 Data-driven Modeling

In contrast to procedurally modeling trees and plants, data-driven approaches allow botanical structures to be reconstructed from real world data or from user defined input. This rather young form of modeling was predominantly driven by new hardware becoming available over the years, such as digital photo cameras or more efficient laser scanning devices. Reconstructing vegetation from real world data exhibits many advantages. Although similarly complex, reconstructed models tend to appear more realistic than those generated with procedural approaches. Often it is desirable to work on precise representations instead of approximations. Especially geo-scientific applications require exact descriptions instead of biologically plausible, yet completely virtual models.

2.2.1 Image-based Modeling

Image-based techniques use sets of images or even just single photographs to produce digital tree models. One of the first methods that used this form of input was proposed by Shlyakhter *et al.* [170], who extract the visual hull of a tree from photographs to guide a L-System-based modeling process. Reche-Martinez *et al.* [150] presented an approach based on several input images to reconstruct trees with sparse foliage. Though more general, the input images have to be registered carefully to generate the desired output.
The method of Neubert *et al.* [130] allows to work on loosely arranged inputs. Here, the main branches are determined by the user and a static model is constructed

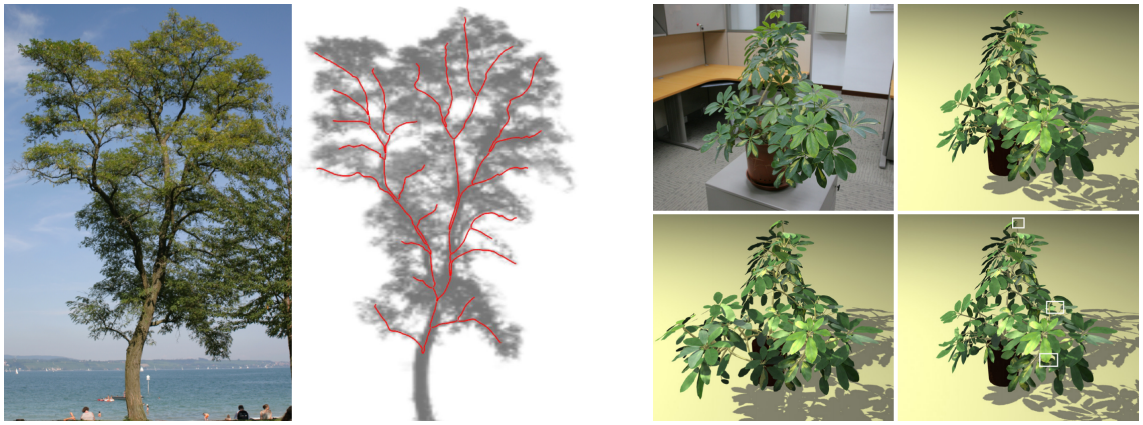


Figure 2.4: Left: An input photograph with a corresponding density estimation and attractor graph (Neubert et al. [130], Figure 2). Right: A set of images from Quan et al. [149] (Figure 7), showing the input image (top left); the recovered model with synthesized leaves from the same view point as the input (top right); the recovered model from images only (bottom right); the model after geometry editing (bottom left).

using a particle flow system with botanical heuristics. Figure 2.4 (left) shows the input photograph with the corresponding density estimation and an attractor graph. While Tan *et al.* [181] and Quan *et al.* [149] focus on semi-automatic techniques to reconstruct trees and plants, Tan *et al.* [180] also utilize a simple sketching method to generate realistic results from a single image. Figure 2.4 (right) shows one of the results by Quan *et al.* [149].
Recently, Li *et al.* [109] proposed a probabilistic approach that allows convincing tree models to be reconstructed from input videos. The method enables the shape to be controlled by drawing a new outline, while motion can be adjusted by modulating dominant frequencies. As video capturing hardware is broadly available, reconstructing trees from videos could potentially become another valuable source of input.
The main advantage of image-based techniques lies in the availability of data. Nowadays it is easily possible to capture and process large sets of images. However, the methods suffer from the fact that trees often only grow in groups or in dense surroundings where it is difficult to capture meaningful images.

2.2.2 Modeling from Laser-scanned Point Sets

The increasing availability of scanning technology motivated the development of approaches for tree reconstruction from point sets [137]. Scanning a tree commonly results in sparsely sampled point sets containing only the dominant characteristics of the corresponding tree. To reconstruct the tree skeleton Verroust and Lazarus [186] as well as Xu *et al.* [198] cluster edges in a spanning graph; leaves are randomly added to the twigs. Later approaches [29, 203] focus on reconstructing

Figure 2.5: A large urban scene (top, left) represented as about 10 million points, 200,000 of which (color coded in the image) capture a number of trees within the scene (top, right). Bottom: the same scene with reconstructed trees (Livny et al. [113], Figure 11).

tree properties such as the main branches and crown shapes to overcome the insufficient sampling density of finer details of the tree. Bucksch *et al.* [23, 24] use space partitioning to cluster points and form a skeleton by connecting adjacent clusters, while Côté *et al.* [37] synthesize minor tree and leaf geometry on the reconstructed branches based on light scattering properties obtained from different intensities of points. Livny *et al.* [113] use global optimizations to automatically reconstruct the branching structure of multiple overlapping trees. Their approach enables entire scenes of vegetation to be reconstructed very efficiently, given that the trees are more or less accessible. Figure 2.5 shows a large urban scene represented as about 10 million points, 200,000 of which capture a number of trees within the scene.
The above tree modeling methods are moving in a direction that favors higher levels of abstraction and control of the intended geometry. Trees are complex natural phenomena due to their structural nuances and random nature. Skeletal structures are highly abstracted properties of trees; using them alone or just augmenting them with random leaves is insufficient to convey the full visual realism. Similar to image-based techniques, faithfully reconstructing a tree from point-sets requires accessing the target from multiple directions.

2.2.3 Sketch-based Modeling

Instead of specifying parameters, sketch-based methods utilize user-defined sketches to support the modeling process. Given that a user is provided with a system that transforms the input sketches to meaningful branching structures, these approaches allow trees and plants to be modeled easily. However, while drawing branches is

Figure 2.6: The tree construction process proposed in Chen et al. [28] (Figure 10 (b)). From left to right: sketch, reconstructed branch mesh, added detailed branches, two views of the final rendering.

more intuitive than specifying a multitude of parameters, it can still be a cumbersome task to define 3D branches in 2D space. The methods proposed by Ijiri *et al.* [88] and Zakaria and Shukri [200] apply sketch-based methods to trees generated by procedural techniques, bridging procedural and image-based techniques.
While Okabe *et al.* [132] combined user-defined sketches with gesture-based editing operations to add, cut or erase branches, Chen *et al.* [28] later proposed a method to convert 2D freehand sketches and biologically motivated branching rules to convincing 3D tree structures (Figure 2.6). Wither *et al.* [197] refined the idea and proposed a method that enables a tree to be sketched according to different scales. While this provides novice users with a high-level tool to generate convincing results by just sketching the silhouette of a model, experts are able to even specify single organs like small twigs and leaves.
Recently, Longay *et al.* [114] introduced a system, which interactively models trees on tablet computers, providing users with a multi-touch interface (Figure 2.7). Allowing the utilization of multiple gestures at a time eases the modeling process and combined with procedural modeling leads to convincing tree models, which can be used in a wide variety of applications.
As these examples show, sketch-based methods provide users with great control of the modeling process, however, in many cases it is desirable to model trees and plants with even less user intervention or completely automatically.

2.3 Biologically-driven Approaches

Although many of the previously introduced techniques generate and reconstruct visually convincing results, they are not necessarily biological plausible. Sketch-based approaches especially, yield models that are unable to exist physiologically in natural environments. While 3D models of plants and trees support botanists, phytologists and - more generally - biologists in better comprehending the relationships among

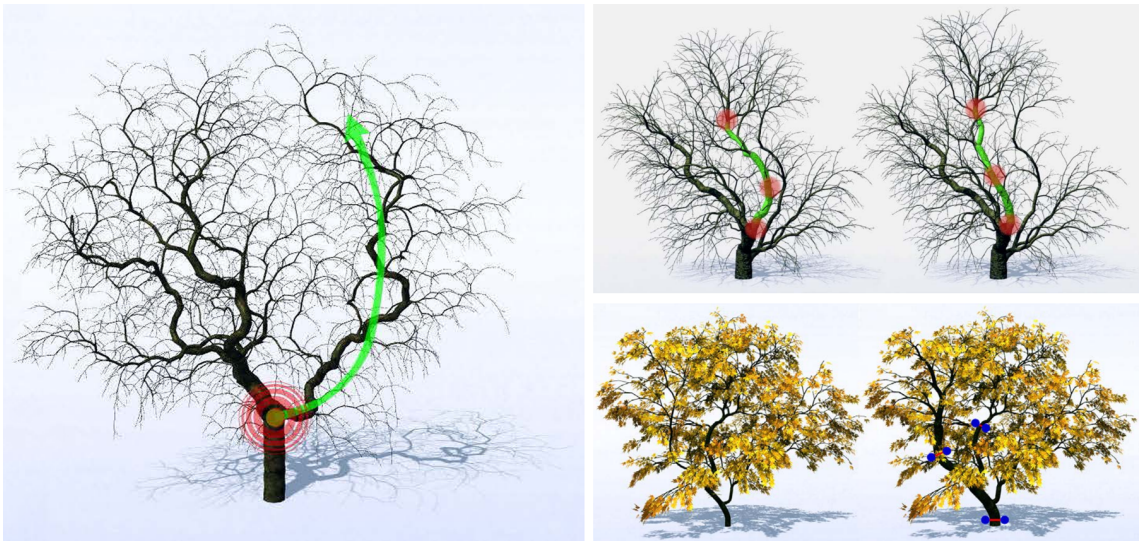


Figure 2.7: Results from Longay et al. [114] illustrating their sketch-based modeling approach. Left: an illustration of selective-growth (Figure 9, right). Top, right: bending a branch with a three-touch gesture (Figure 14). Bottom, right: a model refined by modifying the width of selected branches (Figure 15).

processes in plant functioning and morphogenesis, biologically plausible concepts help improve the quality of digital tree models in graphics applications. However, many of the available theories are not yet part of computer graphics research.

2.3.1 Theoretical Concepts

A number of different theories proved to be useful in understanding plant forms and their development. One of them was proposed as the *Pipe Model Theory* by Shinozaki *et al.* [169], who found that the distribution of leaves is linear proportional to non-photosynthetic tissue. Their theory is based on the idea that a tree consists of unit pipes, or vascular systems, which connect units of leaves to the root. This assembly of pipes builds the structure of a tree and is utilized to transport resources as well as to stabilize itself.
Although real trees are much more diverse in their construction, this model successfully explains many effects in natural trees and allows us to estimate the amount of biomass in forest stands. While Waring *et al.* [194] applied the *Pipe Model Theory* to analyse several western coniferous species, Chiba extended the original theory by a quantitative description of crown structures [30] and later also introduced the *Main-Axis-Cutting Method* [31], an improved version of the *Stratified Clipping Measure* [125], both supporting the *Pipe Model Theory* by allowing the distribution of biomass of trees and plants to be determined. Besides measuring the biomass of plants and forest stands, relationships of different biological measures help to comprehend the developmental process. In computer graphics, these relationships, more commonly known as *Allometric Rules*, help to model the surface mesh of a

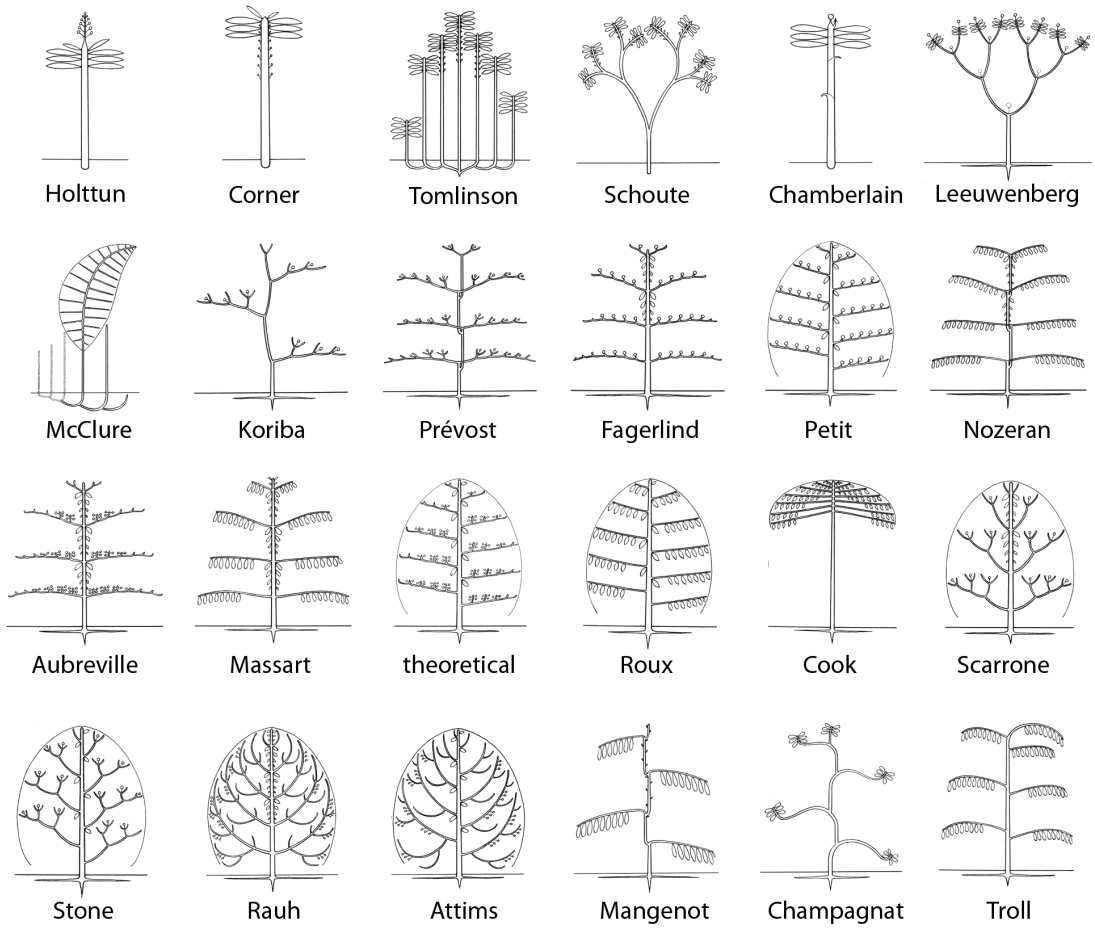


Figure 2.8: Different plant architectures proposed by Hallé [77] and de Reffye et al. [42] (image adapted from [50], Figure 2.5).

tree or a plant and thus allow the approximation of biological dimensions. More generally, allometry represents the study of relationships of body sizes compared to various biological measures. One of the first who found such relationships for trees was Leonardo da Vinci [153]. For example, he discovered that the sum of the cross-sections of two branches is equal to the cross-section area of the parent branch. Later, many other relationships and measures were introduced [131], describing the physiology of trees in various ways [127, 81].
Whilst these methods are only approximations of measures observable on real trees, they allow us to understand and to improve the modeling process of digital trees and plants.

2.3.2 Biological Modeling

Processes that drive the development of plants and their interaction with the environment are complex and not entirely understood. Today, botany research considers various models that describe the behavior of plants and trees at different scales.

To better understand the development of the overall structure, researchers started to classify trees. Hallé *et al.* [77] and later de Reffye *et al.* [42] introduced different types of architectural models of common characteristics among trees (Figure 2.8), facilitating a classification of different species. Barthélémy and Caraglio [8] proposed an architectural analysis, which describes the formation of tree structures as dynamic process of endogenous and exogenous constraints. Their approach embeds several morphological criteria into an architectural model and facilitates better comprehension of their relationships. The global behavior of plant development is easy to detect and hence the interaction with the plant's direct surroundings plays an important role in computer graphics applications.
While light is often considered to be the predominant factor in plant development and biomass production [125, 66, 32], other factors, like the transport of water [64] or carbon [124], and mechanical stress [91] also play an important role in the growth and functioning of the plant. Even the regulation and availability of different hormones has been examined [70]. The hormone Auxin for example, produced and transported towards the root when a bud starts growing into a new shoot, regulates the development of the buds [33].
To increase the visual realism of vegetation in computer graphics applications it is important to utilize more profound models of plant development and functioning. A complete introduction to the field of biological plant models is beyond the scope of this thesis. However, there is a large corpus of related work in the fields of botany and biology. An overview of plant development can be found in [10, 108, 161].

2.3.3 Functional-Structural Plant Models

Functional-Structural Plant Modeling is a rather new concept that combines a 3D plant structure, as already known in computer graphics, with a set of physiological functions. While the early approaches of plant modeling like L-Systems [110] and their subsequent developments [145] form the foundation of plant modeling in botany as well as in computer graphics research, *Functional-Structural Plant Models (FSPM)* constitute a more general description, as yet known predominantly only among botany research communities.
According to Godin and Sinoquet [68] *Functional-Structural Plant Modeling* follows four intentions: (i) a FSPM is commonly associated with a 3D plant model consisting of interconnected organs; (ii) the FSPM represents the spatial distribution of environmental and biological processes, which (iii) rather follow a top-down over a bottom-up approach, meaning that the model is thought to go from organ to plant level or from plant to stand level; (iv) offers enough complexity to process and to represent a variety of plant characteristics.
Vos *et al.* [187] provide a more general description, in which a FSPM consists of an architectural part (plant structure) and a process part (plant functioning) and state that: *"The FSPM paradigm considers that plants respond to their environment by*

not only adapting their functions (e.g. photosynthesis, transpiration, N allocation) but often also their structure (e.g. breaking buds or keeping buds dormant, shape and orientation of organs), which, in turn, modifies the condition (e.g. light) in which functions operate.", [187, p.2102].
Several approaches have been proposed that utilize FSPMs to model different functions of plant development. One of the first methods was introduced as the *LIGNUM* modeling approach proposed by Perttunen *et al.* [136], allowing the growth of trees to be modeled based on simple structural units, which represent the different organs of a plant. Later, Renton *et al.* [152] included physiological aspects in the FSPMs, enabling a plant's response to its environment to be modeled. Although their system does not require a detailed and quantitative understanding of individual plants, it can become quite complex when the interaction between many plants is computed. Another modeling approach was proposed by Cournède *et al.* [38] who introduced *GreenLab*, a mathematical factorization model, which was found to be a popular tool over the years [106, 94, 95]. Their approach does not rely on the number of organs and thus lowers the computational costs of a simulation. FSPMs easily require a multitude of parameters that reflect the complexity of the corresponding biological processes. To address this, Cournède *et al.* [39] proposed a general framework to estimate the parameters for complex models. General overviews about *Functional-Structural Plant Modeling* can be found in several papers [155, 145, 100] two of which were published recently [46, 187].
As these methods indicate, when implemented in computer graphics applications, FSPMs could lead to more sophisticated tree models especially in interactive environments or when used for animations. On the other hand, provided with better visualizations, biologists and botanists could better comprehend the relationships in plant functioning and development. However, similar to *L-Systems*, FSPMs tend to be complex systems with a multitude of parameters, which implies that they are hard to control, computationally expensive and thus inappropriate at least for many of today's applications.

2.4 Level of Detail

In computer graphics, *Level of Detail (LOD)* techniques are used to simplify the representation of objects according to certain metrics, such as the distance to an observer, object-visibility, or even movement speed. Simplified representations and thus reduced amounts of geometry lower the demand for processing power and memory, which is especially important in real-time applications, in which hard time constraints need to be matched. Several different LOD techniques have been proposed over the years [116, 60], but only a small set of algorithms exists that addresses the special requirements and characteristics of vegetation.
Often, changing the level of detail means switching between different types of rep-

Figure 2.9: Left: a detailed close-up of tree foliage. Leaves and branches are modeled as distinct sets of geometry. Right: an image from Behrend et al. [11] (Figure 1 (a)-(c)) illustrating different billboard cloud representations: original model; billboard approximation using the k-means algorithm; approximation using improved clustering and hierarchical model information.

resentations, e.g from full geometry to simple proxy geometry or even just simple textured quads. This allows for a dramatic reduction of the complexity, but it is very difficult to avoid noticable artifacts. This section introduces different forms of representations as well as algorithms, which enable transition artifacts to be reduced.

2.4.1 Forms of Representation

Depending on the requirements of an application, there are various types of representations for trees and plants, ranging from broad overviews of entire forests to detailed representations of single organs or certain plant specifics.
Today, trees are commonly modeled as a set of distinct organs, such as branches, twigs, leaves as well as blossoms or even fruits (Figure 2.9 (left)). This representation allows for modeling in even finer detail, however it also implies the need to process large amounts of geometry. Although vegetation is content of almost all scenarios, it often just replenishes a scene and prohibits to consume a majority of the available resources. Thus, the degree of detail often suffers from carefully made trade-offs between memory consumption and render speed.
As graphics hardware rigorously supports triangle meshes, vegetation is often rendered in this form of representation. Today's applications routinely process complex scenes consisting of billions of triangles, even allowing precisely defined vegetation to be rendered. Early approaches, however, needed this complexity to be reduced by expressing vegetation as textured quadric surfaces [65], commonly known as *Billboards*. Whilst these *Billboards* pose severe problems when complex interaction with light or animations is required they can still be found in today's applications

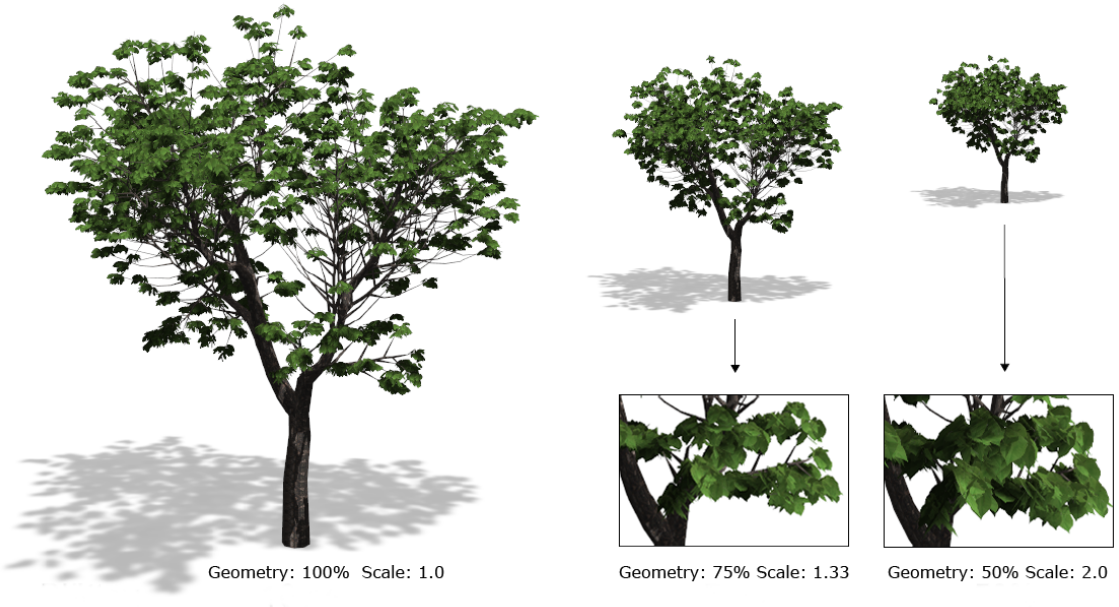


Figure 2.10: The effect of Stochastic Pruning [48, 35]. Left: The original model rendered with full geometry. Right: the simplified representation with scaled geometry. The inset images show the model viewed from afar.

to enrich scenes, especially representing objects far away from the observer. Several approaches have been proposed, which use this form of encoding to represent vegetation with varying degrees of detail [11, 45, 122] (Figure 2.9, right). Especially hybrid forms - a combination of billboards and mesh geometry - continue to be of interest, as they combine the advantage of low amounts of geometry without introducing too many artifacts. Jakulin [90] proposed such an approach by combining cross billboard with mesh geometry. Later, Décoret *et al.* [45] introduced *Billboard Clouds*, groups of plants or leaves that enable complex objects to be simplified massively, while still maintaining their visual plausibility.
Qin *et al.* [148] and Candussi *et al.* [25] proposed systems based on billboard representations, which even account for shading and shadowing. More recently, Lacewell *et al.* [102] utilized billboard clouds to render tree foliage, also supporting smooth transitions between different levels of detail. Billboards and surface meshes are predominantly used in today's applications, but other approaches also allow vegetation to be encoded as voxels [71], particles [151], slices [43], or volumetric textures [44]. A thorough overview of representing and encoding plants is given by Godin [67].

2.4.2 Simplification Algorithms

One of the major goals of *Level of Detail* techniques is to find unnoticeable transitions between representations of the same object with distinct differences in their geometric description. Several LOD algorithms exist [115], but many of them cannot be applied directly to render vegetation.
The surface of a forest consists of millions of triangles, each of which is potentially

Figure 2.11: Renderings from Boulanger et al. [20] (Figure 5.18). The same tree is rendered under varying lighting conditions. The presented light model enables the parameters to be adjusted dynamically in real-time.

moving and also individually interacting with light. Thus, algorithms for vegetation do not only need to severely simplifiy the geometric details, but also to consider the shading and shadowing as well as movements caused by animations. To address this Deussen *et al.* [48] utilized point and line primitives to simplify polygonal meshes, facilitating the interactive rendering of large outdoor landscapes, while maintaining the vegetation's specific characteristics. They also introduced a stochastic simplification approach to reduce the amount of geometry even further. While their method just handles point and line primitives, Cook *et al.* [35] extended the idea to polygonal meshes. This technique, today commonly known as *Stochastic Pruning*, allows for the smooth reduction of aggregate detail geometry, sets of independent objects, often perceived as one entity. While this description predominantly applies to the foliage of trees, swarms, groups of molecules or even crowds can also be categorized as aggregate detail.
Both, Deussen *et al.* [48] and later Cook *et al.* [35], proposed the simple, yet important, strategy involving the stochastic reduction of geometry while proportionally scaling the residuals. As geometry is reduced when the camera moves away from the object, scaling the remaining geometry keeps the screen space area, which the object covers, consistent, thus maintaining the overall appearance. Figure 2.10 illustrates this process.
Approximating polygonal meshes with point primitives proved to be a valuable approach to simplify object representations. More general methods were proposed, which also account for different sampling densities [160, 52] and even allow point samples to be generated on the fly [190, 189, 174].
Billboard and *Impostor* representations facilitate a further reduction of the geometry by encoding details as textures. Many approaches utilize this form of representation [65, 102, 11, 45] the difficulty, however, lies in the fact that noticeable transitions between billboard and full geometry representations are often unavoidable. Especially dynamic shading and shadowing as well as animations hinder their usefulness in many applications.

Figure 2.12: Screenshots from a commercial render benchmark system from Unigine[1]

2.5 Rendering

In general, *Rendering* describes the process of generating images from a geometric description, such as a model or even an entire scene. Although this synthesis of images was content of computer graphics research ever since, especially within the last decade a multitude of techniques, approaches and paradigms were proposed. The increasing popularity of computer graphics in video games and the availability of steadily advancing graphics hardware - a reciprocal relationship - are responsible for astonishingly evolved rendering algorithms. However, only a few methods directly address the rendering of vegetation. Rendering biological objects, even in real-time, is a demanding task that requires careful attention and approaches in a variety of domains.

2.5.1 Real-time Processing

Due to its importance in a variety of applications, real-time rendering of vegetation has frequently been a subject of scientific investigation. One of the major difficulties is still the massive amount of geometry that needs to be processed frequently. As described in Section 2.4, many of the proposed methods try to reduce this complexity by utilizing approximations, such as impostors or billboard clouds [11, 25, 45] and even modern render systems such as the *The Valley* Benchmark from Unigine[1] (Figure 2.12) utilize these approximations.
Other forms of representation go even further, allowing entire forests to be encoded in slices or volumes. Decaudin and Neyret [43] showed how to utilize volumetric textures to represent vegetation and efficiently rendering distant overviews of entire forests with full parallax. Bruneton and Neyrete, later, improved this form of representation by introducing a light model [22] allowing the reproduction of realistic shading effects. Although both approaches create more convincing results than billboard representations, they support neither different levels of detail nor smooth interaction with single instances.

[1] http://www.unigine.com/

Figure 2.13: Result images from Wang et al. [191] (Figure 1, (a)-(d)). A high visual quality is achieved by combining spatially-variant BRDFs with Subsurface Scattering and Precomputed Radiance Transfer.

To increase the realism of single exemplars, Boulanger *et al.* [21] proposed an advanced light model, which considers the spatial distribution of leaves and computes low-frequency, indirect lighting. The algorithm efficiently approximates the complex reflections and transmissions of light occuring in sets of leaves. Figure 2.11 shows the same tree rendered under varying light conditions using the proposed method. Shading and shadowing are commonly treated as separated processes, especially in real-time computer graphics.
Boulanger *et al.* [20] also proposed a stochastic approach to process detailed shadows in tree canopies. Although a lot of work has been devoted to processing trees efficiently, other forms of vegetation also are of interest. Musawir *et al.* [167] introduced an image-based algorithm for the interactive rendering of grass. They construct a bidirectional texture function and combine it with a view-dependent depth map to generate photo-realistic results.
What all these approaches have in common is that they try to meet the hard time constraints required by real-time applications while maintaining visually plausible results. Due to severe simplifications and strong assumptions many of the proposed techniques cannot be applied to a wide range of applications.

2.5.2 Material Properties

Natural trees and plants are inherently complex in their visual appearance. When photo-realism is the objective, the interplay of light as well as different material characteristics need to be carefully replicated. This relationship is commonly defined as *Bidirectional Reflectance Distribution Function (BRDF)*, a multi-dimensional function describing the behavior of light at a certain point. Often, and especially in real-time environments, this BRDF is only a very simple approximation of the real-world behavior.
A variety of methods were proposed to model the material properties of different organs of plants. Leaves particularly exhibit complex characteristics as they often appear diffuse, translucent and specular at the same time. Ma *et al.* [119] were one of the first who examined the optical scattering of a single leaf. While they compare different theoretical measures, Baranoski and Rokne [5, 6] introduced a method for the realistic rendering of leaf tissue by modeling three different components

Figure 2.14: Images from Boulanger [20] (Figure 5.1). The interplay of light and shadows creates variations in the contrast, thus facilitating the perception of structure and depth. Left: photograph of a real tree; right: rendering

of light propagation: surface reflectance, subsurface reflectance and transmittance. Although their method allows generating convincing results, due to its runtime performance it is impractical for interactive and real-time scenarios.
Franzke and Deussen [63] and later Habel *et al.* [73] encode the complex material properties in several layers of textures, with every texture corresponding to different material characteristics, such as albedo, translucency, or normals. As modern graphics hardware allows for the efficient processing of several texture look-ups per pixel, these methods produce appealing results in real-time. More recently Wang *et al.* [191] proposed an improved framework to render plant leaves in real-time. They are able to create compelling results by combining spatially-variant BRDFs with *Subsurface Scattering* and *Precomputed Radiance Transfer*. A sequence of images generated with their method is illustrated in Figure 2.13.
Besides photosynthetic tissue, trunks, branches and twigs are the second predominant feature in the appearance of trees. While techniques such as *Bump-, Parallax-* or *Displacement-Mapping* [16, 93, 192] generally improve the appearance, more sophisticated techniques have been introduced, which directly model and render tree bark. While Hart *et al.* [79] proposed a local modeling technique to define an implicit surface mesh of a tree, other methods focus on modeling bark in texture or texton space [193].
The advantage of such methods lies in the absence of additional geometry, however, they also complicate the shading and lighting. Other approaches directly modify the corresponding surface mesh [57] and even allow cracking behaviour to be modeled as the tree evolves over time [104]. Many of these methods create convincing results, but the inhomogenous appearance of vegetation requires more flexible methods to model and render real-world material characteristics.

Figure 2.15: Improved tree renderings from Luft et al. [117] (Figure 8). Utilizing implicit surfaces enhances depth cues and structure. Left: a photo-realistic rendering; Middle + Right: illustrative renderings.

2.5.3 Shading and Lighting

Contrast variations caused by the interplay of light and shadow are of particular importance for most computer graphics scenarios. Subtle as well as high-frequency changes in the illumination of foliage allow structure and depth to be recognized. While *Shadow Mapping* and *Shadow Volume* techniques are generally applicable to vegetal geometry, they often suffer from drawbacks such as insufficient resolution or runtime performance. Several methods have been proposed, which address this problem by approximatively modeling the influence of light and shadow on vegetation based on textures [73, 20]. As can be seen in Figure 2.14 this produces convincing results. Boulanger *et al.* [21] proposed to stochastically define shadows cast by leaves onto other parts of the foliage, allowing to very efficiently render complex shadows in the tree canopy. Qin *et al.* utilize a set of 2D buffers storing the direct and indirect lighting information [148] and combine the outcome with a two stage shadowing algorithm.
Lacewell *et al.* [101] focused on generating soft shadows from aggregate geometry. They utilize a *Bounding Volume Hierarchy* for storing opacity values that are checked when ray intersections occur, which dramatically reduces the shadow testing costs for complex geometries, however, it is only applicable in offline rendering settings. Hegeman *et al.* [83] introduced an approximation of *Ambient Occlusion* for trees, creating the impression of indirect lighting. The key contribution of their approach lies in utilizing proxy volumes, simple spheres or ellipsoids, which represent the foliage of a tree.
Two other forms of approximating indirect lighting were introduced by Luft *et al.* First, they proposed creating a difference image of the depth buffer and a low-pass filtered version of the same depth buffer for a given scene. This difference image, an unsharp mask, is than utilized to efficiently emphasize structure and depth [118], roughly estimating indirect lighting (see Figure 2.15). Second, Luft *et al.* introduced

Implicit Surfaces to represent the overall shape of the foliage of a tree [117]. As this creates local neighborhoods with similar characteristics it can be used to adjust the surface normal and thus enhances the perception of depth cues and structure, especially for illustrative and non-photorealistic visualizations.
It has been recognized that preserving contrast also plays an important role in rendering or simplification [35]. However, utilizing the strengths and weaknesses of the *Human Perceptual System* to drive simplification algorithms or stochastic visualizations is not yet part of computer graphics research.

3

Modeling Trees through Means of Abstraction

In this chapter a novel tree modeling paradigm is introduced, which captures the main characteristics of individual tree models but without creating too many structural nuances. The resulting representation is extremely memory efficient and thus allows for transmitting and storing massive amounts of tree models.

During the last decades a variety of procedural methods has been developed to design and create models of trees [50, 135]. From a small set of rules, such as those used in L-Systems, these techniques can create visually appealing tree models, which can be extremely complex in geometry and large in size. However, given the high computation expense, such procedural operations cannot be performed during rendering, prohibiting their application in real-time scenarios and interactive applications. Furthermore, controlling the resulting geometric shape, to conform with specific characteristics of individual trees still is a difficult issue [176, 12, 179].

Vegetation in general and trees in particular are extremely complex and require large amounts of vertices when represented as surface meshes. The dominant part of this geometry is covered by the foliage. Large quantities of leaves are rendered as single instances to create visually plausible results. While early attempts tried to encode this complexity in the form of textured quads, today the organs of trees are modeled as single objects with varying degrees of detail. This form of representation with all its detail offers many advantages when used with shading or level of detail algorithms and is considered state of the art in tree modeling. However, although this detail is important to create visually plausible results, it hinders to process large quantities of trees.

The new modeling paradigm addresses both the efficient storage and transmission as well as the faithful modeling and rendering of individual exemplars. The following sections introduce and discuss the new representation and explain its use for today's computer graphics applications.

3.1 Perceptual Motivation

The *Human Visual System (HVS)* [92] enables large amounts of data of very high resolution to be processed at an incredible speed. The *HVS Model* abstracts the capabilities of human vision, providing a better understanding of how humans perceive and process visual characteristics like, luminance and color, contrast and adaptation or masking. Though not entirely understood, it has been recognized that the HVS also exhibits severe limitations when observing complex and intricate scenes. Researchers, especially in the field of computer vision and computational photography, have been trying to utilize these limitations in the design of algorithms and paradigms. A thorough overview of the properties and the restrictions of human perception is provided by Nadenau *et al.* [129].
Unsurprisingly, human perception has also already been utilized in a computer graphics context [36]. The introduced methods predominantly focus on improving level of detail algorithms, such as the *Stochastic Simplification* approach proposed by Cook *et al.* [35]. They presented a stochastic process, which not only makes it possible to adjust geometric complexity smoothly, but also adapts contrast and covered area for a given model. This method proved to be very useful for smoothly reducing geometric complexity, however, as a level of detail algorithm it does not directly affect modeling and therefore does not enable the data complexity to be reduced in general.
Inspired by the existing perceptual metrics [36] and the findings of Cook *et al.* [35], we assume that human perception also underlies certain constraints regarding detail and complexity. Similar to the previous methods, the idea is to utilize the properties of the HVS to enable efficient processing and modeling of vegetation in graphics applications and render systems. Figure 3.1 illustratively sustains these assumptions. Although it is possible to recognize fine details, such as single leaves and twigs in the photograph, the overall structures, such as the main branches, the silhouette, changes in depth and even the interplay of light and shadows, are more important when evaluating the visual appearance of trees. These observations directly yield a possible data structure for trees, an intermediate representation, which covers the overall appearance instead of all possible details (Figure 3.1, right). We introduce a data structure, which follows these assumptions by abstracting the visual properties of tree models. The advantages and capabilities are dicussed in the following sections.

3.2 Advantages of Abstraction

Today, a plethora of applications requires processing vegetation to enhance the visual fidelity. Most of these programs are distributed with the content they later process, such as models, textures, sounds or even videos. However, the development of the internet and the availability of fast computer networks also allow for other

Figure 3.1: Looking at the photograph of a real tree (left), perceptually important parts are the main branching structure and the overall shape of the foliage. While fine details are negligible, it is important to maintain the silhouette and the perception of depth and structure (middle). These considerations directly yield a possible representation for tree models (right).

concepts of distributing content. Particularly, geo-scientific and web-based applications commonly stream their content to millions of users. Transmitting models on demand prohibits large amounts of data, a requirement that overthrows the goal of achieving high visual quality. Represented explicitly as surface meshes, trees and plants consist of enormous amounts of vertices, preventing the efficient transmission of large quantities of exemplars.
Besides transmission, applications also require smooth changes in the level of detail of vegetation. Especially, when processed on low-end mobile devices, models need to natively support different levels of abstraction. Today, model libraries such as *Xfrog, SpeedTree* or *Laubwerk* offer a large variety of different tree and plant models. However, most of these models are defined as detailed surface meshes without considering the afore-mentioned level of detail requirements. Existing approaches predominantly focus on reducing the geometric complexity of detailed models at run-time. This is in opposition to efficient transmission and distribution.
The following sections introduce a new tree modeling paradigm, which addresses both requirements, transmission and level of detail, resulting in an extremely lightweight, yet visually plausible representation of tree models. This new representation not only facilitates efficient storage, transmission, and rendering of massive amounts of trees, but also eases tree modeling, whether for design or reconstruction.

3.3 Cluster-based Tree Representation

The new representation is based on the observation that a tree can be abstracted into an intermediate, abstract representation that approximates its appearance. This representation consists of a skeletal graph, faithfully representing the main branching structure and parts of canonical geometry or - more intuitively - leaf clusters, approximating the overall foliage shape. Figure 3.2 illustrates a tree model and its corresponding intermediate representation.

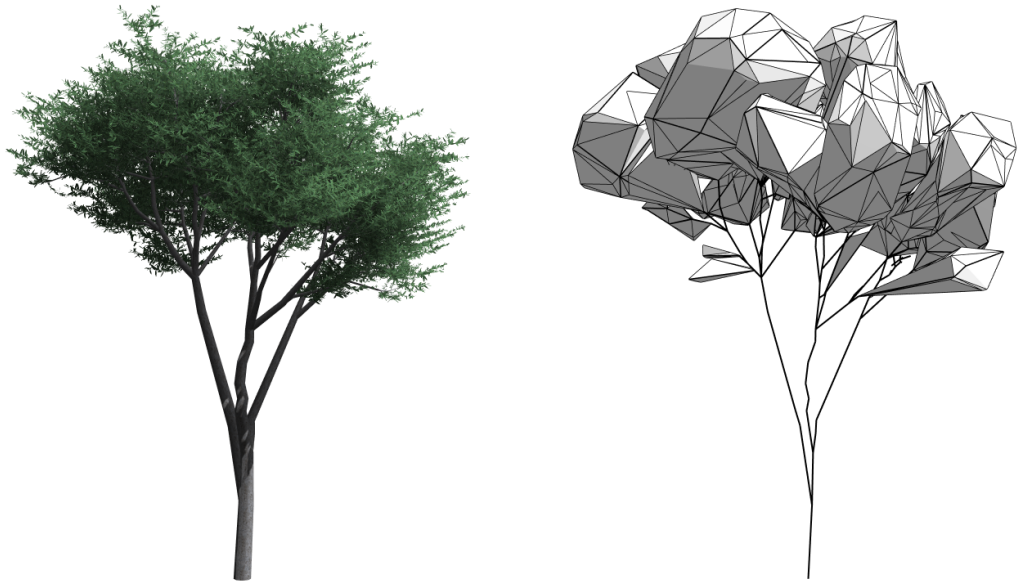


Figure 3.2: A tree model (left) and the corresponding cluster-based tree representation (right). The tree's main characteristics are captured by a set of clusters and the main branching structure. This abstract form of representation requires extremely low amounts of memory and thus enables efficient storage and transmission of individual exemplars.

The skeletal graph, together with a set of clusters and a species-specific signature, forms a cluster-based tree representation, which can be decoded and synthesized back to a full tree model that resembles the input. The cluster-based representation allows encoding arbitrary models from different sources yielding an extremely small memory footprint for individual exemplars.
Decoding a tree model from the cluster-based representation requires reconstructing the main branching structure and the foliage details. While the branch surface mesh can be reconstructed directly from the skeletal graph and the associated allometric information, each cluster is populated with predefined patches, a process similar to instancing or texturing. Every patch or branchlet is a small piece of branch or twig geometry, which is combined by using a discretization of botanical parameters such as branch width and vertical angle.
The cluster-based representation largely supports the reconstruction of tree models from laser-scanned point sets. Generating these point sets already implies a certain degree of abstraction and is therefore perfectly aligned with an abstract representation. But, it can also be utilized to encode existing tree models. The latter case, however, will not result in a precise reconstruction, but rather in an approximation of the input.
Figure 3.3 illustrates the overall tree modeling and synthesis process: the input data, either an existing tree model or a point set, is converted into the cluster-based representation – an encoding process, and subsequently reconstructed from it by texturing the clusters and producing the branch geometry – a decoding process. The encoded tree representation requires only a significantly small memory foot-

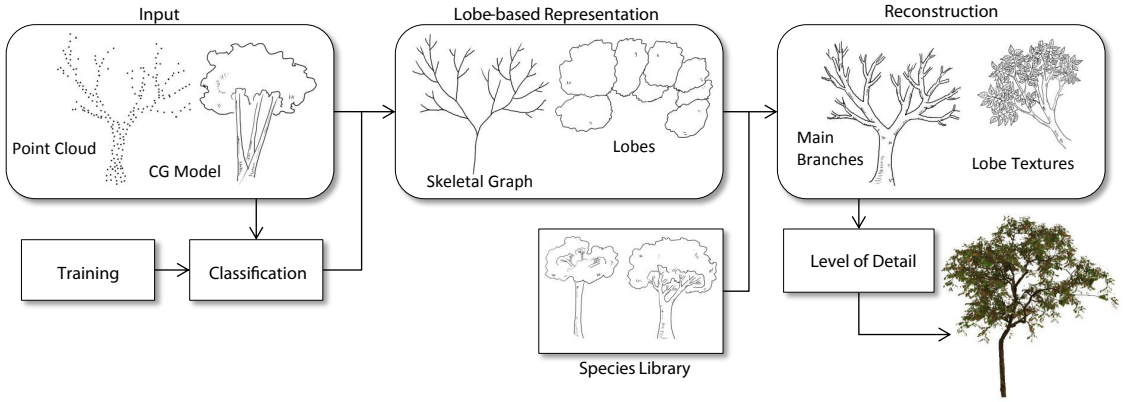


Figure 3.3: Description of the tree modeling and reconstruction process (adapted from [112], Figure 2): the cluster-based representation is computed from the input (point set or CG model). Additionally a classification subsystem is trained and later used to determine the species. A species library was built separately, including procedural elements and parameters for the different species. The tree is reconstructed by rebuilding the branching geometry from the skeletal graph and texturing the clusters using predefined elements from the species library. Level-of-detail, facilitated by cluster-based representation, is applied for interactive rendering.

print for each individual tree. The efficiency of the decoding process, mainly cluster texturing, supports the processing of large quantities of trees.
A key factor for achieving high visual fidelity of the final tree representation is the construction and assignment of cluster textures. This requires pregenerating a species library including parameters that reflect species dependent characteristics. These parameters are used to generate the patches. To automatically identify the species, a given input is classified. The result of this classification subsequently guides the assignment of proper parameters from the species library.
The cluster-based representation allows to encode models from different sources. Laser-scanned point sets as well as digital tree models, given as surface meshes, can serve as input. As a surface mesh provides more information than a point cloud, due to connectivity information, the following sections focus on identifying the skeletal graph and the clusters from point sets.

The following sections describe how to (i) approximate the geometry of a given input and to encode this representation, as well as, (ii) decode the representation and synthesize a detailed tree model with the full amount of geometry required to visually resemble the input.

3.3.1 Skeletal Structure

There are several methods, which enable the skeletal structure to be extracted from point sets. The main branches of a tree are described using a graph with spline functions, which are associated with allometric values for their diameters along the

axes. A typical skeletal graph consists of a few dozen of edges and is extracted from a given point set similar to Livny *et al.* [113] who proposed connecting neighboring points and constructing a shortest-path tree. Each edge (u, v) is assigned an edge weight $\|u - v\|^{\beta}$ which determines how likely its points will belong to the same branch. The parameter β allows for leveraging the edge lengths within Dijkstra's algorithm, which is applied in the next step to get the main tree structure. If β is close to one, the weights reflect the Euclidian distances of the corresponding points. Higher values will assign larger weights on longer edges and thus create a more compact tree structure.
While Livny *et al.* [113] tried to obtain this parameter automatically, which does not work well for leafy trees, it is determined manually for every species and stored along with other parameters in the species library (Table 3.1). In case no species information is provided, a default value is used.
A second parameter is important for computing the cluster-based representation: given the diameter of the trunk at the tree base (d_{root}), a formula for tree allometry is used, which describes how the branch diameters decrease in terms of their distance from the tree base. The general mechanism was already described by Da Vinci who stated that the diameter of a branch is the sum of diameters of all child branches [153, 89]. Based on this observation the diameter $d(u)$ for a node u in the spanning tree can be computed:

$$d(u) = d_{root} \left(\frac{l(u)}{l(root)} \right)^{\gamma} \tag{3.1}$$

where $l(u)$ is the sum of the length of all the edges in the subtree and with root node u, $l(root)$ being the sum of the length of all the edges in the complete spanning tree. Studies support this relation with $\gamma \approx 1.5$ for many cases [198], however, in practice it also depends on the species and on environmental factors such as wind strength, snow mass and elevation of the tree stand. Based on 8–10 scans for each tree species typical values for γ are manually determined (see Table 3.1) and stored in the species library.

Tree Species	β	γ	f_s
Mahogany	1.7	1.3	0.975
Bischofia polycarpa	1.8	1.5	0.986
Delonix	1.7	1.5	0.982
Lagerstroemia	1.6	1.5	0.976
Ailanthus altissima	1.5	1.9	0.980
Palm	1.8	1.3	0.800
Terminalia	1.5	1.5	0.970
Pine	1.3	1.2	0.980
Willow	1.5	1.9	0.980
Ficus Virens	2.0	1.6	0.975

Table 3.1: Typical parameter values of β, γ, and f_s.

3.3.2 Cluster Geometry

The tree-graph and the associated parameter γ allows selection of all the edges for which the average distance of their assigned points is higher than the diameter of the branches. The probability that these points will be directly connected in the original tree structure is low, which, in turn means that confidence in the generated edges is also low.
Thus, the edges in the tree-graph are traced from the root to the leaves until a low-confidence edge (at a point p_i) is reached; traversed edges are marked as the main branching structure. The points belonging to the remaining edges are all connected to one of the points p_i and are collected to form a cluster C_i.
In practice, however, in order to determine the clusters, the branch diameter and respective maximal edge-length has to be modified since it is much harder to scan the interior of leafy trees, due to occlusion of the foliage, in comparison to sparse models. To compensate for this fact the parameter f_s is introduced in Equation (3.2):

$$d'(u) = f_s \cdot d(u) \tag{3.2}$$

This parameter virtually reduces the diameter of the branch at a certain distance, thus enabling us to modify the amount and size of the clusters. Small values of f_s create large clusters, and for a tree without leaves, a value of $f_s \approx 1.0$ is used to generate many small clusters and reconstruct branches from the majority of the points (see Figure 3.4 and Figure 3.11).

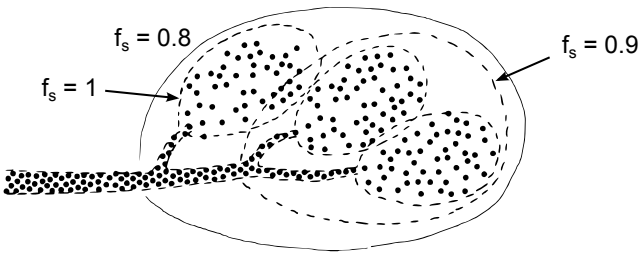


Figure 3.4: Clusters for different values of f_s (Figure 3 from [112]).

The cluster-geometry is computed with the α-Shape approach [53, 203] by using the points of each cluster C_i. α-Shapes are extensions of convex hulls and allow non-convex envelopes to be created. The points are combined to hexagons, with α determining the radius of a virtual ball, which is used to delete all those hexagons that do not fit into the ball.
An appropriate value of α plays an important role in establishing a balance between capturing the accuracy of a cluster on the one hand and the size of the resulting geometry on the other. A large value of α will create the convex hull, while a small value will create a surface with many holes. Since a good value for α depends on the point density, α_{min} is computed as the value in which the points are represented

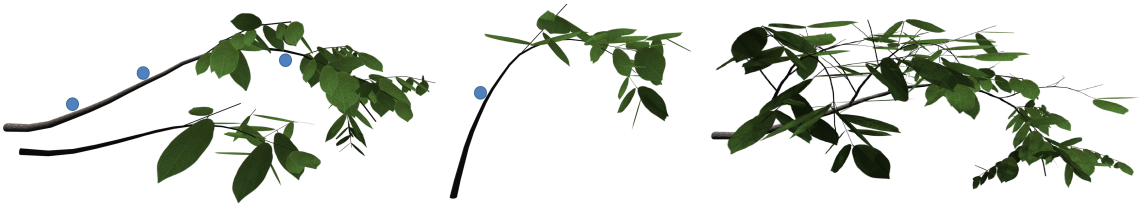


Figure 3.5: Subset of predefined branch patches (small geometric elements) and a composed structure for the Lagerstroemia species. The blue dots denote predefined docking positions for adjacent patches (Figure 4 from [112]).

by a single surface without holes and define our alpha as a multiple of this value. Experiments show that independently from the species a value of $\alpha = 5.0 \cdot \alpha_{min}$ results in sufficiently accurate and simple hulls.

3.4 Model Reconstruction

Given the cluster-based representation and the species-dependent parameters from the library the geometry of a tree can be reconstructed. The main branching structure is produced by creating generalized cylinders [17] for each edge of the skeletal graph using the width information obtained from allometric rules. Additionally, the clusters are represented by the triangulated surfaces produced from the α-Shapes, while the seed points are stored on the skeletal graph for procedural texturing.
To synthesize a species we generate a set of branchlets and store them in a species library (see Figure 3.5). An L-System is used to form a collection of 20–30 complete branchlets for each species varying in thickness, shape, and orientation. Each of the branchlets is then partitioned in order to form smaller incomplete branchlets (patches). Docking points are attached to branchlets where smaller twigs can be placed and each docking point has an assigned orientation and thickness.
The texturing of a cluster from seeds is performed by iterating two steps: patch selection and patch fitting. A branchlet is selected by enforcing thickness and orientation constraints to a seed point s. A subset S_s of patches is selected from the pre-defined patches by taking those with similar initial thickness and orientation to s. The fitting of a patch to a docking position is computed by

$$d = \frac{\max(thickn_r, thickn_d)}{\min(thickn_r, thickn_d)} \times (v_r \cdot v_d) \tag{3.3}$$

where r is the root point of the patch, and d is the docking point, v_r, v_d are the orientation vectors of patch and docking position. The branchlet that additionally corresponds best to the local geometry of the cluster is selected. The distance of the patch geometry and the points of the cluster geometry is measured and the selected branchlet is added to the cluster texture, while its docking positions, if there are any,

are used for the next texturing iterations (see Figure 3.6). The smallest branchlets do not have docking positions and terminate the texturing process. This fitting of branchlets is very similar to patch-based texture synthesis [54] in which a patch (in this case the region of an input image) is also selected in order to fit to a given local environment (the borders of a so far constructed texture). In texture synthesis, patches are selected and combined such that the texture is similar to the input image but has a larger size. Here the "input image" is the predefined branching structure of the species and the "texture" is the reconstructed branching structure in the cluster. Figure 3.7 shows two examples for the texturing of clusters. Starting from an initial branch patch, new ones are selected and added until the cluster geometry is filled, i.e., all docking positions are combined with patches.

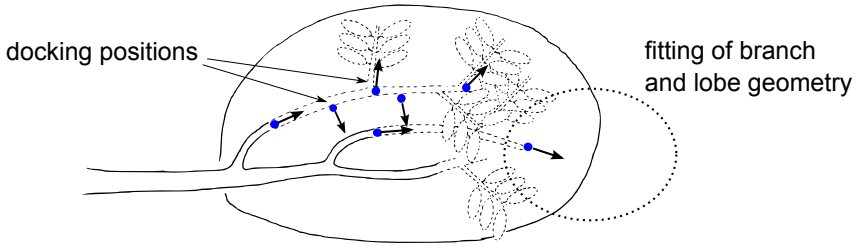


Figure 3.6: Selection of patches for cluster texturing: patches are selected by branch diameter and vertical angle (blue dots with arrows) and have to match the cluster shape (Figure 5 from [112]).

3.5 Application in Acquisition and Modeling

To demonstrate the applicability of the new tree representation, it is used to model massive amounts of trees acquired by a laser scanning device. The key to a high-quality representation with the cluster-based representation is the classification of individual tree models. The classification supports the retrieval of appropriate parameters (e.g., allometric values) for generating the skeletal structure and for assigning the cluster-patches to cluster-geometry.

3.5.1 Tree classification

This section introduces a supervised classification method, which classifies scans of trees taken from real urban environments. Note that tree classification remains a very complex and challenging problem, for which a general and robust algorithm is still hard to find.
The scanned point sets are classified by computing a number of features directly from the input points as well as from an approximate first reconstruction based on the approach from Livny *et al.* [113]. However, botanists also consider leaf shape

Figure 3.7: Texturing a cluster: while the cluster geometry serves as a constraint it is only approximated by the branching structure due to limited cluster-patches provided by the library (Figure 6 from [112]).

and texture when identifying a tree species [1]. Bark texture and multispectral images of the tree canopy also provide additional clues [97]. Thus the presented method cannot be generally applied to all kinds of trees and all kinds of contexts. Nevertheless, it works effectively considering the fact that only a limited number of species exists in a local area and there is only a certain spatial coherence of trees.

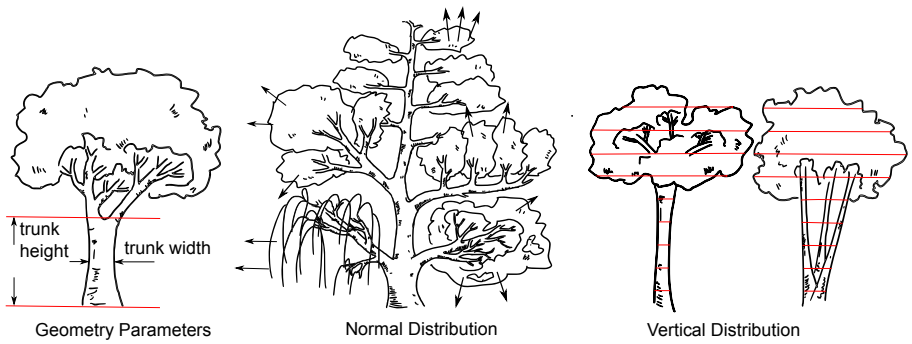


Figure 3.8: Effective features used for classification: tree shape parameters, distribution of normals (trees with different shape structure vary in the normal distribution) and point set distribution (Figure 8 from [112]).

For each tree over 200 features are computed, which are useful to distinguish different tree shapes. The values are combined to a parameter vector p and a *Joint Boost Classifier* [183] is trained with tree models that are typically found in urban environments. *Joint Boost* results in a vector h of probabilities describing the likelihood for a point set to belong to one of the species.

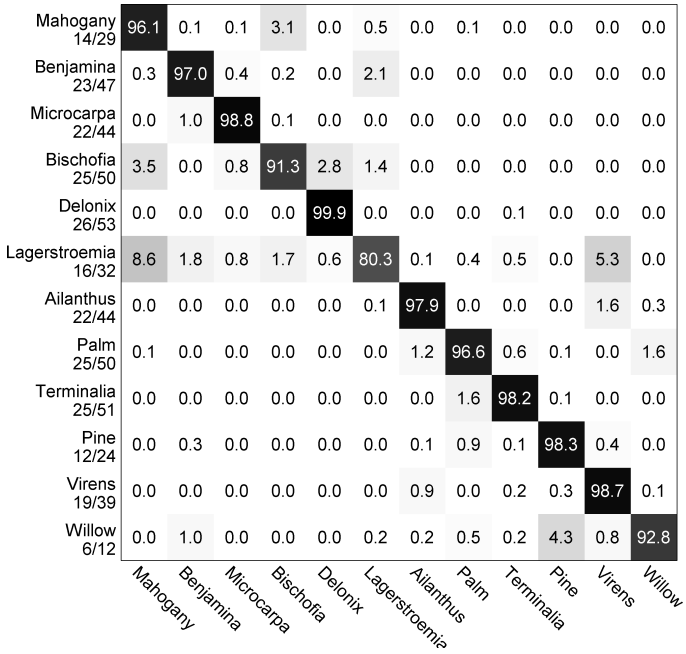


Figure 3.9: Recognition rate for the Joint Boost Classification with 12 species and 12–53 scans for each (training set size and total set size under the tree names). Each row describes the classification results for point sets of one species (Figure 7 from [112]).

To evaluate the effectiveness of the classification 12–53 individuals of twelve given species were scanned and manually classified. Half of them were used as a training set, the other half for testing. The average recognition rate was determined by 100 experiments where the models for the training sets and those for the test set were selected at random.
The average classification result is shown in Figure 3.9. Most species are correctly classified with an average recognition rate of 95.5%, which is high in comparison to usual rates in classification. Analyzing the classification results revealed that the most effective features are related to the following geometric properties (as also illustrated in Figure 3.8):

- trunk width, trunk height, crown width, height, ratio of tree width and height
- distribution of normal directions (computed in a neighborhood of each point)
- density distribution of the point set in vertical direction
- inhomogeneity of point set, number of main stems

Furthermore, trees in urban areas are not planted at random but typically in arrangements of the same species. Thus the Joint Boost classification was amended by a simple spatial voting mechanism to improve the results.
Confidence in a classification of Joint Boost is defined as when $h_1 > 1.5 \cdot h_2$ with h_1 being the highest probability value in the probability vector and h_2 the second

highest. If no confidence is given all classification results for the neighboring trees are collected within a radius of 25 meters (2–3 times the diameter of a typical tree). In case the neighbor examplars provide more reliable classification, a set of possible species is created and the candidate species with the highest probability is selected. The method works very well for trees in an urban landscape, here in most cases an average recognition rate of 98.8% can be reached. However, for mixed scenes with random species the results produce lower recognition rates, from an average of 95.5% to 89%.

3.5.2 Level-of-Detail and Rendering

In order to render dense scenes with many trees interactively, an efficient level of detail (LOD) mechanism is needed [48]. Cluster texturing (see Figure 3.7) lends itself to a progressive LOD mechanism – branchlets are added or removed based on the distance of the cluster to the camera. To avoid popping artifacts with changing LOD, *Stochastic Pruning* [50, 35] is applied, which proved to be an efficient LOD method for rendering trees, grass and other vegetal objects that consist of many small elements (leaves or grass blades). The method stochastically removes geometric elements and enlarges the rest in a way that enables the overall appearance to be maintained. In its original form the method requires that the complete geometry is produced before LOD can be applied. Here, when a dense scene with many trees has to be displayed, this is not possible.
In this event, we utilize interactive geometry production on the GPU combined with a dynamic variant of *Stochastic Pruning* to avoid this problem. The process benefits from the fact that the scene is assembled from a huge number of instances taken from a relatively small set of species and corresponding branchlets (20–30 per species). Each branching patch has a fixed number of leaves, which are stored in random order in a *Vertex Buffer Object (VBO)*. During stochastic pruning, only a prefix of the VBO for each instance of the patch is rendered. The leaves are faded in and out to avoid popping artifacts.

Patch Optimization

The overall number of patch instances affects the run-time performance on the GPU, where the branchlets have to be stored and accessed. For interactivity, it is desirable to minimize the number of instances. This can be achieved by forming larger branchlets from a set of smaller ones belonging to the same cluster. These newly generated branchlets are used for the respective clusters to reduce the total number of GPU calls and thus improve performance at the cost of memory.

Figure 3.10: Advantage of using cluster-based representation for laser scanned tree data (left to right: photograph, point set, conventional skeletal structure reconstruction (e.g., Livny et al. [2010]), cluster-based representation, skeletal geometry from cluster-based representation, full tree geometry after cluster texturing. Results from cluster-based representation better capture the characteristics of the input tree (Figure 9 from [112]).

Skeletal Graph

The segments of the tree skeleton (the edges of the explicit graph) are sorted by their thickness and also stored in a buffer. A prefix of these edges is rendered based on their distance to the camera. On the GPU, the screen space length of an edge is used to control the amount of the generated geometry by the tessellation shader.

3.6 Results

The method of processing trees with the cluster-based representation was tested with point data obtained from laser scanning. Results are illustrated in Figure 3.10. An advantage of clusters reconstructed from point sets is the proper synthesis of details that are not sufficiently captured in the original data, a common situation for scans. As long as points exist to define a cluster geometry, the internal structure of the tree can be synthesized by cluster texturing. Often, the synthesized structures better capture the characteristics of the species. Reconstruction methods, such as global optimization [113], applied to the mere input of point sets may produce structures that appear foreign or even erroneous, due to noisy and insufficient data.
A faithful reconstruction corresponds to the proper selection of the number and size of the clusters. The number of clusters is mainly determined by the 3D arrangement of the main branches of the tree, however, it can also be adjusted by a clustering parameter for each species available in the species library.
Figure 3.11 shows a tree with different amounts of clusters generated by changing the parameter value. When the rightful number of clusters (in this case 20) is used, the overall tree structure is reconstructed faithfully. In this case, the split between the skeletal structure and cluster geometry matches what can be faithfully extracted from the point set and what has to be synthesized by cluster texturing. When only one cluster is used, the tree structure is over-synthesized – the texturing mechanism fails to fill in the whole cluster and the synthesis inevitably mismatches some prominent skeletal structure in the original data. When a very large number of small clusters is used, the tree structure is under-synthesized, resulting in missing

Figure 3.11: Representation of a tree with a different number of clusters (adapted from [112], Figure 11). If only one cluster is used (left), the representation of the tree canopy becomes unspecific and texturing fails to fill the large cluster completely. It is designed for creating small random twigs and cannot build up large structures. With an appropriate number of clusters (middle) details are represented and the overall shape is textured. If too many clusters are used, the occupied volume of the clusters becomes insignificant such that the detailed structure in the crown area is insufficiently synthesized (right).

structural details in the crown area. Figure 3.16 shows that each species exhibits unique structural properties that yield a different number of clusters.
We have also applied our cluster-based representation to computer-generated plant models from the Xfrog library. We perform this experiment by simply converting the Xfrog model into a point cloud similar to laser scans, then apply the aforementioned method. Figure 3.15 shows the results that quite faithfully resemble the original models.
The cluster-based representation allows a variety of tree species to be reconstructed, ranging from leafy trees to special forms such as Weeping Willows, Palms, or Pine trees (see Figures 3.13, 3.16). In the paragraphs below, we present some numerical results obtained in our experiments.

Encoding

The description of a species in the library is about 200 kB and quite independent of the tree type. Less than one percent of the size is taken for the parameters of the system, the rest is needed for textures. As mentioned, the memory foot-

Figure 3.12: The location sometimes makes it impossible to scan a tree from all sides. In such a case, the point set lacks sufficient quality for the back parts, which results in an unbalanced canopy shape of the point cloud. From left to right: photograph of a pine tree; front view of the point set and the corresponding model; side view of the same model.

print of the cluster-based representation is quite small. For all species it is below 40 kB (see Table 3.2), allowing us to transmit many models per second via the internet.

Decoding

Reconstruction of the geometry from the model is also very efficient. Column 4 of Table 3.2 shows the time for a full reconstruction, which is between 500K triangles for the Delonix tree and 3M triangles for the Terminalia. However, it has to be noted that during an interactive session, due to the LOD, only a small portion of the triangles are shown for each frame. Even if the viewer is directly in front of a tree model, its distant parts are reduced by the LOD rendering. This is why we show the times for reconstructing a full model (3 –18 ms) and the model with LOD enabled (1 – 4 ms). A consumer PC with a GeForce GTX 480 graphics board was used for scene reconstruction.
Furthermore, in our implementation the tree models are not reconstructed for every frame but just updated, enabling the system to produce frame rates of about 20 – 30 fps with 40 different tree models and 10 – 15 fps with 250 trees in an urban scene as shown in the accompanying video. Figure 3.14 shows a screen shot of a scene to demonstrate the visual quality of the reconstructed LOD models.

Limitations

The introduced approach can reconstruct large sets of trees comprising a high variation of species and canopy shape. The cluster-based representation enables trees to be stored, transmitted and rendered very efficiently, while maintaining a high degree of visual complexity. However, there are some limitations caused by relying

Species	# of Clusters	Model Size	Time Reconst.	Time LOD
Mahogany	29	15 kB	7 ms	1 ms
Bischofia polycarpa	31	30 kB	9 ms	2 ms
Delonix	39	39 kB	3 ms	0.5 ms
Lagerstroemia	24	22 kB	9 ms	1.5 ms
Ailanthus altissima	24	30 kB	12 ms	2 ms
Palm	1	3 kB	4 ms	0.5 ms
Terminalia	80	19 kB	11 ms	2.5 ms
Pine	86	20 kB	18 ms	4 ms
Ficus Virens	72	23 kB	16 ms	3 ms
Willow	9	5 kB	8 ms	0.5 ms

Table 3.2: Memory footprint and reconstruction time for different species. For similar size trees, each species has a typical number of clusters representing each model.

on the quality of a scanned point set. Some trees have dense foliage (e.g. pine trees) and thus can only be scanned from a single side. In such a case, the point set lacks sufficient quality for the back parts, which results in an unbalanced canopy shape of the point cloud. Figure 3.12 illustrates this problem.
The cluster-based representation reconstructs the canopy shape even in such a situation but discards the inner structure of tiny branches. The difference becomes most obvious when the canopy consists of long and sparse twigs (see Willow in Figure 3.13).

3.7 Summary

In this chapter we introduced a new tree modeling paradigm, which represents a given model by two types of entities: an explicit representation of the main branching structure in form of a graph with allometric parameters and an implicit representation of the finer branching details and biomass. These clusters are textured with predefined patches at render time. Using species information, the models can be produced automatically, and even more faithfully, from scanned point sets as well as from commonly available tree models. This enables the scanning and processing of large areas of trees and the creation of models for interactive applications.
A key feature of the presented method is the classification of trees and the pregeneration of species information. Although the proposed classification already allows for sufficient categorization of points set, real applications require a more robust process. In the future we want to extend the approach to a much larger number of species, which will allow us to process most of the exemplars found in urban areas. Subsequently, classification would have to be adapted. Since the clusters in our model represent biomass we are certain that there is a more compact implicit representation of the clusters, in comparison to using α-Shapes, which would be able to produce geometry even faster. This would allow representation to be reduced even further.

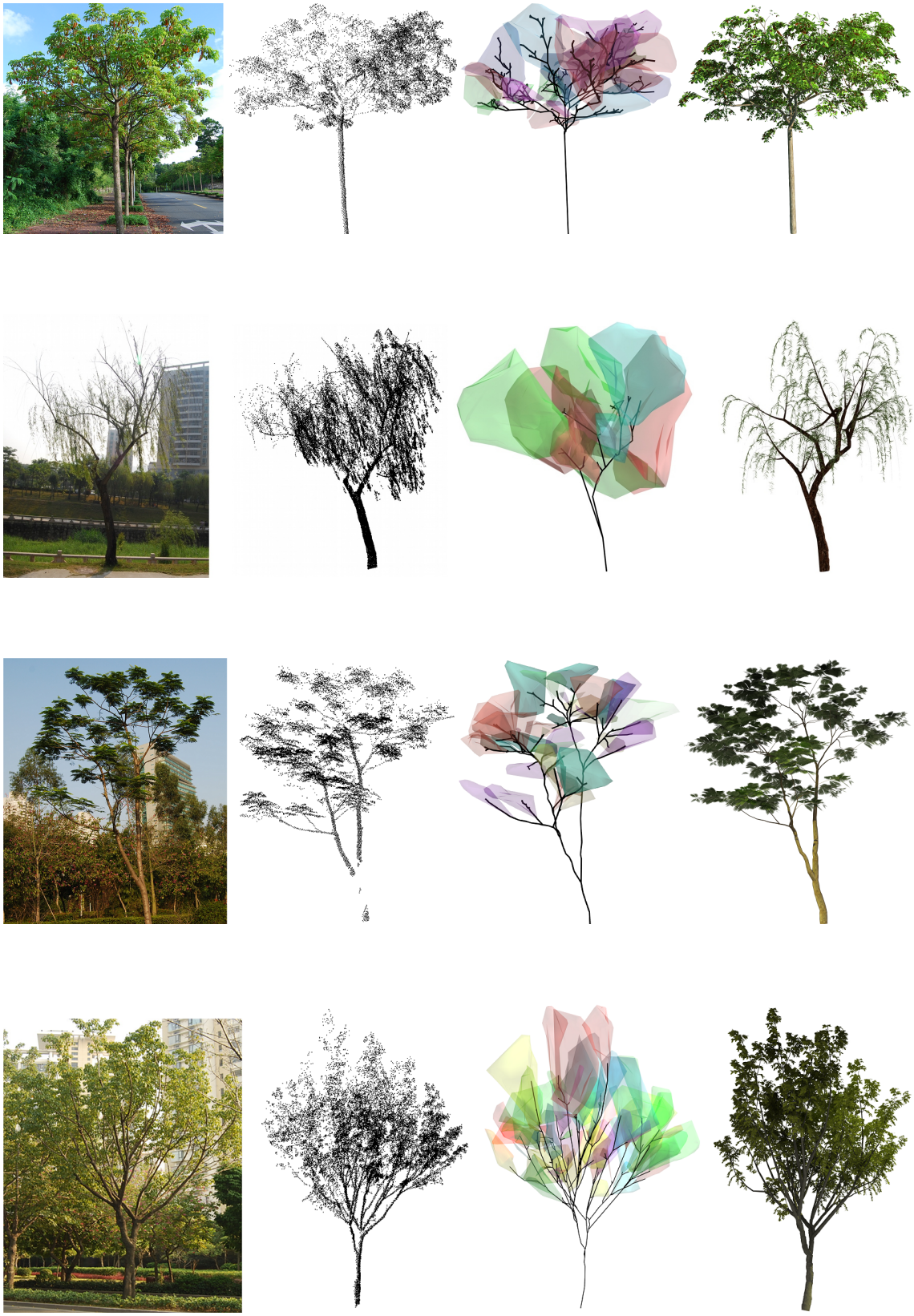


Figure 3.13: Several results of reconstructed trees: Bischofia Polycarpa, Willow, Delonix, Ficus Virens. As can be seen, the cluster-based representation faithfully approximates the trees global characteristics (Figure 10 from [112]).

Figure 3.14: A large scene shown in our interactive system. Thirty trees with the intermediate cluster-based representation (250kB) are synthesized on-the-fly with a total geometry of over 40M triangles, achieving 25 fps on a standard PC using an nVidia GeForce GTX 480 graphics board (Figure 13 from [112]).

(a) (b)

Figure 3.15: Re-representation of two Xfrog plant models. For the two image pairs (a) and (b), the right image is a re-representation of the left one, achieving high fidelity, while significantly reducing representation size (60 kB instead of 56 MB for the tree, and 45 kB instead of 47 MB for the shrub (Figure 12 from [112]).

Figure 3.16: Reconstruction of three tree models with special forms, the cluster-based representation adapts automatically to these situations (Figure 14 from [112]).

4
Interactive Self-Adapting Tree Models

In this chapter a dynamic tree modeling and representation technique is presented, which allows complex tree models to interact with their environment. This new method uses changes in the light distribution and proximity to solid obstacles and other trees as approximations of biologically motivated transformations on a skeletal representation of the tree's main branches and its procedurally generated foliage. Parts of the tree are transformed only when required, which makes it much faster than common algorithms such as *Open L-Systems* or *Space Colonization* methods while maintaining biological plausibility and high visual fidelity.
The input is a skeleton-based tree geometry, which can be computed from common tree production systems such as *Open L-Systems* or *Xfrog*, or models from laser-scanned point sets. The new approach enables content creators to interact directly with trees and create visually convincing ecosystems efficiently. New different interaction types are introduced and evaluated by comparing the transformation-based adaptations to growth simulation techniques.

4.1 Introduction

Botanical tree models are used in many application areas, including architecture, urban modeling, gaming, and movies. However, modeling the variety of tree shapes is a challenging problem because trees react to the surrounding environment in complex ways. Their shape is determined by their endogenous information (individual plants' genetics) and by exogenous influences (their environment). The same tree species that has a well-developed crown when grown in an open space might have a longer trunk and only a small tree crown when standing in a forest. The variety of plant shapes can be captured by growth models, but the growth simulation is time intensive and models typically require a large set of input parameters, which makes such approaches unsuitable for interactive design. As we have seen in Chapter 2 (Section 2.1) many different procedural tree modeling techniques have been published, but most of the resulting models are still static. If trees have to be combined, or if the environment changes, models have to be adapted manually or the procedural creation has to be rerun.

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

Figure 4.1: A 3D model of a tree is imported (a). Our system automatically computes a dynamic model that is able to react interactively to environmental changes such as trees growing together (b) or when obstacles are moved towards the tree and cast shadow on it (c)-(e) (Figure 1 from [139]).

A dynamic modeling and representation technique for trees is shown that aims at incorporating aspects of the tree's genotype into the models to allow them to react to the environment. In the past this was only possible using growth models such as *Open L-Systems* [128] or *Space Colonization* algorithms [135], which regrow the entire model and thus exhibit prohibitively long computing times. In contrast, we propose a technique that approximates growth behavior through biologically motivated transformations, allowing users to even interact with dozens of complex models in a scene. The method works with various kinds of input models, which are represented as polygonal surfaces, the only assumption made is that the models correspond to trees that were designed without the influence of other trees or obstacles. We have successfully applied this method to models generated by *Open L-systems* [128], to manually designed models from *Xfrog* [111], and to laser-scanned and reconstructed models.
To process a model, it is converted to the cluster-based representation described in Chapter 3. Defining a tree as a skeletal graph and a set of clusters allows the structure of the tree model to be modified efficiently when the environment changes. As we have seen, the foliage is represented as a number of leaf clusters. Tiny twigs and leaves inside these clusters are created procedurally using a GPU-based approach. Inspired by real trees, the changes in the light distribution are the most important factor for influencing tree models. The illumination of different parts of the tree is determined and then used to modify their geometric structure and the procedural content of the leaf clusters.

The method introduced in this chapter can be used to interactively model ecosystems consisting of up to some dozens of different trees. An example is shown in Figure 4.1. This model has been procedurally created and then imported to our system. The automatic adaptation of tree models to changing environmental factors releases content creators from dealing with tree parameters and makes it possible to construct complex scenes very efficiently. Because most of the geometry of the processed exemplars can be created on the fly using graphics hardware, the models can also be used in real-time scenarios such as games or simulators.

4.2 Overview

The effect of the environment on the youngest parts of the tree is usually not visible, as these parts have not had enough time to develop. This provides the motivation to simulate the effect of the environment mostly on the trunk and the main tree branches (the tree skeleton). The input is a tree model that was created as if it was grown in an open space. It is converted to the cluster-based representation introduced in Chapter 3. This approximate representation is the key for fast interaction with the environment and an efficient rendering of tree models while maintaining their visual fidelity.
The shape of a tree is the result of the competition for resources, the most important of which is light [162]. If a tree has been grown close to an obstacle, changes in the plant shape, such as bending or shedding, can occur. These effects are simulated by the dynamic representation introduced in this chapter.
The skeletal graph and the set of clusters react differently to environmental changes. The main branching skeleton is bent and pruned according to the light distribution. When the local light conditions change, the procedural content of the leaf clusters is modified. Small branches and twigs bend towards the light and can be pruned if the clusters interact with a solid obstacle. The shape of the clusters can be deformed when the tree is bent, which also affects the procedural content.
The remainder of this chapter is organized as follows. The next section introduces the necessary prerequisites for the method and the processing of the input models. Section 4.4 outlines the transformation-based modeling and interaction as well as their efficient implementation. An evaluation of the method is given in Section 4.5, which presents a number of results and compares them with procedural systems that also allow for interaction with their environment.

4.3 Tree Analysis

The response of individual branches and the sensitivity of the given input tree to changing environmental conditions is calculated from the tree's geometrical and topological information. It is therefore assumed that it has been grown in an isolated space with no external obstacles.
However, the shape of the input tree is affected by self-shadowing even when external obstacles are not present. To account for this effect, the environmental conditions that influenced the tree structure are estimated and intrinsic morphological properties are determined. These properties are used to construct a procedural model that defines the behavior of each branch. This model is controlled by a set of environmental parameters and is thus able to react dynamically to environmental changes.
To estimate morphological parameters of the tree branches, such as their desired orientation or their response to insufficient amounts of light, the environmental conditions that affected the structure of the input tree need to be estimated. As

mentioned above, this focuses on light distribution since this is the most important factor for tree growth. It is assumed that the light distribution is affected only by the tree itself: leaves and branches cast shadows that influence the growth of underlying branches.
The shadows within a tree change as the tree grows because new branches and leaves are constantly created and old ones die off. In order to estimate the growth parameters of the input tree, the temporal light conditions at different stages of its growth need to be computed. Such light conditions affect the local growth rates of branches and thus can be revealed from them.

4.3.1 Computing the Branch Age

An estimate of the branch age can be determined if the growth rates across the entire tree are known. The growth rate of an individual branch is determined by how many internodes (segments without buds) a given branch produces in one season. This growth rate can vary across the tree, as it is influenced by the amount of resources that a given branch receives during its growth. Therefore, to compute an approximation of the branch age it is required to estimate both the internode length and the growth rate of individual branches of the tree. Figure 4.2 (b) illustrates the corresponding data structure with assigned values representing the age of individual edges.
The length of an internode l_i is estimated from the distribution of distances between nearest branching nodes. The mean internode length is used for the estimation which is then the most significant peak point in the distribution found by using mean-shift clustering (Figure 4.2 (a)). To estimate the growth rate ν_s of a given branch segment s, we first compute a relative growth rate $\hat{\nu}_s$ as:

$$\hat{\nu}_s = \frac{d_{s,l}}{d_r - d_{s,r}}, \tag{4.1}$$

where $d_{s,l}$ is the distance from a given segment to its furthest leaf node, $d_{s,r}$ is the distance from the segment to the root of the tree, and d_r is the distance from the root of the tree to its furthest leaf node. The computed value specifies how much slower a given branch has to grow compared to the fastest growing branch in order to ensure that all branches reach their leaf nodes at the same time. The estimation of the relative growth rate loses its accuracy as we move closer to the leaf nodes; therefore, the above estimation is used only for segments whose distance to the leaf node is larger than a threshold distance $d_t = 0.2d_r$. For the remaining branch segments, we copy the relative growth rate from their parent branches.
When the relative growth rate is estimated for all branches, we compute the absolute growth rate $\nu_s = \hat{\nu}_s / \hat{\nu}_{min}$, where $\hat{\nu}_{min}$ is the minimum relative growth rate from all branch segments. The age of each branch t_s is then computed as

$$t_s = \frac{l_s}{l_i}\nu_s + t_{s_p}, \tag{4.2}$$

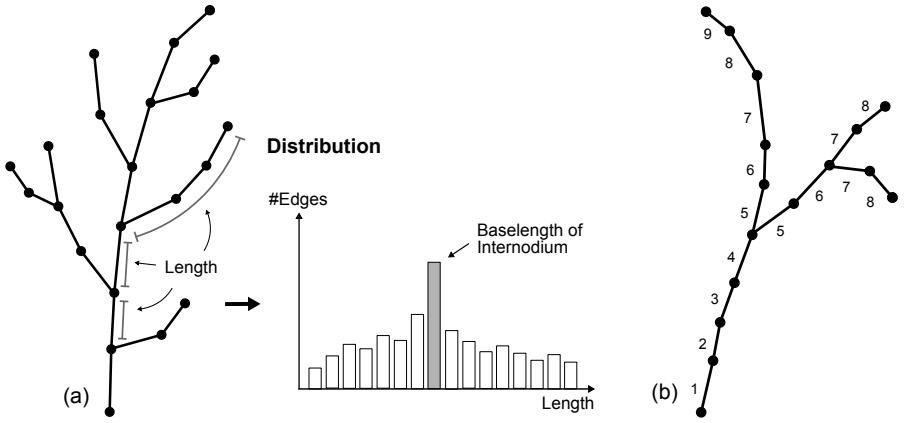


Figure 4.2: (a) The length of an internode is determined based on the distribution of distances between nearest branching nodes. The mean internode length is used for the estimation of the growth rate and the branch age. (b) A simple branching structure indicating the age of an edge assigned to the skeletal graph.

where l_s is the length of the branch segment and t_{s_p} is the age of its parent branch. The final estimated age of each branch segment is then clamped to the nearest lower integer, which represents the season in which a given branch segment was created.

4.3.2 Temporal Light Conditions

Once the branch age is known, the light conditions for the different stages of the tree growth can be estimated. It would be infeasible to simulate the effect of each leaf individually; therefore, the leaf distribution is approximated by virtual leaf clusters. The initial clusters are created from the clusters of the input tree. The clusters cast shadows onto the rest of the tree using a simplified light model, which accounts for the typical incident light within a day as proposed by Palubicki *et al.* [135]. Although advanced illumination models for plants exist [173], we use a simplified model suitable for fast calculation. To compute the light received at a given point $\mathbf{p}$, we integrate the incoming light from a hemisphere representing the sky:

$$i(\mathbf{p}) = c \int_0^{2\pi} \int_0^{\pi} I(\theta, \phi)(1 - O(\mathbf{p}, \theta, \phi)) \sin\theta \, \mathrm{d}\theta \mathrm{d}\phi, \tag{4.3}$$

where I is the amount of light coming from a specific direction (irradiance) and O is the visibility of the hemisphere from the given point $\mathbf{p}$, which is determined by the combined translucency γ of all obstacles that are in the given direction $O(\mathbf{p}, \theta, \phi) = 1 - \prod \gamma(\mathbf{p}, \theta, \phi)$. Finally, c is a normalization constant, which brings the amount of incoming light into range $[0, 1]$. The same approach is also used to compute the average light direction, where the integrand in the above equation is multiplied by a unit vector defined by (θ, ϕ).

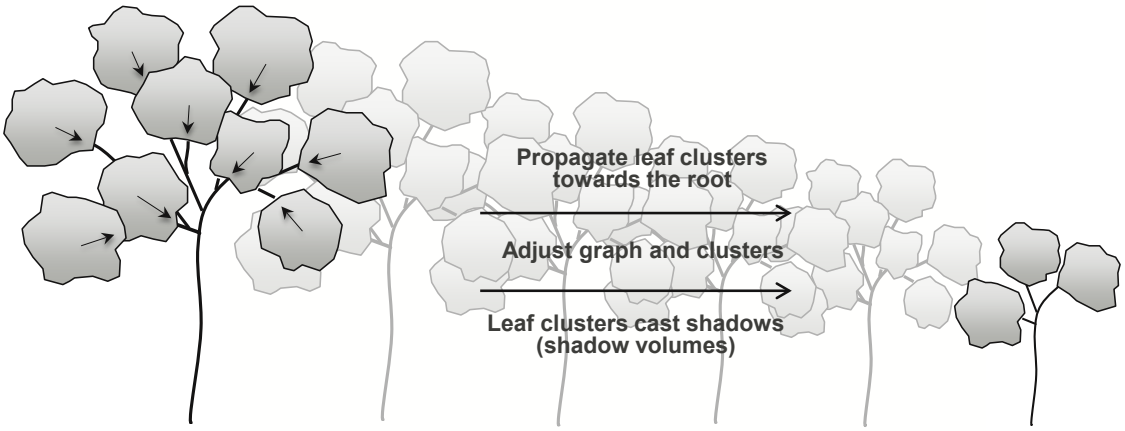


Figure 4.3: Estimation of the light exposition during growth. Each of the leaf clusters is propagated towards the root node. The skeletal graph and the leaf clusters are adjusted iteratively. Each leaf cluster serves as a shadow volume allowing us to estimate the amount of available light.

In this work, we approximate the intensity of the light coming from the sky by using $I(\theta, \phi) = \cos^2(\Delta\sigma(\theta, \phi))$, where $\Delta\sigma$ is the angular distance between a given direction (θ, ϕ) and the direction of the brightest point in the sky.

The shadows are computed and integrated into the scene using shadow volumes attached to each shadow caster. Shadow volumes represent a volume where the influence of the shadow caster is still significant, and they are used to quickly determine which obstacles should be included in Equation (4.3). Figure 4.4 illustrates these shadow volumes for different occluders. Since only simple geometric shapes are used to represent obstacles, it is possible to evaluate the light model analytically; for more complex cases a numerical solution might be employed such as one used in Měch *et al* [128] or Soler *et al.* [173]. To determine the intensity of cast shadows, the translucency of the leaf clusters γ_c is approximated from their radius r_c as:

$$\gamma_c = \gamma_{c_0}^{r_c}, \tag{4.4}$$

where γ_{c_0} is the base unit translucency of a leaf cluster. We use a rough species-independent approximation with $\gamma_{c_0} = 0.5$, while knowing that different species differ in their translucency.
To compute the light conditions at earlier stages of the tree life, the leaf clusters are propagated towards the root as illustrated in Figure 4.3. At each iteration the active threshold age is decreased by one, and all nodes are removed that are older than the threshold. The leaf clusters that contained the removed nodes are then propagated to the nodes that are now leaves. The center of each cluster c_n is determined by

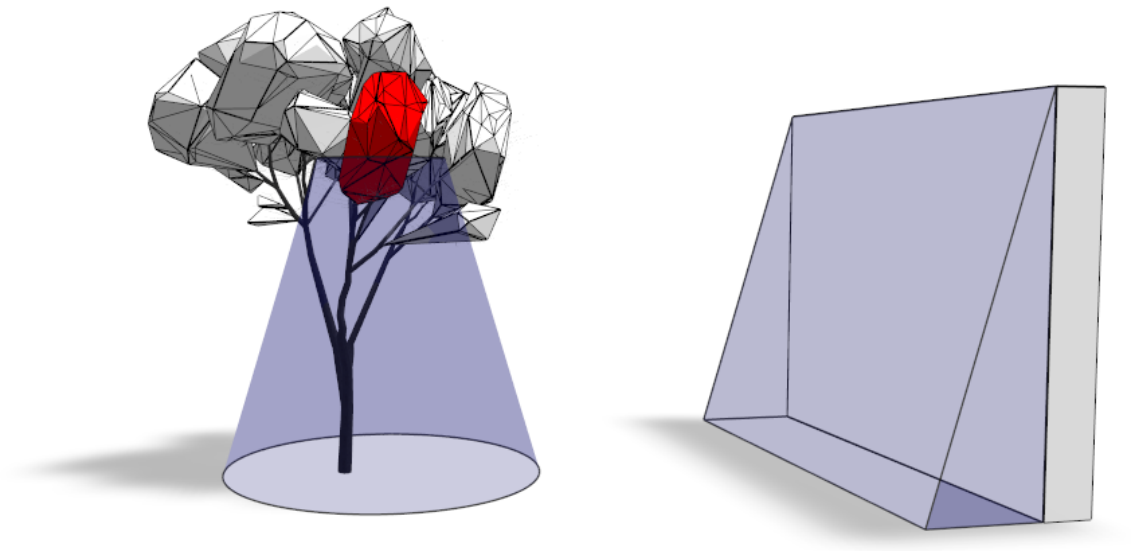


Figure 4.4: A visual representation of the shadow volumes. Each volume represents a region in which the influence of a shadow caster is still significant. Simple shapes enable us to quickly determine which obstacle influences the tree's growth. Leaf-clusters (left) and obstacles (right) cast different types of shadow volumes.

the centroid of the nodes assigned to it. The radius of a new cluster r_n is estimated from the properties of the set of removed leaf clusters in child nodes C_p as:

$$r_n = \sqrt[3]{\sum_{i \in C_p} r_i^3} \prod_{i \in C_p} \frac{d_n}{d_i}, \tag{4.5}$$

where r_i is the radius of a single child cluster, d_i is the distance from the root to the centroid of child leaf cluster i and d_n from the root to the new leaf cluster. The new leaf cluster has the combined volume of all child leaf clusters scaled down by the relative difference of their distances to the root of the tree.

4.3.3 Inverse Tropism

Once the environmental conditions at different stages of the tree's development are known, the effects of the environment on the shape of the input tree can be calculated. The first effect we are able to compute is the influence of tropisms on the tree growth. A tropism is the tendency of the branches to grow towards or away from some entity. In general the structure of the tree can be affected by different tropisms where each tropism τ is defined by a vector $\vec{t_\tau} = w_\tau \vec{t_\tau}$, where $\vec{t_\tau}$ is the unit direction of the tropism and w_τ is its strength.

The focus of the presented method lies on simulating phototropism and gravitropism. Phototropism is the tendency of a given branch to grow towards the direction of the light. The effect of phototropism for each branch is estimated at the time the branch

was growing using the temporal light model described above. Gravitropism controls bending of the branches either away from or towards gravity. While the strength of the tropisms is directly computed from the input tree, it is later exposed as a parameter that the user can modify to control the transformation behavior of the trees. A tropism that acts on a branch segment (growing in normalized direction $\vec{d_o}$) bends the branch into a new direction $\vec{h}$, which is computed as:

$$\vec{h} = w_s\vec{d_o} + (1 - w_s)\frac{\sum w_\tau \vec{t_\tau}}{\sum w_\tau} = w_s\vec{d_o} + (1 - w_s)\vec{t}, \tag{4.6}$$

where $\vec{t}$ is a linear combination of all tropisms, w_τ are the weights with $\sum w_\tau < 1$ and w_s as the weight of the original direction of the branch segment with length l_s. This weight can be determined by:

$$w_s = (1 - \sum w_\tau)^{\frac{l_s}{l_i}}, \tag{4.7}$$

where the exponent represents the accumulated effect of tropisms over the length of the branch segment, normalized by the internode length l_i.
In order to compute the effects of tropisms on the input tree when its environment changes, we first need to identify to what extent the tree structure was influenced by a certain tropism at the time when it was created. This problem can be referred to as computing the *inverse tropism*. The input branches are already affected by different tropisms and thus exibit a certain amount of bending; the goal is to compute how the branches would look if these effects were removed.
Computing the inverse tropism means solving Equation (4.6) for $\vec{d_o}$ with known $\vec{h}$, defined by the orientation of the branches in the input trees. The actual length of $\vec{h}$ in Equation (4.6) can vary because it is computed from a linear combination of different vectors. Therefore, we have to modify Equation (4.6) by introducing a line parameter p:

$$w_s\vec{d_o} = p\vec{h} - (1 - w_s)\vec{t}. \tag{4.8}$$

This equation has the following geometric interpretation (see Figure 4.5): we look for a direction of a vector $w_s\vec{d_0}$ which, when added to $(1 - w_s)\vec{t}$, results in a vector that lies on a line defined by $\vec{h}$. The line parameter p, for which the vector $\vec{d_o}$, has a unit size, can be found if we intersect the line with a sphere whose radius is equal to w_s. Each point located on the sphere fulfils the following equation:

$$(\vec{x} - \vec{t_w})^2 = w_s^2. \tag{4.9}$$

with $\vec{t_w}^2 = (1 - w_s)\vec{t}$. Inserting the line $p\vec{h}$ yields the following equation with parameter p being the maximal solution:

$$w_s^2 = p^2|\vec{h}|^2 - 2p(\vec{h} \cdot \vec{t_w}) + |\vec{t_w}|^2, \tag{4.10}$$

If Equation (4.10) does not have a solution, the first derivative is used to compute a value of p that defines the point on the line that is closest to the sphere. This

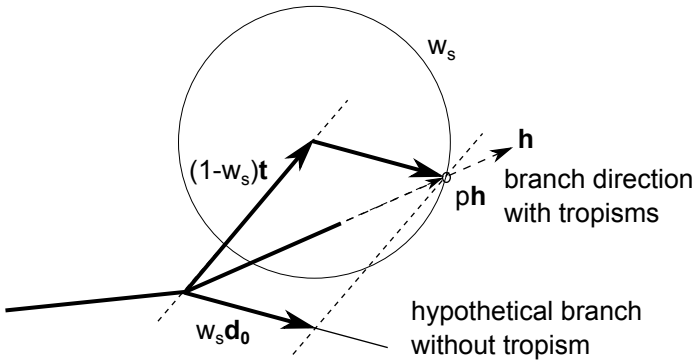


Figure 4.5: Computing the inverse tropism. Please refere to the text for a description of the vectors (Figure 3 from [139]).

value is then inserted into Equation (4.8) to compute the vector $\vec{d_o}$. Knowing this direction for every branch segment of the input tree, Equation (4.6) can be used to adjust the bending of the branches when the environment changes.

4.3.4 Pruning Estimation

Natural pruning influences the tree structure of most species and therefore it is crucial for us to determine when a given branch should be pruned. This is expressed in terms of the resources (photosynthates) gathered by the tree from the light. Please note that we cannot capture topiary.
Apical meristematic cells in a bud produce wood or plant organs according to the amount of light they receive. If the bud is shadowed for a long period of time, it slows down its activity. A branch that does not receive light for a longer period of time eventually dies off.
An approach similar to Palubicki *et al.* [135] is used, where the pruning of a branch is computed based on the sum of node distances to all leaf nodes l_t and the amount of resources gathered by their child leaf clusters ζ_t. A branch is pruned when the ratio ζ_t/l_t is smaller than some threshold value called *pruning factor* ψ. For a given branch segment s, the gathered amount of resources ζ_{t_s} is computed from the light that is received on the leaf clusters C_s that are located on the child nodes of a branch segment s:

$$\zeta_{t_s} = \sum_{c \in C_s} 2\pi r_c^2 i_c, \tag{4.11}$$

where r_c is the radius of a given leaf cluster and i_c is the normalized amount of light that the cluster receives.
Since the input is an already grown tree where the branches have been pruned, it is impossible to compute the pruning factor ψ directly. Instead, a local pruning factor $\psi_s(t)$ is computed for each branch segment s and for every stage of the tree growth t. The local pruning factor allows the minimum pruning factor to be determined when a given branch is not shed during growth. Since only a rough approximation

can be computed, we use the fifth percentile of the distribution of all $\psi_s(t)$ as the reference pruning factor ψ_{ref}. This gives us an estimation. The individual pruning factor is now computed for every branch s by

$$\psi_{s_{min}} = c_\psi \min(\psi_{ref}, \min_{\forall t}(\psi_s(t))), \tag{4.12}$$

where c_ψ is a user-controllable parameter that determines the strength of the pruning. We use a default value $c_\psi = 0.8$.

4.4 Dynamic Interaction

After analyzing the input tree to estimate growth behavior and pruning strength, we can efficiently model the interaction with its environment. During the interaction we first calculate the amount of changes in the environment; the tree response is then expressed by transforming the main branches by updating the shape of the leaf clusters and by modifying their procedural content.

4.4.1 Tree Graph Transformations

The transformations should represent changes in the tree growth. We transform individual branch segments according to their estimated age (Section 4.3.1) and as a reaction to the new light conditions using our temporal light model.
The direction of the incident light is used to generate the bending transformation from Equation (4.6). The rotation of a branch segment is propagated to its child branches. When a branch is transformed, it is also necessary to update those leaf clusters that may cast shadows onto younger branches.
The pruning transformations are computed after all branches of a given age have been transformed. When all branches on level t have been transformed, the resource allocation ζ_{t_s} for all branches with ages between 0 and t (see Equation (4.11)) is computed. The resources are then compared with the total length of all existing child branches and the pruning factor $\psi_s(t)$ is computed. All branches (and their children) with resources smaller than $\psi_{s_{min}}$ are pruned.

4.4.2 Leaf-Cluster-based Modeling

In order to simulate a cluster's response to light, the amount of light each cluster receives is computed. This information is then used to adjust the creation of branches within a cluster, their orientation, and the number of leaves per branch. As described in Chapter 3, the procedural creation is a repeating process of adding branchlets (small branches obtained from the input model) to some initial seed

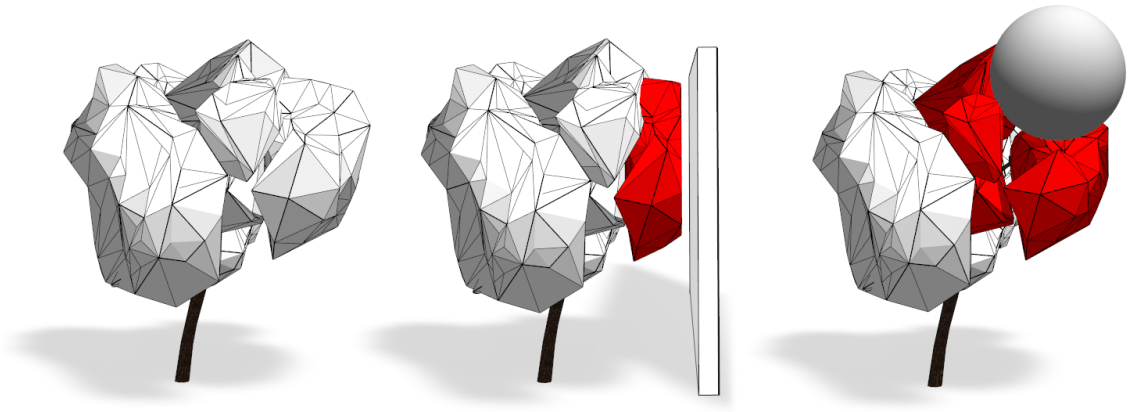


Figure 4.6: Cluster intersection after a collision with solid obstacles. Each leaf cluster dynamically interacts with changing environmental conditions. In the current implementation vertices are just moved away from the obstacle (Figure 4 from [139]).

points of the leaf cluster on the main branching skeleton. This process is parameterized by the desired cluster density. The relationship between the incoming light i and the normalized density ρ_l is denoted by

$$\rho_l = \frac{\rho_{l_0}}{\rho(i_{l_0})}\rho(i). \tag{4.13}$$

for leaf cluster l with an initial density ρ_{l_0} for the initial light valuei_{l_0}.

When a leaf cluster collides with an obstacle it is intersected as shown in Figure 4.7. The selection of branchlets is adjusted according to the new hull of the cluster on a frame-to-frame basis. Due to the approximative nature of the cluster filling process, small branches sometimes grow out of the hull and thus might enter an obstacle. More precise interactions would require complicated boundary checks, which cannot be performed at interactive rates.

In case the object is not solid (e.g. another tree model), the clusters are not intersected but overlap and share the space. Hence it is possible to achieve convincing branch canopies for close tree models (see Figure 4.18).

When a branch is bent or pruned away, the associated leaf clusters (filling geometry and also the envelope shape) are updated. The position of all leaf clusters is updated by computing the average offset of all their nodes from their initial position. This offset is then added to the cluster centroid. If parts of the tree graph that belong to a cluster are pruned, the respective seed points are removed, the procedural filling of the cluster is updated, and its translucency γ_c decreases:

$$\gamma_c' = \gamma_c^{\frac{n_c}{n_0}}, \tag{4.14}$$

where n_c is the current number of existing nodes and n_0 is the initial number. If all nodes of a cluster are removed, the leaf cluster itself is deleted.

Figure 4.7: The cluster-filling process depends on the availability of light. As can be seen in the figures, obstacles close to the tree model yield a lower foliage density.

4.4.3 Types of Interaction

The above-described transformations allow three types of interaction representing the most common scenarios that can be encountered during modeling: tree–obstacle interaction, tree–tree interaction, and global light interaction.
Tree–Obstacle Interaction occurs when a tree is moved close to an obstacle or vice versa. The obstacle then becomes a part of the environment that influences the tree by casting a shadow for the entire life span of its growth.
Tree–Tree Interaction is triggered when two or more trees are moved so close to each other that their mutual shadows influence their growth. For a proper simulation of this kind of interaction, the transformations of individual branches are processed in all involved trees in parallel. Branches are adjusted according to their age. If the maximal age of a tree is higher than its competitors, the processing time for the branches of the younger trees is changed in order to end up at the same completion time for all trees.
Global Light Interaction represents changes in the global light conditions of a scene. Such changes can occur when the scene is moved from a southern hemisphere to a northern one or when the orientation of the scene is altered. These kinds of changes are represented by altering the direction $I(\theta, \phi)$ in Equation (4.3). Whenever global light conditions change, the transformations for all trees have to be recomputed.
In most scenarios the above interactions happen at the same time. For example, when a wall is moved into a forest it transforms nearby trees, which in turn might affect the shape of their neighboring trees.

4.5 Evaluation

The complexity of tree models and the fact that trees exhibit a large amount of randomness complicate a thorough evaluation. To still quantify the introduced transformations of tree graphs compared to other methods known to create similar

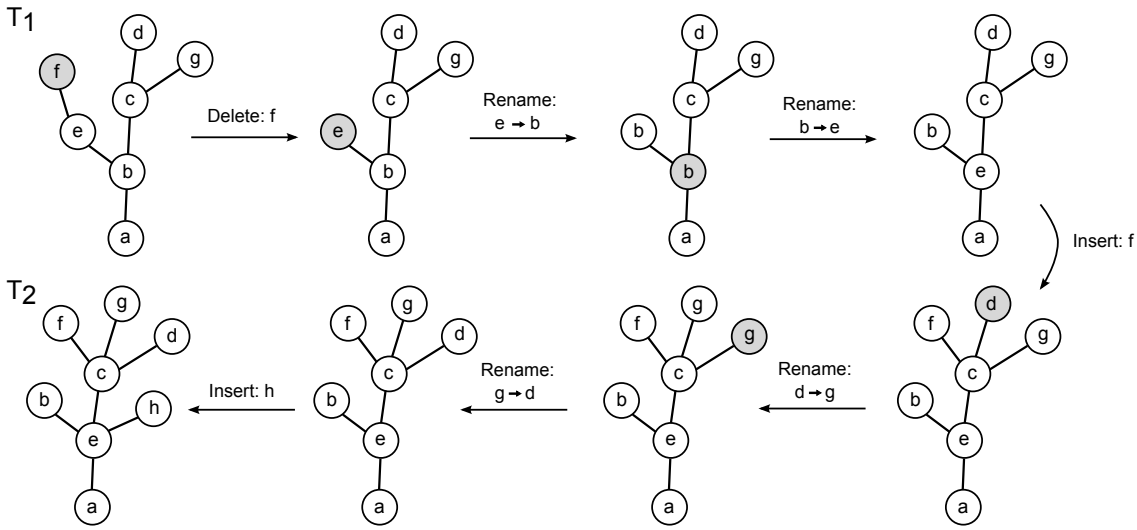


Figure 4.8: Transformation of a graph T_1 into a graph T_2 based on the edit transformations: delete, rename and insert (illustration adapted from [58], Figure 2(a)).

results an approach introduced by Ferraro and Godin [58] seems applicable. They formalize a branching structure into a graph $G = \{V, E\}$ which consists of a number of vertices V and a set of edges E, where each edge is defined by two vertices. Given an edge e_i and its corresponding vertices (v_1, v_2) a hierarchical relationship is produced, with v_1 being the ancestor of v_2. Each vertex is assigned a value or label, making G an unordered labeled graph [201], with one of the vertices being the root node v_{root}.
Hence, similarity of plant individuals can be measured by computing the similarity of their corresponding graphs, a field that has received considerable research attention in the past [178]. To efficiently quantify large sets of tree models only the main branching structure of the cluster-based representation is considered in the evaluation.

4.5.1 Edit Distance

Given two trees as unordered labeled graphs, their similarity can be measured as edit distance, the weighted minimum amount of operations, which converts one graph into the other, with the possible edit operations

- deletion of a node (the children of the node become the children of the parent node and the node is deleted),
- node insertion (inverse of deletion), or
- changing or renaming a node (which assigns a new label to the node).

An example for transforming a graph T_1 into another graph T_2 based on the aforementioned operations is illustrated in Figure 4.8. Each of the operations is assigned

a non-negative value representing the cost for this transformation. For trees the costs are proportional to the branch thickness and thus also to its age. Given a sequence S of n operations $(s_1, s_2, \ldots, s_n)$ the total cost $\gamma(S)$ for transforming one graph into the other is defined by:

$$\gamma_{total}(S) = \sum_{s_i \in S} \gamma(s_i) \tag{4.15}$$

The edit distance $D(T_1, T_2)$ that transforms one graph T_1 into the other T_2 is then denoted as the minimum cost of a sequence in $\mathcal{S}$, with $\mathcal{S}$ being the set possible edit operation sequences:

$$D(T_1, T_2) = \min_{S \in \mathcal{S}} \gamma(S) \tag{4.16}$$

Thus, dissimilarity between two tree models T_1 and T_2 can be expressed by the minimal costs between their graphs [58].

4.5.2 Mapping of Tree Graphs

Although comparing the costs of edit operations seems to be an interesting solution to evaluate the difference between two different graphs, it is impossible to directly apply this method for an evaluation. Finding the set of optimal operations that transform one graph into the other is considered to be an NP-complete problem [99], prohibiting a closed algorithmic solution.

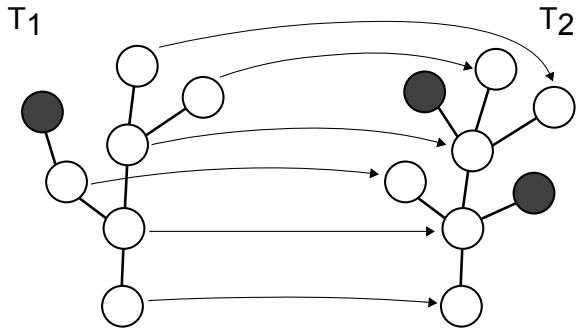


Figure 4.9: Mapping of a tree graph that transforms T_1 into T_2. Dark grey vertices represent inserted or deleted vertices, arrows indicate rename-operations (illustration adapted from [58], Figure 2(b)).

To determine the effect of different sequences of operations, Taï [178] generalized this problem and introduced *mappings* between two tree graphs. A mapping M is a set of ordered pairs (v_1, v_2) of vertices from the graphs $T_1 \times T_2$ and determining the costs of edit operations is synonymous with identifying the costs that are assigned to each mapping:

$$\gamma_{mapping}(M) = \sum_{(v_1,v_2) \in M} \gamma_{ren}(v_1, v_2) + \sum_{v_1 \in \bar{M}_1} \gamma_{ins}(v_1) + \sum_{v_2 \in \bar{M}_2} \gamma_{del}(v_2) \tag{4.17}$$

where M is the set of all possible mappings from T_1 to T_2; $\bar{M}_1$ and $\bar{M}_2$ denote vertices that cannot be found in M. $\gamma_{ren}(v_1, v_2)$ is the cost function for a renaming operation; $\gamma_{ins}(v_1)$ and $\gamma_{del}(v_2)$ represent cost functions for insert and delete operations. An example of a mapping that transforms a tree graph T_1 into a tree graph T_2 is illustrated in Figure 4.9. As shown in Ferraro and Godin [58] finding the mapping with the minimum costs is equivalent to solving an optimization problem:

$$D(T_1, T_2) = \min_{S \in \mathcal{S}} \gamma(S) = \min_{M \in \mathcal{M}} \gamma(M) \tag{4.18}$$

To reduce the algorithmic complexity, Zhang [201] as well as Tanaka and Tanaka [182] proposed to preserve structural properties. These structural preservations also support a biological point of view; when determining similarity not all of the possible mappings are of interest. Only mappings that support valid structural properties are meaningful and allow similarity to be measured. Comparing leaf nodes of one graph to the main branches of the other does not yield plausible results and hence does not require an evaluation of the cost fuction. Considering structural conditions prohibits certain mappings and thus enables us to find an algorithmic solution in polynomial time.
Ferraro and Godin [58] proposed utilizing the relationship of hierarchy in the graph structure to reduce possible mappings. They define sub-trees based on the notion of the least common ancestor (lca) in sets of nodes and state:
"In a tree-graph, the least common ancestor of v_1 and v_2 denoted as $lca(v_1, v_2)$ is a common ancestor of v_1 and v_2 such that every common ancestor w and v_1 and v_2 satisfies $w \leq lca(v_1, v_2)$" [58, p.449].
Thus, a valid mapping is a set of ordered pairs (v_1, v_2) of vertices satisfying: $v_1 \in T_1$, $v_2 \in T_2$ and for every pair (v_1, v_2), (w_1, w_2), (u_1, u_2) in M

$$v_1 = w_1 \Leftrightarrow v_2 = w_2 \tag{4.19}$$

$$v_1 \leq w_1 \Leftrightarrow v_2 \leq w_2 \tag{4.20}$$

$$lca(v_1, w_1) < u_1 \Leftrightarrow lca(v_2, w_2) < u_2. \tag{4.21}$$

According to Ferraro and Godin [58] condition (4.20) and (4.21) ensure a valid ancestor relationship as well as a conservation of the branching system. Figure 4.10 illustrates valid and forbidden matching functions. Given a valid set of mappings $\mathcal{M}$ the dissimilarity of two graphs T_1 and T_2 can thus be determined by

$$D(T_1, T_2) = \min_{M \in \mathcal{M}} (\gamma(M)) \tag{4.22}$$

Measuring the distance between sequences of edit operations requires defining a cost function for each of the available operations. The next section provides a formal definition of such a measure.

4.5.3 Cost Function

Effectively quantifying the similarity of plants and trees requires the careful definition of the cost function of each possible edit operation. Ferraro and Godin [58] proposed utilizing the *Levenstein's Distance* [107], a simple binary distance, which assigns the costs of *one* to the delete and insert operation and the costs of *null* to the renaming operation. Applying this distance considers only the topological structure and thus does not necessarily yield appropriate results.
Each of the vertices in the graphs T_1 and T_2 can have a number n of numerical attributes $a_1(v), a_2(v), ..., a_n(v)$ which can also be considered in the evaluation of the cost function. To account for similar importance of each of the attributes, they need to be standardized. This includes calculating the mean value of each variable over the entire graph G as well as quantifying the distribution based on the standard deviation:

$$s_i = \sqrt{\frac{1}{n-1} \sum_{v \in T_1 \cup T_2} \Big(a_i(v) - n_i \Big)} \tag{4.23}$$

Based on the assumption that s_i is not zero the standardized values for each vertex v are defined by:

$$f_i(v) = \frac{a_i(v) - m_i}{s_i} \tag{4.24}$$

In case s_i is equal to zero, f_i is a constant. The distance between a pair of two vertices (v_1, v_2) can be defined as metric distance in n-dimensional space. For renaming, this distance is often computed as the *Manhatten Distance*:

$$d_{ren}(v_1, v_2) = \sum_{i=1}^{n} |f_i(v_1) - f_i(v_2)| \tag{4.25}$$

The costs for insert and delete can be defined in many different ways, however, it may be chosen to be proportional to the sum of absolute values of the attributes:

$$d_{ins}(v1) = \sum_{i=1}^{n} |f_i(v_1)| \qquad d_{del}(v2) = \sum_{i=1}^{n} |f_i(v_2)| \tag{4.26}$$

The cost function directly relates to the distances of the attributes and thus the natural parameters of trees and plants. In this work, the length and the thicknesses of branches are considered as attributes. A more detailed description of defining the cost function is provided in Ferraro and Godin [58].

4.5.4 Evaluation Process

It would be impractical to compare artificially-generated models to real trees as it is difficult to correctly estimate their environmental conditions. Therefore, we

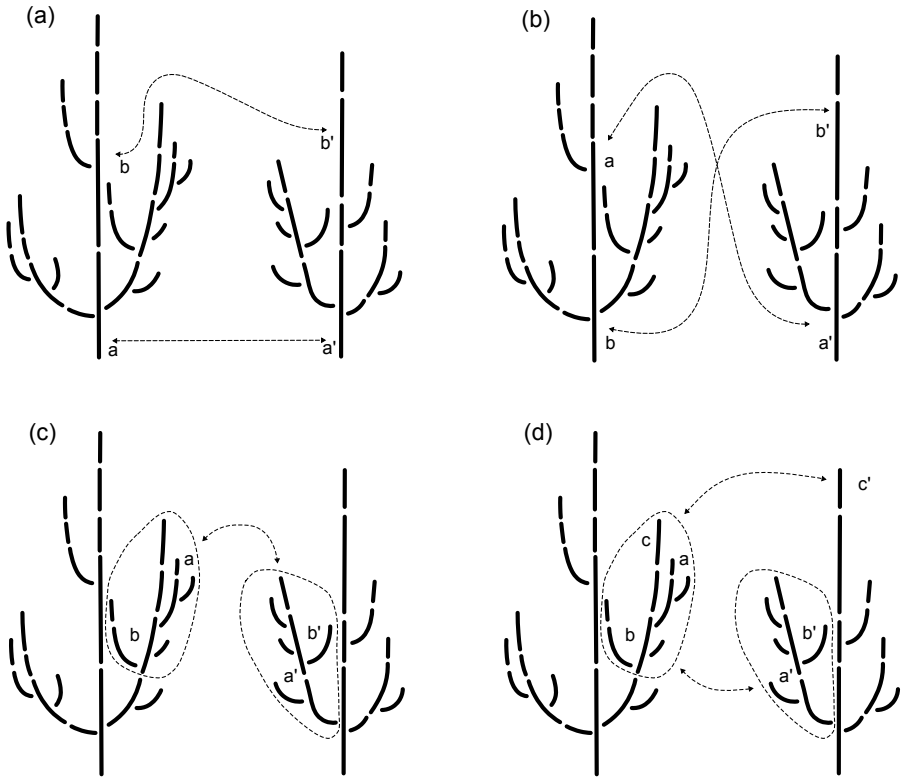


Figure 4.10: Illustration adapted from Ferraro and Godin [58] (Figure 3) showing different mapping scenarios; (a) preservation of anchestor relationship, (b) non-preservation of ancestor relationship, (c) preservation of branching system, (d) non-preservation of branching system.

use trees created by a growth model similar to the one proposed by Palubicki *et al.* [135]. In order to compare the trees, we first create solitary tree models using the growth model and use them as an input for our system. We created three types of trees that differ in their tropisms. Tree A presents 0.25 phototropism and 0.12 gravitropism, the tropisms for Tree B are 0.45 and 0.44, respectively and Tree C presents 0.03 phototropism and 0.23 gravitropism. The first two trees are the same age and present the same pruning factor. The last tree is older having a smaller pruning factor. We created three tree sets for each tree type:

- Set O: Trees created using the growth model with no obstacles where only light and self-shadowing influences the growth.
- Set G: Trees with the same growth parameters as O grown using the growth model close to a wall.
- Set T: Trees with the same growth parameters as O with the shape calculated using our system under the same conditions as G.

Four different groups of fifty values each were computed (see Appendix A). The values represent distances between pairs of trees selected from the sets described

Type	Groups	T-G	G-G	T-O	T-T
A	Mean	236.29	229.65	347.19	225.38
	Standard Deviation	35.27	52.71	91.33	33.94
	p-value	$p = 0.46$		$p < 0.001$	
B	Mean	431.73	396.81	471.63	413.64
	Standard Deviation	112.79	71.32	70.65	90.76
	p-value	$p = 0.06$		$p < 0.001$	
C	Mean	756.27	841.79	927.64	550.63
	Standard Deviation	266.84	239.57	207.85	83.33
	p-value	$p = 0.13$		$p < 0.001$	

Table 4.1: Results obtained for the different groups T-T, T-O, T-G and G-G. The letters represent the groups of the pairs of trees that were compared.

above so that no tree was used twice. The different groups were *T-T*, *T-O*, *T-G* and *G-G*. The letters represent the groups for the pairs of trees that were compared.
In order to show that there is no significant difference between the transformed trees from set T and the trees grown by the growth model from set G, we performed a two-tailed t-test between *T-G* and *G-G* with an alpha of 0.05. To show that the transformed trees from set T are significantly different to the trees from O, a t-test is performed between groups *T-O* and *T-T*. Table 4.1 summarizes the obtained results.
This suggests a significant difference between trees grown without constraints versus trees grown next to a wall. On the other hand, there is no significant difference between the adapted trees grown with the growth model and the trees transformed with our approach under the same conditions.
Furthermore, Figure 4.11 shows a visual comparison of the changes obtained by the growth model and our transformations. The models were grown in an open space and then imported to our system (left). Next, they were grown in different light conditions using the growth model (middle) and also transformed using our approach (right).

4.6 Implementation and Results

The system presented in this chapter is implemented in C++ using OpenGL and GLSL. All examples were generated on a desktop computer equipped with Intel i7 CPU @ 3.7GHz with 16GB of memory. Most of the rendering was done directly on the GPU (nVidia GeForce GTX 580 with a 1.5GB of dedicated memory).
The visual appearance of tree models is mostly determined by the structure of their main branches and not by the exact structure of leaf clusters. Therefore a threshold for the thickness of branches is defined to determine whether or not a given branch should be rendered. All branches with a thickness above the threshold are stored in

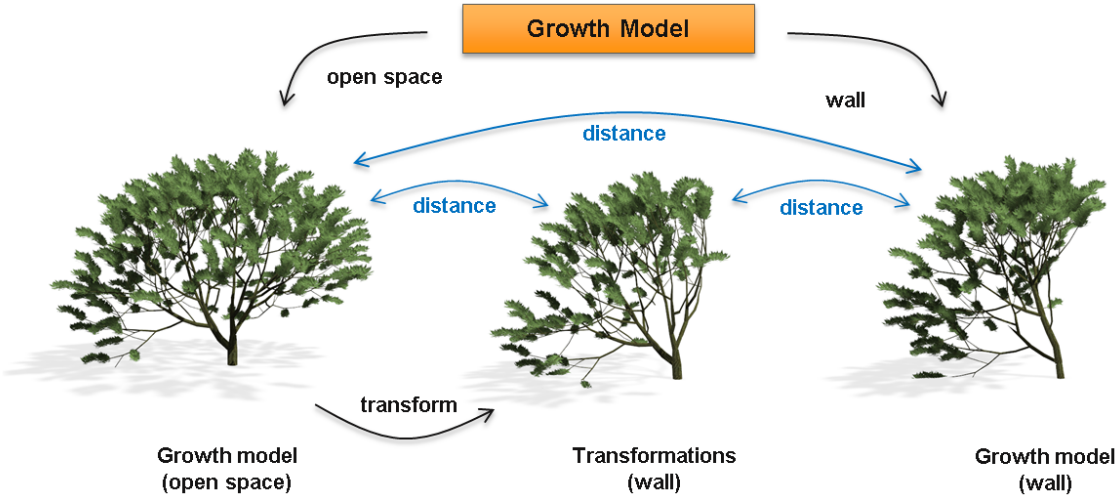
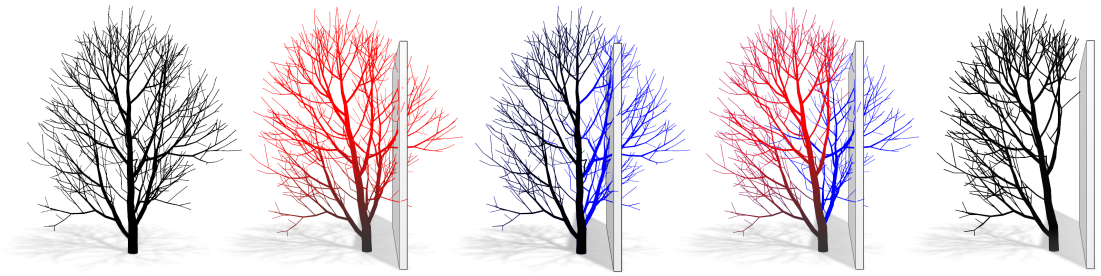


Figure 4.11: Evaluation of transformations: a tree model is generated with a growth model without considering environmental constraints. This model is transformed based on the transformation approach and compared against a model generated grown close to a wall. The similarity of two models is measured by utilizing the constraint edit distance proposed by Ferraro and Godin [58].

a tree graph, while the smaller branches are removed from the tree and converted into the leaf-clusters. To interact with large amounts of tree models in real time, a set of level of detail (LOD) techniques is applied. The amount of produced geometry depends on the cluster size, the light situation, and the LOD stage. If the tree is far away from the camera, only a small subset of leaves is produced and scaled according to stochastic pruning, as introduced by Cook *et al.* [35]. The tree foliage is rendered with *Alpha-to-Coverage*; an efficient method for layers of textures containing large numbers of transparent *Texels* (Chapter 6, Section 6.2).
Since trees exhibit a large amount of self-similarity, it is possible to approximate the leaf clusters using small subsets of branchlets (branch patches), which are instantiated. A set of branchlets from the input model is used to populate the volumes. These patches are stored in a texture buffer on the GPU, and the main branching structure and the patch geometry of the leaf clusters are combined to make a complete graph of the tree.
In order to apply the transformation to the tree graph, the main branching structure is stored in CPU memory, which is mapped to a frequently updated vertex buffer object on the GPU. The geometry of the skeleton is represented by generalized cylinders. To allow rendering of many tree models, the graph and mesh generation is adjusted on a frame-to-frame basis. A more detailed explanation is given in Chapter 6, Section 6.1.

Table 4.2 shows a comparison of construction times using Open L-Systems and the transformation-based approach. Twenty models per group of tree ages 10, 20, and 30 were grown with an Open L-System, leading to an increasingly complex geometry. The growth times were recorded and averaged. Another three groups of twenty trees were input into our system and the transformation times were recorded. It is



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

Figure 4.12: The environmental effect of a shadow cast by a wall (Figure 6 from [139]). The color represents the difference between the input tree (a) and the transformed versions. The amount of bending expressed in red (b), pruned branches colored blue (c), both transformations (d), the final model (e).

Tree age	Average nodes	Growth	Transforms
10 years	530	216.3 ms	3.82 ms
20 years	2541	8855.6 ms	50.5 ms
30 years	9134	61,843.9 ms	222.8 ms

Table 4.2: Complexity and simulation time for trees in different ages.

important to note that the simulation applies to the regrowth of the entire tree from scratch, while the transformations are applied only in the affected areas. Generally, transforming a tree model is two orders of magnitude faster.

4.6.1 Results

The first example in Figure 4.12 demonstrates the environmental effect of the shadow cast on a tree by a wall. A tree model grown in open conditions is compared with transformed models, and the amount of displacement of each node is expressed as color. As expected, the biggest change is at the tip of the tree because the error accumulates through the main skeleton.
Figure 4.13 shows three different species with their reaction to an obstacle. The original trees were reconstructed from LiDAR scans (Chapter 3). The trees react to the proximity of the shadow by bending away (light seeking) and by shedding branches that are close to the wall. Figure 4.14 shows the results for two Xfrog models.
The results shown so far have demonstrated the transformation process with typical European and North American trees. However, more special trees, such as palm trees or pines, are shown in Figure 4.19. Their graph structure has no lateral branches, it is a line for the main branches and another line through each of the leaves; the clusters cannot be applied here. The models were generated using Xfrog, and their reaction to the environment is still predictable. The palm tree bends in the direction opposite

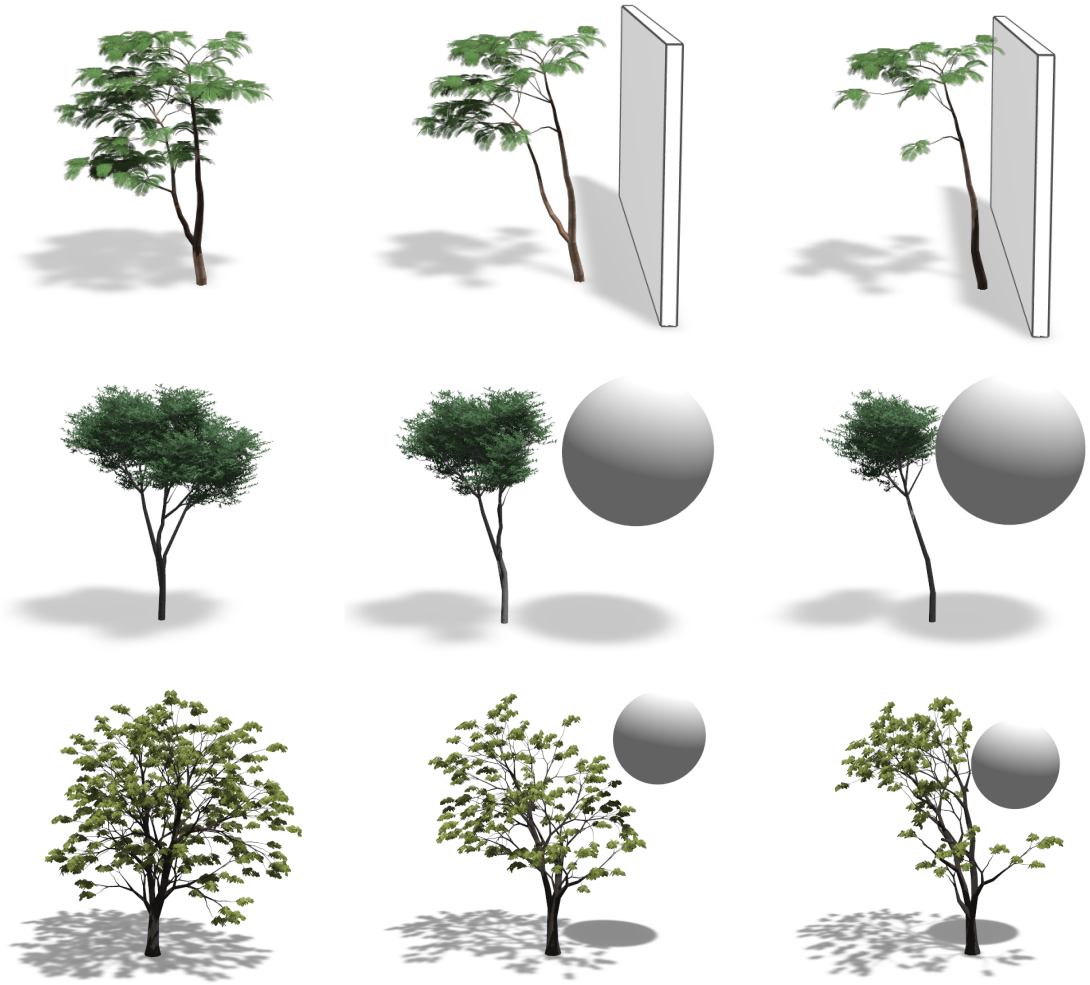


Figure 4.13: Different tree models (LiDAR, Xfrog) exposed to changing conditions. As the obstacle moves close, the tree bends its shape and some branches are pruned (adapted from [139], Figure 7).

to the shadow in an attempt to capture more light. Pine trees do not usually bend and react only to the lack of light on the lower branches. As expected, low branches are pruned while the complete branching structure at the top is maintained.
A number of different input models (Open L-Systems, Xfrog, LiDAR scan) is shown in Figure 4.15, which are frames generated with the implemented prototype. The trees are affected by mutual shadowing as well as by the walls enclosing them from two sides. The branches in the ecosystem fill the available space as they would in the case of a real ecosystem.
The example in Figure 4.18 shows two mahogany trees modeled in Xfrog that are moved close to each other. The competition for resources lets a single crown emerge in which each tree contributes in part. Figure 4.16 shows a box that is moved into a forest. Although it is not possible to show such big scenes with all interactions and transformations interactively (this scene had 5 fps on a test system), the tree shapes adapt convincingly and a user is still able to interact with the scene.

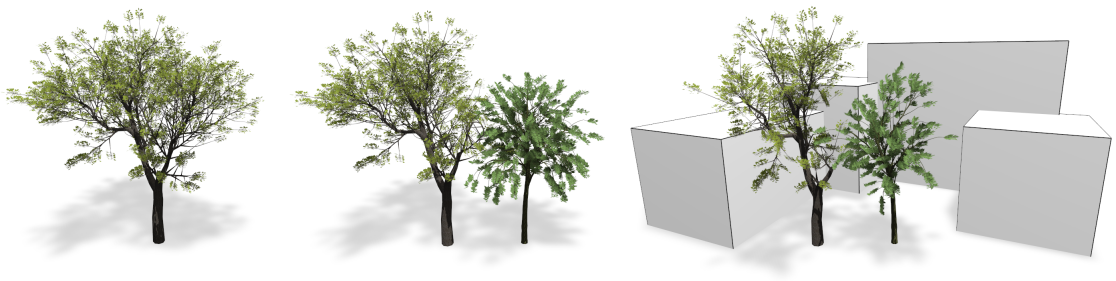


Figure 4.14: Xfrog tree models interacting to changing environmental conditions (Figure 9 from [139]).

4.7 Summary

The introduced framework allows complex tree models to be transformed dynamically and interactively in real-time. All kinds of polygonal input models can be converted into this representation. The influences of tropisms as well as other environmental effects such as shadowing of the input model are estimated sufficiently. When the models are created they react to obstacles and changes in the lighting. An efficient implementation allows even very complex scenes to be manipulated at an interactive rate.

The method, however, does not yet allow new branches to be produced for the main skeleton. Only the basic branching structure is deformed and manipulated. Furthermore, the input is limited to solitary tree models since the method is not able to hypothesize branches of the main skeleton that eventually die off. An example (Figure 4.20, left) shows a LiDAR scanned and reconstructed tree that has already been severely modified either by some early age trauma or by growing close to an obstacle. This tree shows an unnatural bending towards the obstacle that is partially alleviated by the transformations, but still prevails as the main growth direction. The resulting tree (Figure 4.20, right) does not seem very natural.

Another limitation is the number of effects currently supported by the system. So far only environmental factors such as light and tropisms are included, but no wind effects or nutrition changes in the soil. Furthermore, the system could be evaluated better by comparing it to real-world trees.

Currently the system needs user-defined parameters, such as the amount of newly added tropisms for the tree models. So far we do not have a general-purpose GPU-based modeling process for the procedural content. A pre-defined species library is used for the parameters, which allows a number of foliage types - but not all - to be produced.

Figure 4.15: A small ecosystem demonstrating various input models interacting. Models from different modeling paradigms (i.e. Open L-Systems, Xfrog and LiDAR reconstructions) were used in this example (Figure 8 from [139]).

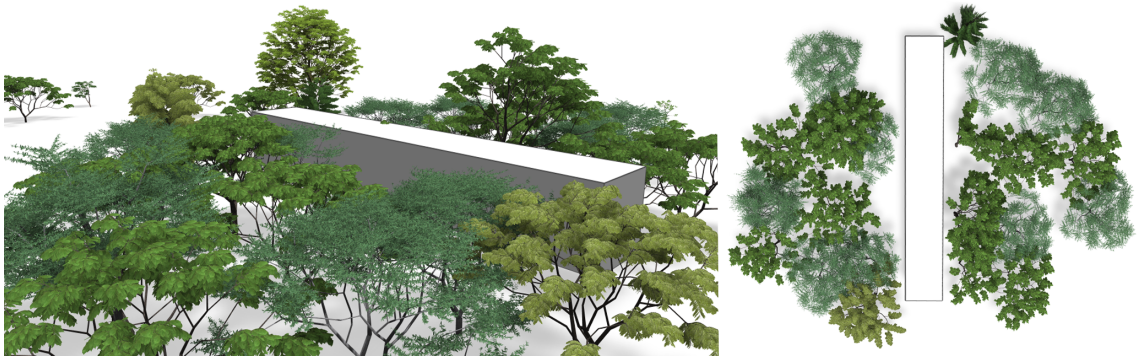


Figure 4.16: Influence of an obstacle moved into a dense ecosystem (Figure 13 from [139]).

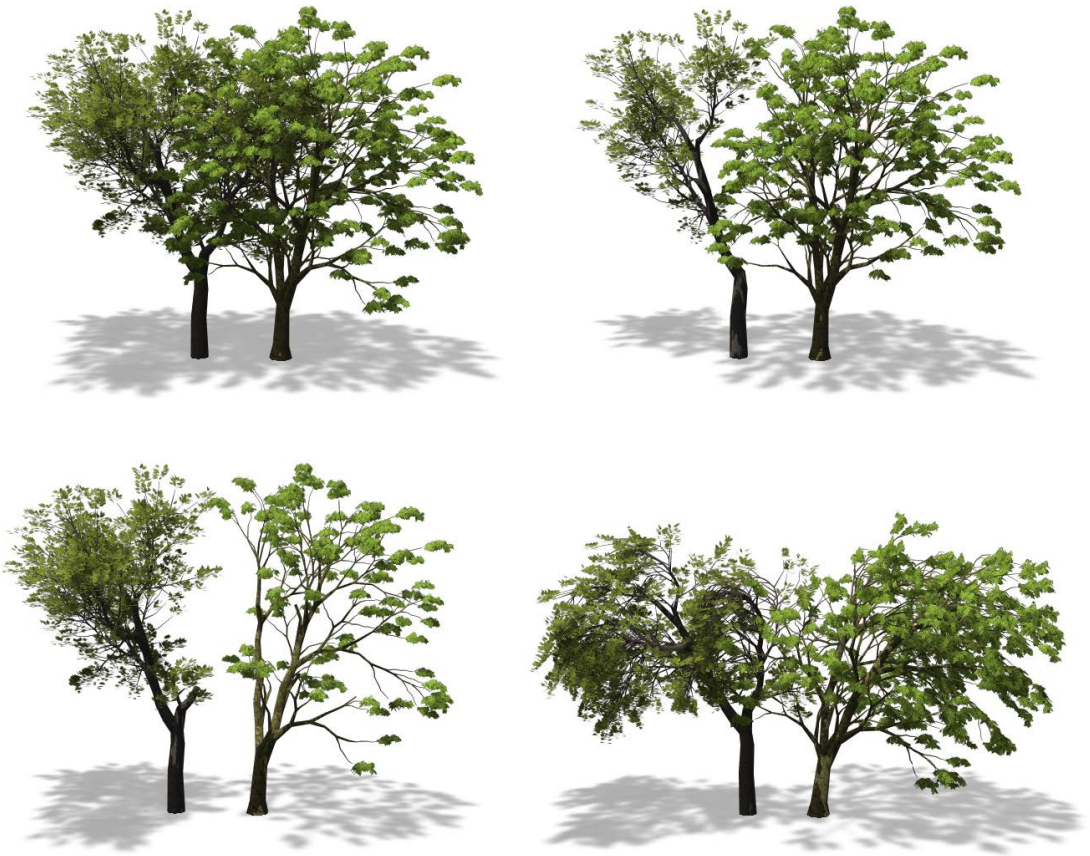


Figure 4.17: Two Xfrog trees (left to right): static; combination of bending and pruning; strong pruning; exaggerated bending (Figure 12 from [139]).

Figure 4.18: Two procedurally generated trees, which grew close to each other, form a crown that resembles a single tree (Figure 11 from [139]).

Figure 4.19: Special tree models interacting with obstacles (adapted from [139], Figure 10).

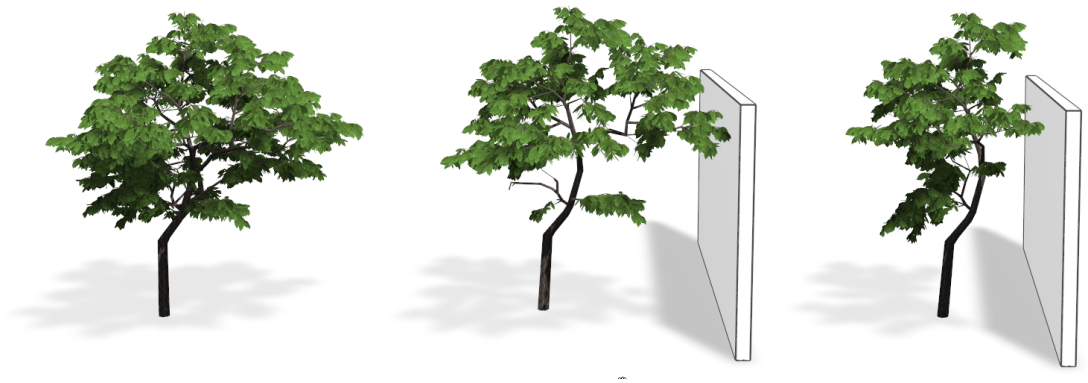


Figure 4.20: A tree reconstructed from LiDAR data that is already bent. Such models often cannot be altered properly (Figure 14 from [139]).

5

Capturing and Animating Tree Growth

In this chapter we introduce an automatic method for generating a continuous number of developmental stages from a single input tree model, approximating the tree's natural growth. The tree model is analyzed and a graph-based description of its skeleton is determined. Based on structural similarity, branches are added where pruning has been applied or branches have died off over time. Botanic growth models and allometric rules enable us to produce convincing animations from a young tree, which converges to the given model. The method is fully automatic and thus even supports animating the growth of entire forest stands. In addition, the method offers new forms of editing through selectively applying growth development to individual parts of a plant. This dramatically reduces the tedium of editing such complex models and opens up what we call a *Growth Space*: a space for developmental edits that influences the model in a natural way.

5.1 Introduction

As trees and plants belong to almost every animated movie or computer game, modeling complex natural scenes and objects was actively investigated for many years. Various methods have been proposed that successfully ease the modeling process while still creating realistic models. Today, thousands of such tree models can be found in large libraries that encompass all sorts of plants. While their visual fidelity has steadily improved over time, most of these models are still stored in the form of static geometries that cannot be altered easily, if needed. Specialized programs have to be used and sets of quite specific parameters have to be known to create variations or even animations – a task typically much too tedious for most content creators.

Most polygonal tree models, generated by *L-Systems*, *Laser-scanning*, *Xfrog* or other systems, can serve as the input to the proposed system. Using a mesh contraction algorithm we reduce them to a graph structure that represents the tree skeleton. The allometry – the geometric relations within the tree – and the branching statistics can be obtained from this data structure.

During tree growth the lower branches of a tree typically die off or are pruned,

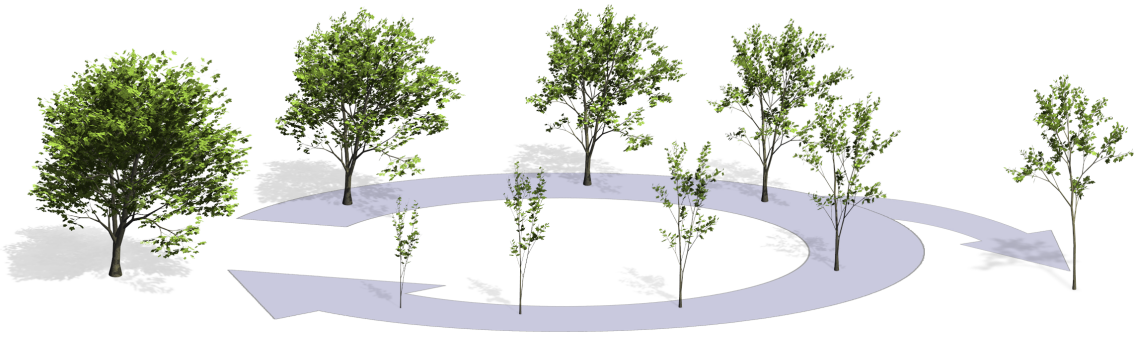


Figure 5.1: The static tree model on the left is converted into a developmental model (middle part), which is able to create arbitrary intermediate stages between a very young model and the given geometry. We define a "growth space" that allows the user to edit the model in an enhanced way. A corresponding model is shown on the right (Figure 1 from [138]).

which entails artificially adding back these branches to the intermediate stages of the model. This is done by analyzing the main branching structure and by using structural similarity – the fact that trees typically repeat branching structures along their main axes. Furthermore, geometric relations are adapted during growth since branches change their thickness and bend as the tree develops.
Botanic growth models and allometric rules produce convincing animations from a young tree, which, over time, converges to the input model. Once a tree model is processed with the presented system, a user can explore all intermediate stages instantaneously. By selectively applying the process to parts of the tree even complex models can be edited easily. This form of reverse engineering enables users to create rich natural scenes from a small number of static tree models. Figure 5.1 illustrates the steps of such a process.
Different growth rates have to be applied to different branches in order to meet botanic rules whereby a branch that receives more light grows faster than others and if light is below a threshold, the branch eventually dies off. Seasons with falling leaves can be integrated, pruned branches can be removed by fading or dropping them, growth can be exaggerated to meet various artists' needs.
After preprocessing the tree models, which is described in the next section, it is shown how to compute the important properties of the models such as growth rates and allometry. A number of models and growth simulations are shown and evaluated, limitations are discussed.

5.2 Processing Tree Models

The developmental model relies on a graph-based data structure, which is computed from the input tree models. Typically, these models are given as static meshes - a grouped list of vertices with connectivity information. Each group represents a different set of organs of the plant such as branches, leaves or blossoms.

5.2.1 Mesh Contraction

In contrast to the tree reconstruction technique introduced in Chapter 3 and the interaction approach presented in Chapter 4 the method described in the following sections does not rely on the cluster-based representation. Instead, a graph of a fully developed model from any source serves as input to the algorithm. Unfortunately, most tree models are not distributed with the underlying branching structure. To process arbitrary inputs with our method, the skeletal graph is obtained from the surface mesh of the tree model based on an algorithm similar to the contraction approach proposed by Au *et al.*[4]. This allows a wide variety of tree models to be processed, ranging from reconstructed trees to models grown by L-Systems and those from commercial libraries.

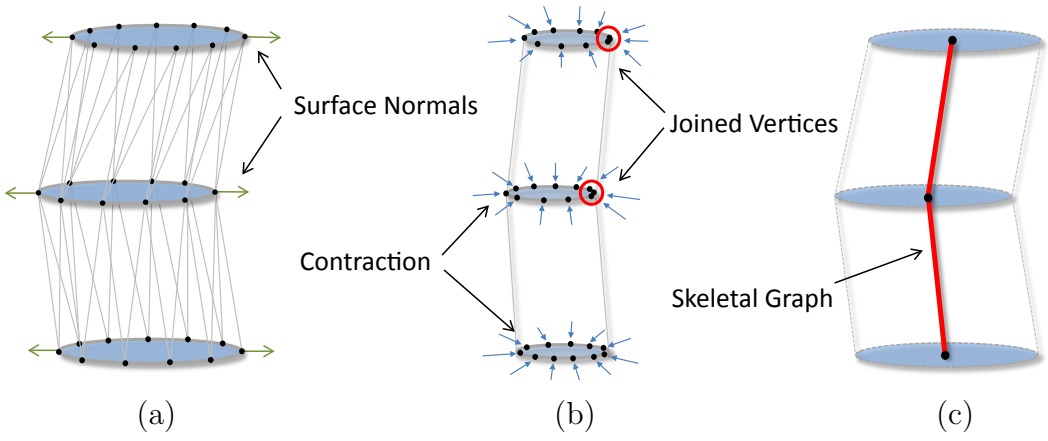


Figure 5.2: A schematic illustration of the contraction process: a surface mesh with connectivity information (a) is contracted based on the normals provided with each vertex; vertices close to each other are iteratively joined (b). The resulting skeletal graph (c) is used for generating and finally animating the growth of a model.

While Au *et al.* [4] proposed a general algorithm for arbitrary surface meshes, we implemented a more rudimentary version, which only processes branching structures. Tree models – even manually modeled exemplars – are commonly represented as generalized cylinders with more or less distinctive characteristics [17]. These cylinders originate from the underlying branching skeleton generated by a modeling procedure. The prerequisite to our algorithm is a mesh $M = (V, E)$, with vertices V and connectivity information in form of edges E. Each of the vertices $V = \{v_1, v_2, ..., v_n\}$ has a number of attributes $A = (P, N, C, T, ...)$, from which the position P and the surface normal N are the ones considered by the algorithm.
The algorithm moves all vertices iteratively towards the opposite direction of the surface normal N. The magnitude of the displacement corresponds to the initial thickness of the branch.

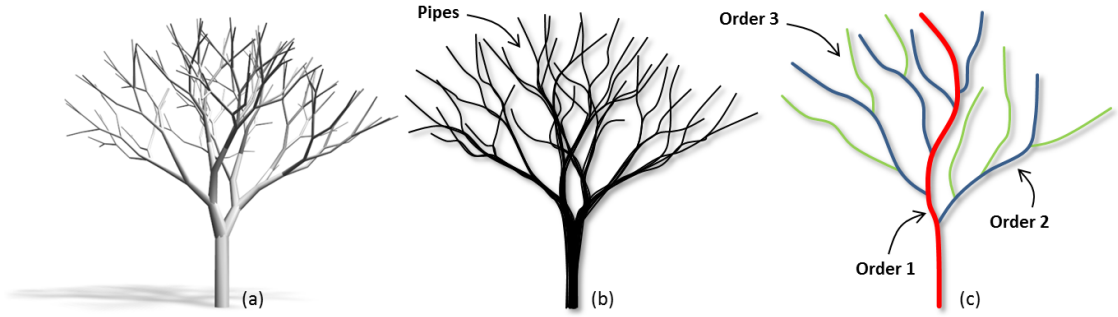


Figure 5.3: A tree model and different representations: a) the original model; b) Pipe Model representation [169]: the tree is seen as a system of pipes of uniform thickness, which supports the leaves. c) Gravelius ordering of the main branches of the tree is shown on the left. [69]. The orders are color-coded, starting with the main branch with order $g = 1$ *shown in red.*

In each iteration, vertices with a distance $d(x, y)$ smaller than a certain threshold t_{dist} are joined:

$$d(x, y) \leq t_{dist} \tag{5.1}$$

with $d(x, y)$ being the *Euclidian Distance*. The resulting vertex V_{new} is the average of the involved vertices:

$$V_{new} = AVG(v_1, v_2, ..., v_n). \tag{5.2}$$

Besides averaging the position and vertex normal, the connectivity information of the involved vertices $V_{inv} = \{v_1, v_2, ..., v_n\}$ is assigned to the new vertex V_{new}. The algorithm terminates when the vertices of a cylinder lying in the same plane are joined. The process is illustrated in Figure 5.2. A more detailed description can be found in Au *et al.* [4]. In contrast to the cluster-based representation introduced in Chapter 3 the resulting branching skeleton does not only represent the main branching structure, but the branches of the entire tree model.

5.2.2 Skeletal Structure

When a tree develops, branching angles of bifurcation change over time. This is due to various kinds of tropisms, i.e. phototropism (growth towards a light source) or gravitropism (growth following gravity - or perpendicular to it) and also due to forces that act on the branching structure and bend the branch. A tree develops additional wood at places with high tension, this also alters the branches.
To effectively model these changes, we use a branch representation in coordinates by two angles α, β which are relative to the parent edge. Please note that branches might be represented by more than one edge, since we allow consecutive sequences of edges without branching.
Botanical branching structures are often described by marking the *Terminal Leader* (main axis) and the *Gravelius Order* of branches; an ordering method originally

developed for binary branching systems to describe rivers and streams [69], but also successfully applied to tree modeling by Holton [84].
Starting from the root edge of the graph, which is assigned a level $g_{root} = 1$, the child edge with the smallest deviation angle within each bifurcation is assigned the same level as the parent edge, while the secondary edges are considered to be one level higher. To account for decurrent tree architectures (trees with weak apical dominance) we assign the same level to all child edges whenever their difference in angle and in radius is below a 5% range. Following the *Gravelius Order* [69] of the branches we are able to determine the main trunk and the side branches of the different orders. This scheme is our primary ordering within the graph.
We are now able to define chains in the branching graph: a chain is a sequence of consecutive edges without any level change. These chains are used to determine the growth rate and to insert missing geometry into the original model. They can reach from the root to the tip of a branch (see Chapter 6, Section 6.1.3 for more details). Additional information is collected about the average branching angle α_{avg} for bifurcations with different child levels and also about the *Internode Length I* (average spacing between leaves) of the terminal branches.
The resulting data structure is a set of edges E with each edge having been assigned the following properties: rotation angles α, β, length l, order g, and radius r and a set of child edges E_c (for all parameters see Table 5.1).

5.3 Reverse Growth Simulation

For the given graph we perform a reverse growth simulation, i.e. parameters are computed for the intermediate stages of the tree that finally lead to the given graph. This involves computing individual growth rates for the branches, interpolation of the branch radii, updating the branch angles, adding pruned branches and also the pruning of existing branches.

5.3.1 Growth Rate

A tree with a constant and equal growth rate for all branches would result in a shape, in which all branches are the same distance from the root. This is neither the case for real trees nor for the models we obtain as our input. Growth at a uniform speed, for these models, would result in intermediate models with some branches stopping their growth at an early stage. Rather than having a constant growth rate across all branches, the growth rate in nature is dictated by a number of influences, but mainly by the amount of light received by a branch (see Figure 5.4). Branches that receive less resources produce less biomass and will eventually die off (for details please refer to [108]).

Name	Description
$(\alpha_i^{(*)}, \beta_i, l_i^{(*)})$	relative coordinate frame (angle pair and length) of the edge e_i
g_i	level assigned to edge e_i according to the Gravelius order.
$r_i^{(*)}$	radius of edge/branch e_i. We limit our description to one radius per edge, although a precise description would have one radius for the start and another for the end.
b_i	radius coefficient of edge e_i (see Section 5.3.3).
$v^{(*)}$	growth rate
$v_\alpha^{(*)}$	angular velocity
r_{avg}	average radius of all terminal edges. Used as initial value for interpolation during growth.
$\alpha_{avg}^{(n)}$	average angle of branches with different assigned levels from n to $n+1$.
I	internode length defined as the average distance between leaves on terminal branches.
CR:	Crown ratio (crown height divided by tree height). Influences growth rate and time of removal of geometry added in growth space.

Table 5.1: System variables used for the tree models. $^{(*)}$ *denotes time-dependent variables.*

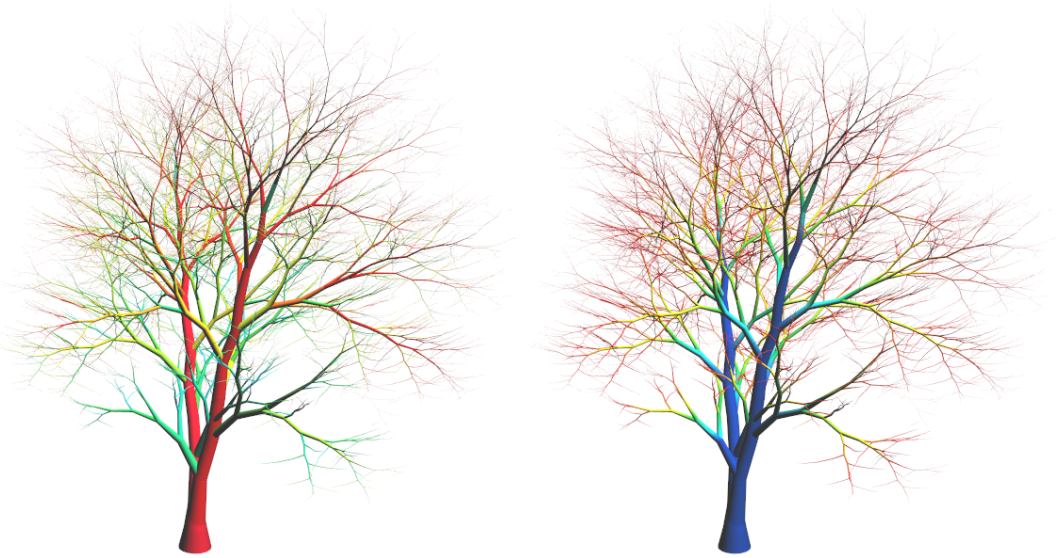
The fact that we want the final model to be reproduced with all branches still growing, leads directly to a recursive formulation with different growth rates for individual branches. The growth rate v_g of a chain is defined as

$$v_g = \frac{\sum_i l_i}{t_{max} - t_{start}} \tag{5.3}$$

with l_i being the lengths of the edges that form the chain, and t_{max} the chosen duration of the growth process. We start with the chain that defines the main axis (edges with order $g = 1$) $t_{start} = 0$. For chains where $g > 1$ the computation of v_g depends on the time $t_{start} = \sum_p l_p / v_p$ (p index of parent edges) where all parent edges reach their full length. The length of a chain at time t is computed by

$$l(t) = v_g \cdot (t - t_{start}). \tag{5.4}$$

This definition provides a constant growth rate per edge with the merit of being fully automatic. However, trees do not often have such a constant growth rate. Modifying Equation (5.4) allows us to define other biologically motivated functions for growth rate and angular speed such as the logistic function or linear acceleration (as discussed in [144]). We found, however, that constant growth already creates plausible animations for many tree models.



(a) (b)

Figure 5.4: (a) The growth speed of a tree visualized by color; red represents the fastest speed, cyan the slowest (Figure 3 from [138]). (b) The bendability of branches encoded with a different set of colors. Blue indicates rigid branches, yellow and red the most flexible ones.

5.3.2 Angle Interpolation

During growth, branching angles change over time. We distinguish between two classes of branches: apical and lateral branches. For apical branches the initial direction is the one of the parent branch, for lateral branches of level g the initial angle is given by $\alpha_{avg}^{(n)}$. The angular velocity v_α of an edge is set to

$$v_\alpha = \frac{\alpha_i - \alpha_{init}}{\Delta t} \tag{5.5}$$

with Δt being the duration the angle changes. This duration of the interpolation depends on the time the edge starts to grow $t = \sum_p l_p / v_p$ (p: all parent edges) and the time the branch reaches a certain radius.
Following the *Pipe Model Theory* of Shinozaki *et al.* [169] (see Section 5.3.4) we stop the angle interpolation as soon as the number of terminal edges of the subgraph reaches an upper limit (a good value is a limit of 5). In this case the branch becomes *solid* and does not bend anymore. The bendability of branches is illustrated in Figure 5.4.

5.3.3 Radius Interpolation

Another recursive formulation can be found for the branch radii, but this time from the tips of the twigs towards the root. According to an observation of Leonardo da

Vinci, the cross-section area of a branch is the sum of the cross-section areas of its child branches [153]. Later Murray [127] empirically found a relation of circumference c at bifurcations (c_1, c_2: circumferences of child branches)

$$c^{2.49} = c_1^{2.49} + c_2^{2.49} \tag{5.6}$$

This was based on measurements for branches of many different species. As a result, Murray proposed that trees statistically follow a 2.5 power law of branching.
We use this simple allometric rule to define branch radii during the growth simulation. However, since hand-modeled trees often ignore such rules (e.g. for efficiency, for artistic reasons or to emphasize some parts of the tree) and our main goal is to eventually match the final tree, we calculate a coefficient b of the original tree model for each edge using

$$b_p = \frac{1}{r_p^u} \sum_i r_i^u \quad u = 2.5 \ . \tag{5.7}$$

with r_i being the radii of child branches and r_p the associated radius of the edge. The coefficient b_p is used for computing the radii of intermediate stages using the following recursive description: For each branch e_p that currently does not have children (a terminal shot) the current radius r_p is initialized by r_{avg} while for non-terminal branches the radius r_p is defined by

$$r_p = \left(\frac{\sum r_i^u}{b_p} \right)^{1/u} \tag{5.8}$$

with r_i being the radii of the child branches.

5.3.4 Pipe Model Theory

A popular model in theoretical biology is the *Pipe Model Theory* by Shinozaki et al. [169]. The theory postulates that tree forms emerge from vascular systems, which have to distribute resources within the tree as well as having to mechanically support themselves. A tree structure is an assembly of leaf units of constant size and corresponding pipes of uniform thickness connecting the leaves to the root (see Figure 5.3 (b)). While real trees have a much more diverse construction, the theory successfully explains many effects in natural trees.
One fundamental finding produced by the theory is the *Profile Diagram* (see Figure 5.5), which maintains its similarity among various plant communities despite their differences in species or morphology.
The *Profile Diagram* represents the vertical distribution of leaves $\Gamma(z)$ and of non-photosynthetic tissue $C(z)$ (contained in a horizontal layer of unit thickness Δz) downward from the top of the plant or community to the ground. $F(z)$ is the cumulated leaf quantity for the whole tree and $T(z)$ the cumulated tissue (leaves plus non-photosynthetic tissue).
These distributions are what we try to maintain when animating tree growth. If the

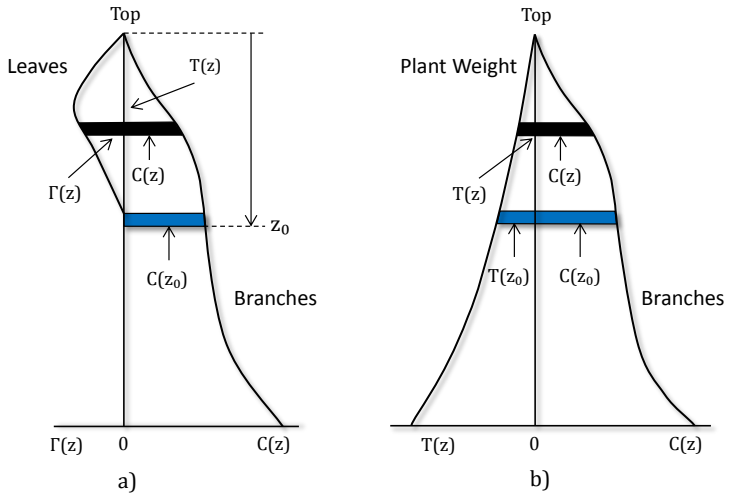


Figure 5.5: a) Profile diagrams for trees (illustration adapted from [30], Figure 1): a) relation between the vertical distribution of mass in leaves ($\Gamma(z)$) and non-photosynthetic organs ($C(z)$); b) relation between the accumulated amount of plant weight ($T(z)$) and the distribution of non-photosynthetic organs ($C(z)$).

distributions of the given model deviate too much from these values, we assume that branches were pruned, so we add branches and let them die off during animation to finally reach the given model.
For empirical investigations Γ and C are determined by applying the so-called *Stratified Clipping Method (STC)* where the biomass for the leaves ($\Gamma(z)$) or the non-photosynthetic tissue ($C(z)$) is determined by selecting a vertical range $[z, z + \Delta z]$ within the tree. Another way of collecting the data for these distributions is the *Main-axis Cutting Method (MAC)* [30, 31]. Here the values associated with a height z are those parts of the plant that emerge from the main-axis segment of length $[z, z + \Delta z]$.
Differences in the biomass distributions obtained by MAC and STC of the input model allow us to detect regions in which additional geometry is needed during earlier developmental stages of the model (we will explain this in more detail in the next section). Such regions are candidates for filling in additional geometry to account for pruned branches during our simulation.

5.3.5 Adding Missing Structures

Trees are affected by several environmental factors that influence their shape. During growth, parts of a tree get lost, e.g. through mechanical influences such as wind, intentionally by human pruning, or due to the lack of resources.
This information is not available by looking at the final tree, which is the basic problem for the input models we use. Most of the available tree models have been developed for certain environments we are lacking information about. A plant was potentially modeled to be part of a tree stand or to be placed in an urban environ-

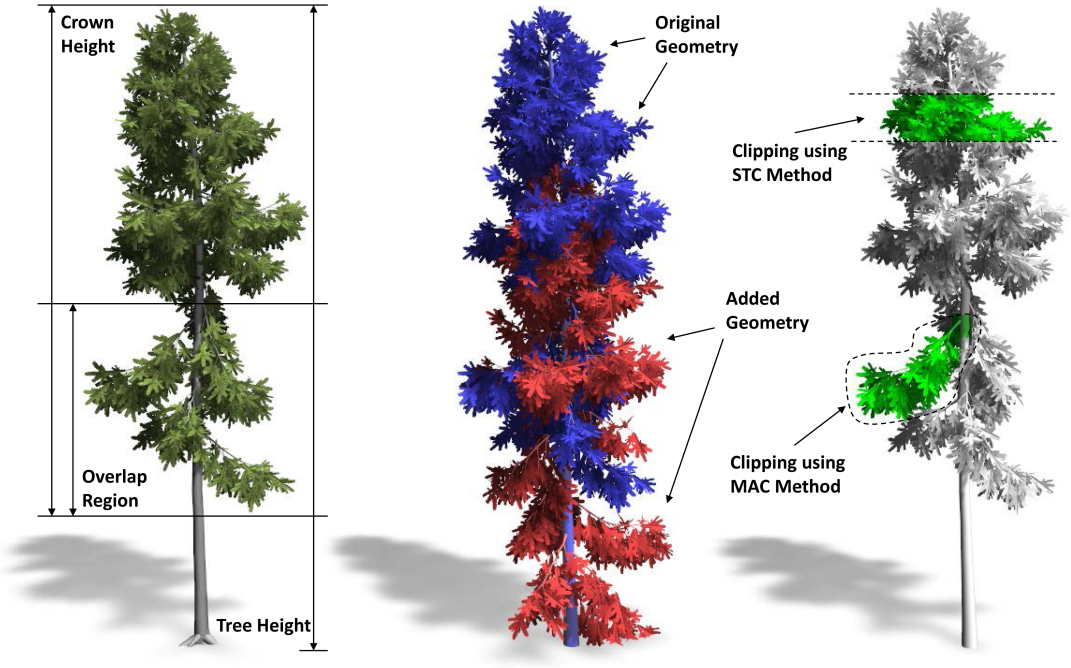


Figure 5.6: Left to right: tree model with marked crown region and overlap region; updated model with added geometry. Original branches are colored blue, added branches are colored red; different clipping methods (Figure 5 from [138]).

ment where lower branches have to be pruned. In both cases we need to fill in the missing geometry in order to obtain a plausible growth animation.
We do this by taking advantage of the principle of self-similarity within trees [59]. This principle has been heavily used for tree modeling in the past, grammar based methods mostly rely on it (see [142, pp. 121] for an overview). We exploit self-similarity and copy branches from the tree to the missing parts. This is done in the following way:

- copy the largest possible continuous part of the branches in the tree canopy to the lower parts of the trunk that are empty.
- fill in the missing information with the fewest possible number of copies.

The regions to be filled with geometry are determined as follows: A well-known dimensionless measure used in forestry is the crown ratio (CR, see Table 5.1). Crown ratio values range from 0 (no crown, dead or defoliated) to 1 (crown extends over the entire tree bole). For trees where the crown ratio is smaller than 1, the region below the crown needs to be filled with additional geometry (see Figure 5.6). Additionally, predicting CR values for intermediate stages allows us to determine when additional geometry is pruned during growth (see next section).
While competing for space and resources such as sunlight, individual branches eventually block out other lateral branches, finally obtaining a larger volume than their competitors that eventually diminish. Without adding additional geometry and

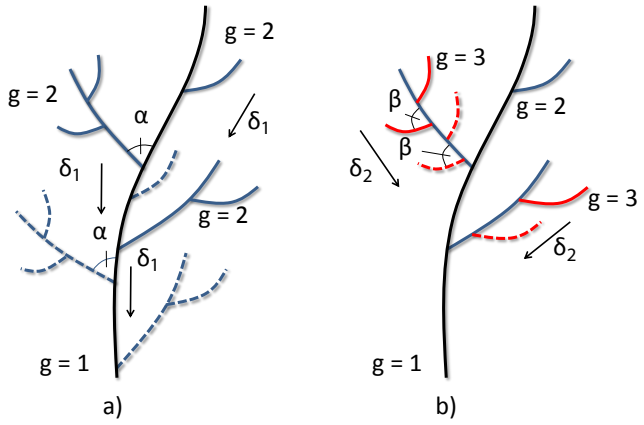


Figure 5.7: Copying of branch geometry (Figure 6 from [138]). a) Branches attached to the root chain (g=1) are copied and added with an offset δ_1 (dashed lines). b) We can apply this process to chains of any order (here g=3 with δ_2).

branches during earlier growth stages these regions become apparent for intermediate models. The above-mentioned STC and MAC measures are taken to detect regions of reduced density within the tree. We call these kinds of regions *Overlap Regions*. They are also filled with additional branches.
Following the first chain (g=1, the main trunk) we copy all attached sub-branches adding an offset along the chain to fill this region (see Figure 5.7). We define the *Overlap Region* to be as large as the filled sub-crown region. We only add branches in this region that originate from parts of the axis that fullfil the following heuristic. The difference between $T(z)$ obtained by MAC and STC is used to analyse the overlap region between copied and original branches. A significantly higher value of $T(z)^{(STC)}$ for a certain segment along the chain then $T(z)^{(MAC)}$ (obtained by MAC) suggests that the expected biomass for this region is concentrated along another lateral branch that originates from another part of the main axis. This analysis is performed once for the original input model.
We want to reconstruct these candidate branches and keep branches for the overlap region that are associated with segments along the chain that either exhibit larger $T(z)$ values generated by STC then by MAC or both values are low. In all other cases these candidate branches are neglected. The additional structures are added to the branching graph (together with the associated edge parameters) and used to generate intermediate stages. In the remainder of this section we show how the additional structures are removed during growth to eventually match the original model.
While pruning and self-thinning during early stages of development only affect the main branch, secondary level branches may be influenced in the same way during later stages. We fill in these missing parts in the same way as we add geometry to the main trunk.
The ability to add missing geometry and hallucinate branches that have died off during growth enables us to form what we call *Growth Space*, a space of possible extension (branches) obtained from the original model.

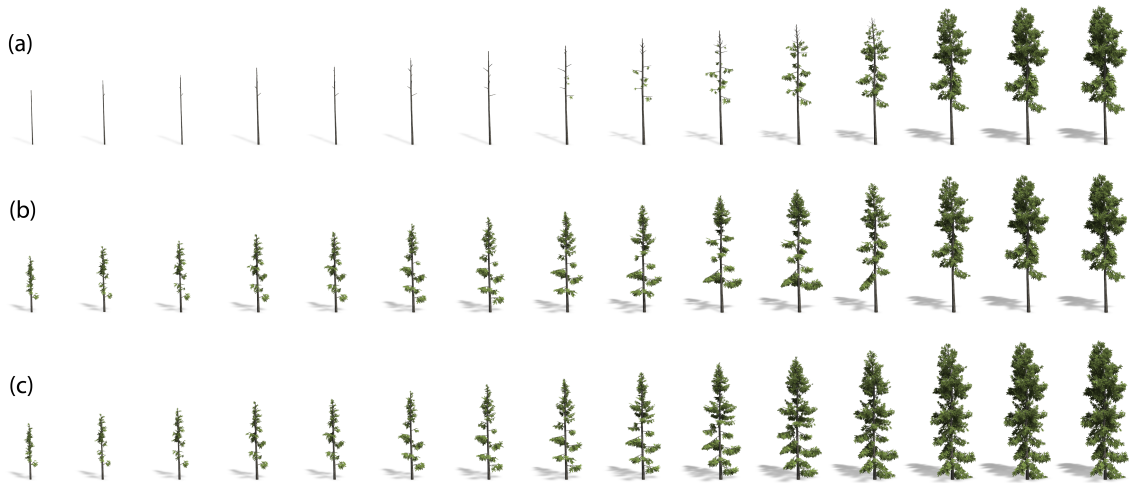


Figure 5.8: Growth animations: (a) without additional geometry the tree remains empty during growth; (b) without the removal of additional geometry the crown ratio is not met; (c) gradually removing additional geometry during growth creates a natural model in all steps.

5.3.6 Removing Added Geometry

The principle of self-similarity also guides us during animation. We try to maintain a plausible crown ratio of the model during growth while starting with a typical young tree. Since branches do not move upward during growth but are replaced by new ones, we use the added geometry to form the crown at early stages. When the tree grows, the early branches have to be pruned to maintain the crown ratio.
Empirical investigations indicate the following logistic function for the development of the crown ratio during growth (see Hasenauer and Monserud [81])

$$CR = \frac{1}{1+e^{-x}}, \qquad x = c_0 + c_1 * SZ + c_2 * CO + c_3 * ST, \tag{5.9}$$

where x is a linear combination of species-dependent and environment-specific input variables: SZ = size of the model, CO = competition for resources, ST = site-specific parameters. We assume competition and site variables to remain constant during development of a tree and use tree height h for SZ. This leads to the simplified version of Equation (5.9):

$$CR = \left(\frac{1}{1+e^{-(c_0'+c_1*h)}}\right)^{\gamma} \tag{5.10}$$

using CR_p and height h_p of the original model and $CR_0 = 0.99$ for $h = 0$ (see Hasenauer and Monserud [81]) we are able to compute the model specific coefficients c_0', c_1 for a standard value of $\gamma = 1$. Evaluating Equation (5.10) for intermediate values of h gives us the time at when the additional geometry for regions below the original crown has to be removed.
This provides a fully automatic method to find plausible values for c_0' and c_1. However, an artist can choose different values and thus change the timing at which

additional geometry is removed (see below). The above function with slightly different values for the target crown ratio at maximal plant height h_p is used to gradually remove additional geometry in the overlap area.
In Figure 5.8 the importance of adding and gradually removing geometry during growth is shown. A natural looking result can be achieved by using only this mechanism. It is important to note that the branches removed during growth contribute to the computation of the branch radii. If this were not the case, the deletion would alter other radii leading to visible artifacts in an animation.

5.3.7 Adding Leaves

Usually trees produce leaves only at terminal branches. However during growth, some edges turn into terminal edges, which were not terminal edges in the original model, and would therefore not have provided leaves for a realistic appearance of the developmental stages. To improve the intermediate appearance for each chain we use the leaf distribution of terminal branches in the original model as a candidate set for temporary leaf positions. During the growth interpolation we make these leaves visible if the edge is terminal and remove them again if the edge becomes a non-terminal edge. Since the number of branches slowly approaches the final number, the leaves also gradually approach the foliage of the input model.

5.4 Results

Figure 5.13 shows six models that were processed and animated using our system. The models were imported from different sources (reconstructed point scans, Xfrog, and L-System generated models). All trees show a visually plausible shape at the various stages of their growth.

5.4.1 Growth-based Editing

A system is presented that automatically generates intermediate models of a given input tree. However, different parameters can be introduced allowing artists to gain additional control over the appearance of intermediate models and influence the resulting animation. The proposed process allows for enhanced editing operations, such as scale, copy and translate operations for tree models.

Changing Growth Speed and Timings: Changing the target crown range in Section 5.3.6 allows us to keep additional geometry for the final model (see Figure 5.11). Solving Equation (5.10) for different values of γ automatically changes the time steps for which additional geometry is removed.

Figure 5.9: Frames of a seasonal growth simulation with typical color change (Figure 9 from [138]).

While the above changes are applied to the whole model, some enhanced editing can also be applied to parts of the tree. This allows for a natural manipulation of the model. The following operations for tree models can be seen as combinations of scale and structural growth where scale translates to growth (see Figure 5.12)

Advanced Scaling: Selecting individual branches and changing their age (while keeping the age of non-selected branches) corresponds to traditional scaling operations, which can be accomplished with standard editing tools. The advantage of our method is that the resulting branch radii, sub-branch angles and lengths are properly transformed to match the new age of the branch.

Advanced Copy&Translate: Copying selected branches and moving them is combined with the growth process and bound to the time parameter. This means that branches that are translated to younger parts of the plant automatically reduce their age, angles and radii, and vice versa. The radii of parent branches are changed according to Section 5.3.3.

This increases the number of variations of the original model and provides a powerful editing tool. Figures 5.11 and 5.12 show the range of editing possibilities for a single input model. After the growth parameters have been computed the user is able to adjust the branches of the model and change growth rate and relative time. This allows us to produce a diverse set of models from a single input exemplar.

5.4.2 Approximating Seasonal Growth

Typically plants exhibit a rhythmic extension of their leafy axes during seasons. It is caused by an alternation of meristem inactivity and active shoot extension [8]. We emulate this growth pattern for animation purposes using a remapping function f of the time parameter t ($t \in [0,1]$ with $t = 1$ for the full model)

$$t_{new} = f(t) = \frac{sin((t \cdot n - s) \cdot 2 \cdot \pi)}{(2 \cdot \pi \cdot n \cdot c)} + t \tag{5.11}$$

with n being the number of years the growth simulation should approximate. The coefficient c defines the behavior during seasons with decreased growth rate ($c \in [1, 1.4]$, for $c = 1$ $f'(t)$ recurrently becomes 0 after a one year period). A shift parameter $s = 0.5$ is used to remap the angle interpolation parameter in order to account for the fact that the highest additional mechanical stress is caused by the biomass of leaves, while $s = 0$ is used to remap growth.
The derivative of the remapping function is used as a factor to change the leaf size of the model for deciduous trees. Figure 5.9 shows frames of a seasonal growth animation with typical leaf coloring.

5.4.3 Performance

We measured the performance of our system to demonstrate its efficiency. Table 5.2 shows the timing in milliseconds using an Intel Dualcore @ 2.66GHz with nVidia GTX 580 graphics board. Even quite complex scenes can be animated at interactive rates using this kind of system (see Figure 5.14).

Species	**#Branches**	**#Leaves**	**Model**	**Render**
Fagus Sylvatica	10,623	59,822	49.1	4.56
Ulmus Laevis	27,236	33,200	56.6	3.87
Aesculus Hippo.	43,466	47,148	76.4	4.05
Picea Abies	3,813	5,901	13.0	5,38
Quercus Petraea	9,585	58,796	41.6	3.84
Quercus Petraea (complex)	14,554	89,913	64.7	4.13

Table 5.2: Time required for modeling and rendering of tree models (in ms). Model size in number of triangles.

5.5 Limitations

We successfully used the proposed method to generate models of intermediate age for a number of different species from various sources. So far, our method is working with monopodial trees only; however, adding other branching structures is not a general problem and will be implemented in the future.
Another limitation is that during simulation we are bound to the environment for which the tree was modeled. This is what defines the crown ratio and specifies the environmental factors that are presumed to remain constant during growth. Changing this (e.g. relaxing the constraint to eventually meet the original model) might result in artifacts such as repetition. This effect is not usually seen since the original geometry and its copies are used with different local parameters, such as time, and therefore look different.

Figure 5.10: Failure case: the Salix, a willow tree, forms branches in a particular way that cannot be reconstructed by our system (Figure 11 from [138]).

Furthermore, there are species that do not develop their branches continuously. Every year, willow trees create hanging twigs that are not always converted into branches in the next year but fall off. It is therefore not possible to produce a continuous animation for this specific case. Figure 5.10 shows the result for a Weeping Willow. While intermediate states still look natural they do not show the typical bending of the small branches and twigs.

5.6 Summary

In this chapter we introduced a new method for creating growth models from a static input tree. The tree is analyzed and self-similarity is used to create branches where others have been pruned during growth (growth space). Growth parameters are determined and interpolated to create convincing animations. Based on these parameters, the tree can be edited as a whole or in parts. The method allows the production of time lapse animations of tree development, providing plausible interpolation of branching angles and growth rates at the same time. Since the introduced method is not a growth simulation, every step in the development of a tree can be created separately. This limits the method to a certain set of applications (e.g. games, movies) since it provides only a plausible approximation of the growth process. However, the method is efficient, and enables real-time evaluation and rendering.

A promising extension of the proposed method would be the inclusion of environmental factors such as obstacles, shading, or competition with neighboring plants - all kinds of limits and influences affecting the natural growth process. However, since the objective is to finally match the original model, the freedom to change the plant during development is limited, especially since some of the environmental factors are implicitly encoded in the original model.

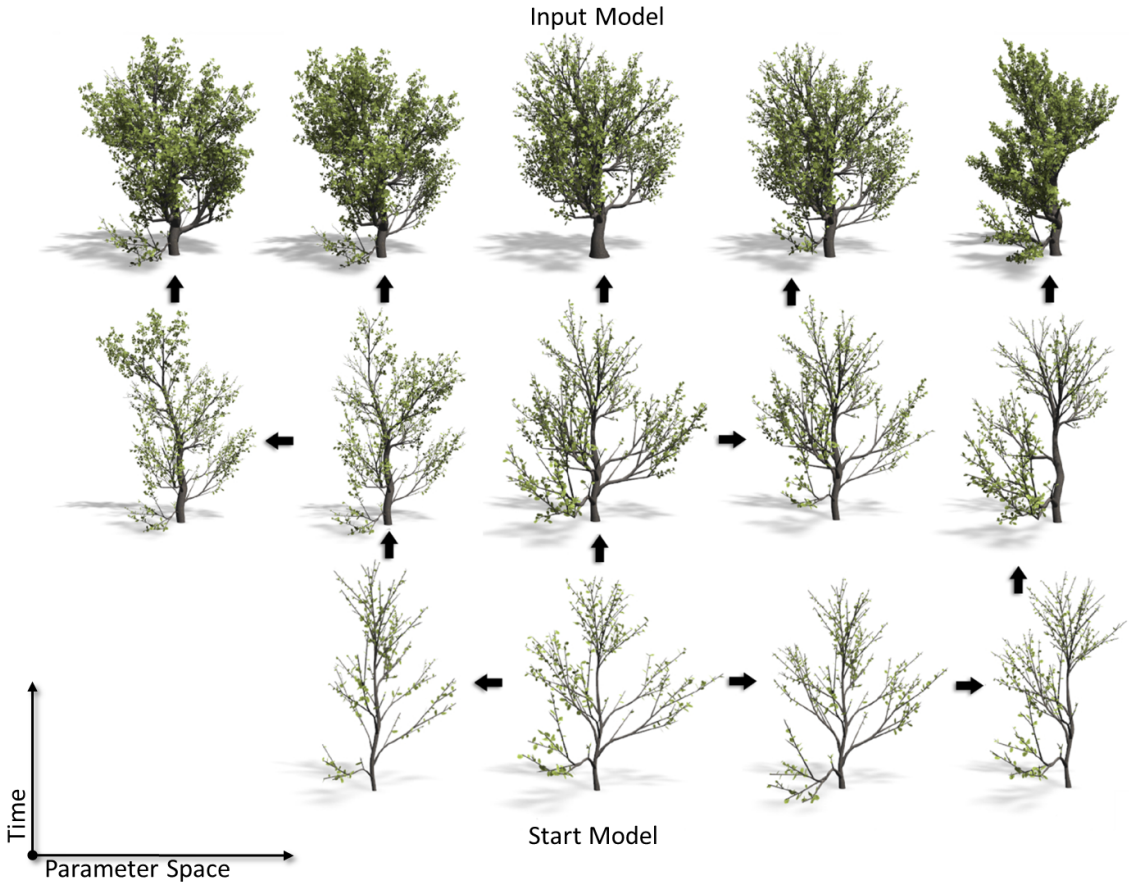


Figure 5.11: By selectively applying growth to individual branches, editing a given model is a simple procedure. Branches can either be emphasized or reduced to create certain effects. This opens a "growth space": a space for developmental edits that influences the model in a natural way (adapted from [138], Figure 8).

Figure 5.12: Different models resulting from an editing session. Users can interactively move through the developmental stages and adjust active branches (shown in red) to edit the appearance of a tree. This process allows a multitude of exemplars to be created from a single input model.

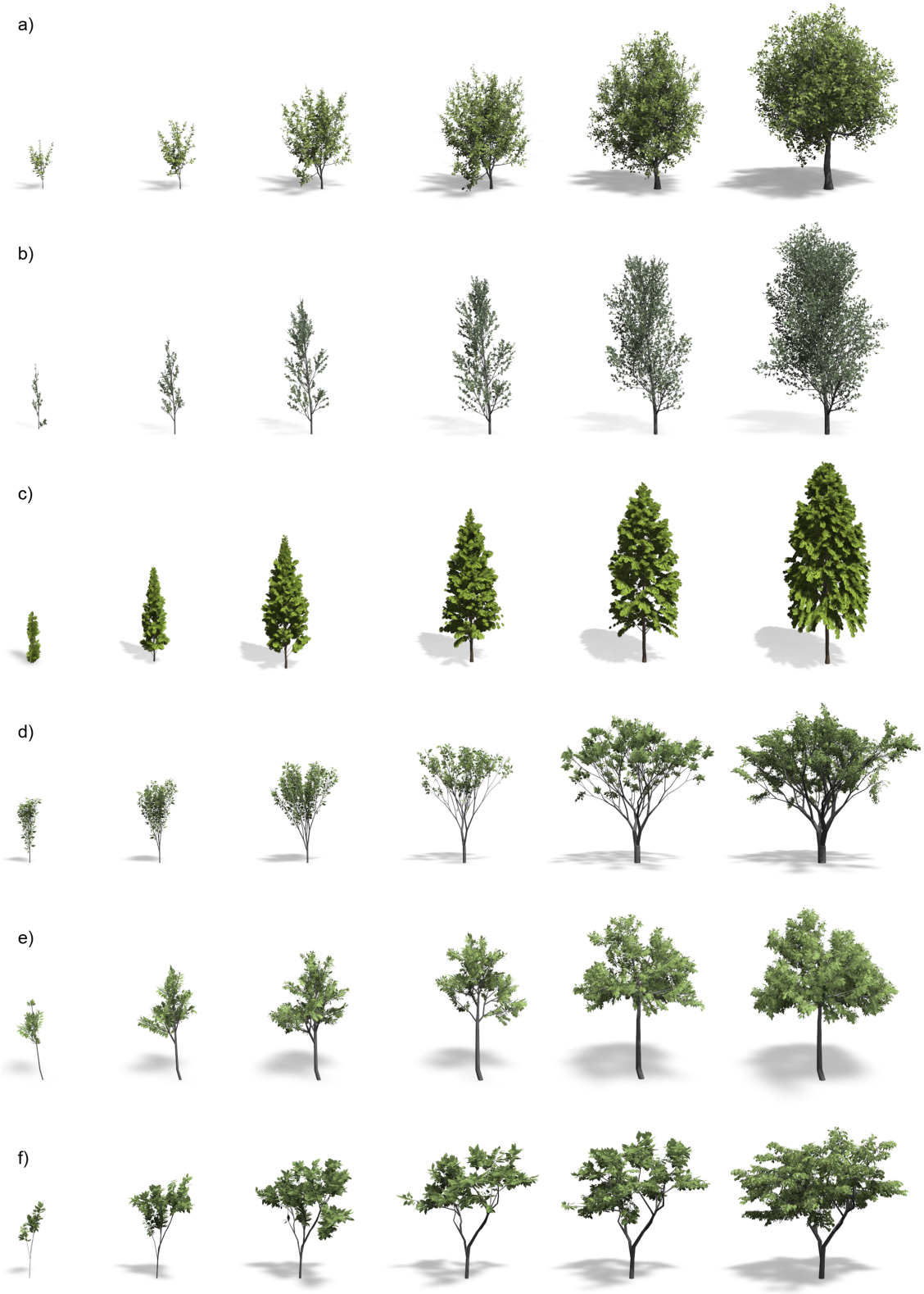


Figure 5.13: Six models processed and animated with our system (Figure 10 from [138]). The largest tree of each species (model on the right) served as input to our method. All other models were synthesized. We used three different model types to test our method; Xfrog: a), b), c); L-System generated model: d); Reconstructed LiDAR Scans: e), f).

Figure 5.14: A sequence of developmental stages of a complex scene consisting of more than 20 trees interactively rendered with our system.

6

Real-Time Rendering Paradigms

The process of synthesizing images is commonly differentiated into two categories: online and offline rendering. These categories have existed since the early beginning of creating computer graphics and represent different techniques and approaches. Offline rendering is mostly concerned with generating images at the best possible quality, while time is a subordinate constraint. Common techniques in this domain, such as *Ray Tracing*, *Path Tracing* or - more generally - *Global Illumination* often approximate physical processes and allow to generate results with a high degree of photo-realism.
Online or more commonly real-time rendering also strives to mimic real-world effects, however, it additionally follows hard time constraints to realize interactive responses. This demand has created a whole set of specialized algorithms, with the focus on processing a single frame in a couple of milliseconds.
Systems to synthesize images are usually based on render pipelines consisting of different stages that serially transform geometric descriptions into pixels. In real-time systems, a rasterizer addresses this process and allows for the efficient generation of images. Today, these pipelines are evolved systems combining complex processes in several paths. Modern graphics hardware dramatically supports this process and allows program code, in the form of shaders, to be executed at several stages in the pipeline. However, such render systems are still overstrained with the enormous detail required to represent natural objects and extensive scenes.
This chapter reviews a number of known techniques and approaches that enable the efficient, time-critical, rendering of vegetation. This encompasses the dynamic generation of surface meshes for different plant organs, rendering techniques to maintain a photo-realistic appearance as well as different methods for shading and shadowing.

6.1 Hardware-accelerated Meshes

The techniques introduced in the previous chapters have shown that modeling, editing and animating vegetation at interactive rates is vital for many application domains. It enables content creators to modify scenes with instantaneous feedback. Although the modeling of realistic or natural effects is most time consuming, up-

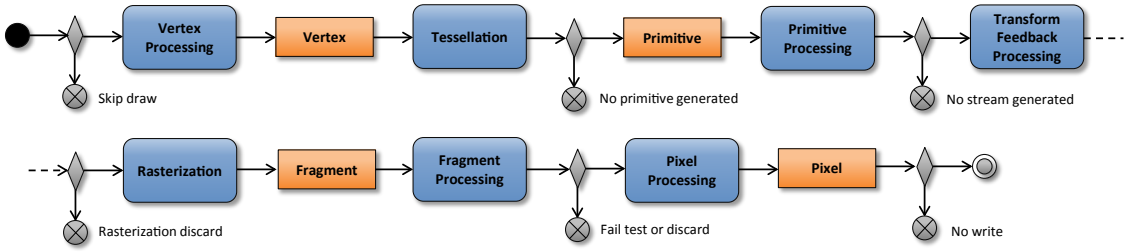


Figure 6.1: The stages of a modern graphics pipeline (OpenGL 4.2) (illustration adapted from [40], inset). Various stages in such a pipeline allow the execution of program code, called shaders.

dating and transforming complex meshes is the other predominant bottle-neck in interactive applications. Thousands or even millions of vertices have to be processed on a frame to frame basis.
In order to counter these problems, surface meshes are commonly stored in high-speed memory directly on the graphics hardware. This limits the number of bus operations while at the same time provides efficient access to the required objects. While today's *Graphics Processing Units (GPU)* handle vertex processing very efficiently, they do not support evolved data structures and programming paradigms as necessary for complex modeling techniques. This results in alternating GPU and CPU operation in the render loop to frequently update buffer objects and other resources.

Modern real-time rendering pipelines include several shader stages, programmable units on the graphics hardware, which provide control of the individual segments of the pipeline. A schematic illustration of a modern (OpenGL 4.0) pipeline is illustrated in Figure 6.1. A more detailed description of modern render systems is out of the scope of this thesis and the interested reader is referred to [40]. The following sections describe how graphics hardware can be utilized to process tree models more efficiently. The main motivation for utilizing the introduced techniques was to find an efficient rendering procedure for tree models that allows even large sets of vegetation to be modeled dynamically.

6.1.1 Tree Graphs

Instead of generating surface meshes, most tree modeling techniques produce a skeletal graph to represent branching structures. This graph $G = \{V, E\}$ commonly consists of a number of vertices V and a set of edges E. Each edge is represented by two vertices $E_n = (v_1, v_2)$ exhibiting a hierarchical relationship, with v_1 being the predecessor of v_2. The techniques presented in this thesis predominantly process monopodial trees, models that have one single main trunk, and thus only one of the

nodes in G is defined as the root node N_{root}. The majority of techniques for editing or animating tree models relies on this intermediate representation to generate the desired effects. Unfortunately, most of the tree models available in various tree libraries are not distributed with such a skeletal graph.
However, several methods exist that allow the skeletal structure to be derived from a branch mesh. A simplified version of the algorithm proposed by Au *et al.* [4] is discussed in Chapter 5 (Section 5.2.1). Other modeling methods directly yield the branching structure as an intermediate stage of a reconstruction process, e.g. as proposed in Livny *et al.* [113] or as dicussed in Chapter 3 (Section 3.3.1).
Interactive operations, animations as well as the instantaneous generation of branch geometry, require frequent updates of models and the corresponding surface mesh. Directly updating the vertices in the GPU's memory is too costly to be performed on a frame-to-frame basis; transforming the vertices in shader programs yields unwanted artifacts. Hence, the most common approach is to perform the modeling operation directly on the tree graph and to dynamically generate the surface mesh on the GPU.
Instead of predefining and then storing the surface vertices in a buffer object, modern graphics hardware supports the creation of new vertices on the fly. This enables generating a full surface mesh from a number of control primitives, the edges of the tree graph. The skeletal graph is stored as a *Vertex Buffer Object (VBO)*. This VBO is processed with a shader program that produces generalized cylinders around the edges of the provided graph, a process similar to that already proposed by Bloomenthal *et al.* [17].

6.1.2 Parallel Transport Frames

In order to generate smoothly connecting cylinders around corresponding edge vertices of a skeletal graph, continuous coordinate frames have to be associated to each vertex in the graph. These frames, each of which is a set of three vectors, define the orientation of the base of a cylinder and thus enable a coherent meshing. A common approach for computing continuous frames along a 3-dimensional curve is the *Frenet Frame Formalism* [15]. A single *Frenet Frame* is defined in the following way [78]: given a 3-dimensional space curve $\vec{x}(t)$, the frame is determined by the tangent, the normal, and the binormal:

$$\begin{aligned} \vec{T}(t) &= \frac{\vec{x}'(t)}{||\vec{x}'(t)||} \\ \vec{B}(t) &= \frac{\vec{x}'(t) \times \vec{x}''(t)}{||\vec{x}'(t) \times \vec{x}''(t)||} \\ \vec{N}(t) &= \vec{B}(t) \times \vec{T}(t) \end{aligned} \tag{6.1}$$

These equations only yield a series of continuous frames in the event of $\vec{x}(t)$ being a thrice-differentiable space curve with a non-vanishing second derivative. This is the

main drawback of using *Frenet Frames* as it prohibits computing a series of frames where the space curve has no curvature. Tree graphs, especially simplified ones, often convey straight lines and these are unable to fulfil this criteria.
A more convenient method is proposed by Hanson and Ma [78] as *Parallel Transport Frames (PTF)*, where a series of smooth frames can even be computed for curves with a vanishing second derivative. They utilize the characteristics of parallel vector fields to transport an orthonormal frame along the corresponding curve. These PTFs convey a more appropriate behavior. In contrast to *Frenet Frames*, where each of the frame vectors is analytically defined, PTFs only require merely determining the tangent of the space curve. The basis $(\vec{N}_1(t), \vec{N}_2(t))$ is chosen arbitrarily. Hence, determining the consecutive frames of a space curve is synonymous with transporting the initial frame along the curve. Following these observations Hanson and Ma [78] show that this can be achieved by determining a rotation axis

$$\vec{B} = \frac{\vec{T}_i \times \vec{T}_{i+1}}{||\vec{T}_i||\ ||\vec{T}_{i+1}||} \tag{6.2}$$

and the angle

$$\theta = arccos\left(\frac{\vec{T}_i \cdot \vec{T}_{i+1}}{||\vec{T}_i||\ ||\vec{T}_{i+1}||}\right) \tag{6.3}$$

.
for a consecutive set of tangent vectors $\vec{T}_i$ and $\vec{T}_{i+1}$. This can be used to iteratively transport the normal vector $\vec{V}_{i+1}$ along the curve:

$$\vec{V}_{i+1} = R(\vec{B}, \theta)\vec{V}_i \tag{6.4}$$

with $\vec{V}_i$ being the normal vector from the previous computation and $\vec{V}_0 \perp \vec{T}_0$. The rotation matrix R is given as

$$R(\theta, \hat{n}) = \begin{bmatrix} c + (n_1)^2(1-c) & n_1n_2(1-c) - sn_3 & n_3n_1(1-c) + sn_2 \\ n_1n_2(1-c) + sn_3 & c + (n_2)^2(1-c) & n_3n_2(1-c) - sn_1 \\ n_1n_3(1-c) - sn_2 & n_2n_3(1-c) + sn_1 & c + (n_3)^2(1-c) \end{bmatrix} \tag{6.5}$$

with $c = cos(\theta)$ and $s = sin(\theta)$ [61].

A comparison of PTFs and *Frenet Frames* is shown in Figure 6.2. As can be seen in the illustration, the PTF approach also provides continuous frames on straight segments and in the event of direction changes. The pseudocode for generating PTFs is provided in Algorithm 1.

6.1.3 Chains

Both approaches, the *Frenet* as well as the *Parallel Transport Frames*, provide the normal, binormal and tangent for single 3-dimensional space curves. Tree graphs,

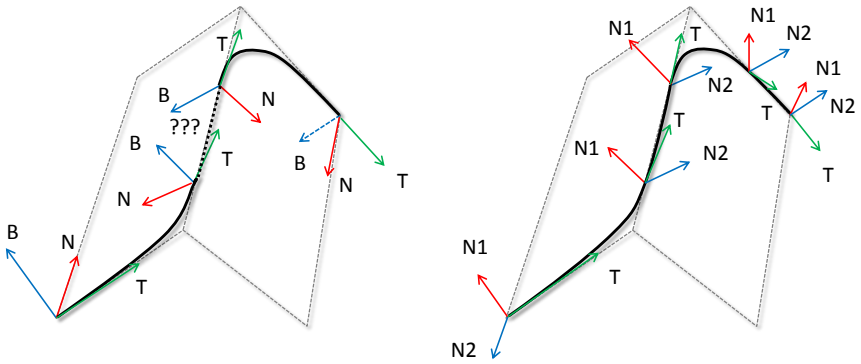


Figure 6.2: Comparison of Frenet and Parallel Transport Frames (PTF) (adapted from Hanson and Ma [78], Figure 3). Left: the series of Frenet frames is interrupted on the straight segment on the space curve, whereas the sequence of PTFs is continuously defined along the entire curve (right).

Algorithm 1 Parallel Transport Frames algorithm from Hanson and Ma [78, p.9].

Input: (1) A list of unit tangent vectors $\{\hat{T}_i\}, i = 0, ..., N$;
(2) An initial normal vector $\vec{V}_0$, $\vec{V}_0 \perp \hat{T}_0$.

Output: A list of parallel-transported normal vectors $\{\vec{V}_i\}, i = 1, ..., N, \vec{V}_i \perp \hat{T}_i$

1: **for** $i \leftarrow 0$ to $N-1$ **do**
2: $\quad \vec{B} \leftarrow \hat{T}_i \times \hat{T_{i+1}}$
3: $\quad$ **if** $||\vec{B}|| = 0$ **then**
4: $\quad\quad \vec{V_{i+1}} \leftarrow \vec{V_i}$;
5: $\quad$ **else**
6: $\quad\quad \hat{B} \leftarrow \vec{B}/||\vec{B}||$;
7: $\quad\quad \theta \leftarrow arccos(\hat{T}_i * \hat{T}_{i+1})$; // $0 \leq \theta \leq \pi$
8: $\quad\quad \vec{V_{i+1}} \leftarrow R(\hat{B}, \theta) * \vec{V_i}$
9: $\quad\quad$ //Rotate by angle θ about $\hat{B}$ (*see Equation* 6.5)
10: $\quad$ **end if**
11: **end for**

however, convey branching structures and thus many intersections. Generating a complete surface mesh for the entire branching structure requires computing a continuous set of frames for each of the branches. To do so we organize the branches as an intermediate data structure, called a chain. Each chain $C = \{E_1, ..., E_n\}$ is represented as a sequence of edges with a hierarchical relationship; a vertex in the chain has only one ancestor. This definition relates directly to the previously introduced space curves.
The chains are built by iteratively traversing the outer child vertices down towards the root vertex. Each node is assigned a flag, indicating whether it has already been traversed. In each iteration, the graph is traversed until a vertex is reached that has already been marked. The algorithm terminates with all outer vertices being

marked as traversed.
Randomly traversing the tree graph from the child vertices towards the root does not always yield plausible results. To achieve visually plausible results for the resulting surface mesh it is important to also consider the branch thickness and the branching angles. An existing approach, the *Gravelius Order* [69], computes hierarchies of rivers based on the angles between main and side arms. We utilize this method for branches and extend it by also considering the thickness and the length of the branches when determining the hierarchy (cf. Chapter 5, Section 5.2.2).

6.1.4 Tessellation and Meshing

Dynamic techniques like the interaction with obstacles (Chapter 4) or the animation of growth (Chapter 5) require frequent updates of the skeletal graph of a tree model. To efficiently process these updates, it is desirable to keep the amount of graph primitives, and thus the CPU-based calculations, low. However, a coarsely defined graph with the minimum set of possible edges yields straight and artificially looking branches. To increase the visual realism it is important to smoothen these edges. The tessellation unit available on today's graphics hardware consists of two programmable shader stages that allow a given geometry to be refined with certain constraints. In the graphics library *OpenGL*[1] these are called *Control-Shader* and *Evaluation-Shader*. While the *Control-Shader* stage enables new vertices to be generated along a given primitive, the *Evaluation-Shader* stage defines where these new vertices are placed. We utilize these hardware capabilities to refine a coarsely defined graph.
Given four subsequent points $P = \{P_{i-1}, P_i, P_{i+1}, P_{i+2}\}$ on a chain C_n we compute an interpolation as the *Cubic B-Spline* [105]:

$$S_i(t) = \sum_{k=0}^{3} P_{i-3+k} b_{i-3+k,3}(t); t \in [0,1] \tag{6.6}$$

which can in matrix-form be written as:

$$S_i = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} Pi-1 \\ Pi \\ Pi+1 \\ Pi+2 \end{bmatrix} \tag{6.7}$$

As it is desirable to maintain the overall structure of the graph, the interpolation is not performed across the intersection of two chains. Only subsequent points belonging to the same chain are interpolated. The result of the tessellation stage is an arbitrarily detailed edge segmented into vertex pairs. Each of these pairs is then

[1] OpenGL 4.3 Core Profile (Feb. 2013): http://www.opengl.org/registry/doc/glspec43.core.20130214.pdf

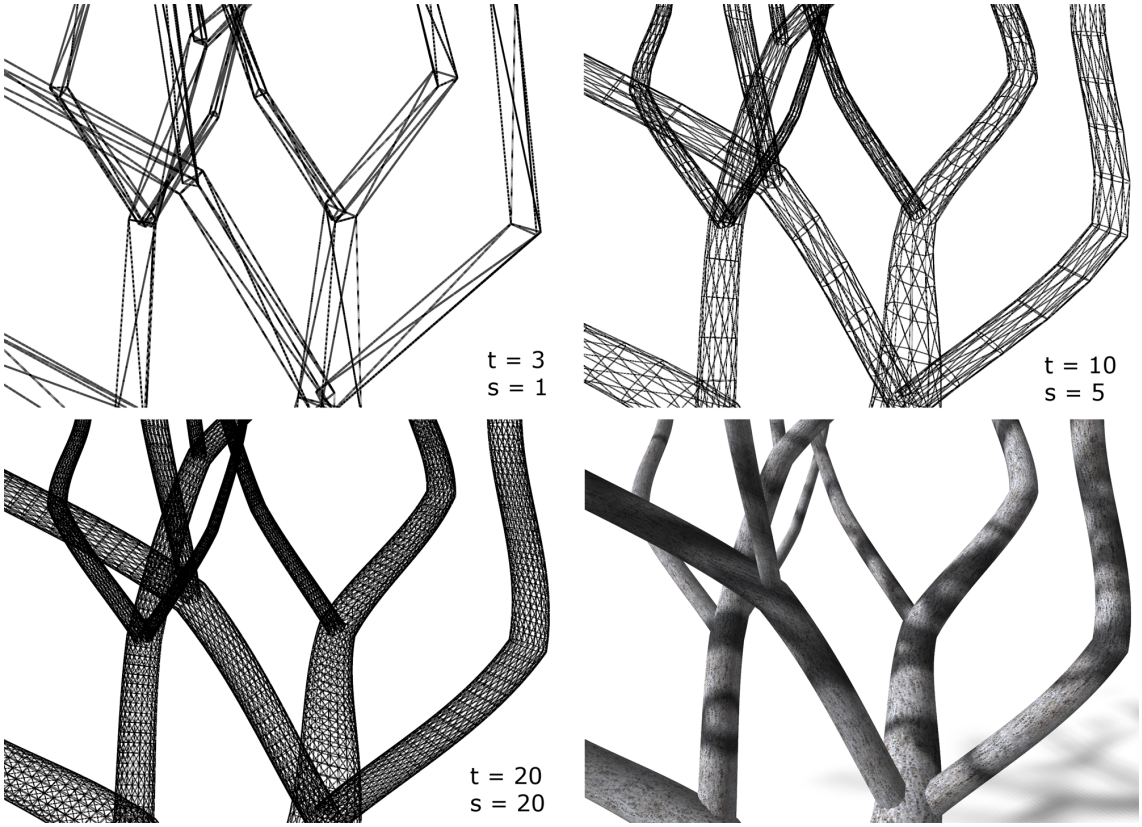


Figure 6.3: A dynamically generated branch mesh with different resolutions (top row and bottom left): t indicates the number of vertices used for the cylinder, s the number of subdivisions for each edge. Bottom right: the final surface mesh rendered with shadows.

further processed by the *Geometry-Shader* stage. Here, new vertices can be created and placed according to positions generated by the program code. We utilize this stage to build generalized cylinders around each of the edge segments.
The complexity of a branch mesh is mainly controlled by the distance between an observer and the tree model. A large distance yields lower amounts of vertices for the mesh of a given edge segment. When a user approaches a given tree model, the number of vertices is unnoticeably increased. This dynamic adaptation of the level of detail allows for the processing of large amounts of individually changing tree models at interactive rates. Figure 6.3 shows a mesh generated with the aforementioned process at different resolutions.

6.1.5 Leaves

Due to the large amount of geometry required to represent the foliage of a tree, leaves are commonly generalized into groups of textured quads, called *Billboard Clouds* [11]. This representation supports efficient processing, however, it also prohibits precise shading and shadowing due to missing surface information.

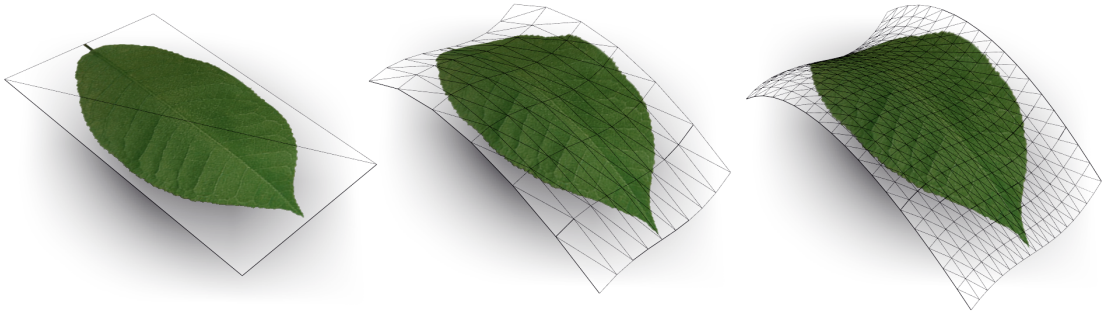


Figure 6.4: A leaf surface rendered with different amounts of geometry. The number of vertices and the curvature of the surface are adjusted based on the distance to an observer.

We, instead, focus on more detailed representations of foliage. Each leaf is expressed as a textured quad that can be dynamically refined to a curved bicubic surface. Such a surface is defined as *Non-uniform rational B-spline (NURBS)* and can be written as [105]:

$$S(u,v) = \sum_{i=1}^{k} \sum_{j=1}^{l} R_{i,j}(u,v) P_{i,j} \tag{6.8}$$

with the rational basis functions:

$$R(i,j) = \frac{N_{i,n}(u) N_{j,m}(v) w_{i,j}}{\sum_{p=1}^{k} \sum_{q=1}^{l} N_{p,n}(u) N_{q,m}(v) w_{p,q}} \tag{6.9}$$

An implementation of bicubic surfaces on the graphics hardware allows us to sample the NURBS dynamically and generate the surface vertices. We again account for a dynamic level of detail based on the distance to the camera. A smaller distance yields a more curved surface and higher amounts of generated vertices. A higher distance levels the surface until it becomes a simple plane and reduces the amount of vertices to four corner points representing a quad.
Bicubic surfaces enhance the visual quality of subsequent lighting algorithms as they provide smoothly changing surface normals for each leaf, however, due to massive amounts of leaves in a single tree crown this approach is infeasible for most applications. Figure 6.4 illustrates a single leaf rendered at different levels of detail.

6.2 Appearance Modeling

As stated in the previous sections, processing vegetation poses unique problems for rendering systems and requires specialized algorithms to achieve visually appealing results. Besides the enormous amounts of complexity, however, other problems occur in subsequent stages of the render pipeline. The following sections describe

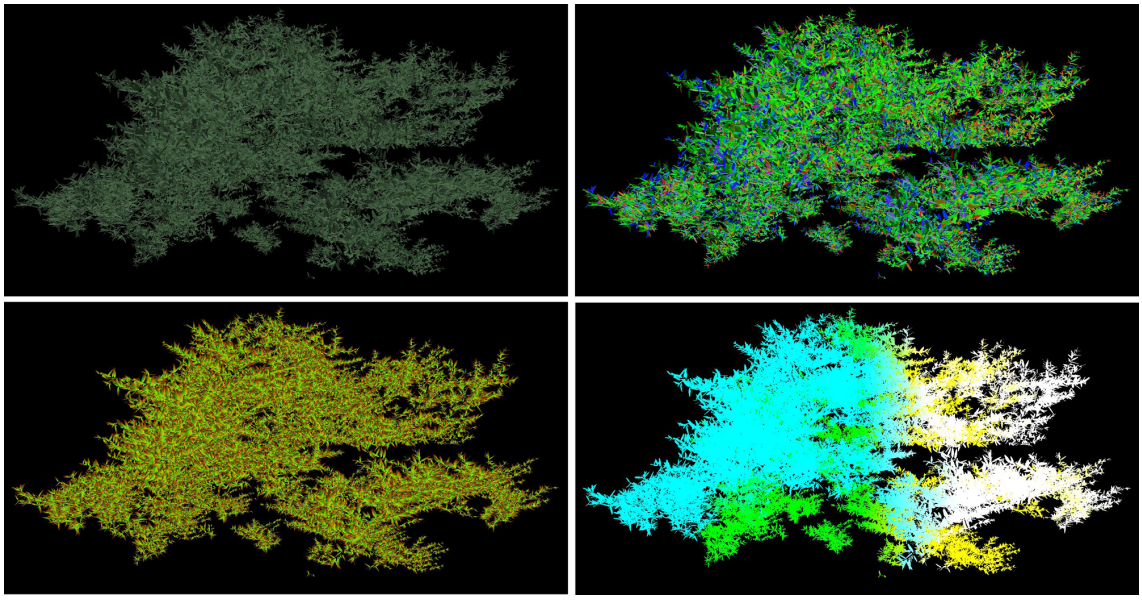


Figure 6.5: Four different deferred buffers generated by rendering leaf geometry. Each of the buffers represents vertex data encoded as color. Top-left: texture color; top-right: normals; bottom-left: texture coordinates; bottom-right: positions

techniques to efficiently model the appearance of trees and plants. This includes a review of commonly used techniques like *Deferred Shading* and *Alpha-to-Coverage* as well as an implementation for *Shadow Mapping*.

6.2.1 Deferred Rendering

Modeling a tree crown causes large amounts of vertices to be rendered relatively close together. The rasterizer transforms these vertices and produces several fragment candidates, potential pixels written to the frame buffer. The fragment processor on the GPU works on these fragments and enables algorithms to be applied to realize complex shading and shadowing effects. However, due to the inherent complexity and the close distance of tree foliage, overdraw occurs. This means that several fragments are generated for a single pixel. The *Depth Test* commonly discards these fragments, but only after the fragment processor has processed them.
To avoid processing these overdrawing fragments, Saito and Takahashi [163] introduced the G-Buffer paradigm. A G-Buffer is a series of screen-sized buffers holding information of the scene's geometry. This set of buffers commonly comprises the position, normal, depth, color and texture coordinates, but can also vary depending on the application's requirements.
Figure 6.5 illustrates a set of buffers of different vertex properties. These buffers are filled in a preceeding rendering pass and are then evaluated and combined to generate the final rendering, a technique commonly named *Deferred Rendering*. The main advantage of this image-based technique stems from the fact that only visible

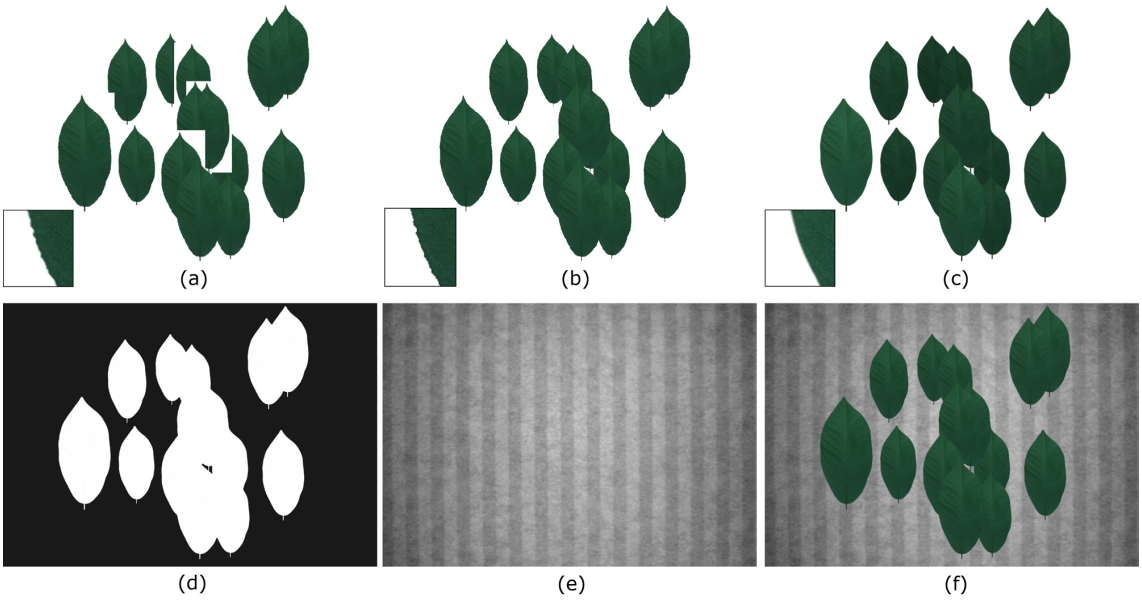


Figure 6.6: Alpha-to-Coverage vs. Alpha Blending and Alpha Testing. (a): Alpha Blending yields strong artifacts for objects with a random order. (b): objects rendered with enabled Alpha Testing exhibit aliased edges. (c): Alpha-to-Coverage creates smooth edges without requiring to sort the objects. (d)-(f): buffers needed to create an Alpha-to-Coverage compositing (mask, background, foliage).

fragments are considered for processing in subsequent shading and shadowing algorithms. This enhances the rendering speed of intricate scenes with large amounts of overdraw. Although the performance of *Deferred Rendering* is dependent on a number of different factors [55], such as the number and the precision of the involved buffers, in general it is linear to the number of fragments of the processed viewport. We utilize *Deferred Rendering* to maintain interactive frame rates when processing large amounts of foliage or grove and forest scenes.

6.2.2 Alpha-to-Coverage

One of the fundamental characteristics of real-time rendering pipelines is the parallel, independent, processing of geometry and fragments. This fact allows for an efficient, and thus real-time, synthesis of images. On the downside, however, it also comprises strong disadvantages for mimicking real-world effects that require the relationship of objects like shadows or transparencies.
Two common techniques allow us to adjust the processing of transparent fragments. *Alpha Blending* defines how the color of a newly processed fragment is blended with the color of a previously written pixel. Textures of leaves commonly contain semi-opaque or even transparent texels. As each of the leaves is independently processed and finally written to the *Depth-* and *Frame Buffer*, artifacts occur where pixels

with transparencies overlap (Figure 6.6 (a)). Hence, *Alpha Blending* requires that the transparent objects are sorted according to the distance of the position of an observer. A prohibitive requirement for thousands of leaves.
Alpha Testing, instead, discards fragments below or above a certain threshold, meaning that they are not further processed and written to the *Frame Buffer*. This avoids the afore-mentioned sorting, but introduces aliasing artifacts at the borders, where the leaf blends with background objects (Figure 6.6, (b)). Both leaf sorting and aliasing are undesireable artifacts when processing vegetation. A number of techniques exist for rendering order independent transparencies [9, 120], however, due to newly introduced drawbacks, they are not always applicable for rendering vegetation.
Alpha-to-Coverage [126, 98] is a multipass rendering technique that allows for an efficient processing of alpha blended fragments without introducing noticeable artifacts. To apply this technique the entire geometry of a scene is separated into foliage geometry and world geometry; both sets are rendered into different *Framebuffer Objects (FBO)*. Rendering the leaves yields a color buffer and a coverage mask (Figure 6.6 (c) and (d)). The alpha values of every single leaf are used to fill the coverage buffer storing the opacity information for each fragment. This takes place without depth testing and with enabled *Alpha Blending*. The leaf color is rendered with enabled *Alpha Testing* to avoid rendering of transparent pixels. The coverage buffer can then be used to smoothly blend the scene color and the foliage color in a final render pass (Figure 6.6 (f)). *Alpha-to-Coverage* provides visually plausible results while avoiding the common artifacts, however, requires support for multi-pass renderings. Figure 6.6 illustrates the content of the different buffers and also shows the final rendering.

6.2.3 Shadowing

In computer graphics, shadows play an important role in generating photo-realistic imagery. The interplay of light and shadows provides cues about the geometry of objects and their spatial relationships [82]. The most common approaches to realize shadows in real-time applications are *Shadow Maps* [196] and *Shadow Volumes* [41]. *Shadow Volumes* allow the precise computation of perceptually correct shadows as they accurately determine the umbra and penumbra regions of a cast shadow. A drawback of this method is that the rendering of the volumes is expensive. Additional geometry needs to be generated for each occluder, which causes large amounts of additional vertices in the render-loop [156].
Shadow Maps, instead, are easy to implement, less sensitive to geometric complexity, and well-supported by today's graphics hardware [51]. The quality of the resulting shadows, however, is bound to the resolution of the target texture, the *Shadow Map*. This easily yields artifacts due to aliasing in applications that require shadows at different scales. Both approaches are not well-suited for generating shadows for trees and plants. The additional geometry required for volume-based approaches is prohibitive in real-time applications and the aggregate structure of tree geometry even

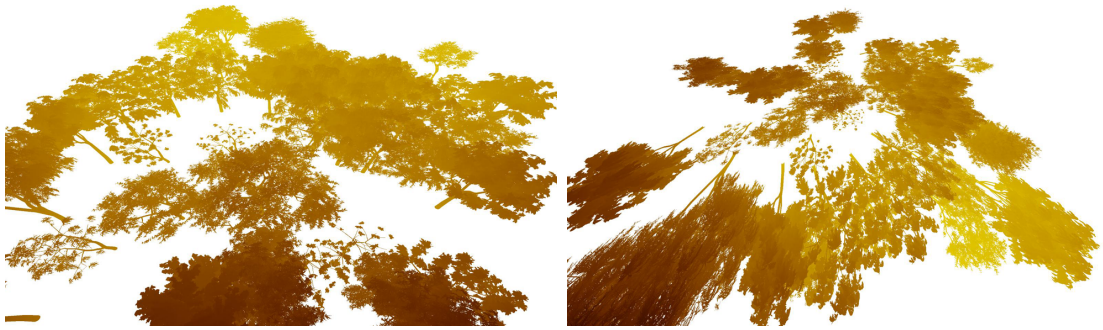


Figure 6.7: Two different types of shadow maps as input for the variance shadow map filtering approach [51]. Left: normal shadow map; right: perspective shadow map. The color gradient represents the depth (red channel) and the depth squared (green channel) of the corresponding scene.

facilitates aliasing when using *Shadow Mapping* techniques. More recent approaches allow the production of high-frequency shadows for tree foliage by stochastically distributing leaf stencils [20]. Although this method enables precise and realistic shadows to be generated within the tree's foliage, it cannot be used to produce ground shadows.
To lower the aliasing problems of shadow maps used at different scales, we implemented the *Perspective Shadow Maps* approach proposed by Stamminger and Drettakis [175]. Their method suggests generating the shadow map after the perspective transformation. As this considers the position of the camera in the process of generating the shadow map, this technique overcomes resolution problems and produces more convincing results. Figure 6.7 illustrates the difference between a normal and a perspectively distorted shadow map.
We combine this technique with the *Variance Shadow Maps* filtering approach proposed by Donnelly and Lauritzen [51] who derive an upper bound on the fraction of occluded fragments and generate smoother and more stable results compared to other filtering approaches. Instead of only storing the depth when creating the shadow map, as done for standard shadow mapping, they propose to also store the depth squared. This allows us to compute the moments M_1 and M_2 over a filtered region as:

$$\begin{aligned} M_1 = E(x) &= \int_{-\infty}^{\infty} xp(x)dx \\ M_2 = E(x^2) &= \int_{-\infty}^{\infty} x^2 p(x)dx \end{aligned} \tag{6.10}$$

and to compute the mean μ as well as the variance σ:

$$\begin{aligned} \mu &= E(x) = M_1 \\ \sigma^2 &= E(x^2) - E(x)^2 = M_2 - M_1^2 \end{aligned} \tag{6.11}$$

Figure 6.8: A number of trees rendered with self- and ground-shadows.

Donnelly and Lauritzen show that the variance can be interpreted as a quantitative measure and they utilize the *Chebychev Inequality* to compute percentage closer filtering [51]:

$$P(x \geq t) \leq p_{max} \equiv \frac{\sigma^2}{\sigma^2 + (t - \mu^2)} \tag{6.12}$$

Figure 6.8 illustrates trees rendered with perspective shadow maps and variance filtering. Although a combination of *Perspective Shadow Maps* and *Variance Shadow Mapping* is not customized for rendering vegetation, it produces visually plausible results.

6.3 Summary

In this chapter we have presented a number of techniques that allow for the efficient processing of vegetation with photo-realistic visual quality. Although the focus lies predominantly on processing trees, most of the approaches can also be applied to plants, bushes or even flowers. Techniques like *Deferred Rendering*, *Alpha-to-Coverage* or *Shadow Mapping* are well established in today's real-time applications and their use is not limited to rendering plants. However, the special characteristics of vegetation, such as relatively small pieces of aggregate geometry, create unique problems that are not entirely addressed by today's techniques.
For instance, none of the existing approaches to generate shadows is well suited to processing vegetation. *Shadow Volumes* are prohibitive as they require even more geometry as is already needed for a tree model. *Shadow Mapping* suffers from aliasing, which is inherent when rendering small details, such as leaves and small branches.
The main limitation in processing vegetation is the enormous amount of geometry required to create enough plausible looking details. Even though a number of modeling paradigms enable a large variety of visually appealing exemplars to be created, the complexity of these models is still quite low. Real-time applications, in particular, require efficient means to handle increasing amounts of geometry while further approaching photo-realism.

7 Summary and Conclusion

In striving to attain photo-realism, processing vegetation has been of interest to computer graphics researchers since the very beginning. As frequent objects in our daily lives, trees and plants are also essential components of nearly all kinds of settings in various application domains. As described throughout this thesis the most limiting factor is the amount of geometry required to express the inherent details of trees and plants. Modeling as well as processing these details requires specialized algorithms, particularly for real-time and interactive applications.
While most applications are distributed with the content they later process, the internet and a diverse set of platforms available today require different paradigms to generate and to distribute computer graphics content. Geo-information-systems (GIS), for instance, on mobile devices require content on demand via low-bandwidth internet connections. Models as available today are not modeled based on these constraints. These application scenarios require the dynamic generation of details on the fly and level of detail approaches that enable the amount of geometry to be reduced while still maintaining the overall visual quality.
So far, models of trees and plants have mostly been static sets of geometry. Manually adjusting a tree model to conform with certain contraints is a time consuming task, which also requires sophisticated knowledge of the underlying biological processes. Content creators, like artists, level designers or architects, need to work with these models without them being all too time consuming. This requires models that adapt interactively to scenes and changing environmental conditions.

7.1 Contributions

In this thesis we have proposed methods, techniques and paradigms to advance the processing of vegetation from a computer graphics point of view. While several approaches presented in the preceding chapters are generally applicable to offline as well as to online rendering, the focus lies in the efficient, time-critical, processing of tree exemplars.
Given this constraint, the goal was to find means that allow to model, interact and animate tree models while maintaining biological plausibility and a photo-realistic appearance. The biological correctness often suffers from severe simplifications made

to meet the time constraints. However, the modeling paradigms presented are also applicable to more complex models and to larger parameter spaces.

The main contributions of this thesis can be summarized as follows:

- In Chapter 3 we presented a new modeling paradigm for tree models, which allows the efficient storage, transmission and rendering of a large variety of tree models. By approximating a given input tree model, this representation reduces the data necessary to describe the model by a couple of orders of magnitude with no negative impact on the overall visual appearance. While the skeletal graph exactly represents the main branching structure, fine details, such as small branches, twigs and leaves are generated in a procedural manner. Besides encoding existing tree models, the representation also allows us to reconstruct trees from laser scanned point sets. We have shown that this new modeling paradigm can be used with a variety of different tree species.

- The efficient interaction of plant models with their environment is of special interest to content creators in various fields. In Chapter 4 we described and discussed a framework, which transforms the skeletal graph of a provided tree model. This new modeling paradigm allows tree models to react more efficiently to their environment while maintaining biological behavior. When moved close to an obstacle a tree model bends or loses its branches and hence maintains a biologically plausible appearance. The results were evaluated against existing modeling paradigms by measuring the edit distance between transformed and grown tree graphs.

- In Chapter 5 we introduced a framework to automatically compute the developmental stages of a given input tree model. The interpolation of these stages enables us to mimic the tree's natural growth, a process that requires an artist hours of manual labor. As an input tree model does not contain branches of its younger developmental stages, we proposed a method that generates these branches based on structural similarity. This process is controlled by a number of measures, such as the *Main-Axis-Cutting (MAC)* and *Stratified Clipping (STC)* method or *Profile Diagrams*, known from the field of biology.

- Several existing methods and techniques to process vegetation in real-time were explored in Chapter 6. The motivation for this chapter was to provide a comprehensive description of state of the art techniques that allow the efficient processing of vegetation with modern graphics hardware. Besides discussing approaches for a dynamic generation of surface meshes from tree graphs, established techniques to maintain a visually plausible appearance, such as *Alpha-to-Coverage*, *Deferred Rendering* and *Shadow Mapping* were explored in the context of photo-realistically rendering tree exemplars.

Some of the results presented in this thesis have already inspired further research in the field of tree and plant modeling and also found their way into commercial applications:

- Jinasena, K. D. S., and Sonnadara, D. U. J. **Stochastic simulation of trees with environmental sensitivity**. In SIGGRAPH Asia 2012 Posters (New York, NY, USA, 2012), SA '12, ACM, pp. 35:1-35:1.

- Wang, Y., Chang, X., Ning, X., Zhang, J., Shi, Z., Zhao, M., and Wang, Q. **Tree branching reconstruction from unilateral point clouds**. In Transactions on Edutainment VIII, Z. Pan, A. Cheok, W. Mller, M. Chang, and M. Zhang, Eds., vol. 7220 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 250-263.

- Kim, J., and Cho, H. **Efficient modeling of numerous trees by introducing growth volume for real-time virtual ecosystems**. Computer Animation and Virtual Worlds 23, 3-4 (2012), 155-165.

- **Laubwerk Player Plug-In for 3DS MAX® and Cinema 4D®**, Laubwerk GmbH[1], August-Bebel-Strasse 27, 14482 Potsdam-Babelsberg, Germany

7.2 Future Work: Geometry

Although recent advances, such as shown in games and movies, might give us cause to think that most of the problems regarding vegetation in computer graphics have already been solved, there exists a plethora of areas to further advance this field from a research point of view. The following sections briefly introduce potential areas of future work with a focus on geometry processing.

7.2.1 Material Properties

Real-time applications, in particular, often require severe simplifications and approximations when processing surface structures. To increase the realism of such objects, it is important to model the surface characteristics more carefully. The surface properties of tree and plant organs is inherently complex and requires large amounts of geometry to replicate the reflectance behavior of real objects.
Methods exist that specifically model the properties of organ surfaces, such as leaves or bark. However, these models are often based on assumptions or neglect underlying biological processes. New approaches need to be found that consider biological insights while at the same time allow for efficient computation.

[1] http://www.laubwerk.com/

7.2.2 Level of Detail

The massive amount of geometry of single trees and plants exceeds the capabilities of today's hardware. Entire forests can only be processed with space partitioning algorithms, such as BSP-, Oct-, or Quadtrees. As hardware performance improves so do our changes of modeling and rendering vegetation in computer graphics applications ever more precisely. However, natural objects convey incredible fine details that need to be modeled efficiently if we want to achieve even higher degrees of photo-realism.
Different platforms and application domains require varying levels of detail and new modeling approaches. The surface of forests consists of a multitude of small surfaces all of which absorb and reflect light. Methods need to be found that allow for smooth transitions in the level of detail while considering the unique material properties of vegetation. Although methods exist to model plants at different scales, one of the major problems continues to be how these techniques can be handled in the same scene.

7.2.3 3D Printing

Physical 3D models enable an experience of a sense of space and structure in several application domains. Especially architects and designers utilize them to educate customers and users. As 3D printers are soon to be broadly available, printed physical models of vegetation will also be objects of interest. Due to their intricate nature, however, these objects are not easy to print. Methods exist that consider complicated object properties (e.g. by printing struts), but they need to be adjusted to also work for vegetation.
Printed physical models of photo-realistic geometry might not be wanted in all situations as they do not blend into a specific design. Methods need to be found that allow us to find different abstracted and printable representations of tree models.

7.3 Future Work: Biology

As shown throughout this thesis, dynamically changing tree models are of special interest to content creators. They ease the setup of virtual sceneries and do not require knowledge of the structure and development of individual exemplars. However, today's modeling and rendering approaches are way beyond a proper biological simulation. Both the visual appearance as well as the modeling of plant organs have to be improved from a biological point of view.
In the following sections we briefly review the current state of the art in modeling plants based on the *Functional-Structural Plant Modeling* paradigm. This rather new form of plant modeling emerged from long existing formalisms such as L-Systems or fractals and generalizes concepts available in various scientific fields. A *Functional-Structural Plant Model (FSPM)* combines a 3D tree model, as already

known in computer graphics, with a set of biological, physiological or physical functions. The main intention behind these models is to better understand the multitude of processes observable in nature that drive a plant's growth and development.
The FSPM concept offers a general and structured concept, which allows the realism of a plant model to be enhanced, a strong requirement in increasingly complex and realistic virtual settings. Besides reviewing FSPMs from a computer graphics point of view we present ideas that help to migrate the existing theoretical concepts to computer graphics applications.

7.3.1 General Architecture

Trees and plants dynamically react to their environment based on multiple factors. While many of these influences can be modeled independently, only their combination allows the complexity of plant morphology and development to be understood more precisely [46].
FSPMs allow us to model not only these influences, but also to adjust the geometry of the model. This relationship between the structure and the functioning is of particular importance as it enables a plant to be built and understood as a system of connected processes. Figure 7.1 provides an overview of the involved structural parameters as well as the different types of functions of an FSPM interacting with its environment. These functions define the plant's development and modify the geometrical properties while at the same time the structure influences the operation of the functions. Due to the connection of structure and functions, FSPMs are more flexible and can be used in a wider range of different applications.

Godin and Sinoquet [68] provide a general definition for FSPMs and define the following four requirements:

- a FSPM is commonly associated with a 3D plant model that consists of interconnected organs;
- the FSPM represents the spatial distribution of environmental and biological processes, which
- follow a top-down rather than a bottom-up approach, meaning that the model is thought to go from organ to plant level or from plant to stand level;
- offers enough complexity to process and represent a variety of plant characteristics.

As stated, the FSPM paradigm allows us to model a plant at different scales. This requirement originates in various scientific fields that have long been interested in modeling vegetation with different view points and range from macroscopic models of plants and entire plant communities [47] to the definition of specific functions or single organs at a microscopic scale [7, 165].

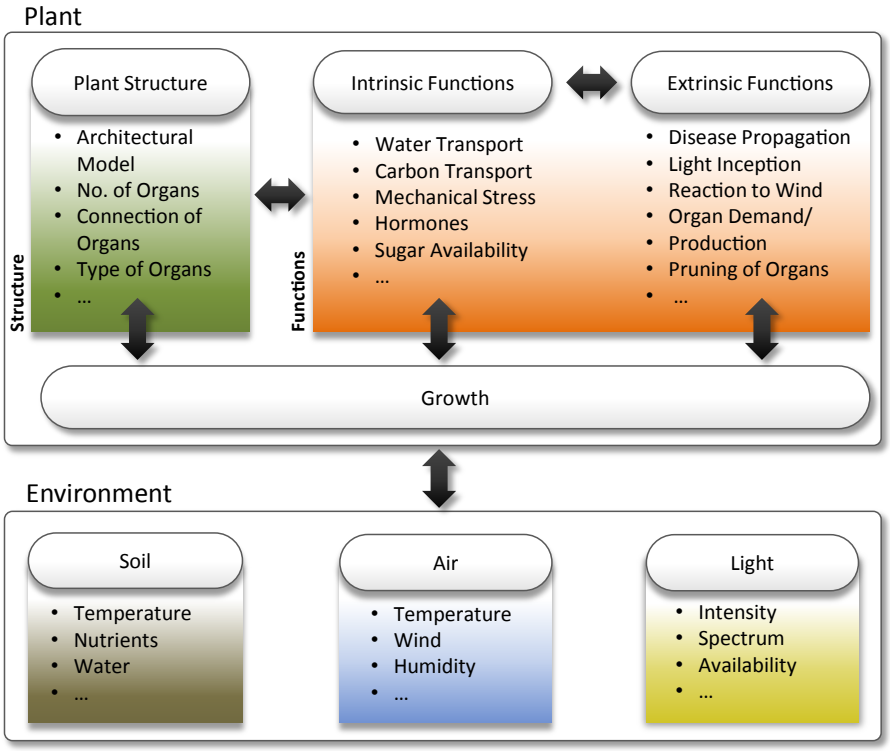


Figure 7.1: A schematic illustration of an FSPM interacting with an environment. Functions can be either intrinsic or extrinsic and define the plant's growth and development. As the illustration indicates the environment influences the development of the plant, but at the same time the plant also adjusts it's environment.

The structural part of an FSPM defines the plant's architecture and specific structural characteristics. This includes the types of organs (Figure 7.2, left), their topological connection (Figure 7.2, right) and a set of geometrical properties, such as leaf angles or curvature [187]. The resulting overall structure is similar and already known to computer graphics applications; the geometry directly yields the visual representation of a plant.
The functional part defines the plant's response to any number of intrinsic and extrinsic physiological, biological or even physical processes, which can best be summarized based on the list of categories provided by Godin and Sinoquet [68]:

- **Plant structure as an interface** - The spatial structure defined by the FSPM paradigm allows us to model the information exchanged by the plant with its environment. This includes the availability of resources such as light and nutrients as well as energy fluxes caused by environmental effects, such as wind attenuation, disease propagation and variations in temperature [27]. The availability of these effects triggers signals that drive the growth and development of a plant. At the same time, however, the plant also adjusts the microclimate in its environment, altogether a complex, interchanging process.

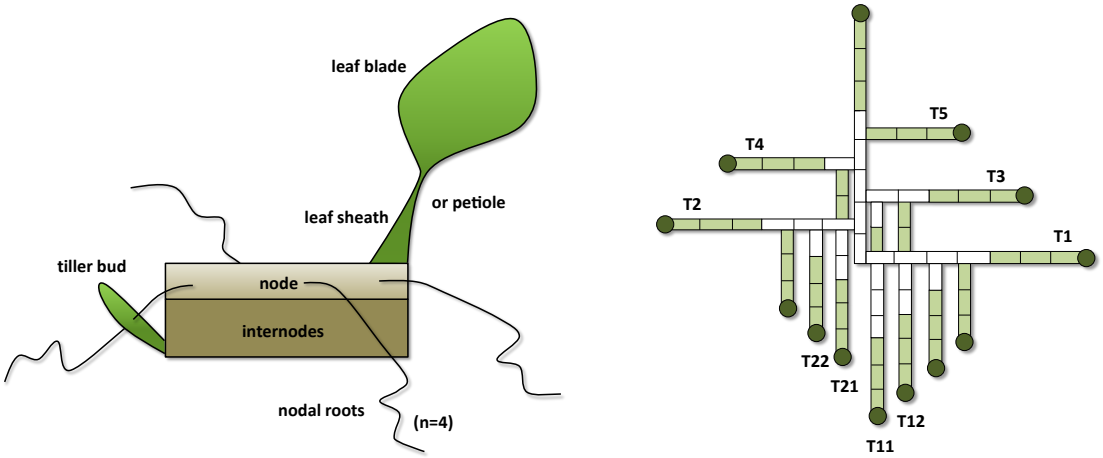


Figure 7.2: Figures from Vos et al. [187]. Left: schematic representation of a phytomer, the basic unit of simulating plant structure (adapted from Figure 1); right: illustration of wheat topology, showing the different types of organs and their connections (adapted from Figure 2, A). The T's represent primary and secondary tillers and shoots.

- **Plant as a network** - The spatial relationship of organs also allows the intrinsic biological development to be modeled. The predominant process of interest is the transport of water within a plant and several approaches have been proposed to model this function [64, 171, 141]. But other fluxes, such as the transport of hormones [34], nutrients [26] and carbon [124], have also been content of scientific investigation.

- **Plant as a developing organism** - Growth and development continuously evolve the spatial structure of a plant and thereby the afore-mentioned intrinsic and extrinsic functions. As the growth of a plant is a complex process, several approaches have been proposed to understand this behavior by modeling certain aspects. This, for example, includes the flushing of buds [177] and the development of plant organs [62]

These categories indicate that the biologically plausible modeling of plants easily comprises a multitude of influences. The FSPM concept offers a general approach for plant development and functioning and thereby helps to structure and clarify methods and techniques from various fields.
A thorough introduction to the biological processes for FSPMs is out of the scope of this thesis and the interested reader is referred to more elaborate papers in this field [46, 68]. Besides providing an overview, Vos *et al.* [187] also introduce guidelines for the construction of dynamic FSPMs.

7.3.2 Software and Concepts

A large number of customized FSPMs as well as specific techniques and software packages exist that allow us to model plants to various needs and according to

the requirements of the scientific fields in which these approaches originate. The following list summarizes the most popular packages and techniques:

- **GREENLAB**[2] - The GREENLAB model is one of the most popular mathematical descriptions aligned with the FSPM paradigm. It was designed to simplify the dynamic modeling of plant morphogenesis and architecture. Hence, one of the goals defining this model was to lower the number of mathematical equations necessary to describe a plant model, while at the same time providing enough complexity to define a large range of botanical and morphological descriptors. GREENLAB allows a large set of functions to be described, corresponding to the *Phenological Process*, the *Modeling of the Organogenesis* and the *Distribution of Biomass* [94, 199].

- **GROIMP**[3] - GROIMP is a modeling platform for creating and visualizing plants according to the FSPM paradigm but focuses on describing the growth process of plants. The GROIMP platform provides means to investigate relational growth grammars expressed in the programming language XL and also allows material properties and geometric descriptions, such as Splines and NURBS, to be specified. GROIMP simplifies the modeling of plant architectures and particularly the investigation of different growth paradigms (Kniemeyer *et al.* [188], Chapter 4).

- **LIGNUM** - In LIGNUM trees are expressed as collections of simple structural components that represent the organs of a tree. The theoretical model focuses on describing the crown of a tree and allows the metabolism taking place in the structural units to be approximated, which produces material for growth. LIGNUM was one of the first models to correspond to the FSPM concept as it models the structure as well as functions for a plant model [136].

- **ALMIS** - The ALMIS model defines a *plant part* and an *environment part* and thereby provides means to model the intrinsic as well as extrinsic functions for a plant model. The plant part is a generic plant model consisting of the components *Meristem, Keaves, Internodes, Roots* and *Roottips*, which allow the modeling of topological, spatial and physiological aspects. The environment part defines a set of microclimatic parameters to specify the plant's surroundings, i.e. soil and air. ALMIS is particularly designed to investigate the flow of carbon for certain tree species [56].

- **Structural Modeling** - Several modeling techniques and software packages exist to model the structural representation of plants. L-Systems [110] are the most popular approach to construct complex branching structures and have a long tradition in modeling plants in computer graphics. Today, a variety of different modeling techniques exist, which enable trees to be reconstructed

[2] http://pma.cirad.fr/GreenLab/
[3] http://www.grogra.de/

from images [130, 150], videos [109], laser-scanned point sets [113, 198], and to be sketched directly [132, 28, 114].
The commercial application *Xfrog*[4] has long been the state of the art in generating plant structures. It provides means to model specific plant characteristics based on a set of rules that are iteratively applied to generate exemplars of a certain tree species. Plants modeled with *Xfrog* convey a high visual quality, however, the result is always a static tree model. *Xfrog* does not support creating, developing or adapting tree models. Newer software packages such as *TreeSketch*[5], *SpeedTree*[6] and *Laubwerk*[7] overcome this limitation and also incorporate tools to sketch and animate models and to have them react to their environment.

As the preceding list indicates, each of the available tools and approaches models plants to specific needs and according to a certain point of view. Although the FSPM concept is a sophisticated theoretical model, so far, no general platform or software system has been developed that incorporates all requirements and aspects. Most of the approaches for generating branching structures originate in computer graphics contexts and by now allow plants to be modeled to a very high level of visual plausibility. However, most of them neglect how to model the plant functioning.

7.3.3 Application Domains and Limitations

A large number of applications need accurate modeling of realistic plants and their developmental behavior at a multiple scale. Researchers of various fields, such as Botany, Biology, Ecology and Agronomy but also Computer Science and Mathematics are interested in realistic representations of plants to better comprehend systems and processes in their field. Most of these domains require precise modeling of the underlying phsyiological, biological or physical behavior in order to derive concrete and representative results.
Trees and plants in computer graphics often serve different purposes. Here, the visual representation is most important, and particularly real-time applications dramatically simplify the geometric description to meet the given time constraints. Although a biological plausible behavior is not required in many cases, knowledge about the underlying processes helps to prepare more realistic exemplars to enable tree models to interact with their surroundings or to even animate plant growth and functioning.
Tree modeling approaches as used in today's computer graphics applications are a long way off sophisticated simulations as carried out in other domains. The FSPM paradigm addresses these limitations by proposing means to model biologically plau-

[4] http://www.greenworks.com/
[5] http://algorithmicbotany.org/TreeSketch/
[6] http://www.speedtree.com/
[7] http://www.laubwerk.com/

sible functions. However, it also introduces a new level of complexity, which cannot be handled interactively by today's computer systems.
Besides the large set of parameters required to model only the structure of plants, as commonly done in a graphics context, a multitude of parameters is required to more precisely define and model biological processes. This contradicts the efforts made in the computer graphics community to lower the number of parameters required for modeling tree and plant exemplars. Content creators, especially for films, games or architecture, require efficient, time-saving tools for designing plants as they often do not account for the main content of a scene. At the same time, however, artists also require models that are biologically plausible and that dynamically react to their environment. So, the goal is to find a balance between the biological correct behavior and the utilization of plants in their application domains.
Advances in hardware technology will allow more parameters to be taken into account in plant models, but it seems more general techniques are required that consider biological plausibility at different levels and at the same time align with the FSPM paradigm.

7.3.4 Functional Models in Computer Graphics

The FSPM paradigm facilitates the generalization of a broad set of already available methods from various scientific fields and eases the development of future approaches, the acquisition of data and the quantification of results. While the biologically plausible modeling is required in many areas, in computer graphics vegetation often needs only to be visually convincing. Today's real-time and interactive applications cannot handle the complexity required to express a multitude of biological processes. On the other hand, higher degrees of visual realism can only be achieved by utilizing more complex and sophisticated theoretical models. This is a contradiction that can only be overcome by carefully designing the biological functions of an FSMP to suit the application's needs.
Hence, from a computer graphics perspective FSPMs do not yet fulfill all requirements. This hinders their broader application and is mostly due to high computation times and a multitude of parameters yielding difficulties for users when working with these models. From the application point of view it is not always necessary to model all aspects of plant development and functioning.
The scalability defined in the FSPM concept describes plant modeling at different geometrical scales, varying from modeling an entire plant to just a single organ. It also allows complex biological, physiological and physical processes to be interconnected in the form of functions. However, it does not provide means to scale the complexity of biological processes. It is for example possible to define a function that models the flow of water within a plant, but such a function can be defined at various levels of biological plausibility. One could use an abstractive flow model to implement this function at a very high level, but could as well define the transport of water based on molecules at a very low level. So far, these techniques only stand for themselves without providing a transition between them.

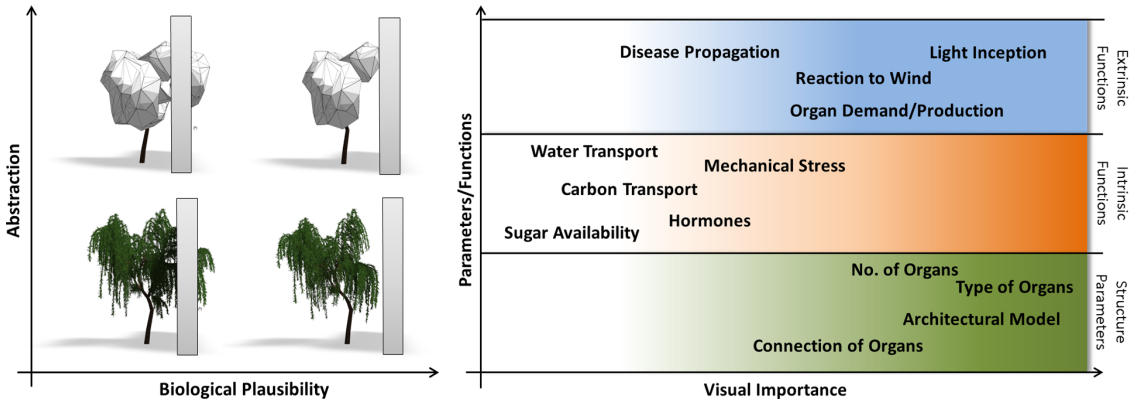


Figure 7.3: Left: a metric correlating the biological plausibility and the abstraction of a tree model. Abstract representations of plant models can also follow biological behavior. Here a tree is shown close to a wall. Increasing the biological plausibility means that the tree model changes according to these envrionmental conditions. Right: a set of parameters describing the structure and the function of an FSPM. Each of these parameters has a different impact in terms of achieving visual plausibility.

Similar to *Level of Detail* techniques, which attempt to lower the geometric complexity of objects in graphics applications, approaches are required to dynamically model biological realism. Smooth transitions between different levels of detail in the functioning of plants would mean that it is possible to remain within the FSPM paradigm consistently, while at the same time a model could be used in a wide variety of applications with varying requirements regarding its biological plausibility. It seems possible to overcome the afore-mentioned limitations by applying level of detail approaches to the functions of a plant model. Dynamically adapting biological plausibility could lower the computational costs and the amount of involved parameters.

In computer graphics, a certain style often needs to be maintained when rendering a scene. *Non-photorealistic Rendering* resembles artistic styles, which often require the abstraction and simplification of the geometrical structure of plant models. Although these applications do not target photo-realism in a visual sense, objects still need to follow realistic rules to mimic real world behavior, including biological plausibility for trees and plants. Figure 7.3 (left) illustrates this correlation for a tree model moved close to a wall. The abstract representation needs to behave in the same way as the photo-realistic representative to avoid artifacts in the rendering. So far, the FSPM paradigm only allows interconnected organs to be modeled at different scales and does not provide means to model abstract or even artificial representations.

Simulations of temperature and wind flow start to be of interest to researchers in the area of urban architecture. They are interested in lowering the energy consumption and try to answer the question if a reduction of energy can be achieved by considering certain layouts of cities or sets of houses. Trees and plants play an important role in such settings. Placing a tree close to a house presumably lowers the energy required to cool the house as it shades its walls from the outside. However, the

opposite effect could also take place, as the tree blocks the air flow. To simulate these effects dozens or even hundreds of plant models need to be managed in the same scene. This again requires abstract representations of plant models that still behave in a biologically plausible way and allow modeling the plant's response to environmental effects. These scenarios require simplified geometrical descriptions and varying levels of detail in the biological behavior of a plant. The FSPM paradigm does not yet support these requirements.

The abstraction of biological behaviour requires that we quantify the impact of certain processes of plant development and functioning. From a graphics point of view, a quantization would target visual realism at tree level. This means that parameters and functions, which modify the visual appearance of the entire model, are most important; processes on a microscopic level play a subordinary role and could be modeled with a coarse level of detail. Figure 7.3 (right) illustrates the importance of functions and structural parameters according to the goal of achieving visual plausibilty. Other scientific areas may define different priorities. Ideally measures to quantify *Biological Correctness* would support this selection by allowing the impact of single functions and entire sets of processes to be verified. However, such measures are not easy to find.

All three points, the scalability of biological functions, the biolocial plausible behavior for abstract representations and the quantization of the functions of a FSPM are interesting areas of research and raise a large number of further questions in the relevant fields.

7.4 Conclusion

As frequent objects in our daily lives, trees and plants play an important role in nearly all virtual scenes and setups. Since the early beginning of computer graphics, researchers and content creators have striven to replicate these objects as means to enhance virtual worlds. With their enormous natural complexity, the processing of trees and plants holds a strong fascination.
Today, models of trees and plants are ubiquitous in a wide range of applications and enhance the virtual realism in a plethora of applications. The realism of these objects has increased dramatically since early attempts; nowadays approaches exist, which can model vegetation at different scales - be it entire branching structures for fully developed trees or subtle material properties to replicate the reflectance characteristics of a leaf surface. Often, however, these techniques are thought and developed from a computer graphics point of view. Interactive and real-time applications, in particular, define hard constraints which put severe limitations on processing vegetation. Photo-realistic representations at multiple levels of detail, which also convey biologically plausible behavior, are still a long way off.
This thesis strives to contribute to the field of processing vegetation by proposing methods, techniques and paradigms to migrate static tree models, as generated by various modeling approaches, to dynamically adapting and developing ones. Improvements in technology will enable more realistic models of trees and plants to be computed, however specialized and efficient algorithms will also be required in the future to realize efficient processing in real-time and interactive applications.

A

Evaluation Results: Self Adapting Tree Models

In order to evaluate the tree transformations introduced in Chapter 4 we created three types of trees. The following tables show the results obtained based on the distance measure discussed in Section 4.5.4. The letters represent different sets of trees. O: growth model, solitary grown; G: growth model, wall; T: transformations, wall.

A.1 Tree Type A

T-G		G-G		T-O		T-T	
358.15	291.59	382.27	368.90	504.51	304.66	339.61	393.49
360.82	389.59	435.42	467.44	459.36	456.41	429.95	394.60
358.60	424.31	314.27	418.89	443.86	483.28	307.51	347.36
576.08	378.80	353.77	525.21	439.31	470.30	439.08	426.15
310.09	647.59	341.99	435.47	293.98	539.50	848.53	348.54
374.98	579.42	386.72	415.38	505.85	359.72	363.43	418.82
352.19	550.25	353.85	491.45	535.17	550.02	366.15	526.84
431.88	478.86	351.48	402.17	430.34	547.39	447.66	418.84
530.03	467.58	326.19	425.34	474.46	556.27	494.31	354.33
684.41	319.26	535.24	373.12	465.67	494.29	441.91	455.26
612.79	493.04	480.10	404.74	534.67	478.38	395.31	409.54
493.19	421.22	514.82	420.44	633.37	448.19	410.36	481.84
426.65	426.35	464.42	363.03	428.63	398.97	379.40	416.22
323.66	365.31	447.51	350.48	382.02	476.00	498.49	314.42
680.88	353.72	577.53	384.10	413.83	458.58	397.55	447.74
495.97	474.79	432.49	369.13	405.74	406.32	326.90	441.48
278.75	314.48	206.19	296.23	462.77	599.63	371.11	325.51
686.74	450.02	439.42	341.00	392.97	515.17	474.09	303.75
361.18	416.86	380.48	336.62	505.48	449.22	350.87	324.03
494.42	308.20	343.85	494.40	512.85	578.64	327.21	607.86
358.16	336.25	427.49	392.80	475.80	405.69	372.50	548.73
464.59	613.14	320.09	410.64	386.24	407.78	347.09	378.27
418.64	352.81	322.45	262.07	503.79	556.84	424.90	421.07
340.22	389.15	440.24	333.72	597.25	498.12	392.90	312.61
303.78	267.45	360.10	419.58	467.79	486.88	457.71	460.25

Table A.1: Phototropism: 0.25, Gravitropism: 0.12.

A.2 Tree Type B

T-G		G-G		T-O		T-T	
206.66	202.73	184.01	320.62	386.74	272.05	145.48	236.27
216.95	248.53	289.15	218.94	267.83	357.81	219.27	161.14
272.61	285.17	288.34	292.76	487.86	287.19	238.99	249.66
214.57	230.28	245.57	210.37	389.59	399.30	212.01	255.39
240.35	311.32	216.33	180.13	392.30	301.88	247.75	204.75
278.61	218.42	238.85	109.50	306.17	387.20	163.77	222.90
287.30	200.67	235.87	186.36	338.19	311.76	243.83	217.07
257.15	192.28	161.98	253.16	333.50	226.89	286.97	268.66
266.95	249.58	130.78	235.48	262.27	325.15	215.99	214.47
203.63	243.64	206.22	120.25	235.47	438.84	220.17	235.23
261.48	210.64	208.59	156.16	414.23	503.84	199.30	270.24
214.84	303.95	293.62	163.01	311.03	275.72	182.15	249.65
209.28	252.19	255.82	246.75	404.39	428.80	210.38	214.93
259.68	188.03	218.46	251.58	325.50	535.53	243.97	286.21
197.16	166.60	176.60	170.86	228.23	511.30	302.73	223.97
232.62	202.01	268.34	333.57	301.92	402.44	169.31	224.58
208.45	220.08	182.14	239.33	294.93	279.82	209.88	199.43
260.05	231.07	316.82	251.85	371.76	634.62	195.61	251.15
233.15	192.10	261.58	220.40	339.24	226.55	240.22	264.52
304.20	207.68	265.43	174.84	340.24	279.54	230.41	193.87
231.20	225.48	256.75	259.51	309.55	261.45	193.93	213.13
255.09	248.43	262.59	188.12	278.63	186.64	217.29	202.67
237.41	314.89	217.93	250.44	495.30	282.10	231.71	189.28
209.65	187.36	305.87	264.63	380.97	356.16	263.21	213.04
264.05	258.67	295.68	200.98	421.26	269.96	290.41	232.43

Table A.2: Phototropism: 0.45, Gravitropism: 0.03.

A.3 Tree Type C

T-G		G-G		T-O		T-T	
424.73	802.51	552.55	987.77	890.91	928.46	533.86	510.21
443.66	584.42	804.41	901.55	930.21	922.11	510.64	601.94
416.80	1091.10	387.81	953.84	1191.30	681.94	584.11	512.80
499.36	544.57	741.46	545.33	837.69	1108.53	622.26	469.63
1091.55	997.88	803.74	732.85	990.95	1101.47	347.31	527.41
823.13	876.26	1043.59	850.54	640.36	940.16	691.77	503.97
671.28	532.85	406.19	649.62	693.61	1151.74	508.70	566.33
893.33	539.39	978.48	806.42	893.46	1128.18	623.56	604.72
779.68	1165.36	843.57	1007.15	867.18	1284.72	503.22	638.44
415.08	615.08	415.36	1073.30	1182.11	1154.67	531.30	520.33
472.91	1012.35	1017.61	889.18	612.88	1000.12	588.58	526.63
1183.33	715.40	900.91	1156.55	498.64	643.11	460.36	482.84
478.52	1067.44	1020.10	1175.79	712.13	1186.70	675.94	509.51
616.25	1022.09	861.27	612.26	909.23	692.09	568.59	623.13
653.77	1300.41	823.68	940.10	1234.93	941.08	687.05	553.02
974.43	581.76	963.27	381.30	1022.64	589.29	521.29	567.53
537.13	688.39	858.03	949.05	1096.01	814.73	591.27	537.98
609.00	1040.44	895.28	957.72	649.86	1010.42	686.27	614.67
585.78	470.07	1127.35	834.60	937.86	1078.40	428.28	504.97
1124.71	1100.24	506.00	887.93	1030.09	893.12	523.88	659.64
1039.94	417.24	942.73	333.00	1076.66	1113.81	404.71	630.22
709.00	1024.69	1301.23	856.21	1116.24	1009.17	386.57	574.84
464.41	1196.37	585.58	1311.92	538.60	984.21	386.59	725.13
398.23	978.12	442.75	1052.21	932.05	1083.67	475.40	570.53
608.47	984.62	1037.75	983.00	445.10	1009.87	538.35	615.67

Table A.3: Phototropism: 0.44, Gravitropism: 0.23.

Bibliography

[1] Agarwal, G., Belhumeur, P., Feiner, S., Jacobs, D., Kress, W., Ramamoorthi, R., Bourg, N., Dixit, N., Ling, H., Mahajan, D., et al. First steps toward an electronic field guide for plants. *Taxon 55*, 3 (2006), 597–610.

[2] Aono, M., and Kunii, T. Botanical tree image generation. *IEEE Computer Graphics and Applications 4* (1984), 10–34.

[3] Arvo, J., and Kirk, D. Modeling plants with environment-sensitive automata. In *Proceedings of Ausgraph'88* (1988), pp. 27–33.

[4] Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., and Lee, T.-Y. Skeleton extraction by mesh contraction. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 44:1–44:10.

[5] Baranoski, G. V., and Rokne, J. G. An algorithmic reflectance and transmittance model for plant tissue. *Computer Graphics Forum 16*, 3 (1997), C141–C150.

[6] Baranoski, G. V. G., and Rokne, J. G. Efficiently simulating scattering of light by leaves. *The Visual Computer 17*, 8 (2001), 491–505.

[7] Barillot, R., Louarn, G., Escobar-Gutirrez, A. J., Huynh, P., and Combes, D. How good is the turbid medium-based approach for accounting for light partitioning in contrasted grasslegume intercropping systems? *Annals of Botany 108*, 6 (2011), 1013–1024.

[8] Barthélémy, D., and Caraglio, Y. Plant architecture: A dynamic, multilevel and comprehensive approach to plant form, structure and ontogeny. *Annals of Botany 99*, 3 (2007), 375–407.

[9] Bavoil, L., and Myers, K. Order independent transparency with dual depth peeling. Tech. rep., 2008.

[10] Beck, C. B. *An Introduction to Plant Structure and Development: Plant Anatomy for the Twenty-First Century, 2nd Edition.* Cambridge University Press, 2010.

[11] Behrendt, S., Colditz, C., Franzke, O., Kopf, J., and Deussen, O. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum 24*, 3 (2005), 507–516.

[12] Benes, B., Stava, O., Mech, R., and Miller, G. Guided procedural modeling. *Computer Graphics Forum 30*, 2 (2011), 325–334.

[13] Beneš, B., Andrysco, N., and Šťava, O. Interactive modeling of virtual ecosystems. In *Proceedings of the Fifth Eurographics conference on Natural Phenomena* (Aire-la-Ville, Switzerland, Switzerland, 2009), NPH'09, Eurographics Association, pp. 9–16.

[14] Beneš, B., and Millán, E. U. Virtual climbing plants competing for space. In *Proceedings of the Computer Animation* (Washington, DC, USA, 2002), CA '02, IEEE Computer Society, pp. 33–ff.

[15] Bishop, R. L. There is more than one way to frame a curve. *The Ameriacan Mathematical Monthly 82*, 3 (1975), 246–251.

[16] Blinn, J. F. Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 286–292.

[17] Bloomenthal, J. Modeling the mighty maple. *SIGGRAPH Computer Graphphics 19* (July 1985), 305–311.

[18] Bokeloh, M., Wand, M., and Seidel, H.-P. A connection between partial symmetry and inverse procedural modeling. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 104:1–104:10.

[19] Boudon, F., Prusinkiewicz, P., Federl, P., Godin, C., and Karwowski, R. Interactive design of bonsai tree models. *Computer Graphics Forum. Proceedings of Eurographics 22*, 3 (2003), 591–599.

[20] Boulanger, K. *Real-Time Realistic Rendering of Nature Scenes with Dynamic Lighting.* PhD thesis, INRIA, University of Rennes, France, 2008.

[21] Boulanger, K., Bouatouch, K., and Pattanaik, S. Rendering trees with indirect lighting in real time. In *Proceedings of the Nineteenth Eurographics conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2008), EGSR'08, Eurographics Association, pp. 1189–1198.

[22] Bruneton, E., and Neyret, F. Real-time Realistic Rendering and Lighting of Forests. *Computer Graphics Forum 31*, 2pt1 (May 2012), 373–382. Special issue: Proceedings of Eurographics 2011.

[23] Bucksch, A., and Lindenbergh, R. Campino – a skeletonization method for point cloud processing. *ISPRS journal of photogrammetry and remote sensing 63*, 1 (2008), 115–127.

[24] Bucksch, A., Lindenbergh, R., and Menenti, M. Skeltre - fast skeletonisation for imperfect point cloud data of botanic trees. In *Proceedings of Eurographics Workshop on 3D Object Retrieval* (2009), pp. 13–27.

[25] Candussi, A., Candussi, N., and Hllerer, T. Rendering realistic trees and forests in real time. *Computer Graphics Forum* (2005).

[26] Chandler, J. W., and Dale, J. E. Photosynthesis and nutrient supply in needles of sitka spruce [picea sitchensis (bong.) carr.]. *New Phytologist 125*, 1 (1993), 101–111.

[27] Chelle, M. Phylloclimate or the climate perceived by individual plant organs: What is it? how to model it? what for? *New Phytologist 166*, 3 (2005), 781–790.

[28] Chen, X., Neubert, B., Xu, Y.-Q., Deussen, O., and Kang, S. B. Sketch-based tree modeling using markov random field. In *ACM SIGGRAPH Asia 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 109:1–109:9.

[29] Cheng, Z., Zhang, X., and Chen, B. Simple reconstruction of tree branches from a single range image. *Journal of Computer Science and Technology 22*, 6 (2007), 846–858.

[30] Chiba, Y. Plant form analysis based on the pipe model theory I. A statical model within the crown. *Ecological Research 5* (1990), 207–220. 10.1007/BF02346992.

[31] Chiba, Y. Plant form based on the pipe model theory II. Quantitative analysis of ramification in morphology. *Ecological Research 6*, 1 (1991), 21–28.

[32] Cieslak, M., Lemieux, C., Hanan, J., and Prusinkiewicz, P. Quasi-monte carlo simulation of the light environment of plants. *Functional Plant Biology 10*, 35 (2008), 837–849.

[33] Cline, M. G., Bhave, N., and Harrington, C. A. The possible roles of nutrient deprivation and auxin repression in apical control. *Trees 23* (Dec 2009), 498–500.

[34] Cline, M. G., and Harrington, C. A. Apical dominance and apical control in multiple flushing of temperate woody species. *Canadian Journal of Forest Research* (Jan 2006), 74–83.

[35] Cook, R. L., Halstead, J., Planck, M., and Ryu, D. Stochastic simplification of aggregate detail. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.

[36] Corsini, M., Larabi, M.-C., Lavou, G., Petrk, O., Va, L., and Wang, K. Perceptual metrics for static and dynamic triangle meshes. In *Eurographics, State of The Art Report* (May 2012).

[37] Côté, J.-F., Widlowski, J.-L., Fournier, R. A., and Verstraete, M. M. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment 113*, 5 (2009), 1067 – 1081.

[38] Cournède, P.-H., Kang, M.-Z., Mathieu, A., Barczi, J.-F., Yan, H.-P., Hu, B.-G., and De Reffye, P. Structural factorization of plants to compute their functional and architectural growth. *Simulation 82*, 7 (July 2006), 427–438.

[39] Cournède, P.-H., Letort, V., Mathieu, A., Lang, M., Lemaire, S., Trevezas, S., Houllier, F., and de Reffye, P. Some parameter estimation issues in functional-structural plant modelling. *Mathematical Modeling of Natural Phenomena 6* (2011), 133–159.

[40] Cozzi, P., and Christophe, R., Eds. *OpenGL Insights*. Taylor & Francis, 2012.

[41] Crow, F. C. Shadow algorithms for computer graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1977), SIGGRAPH '77, ACM, pp. 242–248.

[42] de Reffye, P., Edelin, C., Françon, J., Jaeger, M., and Puech, C. Plant models faithful to botanical structure and development. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 151–158.

[43] Decaudin, P., and Neyret, F. Rendering forest scenes in real-time. In *Rendering Techniques '04 (Eurographics Symposium on Rendering)* (June 2004), pp. 93–102.

[44] Decaudin, P., and Neyret, F. Volumetric billboards. *Computer Graphics Forum 28*, 8 (2009), 2079–2089.

[45] Décoret, X., Durand, F., Sillion, F. X., and Dorsey, J. Billboard clouds for extreme model simplification. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 689–696.

[46] DeJong, T. M., Da Silva, D., Vos, J., and Escobar-Gutirrez, A. J. Using functional-structural plant models to study, understand and integrate plant development and ecophysiology. *Annals of Botany 108*, 6 (2011), 987–989.

[47] Delagrange, S., and Rochon, P. Reconstruction and analysis of a deciduous sapling using digital photographs or terrestrial-lidar technology. *Annals of Botany 108*, 6 (2011), 991–1000.

[48] Deussen, O., Colditz, C., Stamminger, M., and Drettakis, G. Interactive visualization of complex plant ecosystems. In *VIS '02: Proceedings of the Conference on Visualization '02* (2002), pp. 219–226.

[49] Deussen, O., and Lintermann, B. A modelling method and user interface for creating plants. In *Proceedings of the Conference on Graphics Interface '97*

(Toronto, Ont., Canada, Canada, 1997), Canadian Information Processing Society, pp. 189–197.

[50] Deussen, O., and Lintermann, B. *Digital Design of Nature: Computer Generated Plants and Organics*. SpringerVerlag, 2004.

[51] Donnelly, W., and Lauritzen, A. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 161–165.

[52] Drettakis, G., Bonneel, N., Dachsbacher, C., Lefebvre, S., Schwarz, M., and Viaud-Delmon, I. An Interactive Perceptual Rendering Pipeline using Contrast and Spatial Masking. In *Rendering Techniques 2007 (Proceedings of EGSR)* (2007).

[53] Edelsbrunner, H., and Mücke, E. P. Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (1994), 43–72.

[54] Efros, A. A., and Freeman, W. T. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01* (2001), pp. 341–346.

[55] Eissele, M., Weiskopf, D., and Ertl, T. The g^2-buffer framework. In *In Simulation and Visualization* (2004), pp. 287–298.

[56] Eschenbach, C. Emergent properties modelled with the functional structural tree growth model almis: Computer experiments on resource gain and use. *Ecological Modelling 186* (2005), 470–488.

[57] Federl, P. *Modeling fracture formation on growing surfaces*. PhD thesis, University of Calgary, Canada, 2003.

[58] Ferraro, P., and Godin, C. A distance measure between plant architectures. *Annals of Forest Science 57*, 5/6 (2000), 445–461.

[59] Ferraro, P., Godin, C., and Prusinkiewicz, P. Toward a quantification of self-similarity in plants. *Fractals 13*, 2 (2005), 91–109.

[60] Floriani, L., Kobbelt, L., and Puppo, E. A survey on data structures for level-of-detail models. In *Advances in Multiresolution for Geometric Modelling*, N. Dodgson, M. Floater, and M. Sabin, Eds., Mathematics and Visualization. Springer Berlin Heidelberg, 2005, pp. 49–74.

[61] Foley, J., Van Dam, A., Feiner, S., and Hughes, J. Computer graphics: Principles and practice. *Addison-Wesley* (1995), p. 227.

[62] Fournier, C., Durand, J. L., Ljutovac, S., Schufele, R., Gastal, F., and Andrieu, B. A functionalstructural model of elongation of the grass leaf and its relationships with the phyllochron. *New Phytologist 166*, 3 (2005), 881–894.

[63] Franzke, O., and Deussen, O. Rendering plant leaves faithfully. In *ACM SIGGRAPH 2003 Sketches & Applications* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 1–1.

[64] Früh, T., and Kurth, W. The hydraulic system of trees: theoretical framework and numerical solution. *Journal of Theoretical Biology 201* (1999), 251–270.

[65] Gardner, G. Y. Simulation of natural scenes using textured quadric surfaces. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 11–20.

[66] Givnish, T. Adapatation to sun and shade - a whole plant perspective. *Functional Plant Biology 15, 2* (1988), 63–92.

[67] Godin, C. Representing and encoding plant architecture: A review. *Annals of Forest Science 57* (2000), 413–438.

[68] Godin, C., and Sinoquet, H. Functionalstructural plant modelling. *New Phytologist 166*, 3 (2005), 705–708.

[69] Gravelius, H. *Flusskunde.* Grundriss der gesamten Gewässerkunde. G.J. Göschen, 1914.

[70] Gray, W. M. Hormonal regulation of plant growth and development. *PLOS Biology 2* (2004), 9, e311.

[71] Greene, N. Voxel space automata: modeling with stochastic growth processes in voxel space. *SIGGRAPH Computer Graphics 23* (July 1989), 175–184.

[72] Habel, R. *Real-time Rendering and Animation of Vegetation.* PhD thesis, Technische Universität Wien, Austria, 2009.

[73] Habel, R., Kusternig, A., and Wimmer, M. Physically based real-time translucency for leaves. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)* (June 2007), J. Kautz and S. Pattanaik, Eds., Eurographics, Eurographics Association, pp. 253–263.

[74] Habel, R., Kusternig, A., and Wimmer, M. Physically guided animation of trees. *Computer Graphics Forum (Proceedings EUROGRAPHICS 2009) 28*, 2 (Mar. 2009), 523–532.

[75] Haevre, W. V., Fiore, F. D., Bekaert, P., and Reeth, F. V. A ray density estimation approach to take into account environment illumination in plant growth simulation. In *SCCG '04: Proceedings of the 20th spring conference on Computer graphics* (New York, NY, USA, 2004), ACM Press, pp. 121–131.

[76] Haevre, W. V., Fiore, F. D., and Reeth, F. V. Physically-based driven tree animations. *Eurographics Workshop on Natural Phenomena* (2006).

[77] Hallé, A., Oldeman, I., and Tomlinson, J. *Tropical Trees and Forest: an Architectural Analysis*. Springer–Verlag, Berlin/Heidelberg/New-York, 1978.

[78] Hanson, A. J., and Ma, H. Parallel transport approach to curve framing. *Technical Report TR425, Indiana University* (1995).

[79] Hart, J. C., and Baker, B. Implicit modeling of tree surfaces. In *In Proceedings of Implicit Surfaces 96* (1996), pp. 143–152.

[80] Hart, J. C., Baker, B., and Michaelraj, J. Structural simulation of tree growth and response. *The Visual Computer 19*, 2-3 (2003), 151–163.

[81] Hasenauer, H., and Monserud, R. A. A crown ratio model for austrian forests. *Forest Ecology and Management 84*, 1-3 (1996), 49 – 60.

[82] Hasenfratz, J.-M., Lapierre, M., Holzschuch, N., and Sillion, F. X. A survey of Real-Time Soft Shadows Algorithms. *Computer Forum 22*, 4 (Dec. 2003), 753–774.

[83] Hegeman, K., Premože, S., Ashikhmin, M., and GeorgeDrettakis. Approximate ambient occlusion for trees. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM Press, pp. 87–92.

[84] Holton, M. Strands, gravity and botanical tree imagery. *Computer Graphics Forum 13(I)* (1994), 57–67.

[85] Honda, H. Description of the form of trees by the parameters of the tree-like body: effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology 31* (1971), 331–338.

[86] Hua, J., and Kang, M. Functional tree models reacting to the environment. In *ACM SIGGRAPH 2011 Posters* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 60:1–60:1.

[87] Ijiri, T., Měch, R., Igarashi, T., and Miller, G. An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 4 (2008), 429–436.

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

[89] Jaccard, P. Eine neue auffassung über die ursachen des dickenwachstums der bäume, 1913.

[90] Jakulin, A. Interactive vegetation rendering with slicing and blending. In *Proceedings of Eurographics (Short Presentations)* (Interlaken, Switzerland, 2000), A. d. Sousa and J. C. Torres, Eds.

[91] Jirasek, C., Prusinkiewicz, P., and Moulina, B. Integrating biomechanics into developmental models expressed in l-systems. *Plant Biomechanics Germany: Badweiler* (2000), 615–624.

[92] Kaiser, P., and Boynton, R. *Human color vision.* Optical Society of America, 1996.

[93] Kaneko, T., Takahei, T., Inami, M., Kawakami, N., Yanagida, Y., Maeda, T., and Tachi, S. Detailed shape representation with parallax mapping. In *In Proceedings of the ICAT 2001* (2001), pp. 205–208.

[94] Kang, M. Z., Cournède, P. H., de Reffye, P., Auclair, D., and Hu, B. G. Analytical study of a stochastic plant growth model: Application to the greenlab model. *Mathematics and Computers Simulation 78*, 1 (June 2008), 57–75.

[95] Kang, M.-Z., Heuvelink, E., Carvalho, S. M. P., and De Reffye, P. A virtual plant that responds to the environment like a real one: the case for chrysanthemum. *New Phytologist 195* (2012), 384–395.

[96] Kawaguchi, Y. A morphological study of the form of nature. *SIGGRAPH Computer Graphics 16*, 3 (July 1982), 223–232.

[97] Key, T., Warner, T., McGraw, J., and Fajvan, M. A Comparison of Multispectral and Multitemporal Information in High Spatial Resolution Imagery for Classification of Individual Tree Species in a Temperate Hardwood Forest. *Remote Sensing of Environment 75*, 1 (2001), 100–112.

[98] Kharlamov, A., Cantlay, I., and Stepanenko, Y. Gpu gems 3 - next-generation speedtree rendering. *Addison-Wesley Professional* (2007).

[99] Kilpeläinen, P., and Mannila, H. The tree inclusion problem. In *TAPSOFT, Vol.1* (1991), pp. 202–214.

[100] Kurth, W. Morphological models of plant growth: Possibilities and ecological relevance. *Ecological Modelling 75–76*, 0 (1994), 299 – 308. State–of–the–Art in Ecological Modelling proceedings of ISEM's 8th International Conference.

[101] Lacewell, D., Burley, B., Boulos, S., and Shirley, P. Raytracing prefiltered occlusion for aggregate geometry. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (Aug.), pp. 19–26.

[102] Lacewell, J. D., Edwards, D., Shirley, P., and Thompson, W. B. Stochastic billboard clouds for interactive foliage rendering. *Journal of Graphics, GPU, and Game Tools 11*, 1 (2006), 1–12.

[103] Lam, Z., and King, S. A. Simulating tree growth based on internal and environmental factors. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2005), GRAPHITE '05, ACM, pp. 99–107.

[104] Lefebvre, S., and Neyret, F. Synthesizing bark. In *Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), EGRW '02, Eurographics Association, pp. 105–116.

[105] Lengyel, E. Mathematics for 3d game programming and computer graphics. *Course Technology* (2003), 453–497.

[106] Letort, V., Cournde, P. H., Mathieu, A., De Reffye, P., and Constant, T. Parametric identification of a functional-structural tree growth model and application to beech trees (fagus sylvatica). *Functional Plant Biology 35, 10* (2008), 951–963.

[107] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. Tech. Rep. 8, 1966.

[108] Leyser, O., and Day, S. *Mechanisms in Plant Development*. Blackwell Publishing, 2003.

[109] Li, C., Deussen, O., Song, Y.-Z., Willis, P., and Hall, P. Modeling and generating moving trees from video. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 127:1–127:12.

[110] Lindenmayer, A. Mathematical models for cellular interaction in development: Parts i and ii. *Journal of Theoretical Biology 18* (1968).

[111] Lintermann, B., and Deussen, O. Interactive modeling of plants. *IEEE Computer Graphics Applications 19*, 1 (Jan. 1999), 56–65.

[112] Livny, Y., Pirk, S., Cheng, Z., Yan, F., Deussen, O., Cohen-Or, D., and Chen, B. Texture-lobes for tree modelling. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 53:1–53:10.

[113] Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., and El-Sana, J. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics 29* (December 2010), 151:1–151:8.

[114] Longay, S., Runions, A., Boudon, F., and Prusinkiewicz, P. Treesketch: interactive procedural modeling of trees on a tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2012), SBIM '12, pp. 107–120.

[115] Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., and Huebner, R. *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2012.

[116] Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. *Level of Detail for 3D Graphics*. Elsevier Science, 2002.

[117] Luft, T., Balzer, M., and Deussen, O. Expressive illumination of foliage based on implicit surfaces. In *Proceedings of the Eurographics Workshop on Natural Phenomena, NPH 2007, Prague, Czech Republic, 2007* (2007), Eurographics Association, pp. 71–78.

[118] Luft, T., Colditz, C., and Deussen, O. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics 25*, 3 (jul 2006), 1206–1213.

[119] Ma, Q., Ishimaru, A., Phu, P., and Kuga, Y. Transmission, reflection, and depolarization of an optical wave for a single leaf. *Geoscience and Remote Sensing, IEEE Transactions on 28*, 5 (Sept.), 865–872.

[120] Mammen, A. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Comput. Graph. Appl. 9*, 4 (July 1989), 43–55.

[121] Mandelbrot, B. *The fractal geometry of nature*. W. H. Freeman and Comp., New York, 1983.

[122] Max, N. Hierarchical image-based rendering using texture mapping hardware. In *In Eurographics Workshop on Rendering 99* (1999), pp. 57–62.

[123] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *Journal of Medical Physics 21*, 6 (1953), 1087–1092.

[124] Minchin, P. E. H., and Lacointe, A. New understanding on phloem physiology and possible consequences for modelling long-distant carbon transport. *New Phytologist 166* (2005), 771–779.

[125] Monsi, M., and Saeki, T. über den lichtfaktor in den pflanzengesellschaften und seine bedeutung für die stoffproduktion. *Annals of Botany 14* (1953), 22–52.

[126] Moore, J., and Jefferies, D. Covering new ground: Foliage rendering in pure. *SIGGRAPH - 3D Graphics and Games Courses* (2009).

[127] Murray, C. D. A relationship between circumference and weight in trees and its bearing on branching angles. *The Journal of General Physiology 10*, 5 (May 1927), 725–729.

[128] Měch, R., and Prusinkiewicz, P. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 397–410.

[129] Nadenau, M. J., Winkler, S., Alleysson, D., and Kunt, M. Human Vision Models for Perceptually Optimized Image Processing – A Review. In *Proceedings of the IEEE* (2000).

[130] Neubert, B., Franken, T., and Deussen, O. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics 26*, 3 (July 2007).

[131] Niklas, K. J. *Plant Allometry: The Scaling of Form and Process*. University Of Chicago Press, 1994.

[132] Okabe, M., Owada, S., and Igarashi, T. Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[133] Oppenheimer, P. E. Real time design and animation of fractal plants and trees. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 55–64.

[134] Ota, S., Tamura, M., Fujita, K., Fujimoto, T., Muraoka, K., and Chiba, N. 1/f beta; noise-based real-time animation of trees swaying in wind fields. In *Computer Graphics International, 2003. Proceedings* (july 2003), pp. 52 – 59.

[135] Palubicki, W., Horel, K., Longay, S., Runions, A., Lane, B., Měch, R., and Prusinkiewicz, P. Self-organizing tree models for image synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH '09) 28* (July 2009), 58:1–58:10.

[136] Perttunen, J., Sievänen, R., Nikinmaa, E., Salminen, H., Saarenmaa, H., and Väkevä, J. Lignum: a tree model based on simple structural units. *Annals of Botany 77* (1996), 87–88.

[137] Pfeifer, N., Gorte, B., and Winterhalder, D. Automatic reconstruction of single trees from terrestrial laser scanner data. *ISPRS XX th Congress. Istanbul. Turkey. 12.07.- 23.07.2004 Pitas, I* (2004), 0–47137739.

[138] Pirk, S., Niese, T., Deussen, O., and Neubert, B. Capturing and animating the morphogenesis of polygonal tree models. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 169:1–169:10.

[139] Pirk, S., Stava, O., Kratt, J., Said, M. A. M., Neubert, B., Měch, R., Benes, B., and Deussen, O. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics 31*, 4 (July 2012), 50:1–50:10.

[140] Power, J. L., Brush, A. J. B., Prusinkiewicz, P., and Salesin, D. H. Interactive arrangement of botanical l-system models. In *Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM Press, pp. 175–182.

[141] Prieto, I., Armas, C., and Pugnaire, F. I. Water release through plant roots: new insights into its consequences at the plant and ecosystem level. *New Phytologist 193*, 4 (2012), 830–841.

[142] Prusinkiewicz, P. Modeling of spatial structure and development of plants: a review. *Scientia Horticulturae 74*, 1- 2 (1998), 113 – 149.

[143] Prusinkiewicz, P. Modeling plant growth and development. *Current Opinion in Plant Biology 7*, 1 (Feb. 2004), 79–83.

[144] Prusinkiewicz, P., Hammel, M. S., and Mjolsness, E. Animation of plant development. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 351–360.

[145] Prusinkiewicz, P., and Hanan, J. Visualization of botanical structures and processes using parametric l-systems. In *Scientific Visualization and Graphics simulation'90* (1990), D.Thalmann, Ed., vol. 22(4), J.Wiley & Sons, Ltd, pp. 183–201.

[146] Prusinkiewicz, P., Hanan, J., Hammel, M., and Měch, R. L-systems: from the theory to visual models of plants. *Machine Graphics and Vision 2(4)* (1993), 12–22.

[147] Prusinkiewicz, P., Mündermann, L., Karwowski, R., and Lane, B. The use of positional information in the modeling of plants. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 289–300.

[148] Qin, X., Nakamae, E., Tadamura, K., and Nagai, Y. Fast photo-realistic rendering of trees in daylight. *Computer Graphics Forum 22*, 3 (2003), 243–252.

[149] Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., and Kang, S. B. Image-based plant modeling. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 599–604.

[150] Reche-Martinez, A., Martin, I., and Drettakis, G. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 720–727.

[151] Reeves, W. T., and Blau, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Computer Graphics 19*, 3 (1985), 313–322.

[152] Renton, M., Hanan, J., and Kaitaniemi, P. The inside story: including physiology in structural plant models. 95–ff.

[153] Richter, J. P. *The Notebooks of Leonardo da Vinci*, vol. 1. Dover Publications Inc, New York, 1970. reprinted 1888.

[154] Ripperda, N., and Brenner, C. Reconstruction of facade structures using a formal grammar and rjmcmc. In *Proceedings of the 28th conference on Pattern Recognition* (Berlin, Heidelberg, 2006), DAGM'06, Springer-Verlag, pp. 750–759.

[155] Risto Sievänen, Eero Nikinmaa, Pekka Nygren, Harry Ozier-Lafontaine, Jari Perttunen, and Harri Hakula. Components of functional-structural tree models. *Ann. For. Sci. 57*, 5 (2000), 399–412.

[156] Roettger, S., Irionar, A., and Ertl, T. Shadow Volumes Revisited. In *Proceedings of WSCG '02* (2002), pp. 373–393.

[157] Rudnick, S., Linsen, L., and Mcpherson, E. G. Inverse modeling and animation of growing single-stemmed trees at interactive rates. In *in The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007, 2007* (2007), pp. 217–224.

[158] Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A.-G., and Prusinkiewicz, P. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics 24*, 3 (2005), 702–711.

[159] Runions, A., Lane, B., and Prusinkiewicz, P. Modeling trees with a space colonization algorithm. In *Proceedings of the Third Eurographics conference on Natural Phenomena* (Aire-la-Ville, Switzerland, Switzerland, 2007), NPH'07, Eurographics Association, pp. 63–70.

[160] Rusinkiewicz, S., and Levoy, M. QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH '00* (2000), pp. 343–352.

[161] Russell, G., Marshall, B., and Jarvis, P. G. *Plant canopies: their growth, form and function.* Society for Experimental Biology, 1990.

[162] Sachs, T., and Novoplansky, A. Tree form: Architectural models do not suffice. *Israel Journal of Plant Sciences 43* (1995), 203–212.

[163] Saito, T., and Takahashi, T. Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 197–206.

[164] Sakaguchi, T., and Ohya, J. Modeling and animation of botanical trees for interactive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1999), VRST '99, ACM, pp. 139–146.

[165] Sellier, D., Plank, M. J., and Harrington, J. J. A mathematical framework for modelling cambial surface evolution using a level set method. *Annals of Botany 108*, 6 (2011), 1001–1011.

[166] Sen, S. I., and Day, A. M. Tutorials/surveys: Modelling trees and their interaction with the environment: A survey. *Computer Graphics 29*, 5 (Oct. 2005), 805–817.

[167] Shah, M. A., Kontinnen, J., and Pattanaik, S. Real-time rendering of realistic-looking grass. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2005), GRAPHITE '05, ACM, pp. 77–82.

[168] Shin, O., Machiko, T., Tadahiro, F., Kazunobu, M., and Norishige, C. A hybrid method for real-time animation of trees swaying in wind fields. *The Visual Computer 20*, 10 (Dec. 2004), 613–623.

[169] Shinozaki, K., Yoda, K., Hozumi, K., and Kira, T. A quantitative analysis of plant form - the pipe model theory, parts I and II. Basic analysis and further evidence of the theory and its application in forest ecology. *Japanese Journal of Ecology 14* (1964), 97–104,133–139.

[170] Shlyakhter, I., Rozenoer, M., Dorsey, J., and Teller, S. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics Applications 21*, 3 (May 2001), 53–61.

[171] Smirnoff, N. The role of active oxygen in the response of plants to water deficit and desiccation. *New Phytologist 125*, 1 (1993), 27–58.

[172] Smith, A. R. Plants, fractals, and formal languages. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 1–10.

[173] Soler, C., Sillion, F. X., Blaise, F., and Dereffye, P. An efficient instantiation algorithm for simulating radiant energy transfer in plant models. *ACM Transactions on Graphics 22*, 2 (2003), 204–233.

[174] Stamminger, M., and Drettakis, G. Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques 2001 (Proceedings of Eurographics Workshop on Rendering* (2001), pp. 151–162.

[175] Stamminger, M., and Drettakis, G. Perspective shadow maps. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 557–562.

[176] Stava, O., Benes, B., Mech, R., Aliaga, D. G., and Kristof, P. Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum 29*, 2 (2010), 665–674.

[177] Sterck, F. J., Schieving, F., Lemmens, A., and Pons, T. L. Performance of trees in forest canopies: explorations with a bottom-up functional-structural plant growth model. *New Phytologist 166*, 3 (2005), 827–843.

[178] Tai, K.-C. The tree-to-tree correction problem. *J. ACM 26*, 3 (July 1979), 422–433.

[179] Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. Metropolis procedural modeling. *ACM Transactions on Graphics 30* (2011), 11:1–11:14.

[180] Tan, P., Fang, T., Xiao, J., Zhao, P., and Quan, L. Single image tree modeling. In *ACM SIGGRAPH Asia 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 108:1–108:7.

[181] Tan, P., Zeng, G., Wang, J., Kang, S. B., and Quan, L. Image-based tree modeling. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.

[182] Tanaka, E., and Tanaka, K. The tree-to-tree editing problem. *International Journal of Pattern Recognition and Artificial Intelligence 02*, 02 (1988), 221–240.

[183] Torralba, A., Murphy, K. P., and Freeman, W. T. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell. 29* (2007), 854–869.

[184] Ulam, S. Patterns of Growth of Figures: Mathematical Aspects. In *Module, Proportion, Symmetry, Rhythm*, G. Kepes, Ed. Braziller, 1966, pp. 64–74.

[185] Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Waddell, P. Inverse design of urban procedural models. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 168:1–168:11.

[186] Verroust, A., and Lazarus, F. Extracting skeletal curves from 3d scattered data. In *Proceedings of the International Conference on Shape Modeling and Applications* (Washington, DC, USA, 1999), SMI '99, IEEE Computer Society, pp. 194–ff.

[187] Vos, J., Evers, J. B., Buck-Sorlin, G. H., Andrieu, B., Chelle, M., and de Visser, P. H. B. Functional-structural plant modelling: a new versatile tool in crop science. *Journal of Experimental Botany 61*, 8 (2010), 2101–2115.

[188] Vos, J., Marcelis, L. M., de Visser, P. B., Struik, P., and Evers, J. B. *Functional-Structural Plant Modelling in Crop Production*. SpringerVerlag, 2007.

[189] Wand, M., Fischer, M., Peter, I., Meyer auf der Heide, F., and Strasser, W. The randomized z-buffer algorithm: interactive rendering of highly complex scenes. In *SIGGRAPH '01* (2001), pp. 361–370.

[190] Wand, M., and Strasser, W. Multi-resolution rendering of complex animated scenes. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3 (2002).

[191] Wang, L., Wang, W., Dorsey, J., Yang, X., Guo, B., and Shum, H.-Y. Real-time rendering of plant leaves. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[192] Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., and Shum, H.-Y. Generalized displacement maps. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2004), EGSR'04, Eurographics Association, pp. 227–233.

[193] Wang, X., Wang, L., Liu, L., Hu, S., and Guo, B. Interactive modeling of tree bark. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), PG '03, IEEE Computer Society, pp. 83–.

[194] Waring, R. H., Schroeder, P. E., and Oren, R. Application of the pipe model theory to predict canopy leaf area. *Canadian Journal of Forest Research 12(3)* (1982), 556–560.

[195] Weber, J., and Penn, J. Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH '95* (1995), pp. 119–128.

[196] Williams, L. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 270–274.

[197] Wither, J., Boudon, F., Cani, M.-P., and Godin, C. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum 28*, 2 (2009), 541–550.

[198] Xu, H., Gossett, N., and Chen, B. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics 26*, 4 (Oct. 2007).

[199] Yan, H.-P., Kang, M. Z., De Reffye, P., and Dingkuhn, M. A dynamic, architectural plant model simulating resource-dependent growth. *Annals of Botany 93*, 5 (2004), 591–602.

[200] Zakaria, M. N., and Shukri, S. R. A sketch-and-spray interface for modeling trees. In *Proceedings of the 8th international symposium on Smart Graphics* (Berlin, Heidelberg, 2007), SG '07, Springer-Verlag, pp. 23–35.

[201] Zhang, K. A constrained edit distance between unordered labeled trees. *Algorithmica 15*, 3 (1996), 205–222.

[202] Zhang, Q.-L., and Pang, M.-Y. A survey of modeling and rendering trees. In *Proceedings of the 3rd international conference on Technologies for E-Learning and Digital Entertainment* (Berlin, Heidelberg, 2008), Edutainment '08, Springer-Verlag, pp. 757–764.

[203] Zhu, C., Zhang, X., Hu, B., and Jaeger, M. Reconstruction of tree crown shape from scanned data. In *Proceedings of the 3rd international conference on Technologies for E-Learning and Digital Entertainment* (Berlin, Heidelberg, 2008), Edutainment '08, Springer-Verlag, pp. 745–756.