BORIS NEUBERT

COMPUTER GRAPHICS AND NATURE

SIMULATION-BASED AND PROBABILISTIC METHODS FOR IMAGE- AND SKETCH-BASED MODELING AND ADVANCED STOCHASTIC PRUNING

DISSERTATION

zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
an der Universität Konstanz, Mathematisch- Naturwissenschaftliche Sektion,
Fachbereich Informatik und Informationswissenschaft,
vorgelegt von Boris Neubert.

Konstanz, 2011
UNIVERSITY OF KONSTANZ

Reviewers/ Referenten:

1. Referent: Prof. Dr. Oliver Deussen, University of Konstanz, Germany

2. Referent: Prof. Dr. Daniel Keim, University of Konstanz, Germany

Date of Defense/ Tag der mündlichen Prüfung:

7.5.2012

für Daniela

Abstract

This thesis presents new methods for modeling and efficient rendering of botanical scenes and objects. The first method allows for producing 3D tree models from a set of images with limited user intervention by combining principles of image- and simulation-based modeling techniques. The image information is used to estimate an approximate voxel-based tree volume. Density values of the voxels are used to produce initial positions for a set of particles. Performing a 3D flow simulation, the particles are traced downwards to the tree basis and are combined to form twigs and branches. If possible, the trunk and the first-order branches are determined in the input photographs and are used as attractors during the particle simulation. Different initial particle positions result in a variety, yet similar-looking branching structures for a single set of photographs. The guided particle simulation meets two important criteria improving common modeling techniques: it is possible to achieve a high visual similarity to photographs and at the same time allows for simple manipulations of the resulting plant by altering the input photographs and changing the shape or density, providing the artist with an expressive tool while leveraging the need for manual modeling plant details. Following paper based on guided particle simulations coined the term *self-organizing* tree models.

The second method improves the concept of sketch-based modeling tools for plants. The proposed system converts a freehand sketch of a tree drawn by the user into a full 3D model that is both, complex and realistic-looking. This is achieved by probabilistic optimization based on parameters obtained from a database of tree models. Branch interaction is modeled by a Markov random field, which allows for inferring missing information of the tree structure and combining sketch-based and data-driven methodologies. The principle of self-similarity is exploited to add new branches before populating all branches with leaves.

Both modeling methods presented in this work, produce very complex tree models. While this richness is needed to model highly realistic scenes, it leads to a complexity that makes real-time rendering impossible. We present an optimized pruning algorithm that considerably reduces the geometry needed for large botanical scenes, while maintaining high and coherent rendering quality. We improve upon previous techniques by applying model-specific geometry reduction functions and optimized scaling functions. We propose the use of Precision and Recall (PR) as a measure of quality to rendering and show how PR-scores can be used to predict better scaling values. To verify the measure of quality we conducted a user-study allowing subjects to adjust the scaling value, which shows that the predicted scaling matches the preferred ones. Finally, we extend the originally purely stochastic geometry prioritization for pruning in order to account for a view-optimized geometry selection, which allows to take global scene information, such as occlusion, into consideration. We demonstrate our method for the rendering of scenes with thousands of complex tree models in real-time.

Zusammenfassung

Diese Arbeit beschreibt neue Methoden zur Modellierung und Bilderzeugung von komplexen Landschaften und botanischen Modellen, die immer häufiger im Kontext von Landschaftsvisualisierung und in Filmen eingesetzt werden. Mit steigender Rechen- und Speicherkapazität wurden Effizienzbetrachtungen bei der Modellierung zunehmend unwichtiger und Benutzbarkeit und Expressivität rücken in den Fokus aktueller Forschung.

Die beschriebene Methode zur bildbasierten Modellierung beschreibt den Prozess 3D Modelle mit Hilfe von Fotografien zu erzeugen. Das vorgestellte Verfahren verbindet dieses Prinzip mit Methoden simulationsbasierter Modellierung. Basierend auf einer geringen Anzahl Fotografien werden 3D Modelle von Pflanzen erstellt. Hierfür wird zunächst approximativ das Volumen in Kombination mit Dichteinformationen bestimmt, die das Modell beschreiben. Innerhalb des Volumens werden proportional zur ermittelten Dichte Startpositionen für Partikel erzeugt. Die Trajektorien der Partikel während einer Partikelsimulation erzeugen im späteren Model die Zweige und schliesslich den Stamm. Die Bewegungsrichtung der Partikel während der Simulation wird vom Verlauf der Hauptäste des fotografierten Baumes beeinflusst und ermöglicht so ein ähnliches Abbild der fotografierten Pflanze.

Eine weitere Modellierungsmethode kombiniert Prinzipen der datengetriebenen und skizzenbasierten Modellierung. Die fehlenden Informationen einer 2D Pflanzenskizze, wie beispielsweise 3D Positionen der Äste, werden ermittelt, indem Informationen verwendet werden, die aus einer Datenbank mit Modellen gewonnen werden. Die Informationen der Modelle in der Datenbank werden dabei mit Hilfe von Markov Random Fields ausgewertet. Faktor Graphen ermöglichen es, die wahrscheinlichste Konfiguration der fehlenden Parameter basierend auf den Daten eines Models der Datenbank zu ermitteln und so ein vollständiges 3D Modell zu erzeugen.

Die vorgestellten Methoden ermöglichen es detaillierte und komplexe Modelle zu erzeugen. Echtzeitdarstellung erfordert aber die Komplexität auf ein Maß zu beschränken, das eine interaktive Darstellung ermöglicht. Wir stellen ein Verfahren vor, dass modell- und szenenspezifisch die Komplexität der darzustellenden Modelle verringert. Um eine hohe visuelle Qualität zu garantieren bedienen wir uns einem aus dem Informationretrieval bekannten Maß: Precision und Recall. Wir zeigen wie ein auf PR basierendes Qualitätsmaß verwendet werden kann, um optimale Parameter für dieses *Level of Detail* Verfahren zu bestimmen. Ein Vergleich mit einer durchgeführten Benutzerstudie zeigt die Plausibilität der ermittelten Werte. Eine mögliche Anwendung des vorgestellten Qualitätsmaß ist die nichtstochastische, priorisierte Auswahl von Geometrie, die eine modellabhängige automatische Auswertung ermöglicht. Wir zeigen, dass die Modelle eine signifikante Reduzierung der Komplexität und damit eine Echtzeitdarstellung ermöglichen.

Acknowledgements

This work was possible with the kind help and support of various friends, colleges, and collaborators.

First of all I thank my advisor, Professor Dr. Oliver Deussen for the opportunity to work with his group. He not only influenced my development as a researcher, but supported me as an advisor during Bachelor and Master theses and arouse my interest in scientific research and the inspiring field of computer graphics. I also thank my second advisor Professor Dr. Daniel Keim for his comments and reviewing the thesis and for keeping interested in other topics and influencing the way I approach research topics and to Professor Dr. Carsten Dachsbacher for his support and valuable discussions.

Additional thanks to my advisor for giving me the opportunity to work on various interesting topics and especially supporting various research visits at other labs, learning different views and approaches to answer research questions. Mike Sips and Pat Hanhrahan from Stanford University allowed me to work in the field of information visualization and made it possible to work with various interesting researchers. Thanks to Michael F. Cohen for the enriching and inspiring time at Microsoft Research together with Bill Chen and Eyal Ofek and the great opportunity to meet Pravin Bhat, who I later had the pleasure to meet again at Weta Digital Research together with J.P. Lewis, Eugene d'Eon and many more. Thanks for this inspiring and pleasant time.

Many thanks to my colleges at the computer graphics lab in Konstanz, Thomas Franken, Till Niese, Joachim Böttger, Sören Pirk, Michael Balzer, Daniel Heck, Thomas Schlömer, and Johannes Kopf for the nice time and interesting discussions.

I am deeply thankful for the support and patience of my mother and would also like to thank my family for the support they provided me through my entire life and in particular my wife, Daniela, without her love and encouragement, I would not have finished this thesis.

In conclusion, I recognize that this research would not have been possible without the financial assistance of the "Information Technology Baden-Württemberg (BW-FIT)" program and the "Information at your fingertips – Interactive Visualization for Gigapixel Displays" project.

Contents

Introduction

1

Introduction

1.1 Scope of the Thesis and Motivation

Modeling and rendering botanical scenes has always been an important topic in computer graphics research. Whereby both, photo-realistic display of man-made objects and natural phenomena, such as smoke, clouds, skin, hair, and plants have been a core objective in computer graphics research.

However, in comparison to man made objects that are already designed with the help of computers, some properties of natural objects, especially plants and large botanical scenes, are difficult to capture. This equally applies to rendering of such scenes as well as modeling.

Nature is incredible complex on different scales. While complex models of man made objects can often be presented in a computer readable form with a few thousand polygons, natural objects like trees, demand a much higher geometric complexity.

The early solution to handle this complexity was to use mathematical descriptions that are capable of producing detailed models based on very efficient mathematical description. However, working with these descriptions not only requires a solid mathematical understanding, but also a thorough knowledge in botanics, which makes modeling a cumbersome and complicated task.

Recent developments in model synthesis foster the change from experts to artists with the increasing demand of realistic models for computer games and computer graphics application to movies. The increasing capacities over the last years, both in computational power and memory space, relaxed the constrains on high efficient descriptions of natural models and instead raised the need for modeling methods that are easy to use and that support the creativity of artists. Modeling tools suitable for modeling natural objects

had a stronger focus on expressiveness and ease of user versus efficiency considerations.

Another important aspect in the paradigm shift was introduced with the development of digital capturing devices such as digital cameras and laser scanners. Image based modeling techniques were introduced with the intuitive idea of using real world examples and capture devices to obtain digital representations of the models rather than modeling certain objects. However, while this method is very efficient for large scale objects such as architecture, for image based capturing of plants, it is still difficult to obtain digital representations of models fully automatic. This is due to non-unique features that make registration of images very hard and due to the high degree of occlusion for some parts of natural objects that raise problems for algorithms based on digital images as well as methods based on laser scanner.

While complexity considerations with respect to model synthesis were approached by helping the user to generate complex models without the tedium of manual modeling every repeated detail, complexity still affects image synthesis. Usually even small scenes exceed the capabilities of modern graphics hardware. These problems are usually addressed by simplifications algorithms reducing the geometric complexity of models according to distance to the viewer. However, most of the common level-of-detail approaches render impractical for natural scenes, while other methods provide no mechanism of measuring the quality of the simplified model.

Developing an image-space quality measure is an important first step to answer the basic question: How to measure the quality of an image?

1.2 Contributions and Publications

The main contributions in the field of rendering and modeling of nature summarized in this thesis are

- An algorithm combining principles of image-based and simulation-based modeling. The idea of using particle simulations to model trees was recently extended in an interesting work by Runions *et al.* [Runions et al., 2007] and later Palubicki *et al.* [Palubicki et al., 2009]. A variant of the proposed algorithm has also recently been used for modeling vegetation in Disney's feature film "Tangled".
- Combining sketch-based and data driven modeling. Successfully application of probabilistic graphics models to capture the properties of a data base of tree models and use this data to deduce missing information to generate full 3D models from 2D sketches. Data-driven modeling describes the idea to use available data from existing models to interpret the

sketches and deduce missing information in a plausible manner following an expectation maximization approach.

- Proposing the use of Precision and Recall, well known measures from Information Retrieval, to predict the visual quality of simplified models as used in level-of-detail algorithms and answers the question of how to adapt parameterized, continuous LOD algorithms to different models and scene configurations. Precision and Recall are used to establish an image-space quality measure that faithfully predicts optimal, model dependent LOD parameters. While being especially suited for the rendering of complex aggregate detail, it raises an interesting question of general interest to the computer graphics community: How to measure the quality of an image?

The result presented in this thesis and additional research results have been published in the following publications in accordance with the practice for a cumulative thesis:

Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:88:1 – 88:10, July 2007

Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based tree modeling using markov random field. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '08)*, 27:109:1–109:9, December 2008

Johannes Kopf, Boris Neubert, Billy Chen, Michael F. Cohen, Daniel Cohen-Or, Oliver Deussen, Matt Uyttendaele, and Dani Lischinski. Deep photo: Model-based photograph enhancement and viewing. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '08)*, 27:116:1–116:10, December 2008

Billy Chen, Boris Neubert, Eyal Ofek, Oliver Deussen, and Michael F. Cohen. Integrated videos and maps for driving directions. In *Proc. of the ACM symposium on User interface software and technology*, UIST '09, pages 223–232. ACM Press, 2009

Mike Sips, Boris Neubert, John P. Lewis, and Pat Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum (Proc. of EuroVis '09)*, 28(3):831–838, 2009

Boris Neubert, Soeren Pirk, Oliver Deussen, and Carsten Dachsbacher. Improved model- and view-dependent pruning of large botanical scenes. *Computer Graphics Forum*, 30(6):1708–1718, 2011

Sören Pirk, Ondrej Stava, Julian Kratt, Michel Abdul Massih Said, Boris Neubert, Radomír Měch, Bedrich Benes, and Oliver Deussen. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics (Proc. of SIGGRAPH '12)*, 31(4):50:1–50:10, July 2012b

Sören Pirk, Till Niese, Oliver Deussen, and Boris Neubert. Capturing and animating the morphogenesis of polygonal tree models. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '12)*, 2012a

1.3 Outline

The remainder of this thesis is organized in two parts reflecting the two major areas of interest regarding the synthetic imagery of natural scenes: modeling (Part I: *Modeling of Natural Objects*) and rendering (Part II: *Rendering of Complex Scenes*).

The first chapter provides an introduction to the field of modeling and discusses related work. Chapter 3 *Modeling through Simulation* introduces the idea of simulation-based modeling in combination with image-based methods. Chapter 4 *Data-driven Plant Modeling* deals with a method to increase the modeling ease of trees following the sketch-based paradigm and discusses the use of probabilistic graphical models to generate 3D models from 2D sketches. Data-driven modeling describes the idea to use available data from existing models to interpret the sketches and deduce missing information in a plausible manner following an expectation maximization approach. The final step generating 3D geometry from the intermediate graph representation of the plant skeleton is described in Chapter 5: *From Graphs to Models*.

The first chapter of the second part discusses related work in the field of level-of-detail rendering algorithms relevant to natural scenes and objects. Chapter 7: *Realtime Rendering – Improved Model- and View-Dependent Pruning of Large Botanical Scenes* answers the question of how to adapt parameterized, continuous LOD algorithms to different model and scene configurations. Precision and Recall, well known measures from Information Retrieval, are used to establish an image-space quality measure that faithfully predicts optimal, model dependent LOD parameters.

Chapter 8 presents a summary of the results and introduces possibilities for future research.

Part I

Modeling of Natural Objects

2

Introduction to Modeling Nature and Related Work

The development of computer graphics research as an independent discipline within computer science found its starting point with the groundbreaking work of Ivan Sutherland [Sutherland, 1963] [1] a student at the Massachusets Institute of Technology (MIT) in the early 1960s, who then founded, together with Dave Evans, the first computer graphics research lab at the University of Utah in 1968. Not surprisingly, natural phenomena and objects came into the focus of cg research with the emerging goal to render photorealistic models soon after the foundations of computer graphics were laid. Benoit Mandelbrot's publication "The fractal geometry of Nature" [Mandelbrot, 1983] [2] was amongst the most influential publications on this topic at that time.

[1] Ivan E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, Lincoln Lab, 1963

[2] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983

Without the nowadays common possibilities and software that allow for detailed manual modeling, and with the need to have extremely memory efficient model descriptions, describing nature in mathematical terms was a very important step to be able to efficiently use the data and render pictures within a computer graphics framework. In 1968 Aristid Lindenmayer [Lindenmayer, 1968] [3] formulated a method to model natural objects based on mathematical rules. Naturally, these were amongst the first methods used to provide detailed geometry of natural objects.

[3] Aristid Lindenmayer. Mathematical models for cellular interaction in development,parts I and II. *Journal of Theoretical Biology*, 18: 280–315, 1968

With the fundamental step to the Reyes (*Renders Everything You Ever Saw*) rendering architecture [Cook et al., 1987] [4] and the introduction of triangles (in contrast to mathematical descriptions of surfaces and objects) as rendering primitives more advanced modeling methods became interesting.

[4] Robert L. Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. *Computer Graphics (Proc. of SIGGRAPH '87)*, 21:95–102, August 1987

This chapter will briefly introduce some of the fundamental works within natural modeling and retrace the development from purely mathematical descriptions, to advanced user centered approaches, and finally to image-based methods that focus on the capturing of real world plants.

2.1 Mathematical Descriptions

Even before computer graphics became an independent research discipline mathematicians developed methods to formally describe various natural phenomena. Techniques for mathematical descriptions of plants were introduced as early as 1966 by Ulam [Ulam, 1966] [5] . They were soon followed by the introduction of *L-systems* as a formalism for simulating the development of multicellular organs in terms of division, growth, and death of individual cells [Lindenmayer, 1968] [6] . Naturally, these descriptions were the first techniques adapted by the computer graphics community to generate natural objects and were later refined to meet the increasing demand for highly realistic models.

[5] Stanislaw M. Ulam. Pattern of growth of figures: mathematical aspects. In G. Keps, editor, *Module, Proportion, Symmetry, Rhythm*, pages 64–74. Braziller, New York, 1966

[6] Aristid Lindenmayer. Mathematical models for cellular interaction in development,parts I and II. *Journal of Theoretical Biology*, 18: 280–315, 1968

> The central concept of L-systems is that of rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of rewriting rules or production.
> (from [Prusinkiewicz and Lindenmayer, 1996])

L-Systems

Lindenmayer systems, or *L-Systems*, [Lindenmayer, 1968] allow the description of a complex branching pattern based on simple rewriting rules, formalized as grammars very similar to *semi-Thue grammars*, that are applied to an initial state [Prusinkiewicz and Lindenmayer, 1996] [7] , defined as the tuple

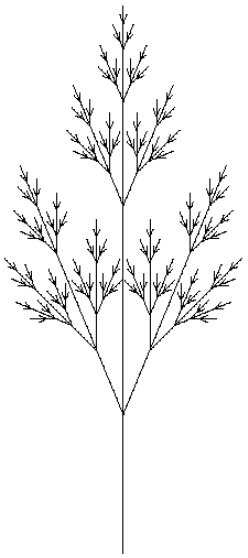
[7] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996

$$G = (V, \omega, P) \tag{2.1}$$

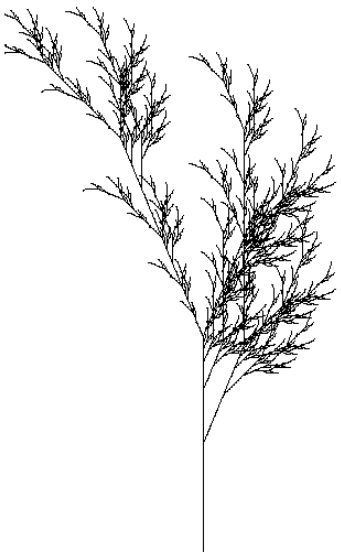
where V is the alphabet (a set of symbols), ω is the initiator consisting of a string of symbols from V (initial state) and P is a set of production rules defining how symbols can be replaced with other strings containing symbols from V (see Fig. 2.1 for some examples of P). Prusinkiewicz *et al.* [Prusinkiewicz et al., 1996] [8] later derived a systematic description for

[8] Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomír Měch. L-systems: from the theory to visual models of plants. In *Proc. of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, volume 3, pages 1–12. CSIRO Publishing, 1996

(a) $n = 5$, $\delta = 25.7$,
$P = \{F \rightarrow F(+F)F(-F)F\}$



(b) $n = 5$, $\delta = 25.7$,
$P = \{X \rightarrow F(+X)(-X)FX, F \rightarrow FF\}$



(c) $n = 7$, $\delta = 22.5$,
$P = \{X \rightarrow F((X)-X)F(+FX), F \rightarrow FF\}$

Figure 2.1: Lindenmayer System with symbol set $V = \{F, X, +, -, (,)\}$

using L-systems to model plants.

In a classical L-System, the rule basis has to be written by the user. Since rules work locally, small changes of values can cause large changes in the overall shape, that are hard to predict. Such behavior makes modeling quite cumbersome. Figure 2.1 shows an example of a grammar-based branching system and the underlying grammars. Various extensions of L-Systems have been proposed since then, e.g. parametric [Prusinkiewicz and Lindenmayer, 1996], open [Měch and Prusinkiewicz, 1996] [9] and differential L-Systems [Prusinkiewicz et al., 1993] [10] . These extensions are able to create a variety of effects, but also result in additional parameters, that need to be defined by the user. Prusinkiewicz *et al.* [Prusinkiewicz et al., 2001] present a modeling interface for L-Systems to enhance the modeling ease, but a large set of parameters still has to be defined by the user and the induced changes are difficult to predict.

[9] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proc. of SIGGRAPH '96*, pages 397–410, New York, NY, USA, 1996. ACM

[10] Przemyslaw Prusinkiewicz, Mark Hammel, and E. Mjolsness. Animation of plant development. In *Proc. of SIGGRAPH '93*, pages 351–360, New York, NY, USA, 1993. ACM

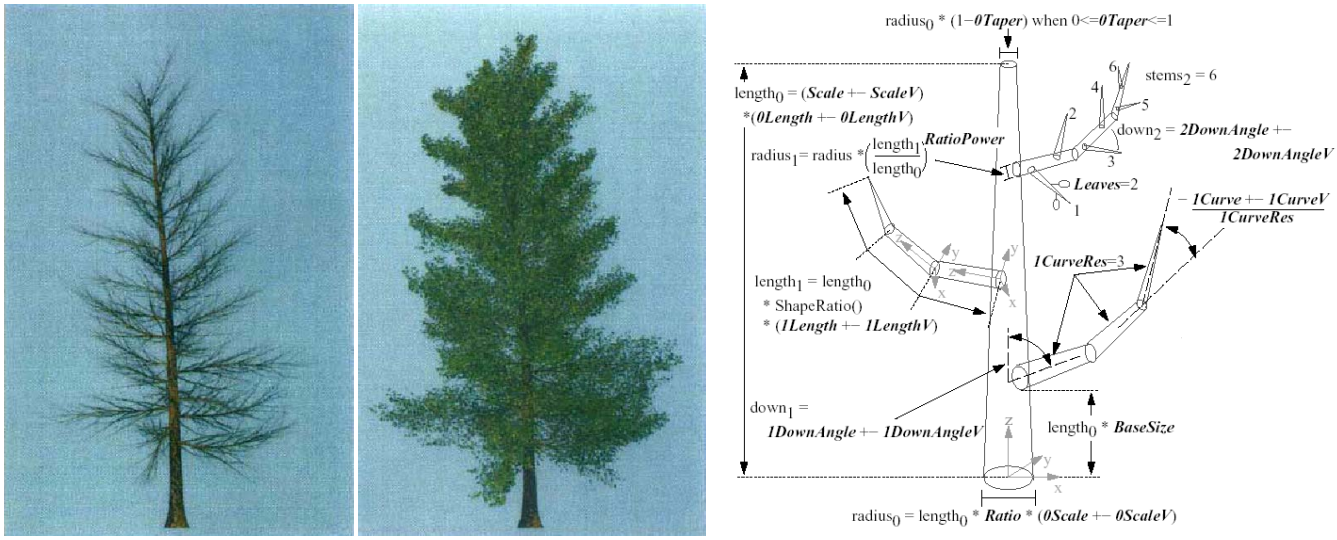


Figure 2.2: Procedural plant modeling [from Weber and Penn, 1995]

2.2 *Procedural Approaches*

Procedural approaches are usually restricted to produce a limited number of forms. They are also able to limit the number of adjustable parameters for the user. However, with increasing model complexity, the number of parameters increases, too. While Oppenheimer [Oppenheimer, 1986] [11] uses only some basic parameters, following approaches such as the one presented by Weber and Penn [Weber and Penn, 1995] have dozens (see Fig. 2.2 for a schematic description of the parameters used).

In the xfrog system [Lintermann and Deussen, 1999] [12] procedural elements are combined using a simple rule system, which allows faster modeling. However, the number of parameters is still large. Ijiri *et al.* [Ijiri et al., 2005] [13] use interactive editors based on botanical rules to create plant models. While these editors allow for an efficient production of flowers and phyllotaxis, the production of complex trees is not their strength. Okabe *et al.* [Okabe et al., 2005] [14] present a sketch-based interface for trees. Here, the user draws the outline of a tree skeleton and its shape. However, again many parameters have to be adjusted to achieve specific species.

[11] Peter E. Oppenheimer. Real time design and animation of fractal plants and trees. In *Computer Graphics (Proc. of SIGGRAPH '86)*, volume 20, pages 55–64, New York, NY, USA, August 1986. ACM

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

[13] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics (Proc. of SIGGRAPH '05)*, 24(3):720–726, July 2005

[14] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3): 487–496, 2005

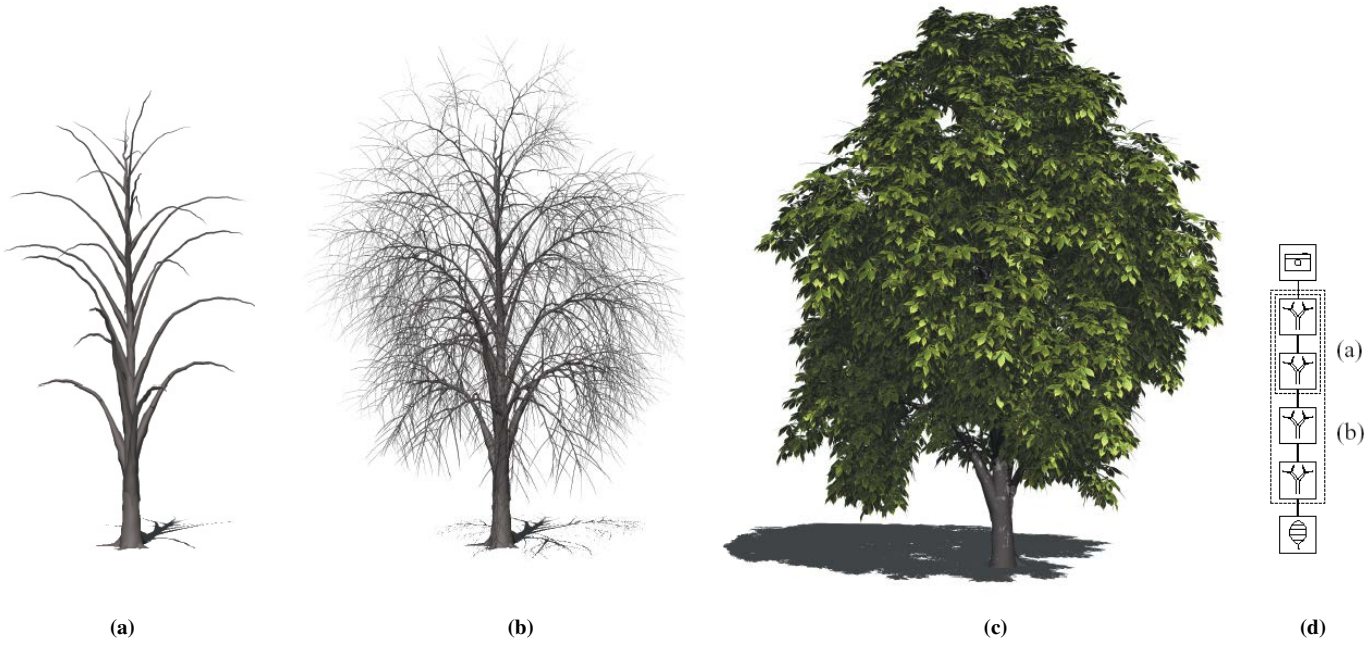


Figure 2.3: Combination of procedural and rule-based plant modeling [from Deussen and Lintermann, 2005].

Rule-based systems are difficult for the novice user to operate because they require not only specialized knowledge on biomechanics and biology for effective parameter specification. The user must also understand how the rules are applied or even formulated consistently. In a number of such systems, the global shape of trees is difficult to control—slight changes in local rules may result in significant changes in the global shape.

The xfrog system [Lintermann and Deussen, 1999] [15] and subsequent graphical L-System editors [Prusinkiewicz et al., 2001] [16] allow the user to manipulate complex parameters graphically. Despite the increased ease of use, such systems still require the user to specify the less intuitive function plots, curves, and surface parameters that govern appearance.

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

[16] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *Proc. of SIGGRAPH '01*, pages 289–300, New York, NY, USA, 2001. ACM Press

2.3 Image-based Modeling

Rather than requiring the user to manually specify the plant model, there are approaches that instead use images as a basis to generate 3D models. For example, Shlyakter *et al.* [Shlyakhter et al., 2001] [17] use an L-system-based growth mechanism that is constrained by the visual hull produced

[17] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21:53–61, May 2001

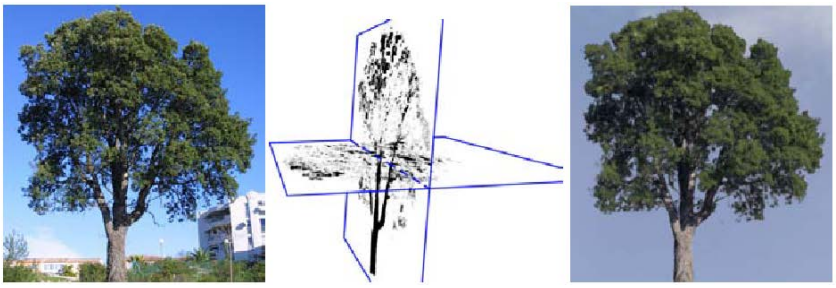


Figure 2.4: Image-based volumetric rendering. Left: one of the input images. Center: two slices of the reconstructed density volume. Right: resulting model. [from Reche-Martinez et al., 2004]

from photographs. A visual hull is reconstructed from the registered input images. The medial axis diagram of the hull is constructed and used as the tree skeleton. The smaller branches and leaves are added using an L-System.

A very precise but complex image-based modeling approach is described by Reche-Martinez *et al.* [Reche-Martinez et al., 2004] [18] . In this case a set of carefully registered photographs is used to determine the volumetric shape of a given tree. The data is stored as a huge set of volume tiles and therefore requires a significant amount of memory. Furthermore, its generic volumetric representation makes it hard to edit. The volume is divided into cells; for each cell a valid visual representation is computed by a set of textures. Figure 2.4 shows one of input images together with the volumetric model rendered from the same viewing direction. The complete set of textures represents the tree quite faithfully. However, a large amount of texture space on the order of tens of megabytes is needed. Also, it is not easy to show the tree under various lighting conditions since the lighting is already incorporated into the textures. This is avoided in the approaches described in the following chapters (Chap. 3 and 4) by creating a 3D surface model using images or a 2D sketch. Also, in the case of the image-based approach described in Chapter 3 we do not need an exact registration of the input photographs and instead create an approximate shape of the given tree.

[18] Alex Reche-Martinez, Ignacio Martin, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (Proc. of SIGGRAPH '04)*, 23(3):720–727, August 2004

Another image-based method for modeling smaller plants was presented by Quan *et al.* [Quan et al., 2006] [19] . Here, an image sequence is recorded and the model is computed from these images in a semi-automatic way. Although the results are of high quality, the amount of required user input prevents modeling complex objects where such a method would be particularly interesting. This is mainly due to the fact that complex plants consist of many at least partially occluded areas and the required clustering needs to be manually adjusted.

[19] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH '06)*, 25:599–604, July 2006

Tan *et al.* [Tan et al., 2007] [20] proposed another method to reconstruct the 3D model from multiple images. The user can also manually edit the branch structures. However, the system requires a significant amount of preprocessing work such as segmentation of branches, leaves, and background. User intervention may be required to correct errors in the 3D branch extraction or seed branch growth. However, mismatches between the construction of hidden branches and observed leaves tend to produce floating leaves.

[20] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:87:1 – 87:8, July 2007

Image-based approaches have the best potential for producing realistic-looking plants, since they rely on images of real plants. At the same time, they can produce only models of preexisting plants. Designing new plants would be a major issue without manual editing. Both the approaches in the following chapters provide a good compromise that requires only a small amount of intuitive input to produce new and realistic-looking trees.

2.4 Sketch-based Approaches

Sketch-based systems were developed to provide a more intuitive way of generating plant models. For example, Okabe *et al.*'s system [Okabe et al., 2005] [21] reconstructs the 3D branching pattern from 2D drawn sketches in different views by maximizing distances between branches. They use additional gesture-based editing functions to add, delete, or cut branches. Moreover, example-based editing is supported to generate branches or leaves using some existing tree models.

[21] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3): 487–496, 2005

Their system, however, requires the user to draw many branches to describe detailed structures. Because their system does not support automatic propagation of branches, a complex tree would require extensive user interaction. In comparison, our system allows the user to generate 3D models by merely drawing a few strokes in single view and optionally defining the overall shape of the tree by loosely sketching the contour of the crown.

Ijiri *et al.*'s system [Ijiri et al., 2006] [22] is based on L-systems. The user draws a single free-form stroke to control the growth of a tree. The change in the shape of the stroke is used as a graphical metaphor for modifying the L-system parameters. However, this system supports only two simple production rules, and the user is not allowed to control the overall shape of the tree. This severely limits the expressive power of the system.

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

2.5 Simulation-based Modeling

The following chapter presents a method for producing 3D tree models from input photographs with only limited user intervention, as a combination of sketch- and image-based modeling. An approximate voxel-based tree volume is estimated using image information from loosely registered input photographs. The density values of the voxels are used to produce initial positions for a set of particles. Performing a 3D flow simulation, the particles are traced downwards to the tree basis and are combined to form twigs and branches. If possible, the trunk and the first-order branches are determined in the input photographs and are used as attractors for a particle simulation. The geometry of the tree skeleton is produced using botanical rules for branch thicknesses and branching angles. Finally, leaves are added. Different initial particle positions lead to diverse yet similar-looking branching structures for a single set of photographs.

The idea of using particle simulations to model trees was recently extended in an interesting work by Runions *et al.* [Runions et al., 2007] [23] and later Palubicki *et al.* [Palubicki et al., 2009] [24] .

[23] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Proc. of the Eurographics Workshop on Natural Phenomena*, pages 63–70. Eurographics Association, 2007

[24] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch, and Przemyslaw Prusinkiewicz. Self-organizing tree models for image synthesis. *ACM Transactions on Graphics (Proc. of SIGGRAPH '09)*, 28: 58:1–58:10, July 2009

3

Modeling through Simulation – Approximate Image-Based Tree-Modeling using Particle Flows

Simulation-based design [Ebert et al., 2003] [1] has a long tradition within computer graphics research. It is used for simulation of all kind of natural effects such as water simulation, smoke, or clouds. The obvious benefit from simulation-based design is that instead of dealing with tedious definitions of rules for one instance or manual modeling of a single model, the inherent properties of the models are encoded within the simulation rules. With a hypothetical, perfect simulation system all observed phenomena would therefore be emergent properties of the simulated results. However, simulations always provide only a simplified model of the underlying system either to reduce the simulation complexity or because of a lack of understanding of all underlying principles.

This chapter proposes the use of particle simulations for the modeling of branching structures of trees.

[1] David S. Ebert, Oliver Deussen, Ronald Fedkiw, F. Kenton Musgrave, Przemyslaw Prusinkiewicz, and Jos Stam. Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH '03: Course Notes 41, 2003

Figure 3.1: A tree is modeled using a set of input photographs. We show some examples of input and resulting 3D tree models. If image information is not available, e.g. the foliage is missing, the user is able to sketch it (right). The models approximate the input images while forming botanically plausible branching structures.

Modeling complex botanical tree geometry has posed a challenge for computer graphics for decades. Beginning with abstract branching structures, the complexity and visual appearance has been enhanced over the years in such a way that today many tree models appear photo-realistic to us. However, creating these types of models is still cumbersome. To mimic a specific tree or a given tree shape, many parameters have to be manually adjusted. Image-based modeling methods try to overcome this problem by using a set of photographs to create the geometry directly.

In recent years some techniques have been published that try to create exact 3D representations for given trees. In contrast to these methods the proposed approach produces qualified approximations, thus avoiding exact registration and many numerical problems while still achieving plausible branching structures. The models still show differences to the input; however, it is possible to create a variety of similar models by changing initial parameters of the system.

The input is a small set of photographs of a tree taken from different views. Usually two images are sufficient for a good approximation. In contrast to other approaches, we do not need an exact registration of these images and interactively arrange them in the system. Since the images typically contain unwanted background, separating the trees using common algorithms for alpha matting is necessary.

The general idea of this approach is to combine a bottom-up construction with internal and external constraints. We compute a voxel-model of the tree volume with each voxel containing a density estimate of the tree's biomass. Proportional to this density, particle positions are initialized and traced downwards to the tree basis using a 3D flow simulation. Simple rules direct and force them to form twigs and subsequent branches. This creates plausible tree skeletons and, by later adding leaves, complete trees.

The particle simulation is directed by the main branching structures found in the input images to achieve a high similarity of the models to the given input photographs. Therefore, if possible, the trunk and the main branches are extracted from the photographs and used as two-dimensional attractor graphs for each input image. These attractor graphs are combined to influence the 3D particle simulation by modifying their directions. Subsequently, a triangular mesh is built around the resulting 3D graph using allometric rules as described in Chapter 5. While primarily focussing on automatic image-based construction, it is also possible to interactively guide the method to a desired result. By painting densities and by changing directions for particle simulation, the produced geometry can be modified in various ways. The user is able to locally adapt the model without having to adjust many parameters.

3.1 Overview

The proposed approach can be summarized as a particle simulation with external constraints from input images and internal botanical restrictions. It can be divided into five steps, which are computed one after the other. The outline below also reflects the further structure of the chapter:

1. **Pre-processing:** For each given input image the tree is separated from the background and a 2D attractor graph is computed or sketched by the user.
2. **Creation of the voxel model:** A voxel-grid is filled with density values by back-projecting the input images. The values of the voxels are an estimate of the tree density.
3. **Computation of direction fields:** Direction fields from 2D attractor graphs are used to transfer information of the input images to the particle simulation. These fields provide direction vectors for each input image plane. Combining these vectors for all image planes yields vectors that direct the particles in the subsequent 3D flow simulation.
4. **Particle simulation:** Random initial positions in proportion to the density values are set for particle flow simulation. The traces of the particles are influenced attraction forces to nearest neighbors and by the direction fields. As a result the main tree skeleton is obtained in form of a 3D graph.
5. **Production of geometry:** The tree skeleton is now converted into 3D geometry using allometric rules. Tiny branches and leaves are added and create the final foliage.

3.2 Pre-processing

Usually the input images contain background objects. Therefore the first step is to separate the tree from the background. This is particularly complicated for natural objects with many holes. Fortunately, in recent years a number of methods for performing alpha matting have been published [Ruzon and Tomasi, 2000, Pérez et al., 2003, Sun et al., 2004] [2] . Usually these methods are not fully automatic. A common setup lets the user create an initial trimap that specifies the pixels of the object, pixels of the background and an uncertainty region. In this case, pixels with appropriate colors are selected in the input images. The separation algorithm fills the uncertain regions with either opaque (foreground), transparent (background) or partly transparent pixels based on an interpolation, which takes the image gradients into account . The results from the matting procedure, as described by Sun

[2] Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 18–25, 2000; Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (Proc. of SIGGRAPH '03)*, 22:313–318, July 2003; and Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Transactions on Graphics (Proc. of SIGGRAPH '04)*, 23, August 2004

et al. [Sun et al., 2004], are interpreted as a density estimation of the tree for the corresponding view (see Figure 3.2(b)).

3.2.1 Alpha Matting

An important pre-processing step is the separation of the tree from the background. In recent years there has been a number of algorithms dealing with *alpha matting* of natural objects.

Most algorithms have in common that the Intensity I of a pixel in the image is seen as a linear combination of foreground color F and background color B with

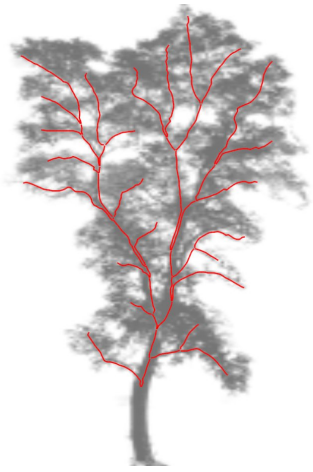
$$I = \alpha F + (1 - \alpha)B.$$

For a given Image I we need to find the fore- and background color and the corresponding alpha values.

The first approaches in this area simplify the problem by introducing a known background color distribution that is either blue for the *Bluescreen Matting* as introduced by Smith and Blinn [Smith and Blinn, 1996] [3] or given by an additional picture taken without the foreground object known as *Difference Matting*. Current methods are not relying on additional informations about the background color distribution and allow alpha matting of natural objects on arbitrary backgrounds [Chuang et al., 2000, Zongker et al., 1999, Chuang et al., 2001]. These methods typically start with a manually edited trimap indicating fore-, background, and uncertain regions.

[3] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Proc. of SIGGRAPH '96*, pages 259–268, New York, NY, USA, 1996. ACM

(a)



(b)

Figure 3.2: a) Input image; b) tree density estimation with corresponding attractor graph.

In 1984 Porter and Duff [Porter and Duff, 1984] [4] introduced the alpha channel concept that allows for complex image compositions. One of the image operations is the *Over* composition defined as

$$C = \alpha F + (1 - \alpha)B$$

with the resulting color of the current pixel C, foreground color F, background color B, and α as blending value.

This is the basic concept used for Bluescreen Matting [Smith and Blinn, 1996], one of the first matting techniques. The foreground object is captured in front of an unicolored background and then the under determined problem is solved

$$\alpha = 1 - \alpha_1 (C_b - \alpha_2 C_g)$$

with C_b blue and C_g green channel, using the additional condition

$$0.5 \leq \alpha_2 \leq C_b \leq \alpha_2 C_g.$$

The parameter α_1 and α_2 are manually set by the user.

Other algorithms which rely on information about the background color distribution are the so called *Difference Matting* techniques. Here the difference between known background and the given image is used to estimate α with $\alpha = 1$ or $\alpha = 0$, depending on a threshold value.

Recent techniques that are solving the more general problem of alpha matting objects in front of arbitrary unknown backgrounds (*Natural Image Matting*) are proposed by Ruzon *et al.* [Ruzon and Tomasi, 2000] [5] and

[4] Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics (Proc. of SIGGRAPH '84)*, 18:253–259, January 1984

[5] Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 18–25, 2000

(a) Input image **(b)** Resulting alpha map **(c)** Foreground object

Figure 3.3: Alpha matting input images.

Chuang *et al.* [Chuang et al., 2001] [6] . They generally rely on a manually edited trimap. The results are foreground color F, background color B, and the α value for each pixel of the uncertainty region. Both algorithms (Ruzon *et al.* and Chuang *et al.*) are based on probability distributions. Ruzon *et al.* define a transition region that is split into several smaller regions that are used as samples to define the foreground color distribution $P(F)$ and the color distribution of the background $P(B)$. The observed color C is based on a color distribution of the current image $P(C)$ that is a combination of $P(B)$ and $P(F)$ and is used to estimate α for a given pixel.

[6] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '01)*, volume 2, pages 264–271. IEEE Computer Society, December 2001

Alpha matting is performed for all input images. The resulting alpha maps (the alpha values of each pixel of the image – see Fig. 3.3(b)) and foreground images (Fig. 3.3(c)) are used in the following to deduce information about the branching pattern of the tree and to initialize the 3D density distribution.

3.2.2 Branching Pattern

In a second step, the images are used to sketch an estimate of the underlying tree skeleton in the corresponding view using the *Livewire* approach [Chodorowski et al., 2005] [7] . The algorithm needs a target point – usually the foot point of the tree or the position of the first branching on the trunk (see Fig. 3.2(b)). Additionally, seed points for the branches have to be determined. This can be done automatically by randomly selecting points on the tree silhouette or by manually introducing seed points, which usually creates better results.

[7] A. Chodorowski, U. Mattsson, M. Langille, and G. Hamarneh. Color lesion boundary detection using live wire. In *Proceedings of SPIE Medical Imaging: Image Processing vol. 5747*, pages 1589–1596, 2005

The attractor graph is now computed with every branching point forming a node in the graph. The algorithm starts at each seed point and finds a path through the visual structures of the input image to reach the target point (this is in fact another kind of particle simulation). The result reflects the main branching structure of the tree in the image. This is done for all photographs and we call the resulting graphs *attractor graphs*, see Figure 3.2(b) and 3.4 (a,b). If no information is contained in the image, an arbitrary skeleton is produced or the user can provide the information as a manually sketched graph. This provides an intuitive way to influence the resulting model and gives the user an additional modeling tool.

Attractor Graph

3.3 Computing the Tree Density

For now let us assume the camera model of the input images to be a parallel projection and to have two input images at a right angle. In the previous step we obtained the alpha values of the input photographs, and now we construct an initial 3D estimation of the plant volume that encompasses a voxel grid. Initial density values for the voxels are estimated from the input photographs. A discretized version of the volume-rendering equation allows us to compute the desired solution. Solving for a least-square solution results in a refined density volume that is later used for the branching structure and as a bounding volume for the final foliage. Each voxel V_i in the grid is assigned a density value α_i. An iterative algorithm is used to refine the values after initialization. In the final density distribution, voxels with higher α values will belong to parts of the plant that contain many leaves or branches.

3.3.1 The Volume Rendering Equation

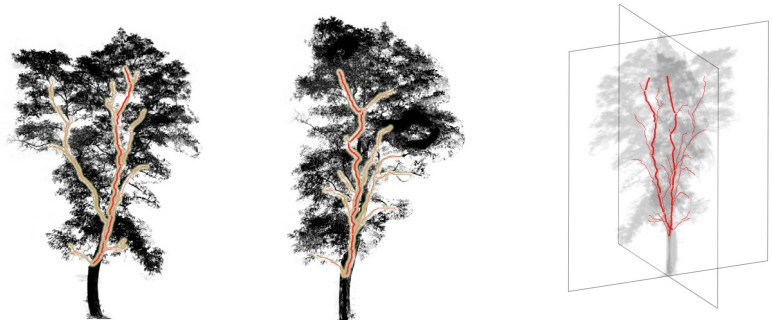
A well known method to generate images from volumetric representations is found in [Sabella, 1988, Max, 1995] [8] using a discrete form of the volume-rendering equation for an emission-absorption model without scattering:

$$I(s_n) \quad = \quad \sum_{k=0}^{n} b_k \prod_{j=k+1}^{n} \theta_j. \tag{3.1}$$

The value θ_j is the transparency of voxel j and b_k is the light emitted from the k-th voxel.

Considering the given input images as a solution for Eq. 3.1, it is possible to reconstruct the density values of the volume grid V. Reche-Martinez

[8] Paolo Sabella. A rendering algorithm for visualizing 3d scalar fields. *Computer Graphics (Proc. of SIGGRAPH '88)*, 22: 51–58, June 1988; and Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG '95)*, 1(2):99–108, 1995. ISSN 1077-2626



(a) (b) (c)

Figure 3.4: Example attractor graphs (a-b); (c) attractor graphs embedded in the image planes.

et al. [Reche-Martinez et al., 2004] [9] use a similar approach to reconstruct the volume model for their image-based rendering algorithm and rely on the same assumptions. However, it is important to note that in this case an accurate reconstruction is not needed, since in the proposed approach images are not directly rendered from the volumetric reconstruction. Instead we use the density information to subsequently produce the density of a surface representation of the plant. This allows us to apply a simple reconstruction method and a relatively coarse grid with typically $25 \times 25 \times 25$ voxels.

To initialize the density α_i for the i-th voxel V_i we project the voxel back onto each input image. We determine the average density of the projected area and use the minimum value as an initialization for α_i. Note, that the resolution of the voxel grid is typically much coarser than the input images we use, so many pixels are combined for one projected area.

This initial value would only be correct if all other contributing voxels had zero density and therefore is an upper limit for the transparency. Voxels with initial density values below a predefined threshold are rejected and not considered in the following steps. This can be seen as implicit space carving and reduces the complexity in the following steps considerably without reducing the quality of the resulting density estimate.

[9] Alex Reche-Martinez, Ignacio Martin, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (Proc. of SIGGRAPH '04)*, 23(3):720–727, August 2004

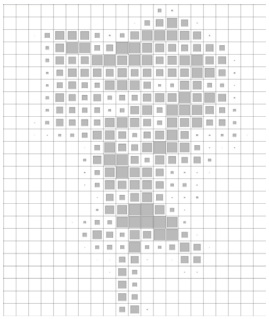
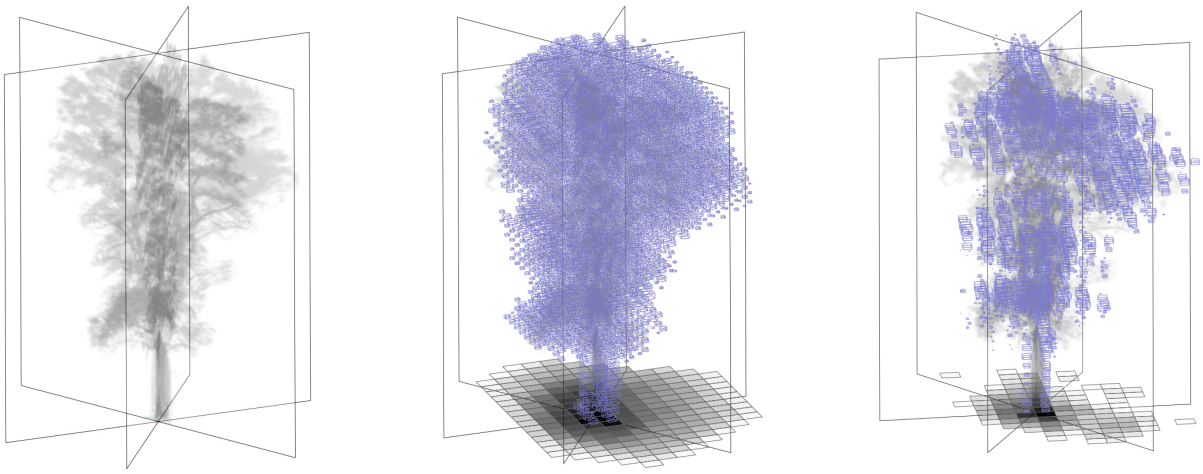


Figure 3.5: Density values for one image plane. High density values are marked by large squares.



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

Figure 3.6: Estimating the tree density: (a) Initial density values for three input images; (b) Voxel grid by back-projection; (c) Refined voxel grid

Solving the system of linear equations in an iterative way similar to Reche-Martinez *et al.*[Reche-Martinez et al., 2004] leads to the refined 3D density distribution. In Figure 3.6 a) a set of three input density images is shown. The initial alpha values of the voxel grid can be seen in Figure 3.6 b); the resulting least square solution is shown in Figure 3.6 c). In the solution, negative results are considered empty voxels. In Figure 3.5, the computed

density values for one image plane are shown. Their density values are indicated by the size of grey squares.

3.3.2 Computing the Linear Equation System

Based on Eq. 3.1 a system of linear equations is set up in the following way: For each pixel in each input image the voxels that contribute to the intensity value of this pixel are determined. This is accomplished by following a ray from a center of projection (or orthogonally to the input image in the case of an orthogonal projection) and marking all visited voxels. Since we use an absorption-only model, the density value of the pixel α_p is related to the density values α_i of the traversed voxel and the transparency relation $\alpha_k = 1 - \theta_k$:

$$\alpha_p = \alpha_n + (1-\alpha_n)(\alpha_{n-1} + (1-\alpha_{n-1})(\dots(1-\alpha_1)\alpha_0)\dots)$$

By substituting the density values with the corresponding transparency values $\theta_i = (1-\alpha_i)$ we achieve

$$\theta_p = \prod_{i=0}^{n} \theta_i. \tag{3.2}$$

A linear equation for each pixel in each input image can be found after applying the logarithm to both sides of the equation. The combination of all equations results in a highly overdetermined set of equations, since the resolution of the voxel system is rather small with respect to the resolution of the input images. The system is denoted by:

$$Ax = b \tag{3.3}$$

with $A \in \mathbb{R}^{psize \times voxel}$, $x \in \mathbb{R}^{voxel}$ and $b \in \mathbb{R}^{psize}$ with $psize = images \cdot pixel$, a vector that contains all pixels of all input images. For A we have $a_{i,j} = 1$, if the j-th voxel is hit by a ray through the i-th pixel.

The solution for Eq. 3.3 is found by using a least square fitting, thus by solving

$$A^T Ax = A^T b.$$

In Figure 3.6 a) a set of three input density images is shown. The initial alpha values of the voxel grid can be seen in Figure 3.6 b); the least square solution is shown in Figure 3.6 c). In the solution negative results are considered to be empty voxels.

In the case of using a perspective camera model, the back-projection of the pixels requires a more precise registration of the images. Not only the orientations, but also the position of the center of projection as well as the focus angles have to be known. For a detailed description of how to deal with such a situation please refer to Reche-Martinez *et al.* [Reche-Martinez et al., 2004]. However, as stated above, for the proposed method only an approximate registration is needed, since density estimation and particle tracing are tolerant against inconsistencies. This is demonstrated in the accompanying video by showing models with completely uncorrelated input images and models that were created from only a single image.

3.4 Particle Tracing

Particle positions for the main tree skeleton are initially set using the 3D density values. The particles are placed randomly in the voxels in proportion to their density. Since little is known about specific botanic measurements of the total number of branches in different tree species, a heuristic is used that can be manually adapted if the resulting tree model is not convincing. For medium-sized trees between 500 and 1000 particles are used and for large models usually between 1000 and 2000.

As mentioned above, the proposed method extends the method by Rodkaew *et al.* [Rodkaew et al., 2003] [10] in order to achieve specific tree skeletons and shapes according to input photographs. This is done by introducing attractor graphs to particle tracing and by establishing additional rules. In the following section the general particle simulation step is discussed, followed by a discussion of the introduced refinements.

[10] Yodthong Rodkaew, Prabhas Chongstitvatana, Suchada Siripant, and Chidchanok Lursinsap. Particle systems for plant modeling. In B. Hu and M. Jaeger, editors, *Plant Growth modeling and Applications (Proc. of PMA '03)*, pages 210–217, 2003

A particle p_i is represented by a position x_i, velocity x_i', and mass m_i. It moves under the influence of time-dependent forces represented by $f_i(x_i, x_i', t)$. The Newtonian law gives us $f_i = m_i \cdot x_i''$ and $x_i'' = f_i / m_i$ which is a well studied differential equation of second order [Hockney and Eastwood, 1988, Witkin and Baraff, 1997] [11] and usually written in the form of coupled first order differential equations:

$$[x_i, x_i'] = [x_i', f_i / m_i] \tag{3.4}$$

[11] Roger W. Hockney and James W. Eastwood. *Computer simulation using particles*. Taylor & Francis, Inc., 1988; and Andrew Witkin and David Baraff. Physically based modeling: Principles and practice. SIGGRAPH '97: Course Notes, 1997

This system can be solved iteratively using an explicit Runge-Kutta method as found in Press *et al.* [Press et al., 1992] [12] . During simulation, the

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992

positions of the particles are updated according to their mass and external forces. The mass can be seen as some kind of reluctancy of a particle to change the current orientation and/or velocity.

One important aspect of the force is to implement the particle attraction. This attraction is needed to form the tree skeleton similar to the given images. Particles close to each other are forced to join and subsequently move forward together. This is implemented by searching the nearest neighbor for each particle and combining them if their distance is below a given threshold. The basic particle tracing mechanism can be written as follows:

```
Initialize particle positions according to voxel density
while particles not at tree basis do
    foreach particle p_i do
        determine forces f_i
        determine velocities and positions (Eq. 3.4)
    end
    determine nearest neighbor and join if close enough
end
```

Algorithm 1: Particle tracing

The particle traces are stored as a 3D branching graph, which describes the main tree skeleton. This basic simulation is now refined in order to match our modeling requirements.

3.4.1 Solving Initial Value Problems

Euler Method

The *forward Euler method* is the simplest *explicit* method to solve an *initial value problem* (IVP, see Press et al. [1992, 707 ff] and Stoer and Bulirsch [1978, 115 ff]) described as

$$y' = f(x,y) \qquad y(x_0) = y_0 \tag{3.5}$$

with $f(x,y(x))$ being the derivative $y'(x)$ of the solution $y(x)$ of Eq. 3.5. For $h \neq 0$,

$$\frac{y(x+h)-y(x)}{h} \approx f(x,y(x))$$

or

$$y(x+h) \approx y(x) + hf(x,y(x)).$$

holds.

Choosing a step size h and starting with the initial value x_0 and $y_0 = y(x_0)$ one gets at $x_i = x_0 + ih, i = 1, \ldots, n$ approximate solutions η_i for $y_i := y(x)$ of the exact solution $y(x_i)$:

$$\eta_0 = y_0$$

and for $i = 0, 1, 2, \ldots$

$$\begin{aligned} \eta_{i+1} &:= \eta_i + hf(x_i, \eta_i) \\ x_{i+1} &:= x_i + h. \end{aligned}$$

A general description of these methods is given by

$$\Phi(x, y, h, f)$$

and starting with x_0, y_0 as initial values approximate solutions η_i for $y_i := y(x_i)$ of the exact solution $y(x)$ are found with

$$\begin{aligned} \eta_0 &:= y_0 \\ \eta_{i+1} &:= \eta_i + h\Phi(x_i, \eta_i, h, f) \\ x_{i+1} &:= x_i + h \end{aligned}$$

.

For the Euler-Cauchy method we get $\Phi(x, y, h, f) := f(x, y)$. Single step methods of higher order can be described using more complex Φ, as for example the second order Heun method with

$$\Phi(x, y, h, f) := \frac{1}{2}[f(x, y) + f(x + h, y + hf(x, y))]$$

or the fourth order Runge-Kutta method with

Fourth order Runge-Kutta method

$$\Phi(x, y, h, f) := \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \tag{3.6}$$

and

$$
\begin{aligned}
k_1 &:= f(x,y) \\
k_2 &:= f(x+\frac{1}{2}h, y+\frac{1}{2}hk_1) \\
k_3 &:= f(x+\frac{1}{2}h, y+\frac{1}{2}hk_2) \\
k_4 &:= f(x+h, y+hk_3)
\end{aligned}
$$

Generally higher order methods provide higher mathematical *stability* and *stiffness* at the costs of multiple evaluations of f (see Eq. 3.6). Other methods for the solution of initial value problems such as multistep or exploration methods are not described here, since the Runge-Kutta method with adaptive stepsize is sufficient for the class of problem in this case.

3.5 The Direction Field

The simulation so far combines the particle traces to small branches that merge and form the skeleton. The particles are directed towards the tree basis and towards their respective nearest neighbors. Additionally, for each 2D attractor graph from the input images (see Section 3.2) we create a two-dimensional discrete vector field in the according image plane. Our goal is to direct the particles in such a way that they simultaneously move towards the attractor graphs in any of the image planes that correspond to these input images.

We project the particle position into each of the image planes using the respective projection. Then we determine the direction vector from the vector fields in the global coordinate system. With two input images at a right angle, we would obtain two vectors perpendicular to each other. Generally speaking, we get a set of direction vectors for which we compute the average. This vector, when projected back onto the image planes, is the average direction and is used as part of the external force that is applied to the particle. Theoretically there is a chance that all direction vectors might sum up to zero, however our practical experiments have never supported such a case.

The direction field is computed by applying a distance transform to the attractor graph. For each position $x_{i,j}$ in the direction field the closest point on the attractor graph $g_{i,j}$ is computed; let $\mathbf{v}_{i,j} = g_{i,j} - x_{i,j}$ be the vector pointing towards the $g_{i,j}$. Additionally we compute the tangential vector of the graph $\mathbf{t}_{i,k}$ at position $g_{i,j}$.

Normalized versions of these two vectors, $\bar{\mathbf{v}}_{i,j}$ and $\bar{\mathbf{t}}_{i,j}$, are used for computing the forces on a particle close to $x_{i,j}$. We modeled this force as a

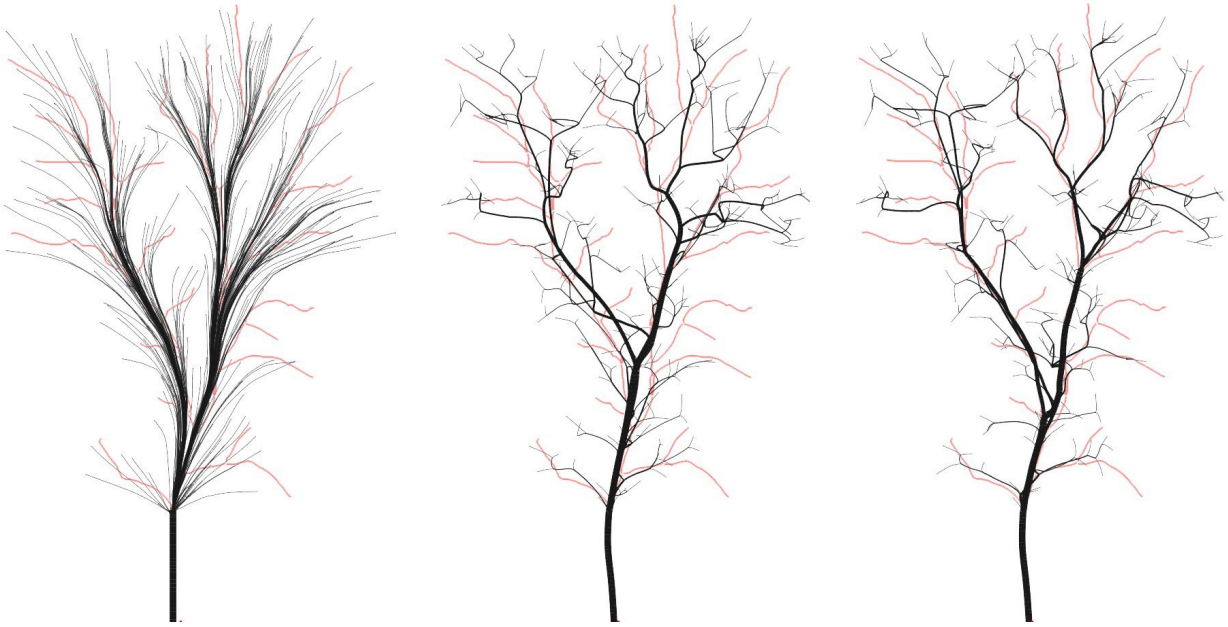


Figure 3.7: (a) Result for linear blending without particle attraction; (b) result for constant blending including particle attraction; (c) result for linear blending including particle attraction.

weighted sum using the blending function $h(d)$ which depends on the distance $d = |\mathbf{v}_{i,j}|$ to the graph:

$$f_{i,j} = h(d) \cdot \bar{\mathbf{v}}_{i,j} + (1 - h(d)) \cdot \bar{\mathbf{t}}_{i,j} \tag{3.7}$$

If the blending function is linear, particles far away from the graph are directed towards $g_{i,j}$ and particles close to the graph into the tangential direction $\mathbf{t}_{i,j}$ in $g_{i,j}$. If instead we use a constant function $h(d)$ for all $d > 0$, the particles are directed at a fixed angle towards the closest point $g_{i,j}$ on the nearest graph segment.

The direction vectors $f_{i,j}$ are computed for a two-dimensional grid which is embedded in each of the image planes. The resolution for these grids does not need to match the resolution of the voxel grid nor the resolution of the images. In our practical tests a grid of 100 by 100 proved to be sufficient.

In Figure 3.7 and 3.8 we demonstrate the particle flow for a two-dimensional case. Figure 3.8(a) shows a direction field for a linear blending function $h(d)$, in (b) for a constant function. Figure 3.7 (a) shows the result for a flow simulation using the field of subfigure (a), but without attraction between particles. In (b) the field of Figure 3.8(b) is used, this time including attraction. The branches now have a nearly fixed branching angle due to the given field. Finally, Fig. 3.7 (c) shows particle flows with linear $h(d)$ and attraction, the

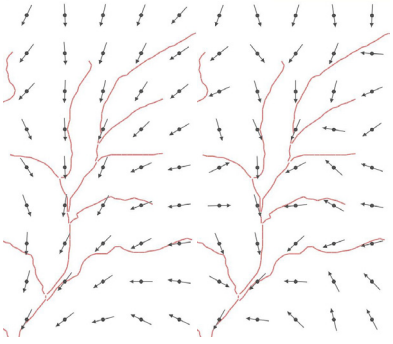


Figure 3.8: (a) Direction field for a given graph and linear blending function $h(d)$, the vectors close to graph segments point down the graph; (b) for a constant blending function, the vectors point at a given angle towards the nearest segment.

Figure 3.9: Two results for a given direction field and different initial particle positions (see Figure 3.12).

most often used case. In this case the graph follows the given direction field quite closely.

3.6 Results and Discussion

Figure 3.10, 3.11 and 3.12 show three tree models with their corresponding input photographs. Animations of the models can be found in the accompanying video. The geometric complexity of the models is 555,000 triangles for the first tree and 285,000 triangles for the second. In both cases the given shape and structure of the input is approximated quite faithfully by the models. In Figure 3.11 a pine tree is created from two quite poor photographs.

The modeling works in most parts at interactive rates. The pre-processing separates the tree in the input images and creates the attractor graph interactively. The volume model is created within a few seconds. The flow simulation of the particles needs about 5–10 seconds for 1000–2000 particles and 200 iterations. The branch geometry can be produced within a matter of seconds and the leaves are also added within seconds even for complex models. All times were recorded on a standard PC with 3 GHz.

In many cases the trees in the images are partly occluded by other objects or have a shape that is deformed due to natural reasons. In this case it is easy to fix the problem in the input images by drawing and removing density (see Figure 3.12). Another situation is due to the seasons. We had three photographs of an oak at winter time. So we created a tree skeleton (Figure

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

Figure 3.10: One example of the created tree models: (a) input photographs; (b) 3D model in view corresponding to upper photograph; (c) view corresponding to lower photograph; (d) another view.

3.9) and added density to the images by simply painting some strokes. The result is a tree model that looks natural and reflects the sketched density by its branches and foliage.

3.6.1 Limitations of the Method

While the proposed method creates high quality models, it has some limitations: As outlined above, the branching patterns at smaller branches are influenced not only by branching angles but also by specific patterns in which branches appear. Such branching patterns are hard to simulate using particle flows since in this case complicated interaction of particles during the flow simulation is needed or a post-processing step after simulation has to be performed. A moved branch might interact with other branches, or curvature might need to be adapted, etc. Following strictly the simulation-based modeling paradigm, it is impossible to solve this issue completely, but to minimize these problems the above mentioned compound leaves are introduced: a number of simple leaves arranged procedurally on a small twig.

It is also not possible to create all tree species as convincingly as others using simulation-based modeling. Some trees change their shape and structure among branching levels. This is hard to simulate without creating different direction fields and particle rules for different branching levels. That said, a more complex simulation system could be able to capture these properties as well, which is an interesting field for future research.

Figure 3.11: Pine tree example.

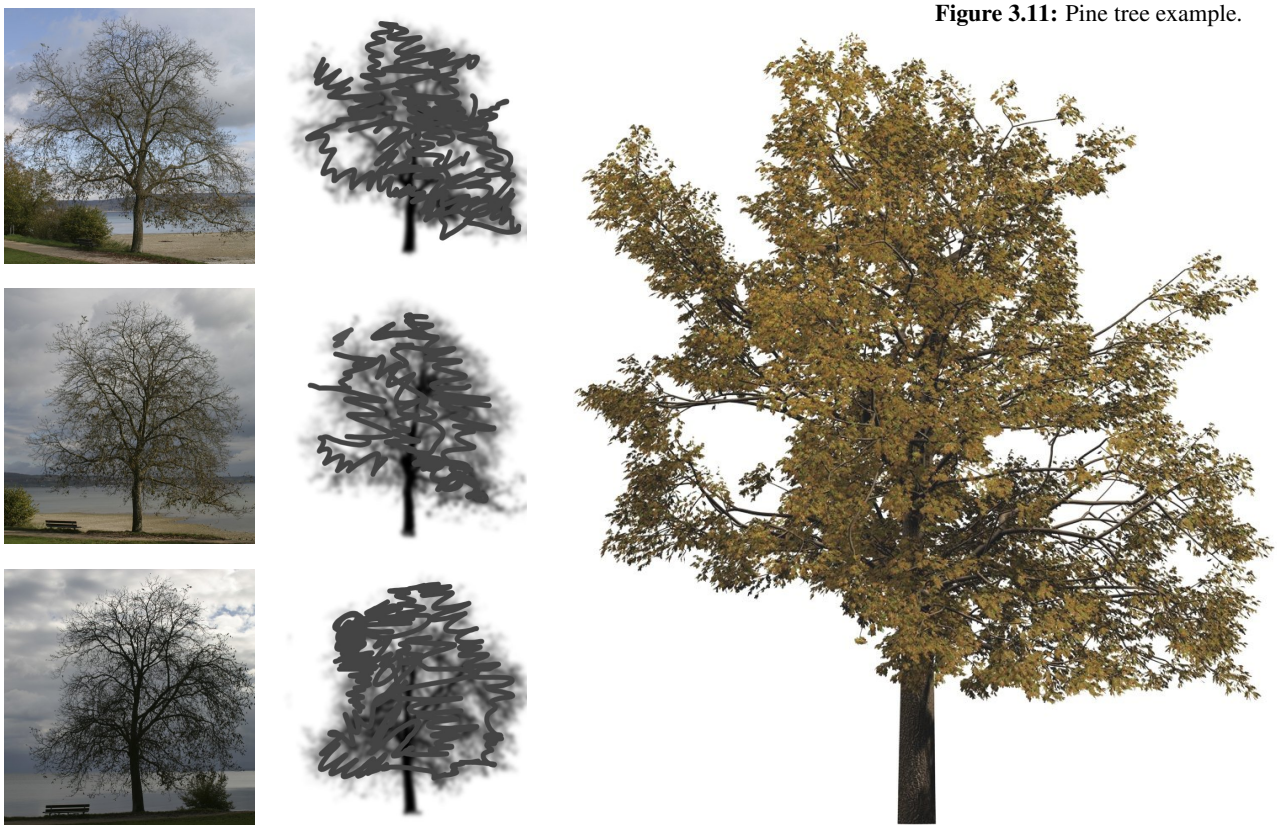


Figure 3.12: Sketch-based modeling of an oak. Left: input photographs; middle: sketched density; Right: resulting model. Please note the hole in the left part of the tree that can also be found in the sketched density of the corresponding view. The skeleton of the oak is shown in Figure 3.9(d).

3.7 Conclusions

In this chapter a new image- and simulation-based modeling method for 3D tree geometry is presented. By imposing image constraints in the form of captured branching patterns and density distributions, the proposed method is able to adapt the particle simulation to a given set of input images. This allows to create convincing tree models that approximate a given shape. Using different initial positions for the particles, different but similar tree skeletons can be produced. Tiny twigs and leaves are added, the method runs at interactive rates and also allows for the modeling of trees by manually altering input images and direction fields for the flow simulation. The approach has been tested with several sets of input images.

Coupling the approach with Level-of-Detail data structures is an interesting focus for future research, since the produced models are often too complex for interactive applications. The branching graph already incorporates a hierarchy since it has larger and smaller branches. For distant models only the large branches might be shown together with an visual approximation of the foliage. Smaller branches and leaves might be generated on the fly when needed. The presented simulation based modeling paradigm makes it possible, with little adaptions to the simulation system, to generate model details on demand, without the need of storing intermediate model representations and even without generating the full detailed model. This property makes simulation-based modeling especially interesting for streaming applications.

4

Data-driven Plant Modeling

Figure 4.1: A tree sketch by Leonardo da Vinci [Richter, 1970, Plate XXVII].

It is possible to interpret sketches of plants and trees using statistical properties of given 3D models. Sketches are not much different from pictures or photographs. However, humans have the ability to abstract information found in pictures and preserve this information in sketches. Automatically deducing 3D information from pictures typically requires multiple instances of images of the object, which is generally not possible for sketches due to the aforementioned abstraction. This chapter proposes an algorithm that utilizes prior information from 3D models to interpret single sketches of tree branching structures and deduce a 3D branching model from one freehand sketch, successfully filling the missing information needed to get a full 3D model, which then can be used to render arbitrary views of the sketched tree.

It is an easy task for humans to find a plausible 3D structure from a single source. This is usually done by utilizing world knowledge.

Providing world knowledge not only helps in the interpretation of the sketch, but also gives a powerful tool to change the appearance of the final model by changing the data used as prior.

To model the world knowledge this chapter introduces the use of *Markov*

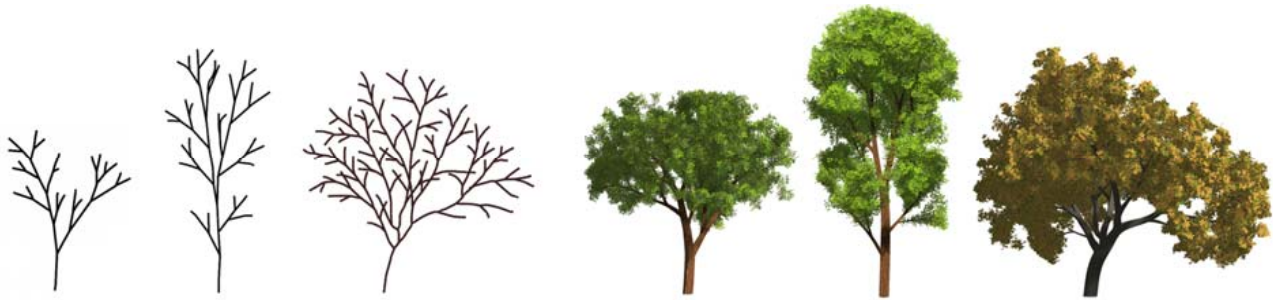


Figure 4.2: Examples of 3D tree models generated from freehand sketches. From left to right: three sketches, and the corresponding tree models generated with our algorithm.

Random Fields (MRF) based on [Chen et al., 2008] [1] . MRFs not only allow the capture of statistical properties of prior models from a large database of full 3D plant models, but also provide well studied methods to find the configuration of all possibilities that are most likely, given the underlying statistical data coming from the tree model.

[1] Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based tree modeling using markov random field. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '08)*, 27:109:1–109:9, December 2008

4.1 Introduction

Achieving realism is one of the major goals of computer graphics, and many approaches ranging from physics-based modeling to image-based rendering have been proposed. Unfortunately, creating new content for realistic rendering remains tedious and time consuming. The problem is exacerbated when the content is being *designed*. 3D modeling systems are cumbersome to use and therefore ill-suited to the early stage of design (unless the design is already well-formulated).

Designers therefore continue to favor freehand sketching for conceptual design. Sketching appeals as an artistic medium because of its low overhead in representing, exploring, and communicating geometric ideas. Indeed, such speculative representations are fundamentally different in spirit and purpose from the definitive media that designers use to present designs. What is needed is a seamless way to move from conceptual design drawings to presentation rendering, which is the focus in this chapter: extending sketch-based modeling to complex and realistic-looking 3D tree models and thus addressing one of the still existing gaps in this area.

This chapter introduces a new, easy-to-use, and flexible method for creating tree models from freehand sketching. It allows the user to very quickly generate a variety of unique 3D models of trees. Given the 2D sketch, the system searches for a 3D interpretation whose projection matches the sketch *and* is natural-looking. The problem is formulated within a graphical modeling framework, using a database of trees as priors. Inference is performed in two steps. First, the initial shape is obtained by bottom-up local optimization at each branch segment from tree root to the rest of the drawn branches. Second, this shape is refined to avoid interpenetration between branches.

Once the 3D branches have been recovered, the model is augmented with more branches using self-similarity as a guiding principle. Subtrees are randomly selected and appropriately scaled and oriented before being attached to end branches. Finally, the user can select the leaves to be automatically added to the branches based on botanical rules. This completes the 3D tree model. Examples of tree models generated from sketches can be seen in Figure 4.2.

4.2 Probability Theory and Probabilistic Graphical Models

Most problems based on observations introduce uncertainty either through noise on measurements or through the finite number of observations. Probability theory provides the mathematical framework to deal with uncertainty of events and resulting probability distributions.

Graphical models provide a declarative representation within the field of probabilistic reasoning and encode the knowledge of how a system works. The key feature of graphical representations is the separation of this representation from algorithms that can be applied to obtain meaningful conclusions and answer the question about what is possible and, even more importantly, about what is probable. Probability theory provides the mathematical framework to answer these questions based on two simple equations: the sum rule and the product rule. In the remainder of this section the basic terminology of probability theory is introduced. For an excellent and more detailed introduction to probability theory and machine learning see Bishop [Bishop, 2006] [2] or Koller and Friedman [Koller and Friedman, 2009] [3] .

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006

[3] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009

Considering two random variables X and Y, that take the values x_i with $i = 1,\ldots,M$ and respectively y_j with $j = 1,\ldots,L$, the total number of instances $N = M + L$, n_{ij} being the number of instances for which $X = x_i$ and $Y = y_j$. Then the *joint* probability of $X = x_i$ and $Y = y_j$ becomes

Joint Probability

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}. \tag{4.1}$$

Summing up all instances for $X = x_i$, regardless of the outcome of Y with $c_i = \sum_j n_{ij}$, the *marginal* probability for $X = x_i$ becomes

Marginal Probability

$$p(X = x_i) = \frac{c_i}{N}. \tag{4.2}$$

Combining 4.1 and 4.2 gives the *sum rule* of probability

Sum Rule of Probability

$$p(X = x_i) = \sum_{j=1}^{L} p(X = x_i, Y = y_j). \tag{4.3}$$

The *conditional* probability of $Y = y_j$ given that $X = x_i$ is defined as

Conditional Probability

$$p(Y = y_j|X = x_i) = \frac{n_{ij}}{c_i}. \tag{4.4}$$

The *product rule* of probability can be formulated from 4.1, 4.2 and 4.4 as

Product Rule of Probability

$$\begin{aligned} p(X = x_i, Y = y_j) &= \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} && (4.5) \\ &= p(Y = y_j|X = x_i)p(X = x_i). && (4.6) \end{aligned}$$

For the sake of simplicity the less accurate but simpler notion of $p(X)$ for $p(X = x_i)$ is used in the following chapters. Solving the product rule for $p(Y|X)$ then gives

$$p(Y|X) = \frac{p(X,Y)}{p(X)} \tag{4.7}$$

and, since the symmetry property $p(X,Y) = p(Y,X)$ holds, translates into *Bayes' theorem*:

Bayes' Theorem

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \tag{4.8}$$

So far we have been looking into discrete probability distributions for which X is defined over a discrete set of events x_i. The extension to continuous distributions is straightforward

$$p(x \in (a,b)) = \int_a^b p(x)\partial x \tag{4.9}$$

considering that the probability of the real valued variable x is part of the interval (a,b).

All introduced rules apply equally to discrete and continuous distributions and combinations thereof.

Probability theory uses graphical models as representation to visualize the conditional independence structure between random variables. These diagrammatic representations of probability distributions play an important role in Bayesian statistics and can be grouped into Bayesian networks and Markov networks, both being heavily used for machine learning with applications in computer graphics and vision [Szeliski, 2010] [4] . Both methods provide insight into the model they represent and encompass the properties of

[4] Richard Szeliski. *Computer Vision: Algorithms and Applications (Texts in Computer Science)*. Springer-Verlag New York Inc, 1st edition, November 2010

factorization and independence; their difference is the set of independences they can encode and the factorization of the distribution that they induce. *Bayesian networks* operate on directed graphs while *Markov random fields* are defined over undirected graphs.

4.2.1 Bayesian networks

Given a joint probability distribution $p(a,b,c)$ over three variables a,b,c and applying the product rule (Eq. 4.5) gives

$$p(a,b,c) = p(c|a,b)p(a,b). \tag{4.10}$$

Applying the rule a second time to $p(a,b)$ Eq. 4.10 translates into

$$p(a,b,c) = p(c|a,b)p(b|a)p(a). \tag{4.11}$$

This can be translated into a graphical model of this distribution in the following way. For each random variable a,b,c a node in the graph is generated. Then for each conditional distribution on the right hand side of Equation 4.11, directed edges are added to the graph connecting the nodes of the affected random variables in the following way: for the factor $p(c|a,b)$ edges from node a and node b to the node denoted by variable c are introduced. Factor $p(b|a)$ translates into a directed edge from a to b while $p(a)$ does not generate an additional edge (see Fig. 4.3 and Fig. 4.4 as examples).

Given any directed acyclic graph it is possible to find a decomposition of the corresponding joint probability distribution by introducing a conditional distribution as a factor for each node. Taking the node's variable a as probability distribution conditioned on all variables of nodes having connections **to** node a.

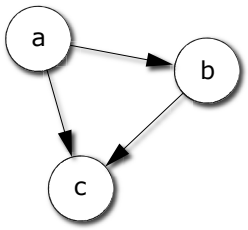


Figure 4.3: Simple Bayesian network representing the joint probability distribution (see Eq. 4.11): $p(a,b,c) = p(c|a,b)p(b|a)p(a)$

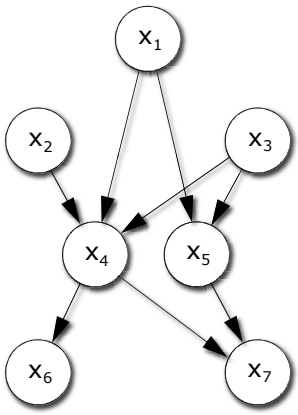


Figure 4.4: Complex Bayesian network representing the joint probability distribution $p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)$ $p(x_6|x_4)p(x_7|x_4,x_5)$

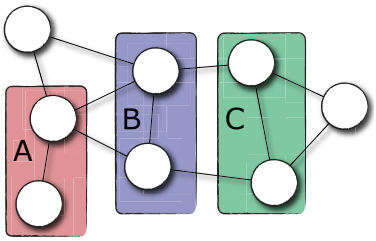


Figure 4.5: Complex Markov network: Conditional independence can be deduced as all path from subset A to subset C pass through a group of nodes B.

4.2.2 Markov Random Fields

Another class of graphical models is described by Markov random fields (MRF). In contrast to Bayesian networks MRFs are described by undirected graphs and additionally allow for representing cyclic dependencies. In MRFs, nodes are defined similarly for random variables and a set of edges that connect two nodes. Conditional independence of random variables can be identified by parts of the undirected graph being separated, i.e. all paths from one set A of nodes to another set C pass through a group of nodes B (see

Fig. 4.5). For the inference of the branching parameter in this chapter we used MRF and therefore limit the further explanations to the latter.

4.2.3 *Factor Graphs*

Both graphical models can be expressed in terms of factor graphs. These graphs are undirected, *bipartite* graphs with two different kinds of nodes directly representing the factorization of a function, with each factor directly representing the probability distribution over the variables it is connected to and therefore being less ambiguous. In addition to the nodes representing the random variables, another kind of node represents a factor (usually indicated by a squared node) of a subset of the random variables of the joint distribution (see Fig. 4.6).

Considering a given factorization in the form of

$$p(x) = f_a(x_1,x_2)f_b(x_1,x_2)f_c(x_2,x_3)f_d(x_3) \tag{4.12}$$

the resulting factor graph $G = (X,F,E)$ will have one node for each variable $x_i \in X$ (denoted by a round node), one node for each factor $f_j \in F$ (denoted by a square node) and one edge $e_k \in E$ connecting each factor to the random variables on which it depends (see Fig. 4.6 for the resulting factor graph of Eq. 4.12).

For a nice introduction to factor graphs see [Loeliger, 2004] [5] .

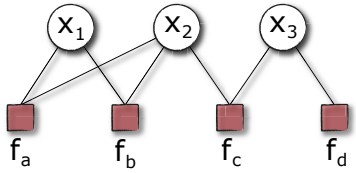
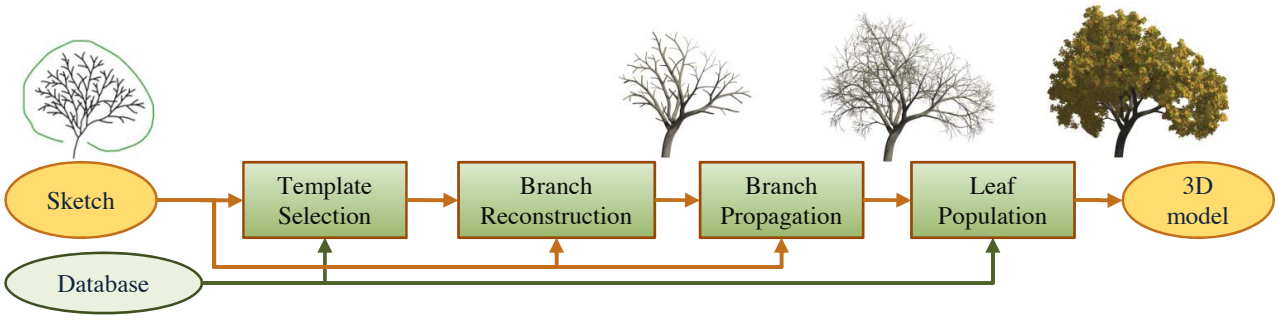


Figure 4.6: Factor graph representing the factorization $p(x) = f_a(x_1,x_2)f_b(x_1,x_2)f_c(x_2,x_3)f_d(x_3)$.

[5] Hans-Andrea Loeliger. An Introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, January 2004

4.3 *Overview of System*



The components of the proposed tree sketching system are shown in Figure 4.7. The user needs to simply provide a few strokes of branches, and

Figure 4.7: Overview of the sketch-based tree modeling system.

optionally the crown of the tree. The database contains typical tree exemplars and their associated global parameters. Based on the shape of the sketch, the system first selects the closest tree exemplar ("template"); the template's global parameters are subsequently used as a prior for constructing the 3D geometry.

Assuming that the sketch is drawn under orthographic projection allows the problem of constructing the 3D geometry of the sketch to be reduced to estimating the depths of branch segment endpoints. A reasonable shape of a tree can be reconstructed from its projection because trees have characteristic shapes, which provide powerful priors for 3D reconstruction. Such priors allow humans to perceive tree shapes from sketches. The location of each branch depends on the location of adjacent branches but the overall shape is dictated by global tree parameters. The local interconnectivity of information and imposition of global priors makes the reconstruction problem a natural fit for the use of Markov random fields. There is a direct mapping of branches to graph nodes, local interconnectivity of branches to interaction between nodes, and global tree parameters to data terms at nodes.

As such, the problem can be formulated as a Markov random field, with each branch segment as a node and its depth as a variable. More specifically, we are dealing with a Markov tree.resu (For a detailed description, please refer to textbooks such as [Bishop, 2006] [6]). In addition, we impose rules governing the tree shape as spatial relationships between neighboring nodes. These relationships are made explicit by introducing additional nodes, producing a *factor graph*.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006

Solving the factor graph produces the 3D shape of the drawn branches. The system then propagates branches using the principle of self-similarity: randomly selecting replication blocks, scaling, and reorienting the geometry, and then attaching them to open branches. If drawn, the crown constrains the overall shape of the tree during branch propagation. If the crown is not drawn, the branches are propagated by a fixed number of generations. To complete the tree model, the user can either select a leaf template from the tree database, or use the default leaf associated with the preselected template. The system populates the tree based on botanical rules. While this is only an approximation of natural diversity, the variety of trees shown in this chapter demonstrates the visual modeling power and expressiveness of the proposed system.

4.4 From 2D Sketches to Factor Graphs

Sketches can be seen as projections of a 3D model to a two dimensional image plane. Inferring the intended 3D structure from a given sketch can therefore be reduced to the problem of identifying plausible depth values for each branching point in the sketch.

The strokes in the sketch are first split into a set of inter-connected branch segments b_i, that are each represented by the position of their end point $\mathbf{p}_i$ and their direction $\mathbf{v}_i$ (see Fig. 4.8). Given a parent segment b_p and its child segment b_c, $\mathbf{v}_c$ is defined as $\mathbf{v}_c = \mathbf{p}_c - \mathbf{p}_p$. Since we assume orthographic projection we only need to extract the depth z associated with the end point of this segment.

From the sketch, 2D coordinates of the branch nodes $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$ are observed for the N branch segments of the tree. Thus a first formulation to maximize the posteriori given the sketch including $Z = \{z_1, \ldots, z_N\}$ is

$$p(Z|X,Y) = \prod_{i=1}^{N} f_i(\mathbf{v}_i, \mathbf{v}_j) \tag{4.13}$$

with b_j being the parent branch of b_i assuming independence between child segments. The posteriors in Eq. 4.13 are made explicit by introducing the factor nodes in the resulting factor graph (see Fig. 4.8 c)).

As one can see from this formulation, we need to make certain assumptions about the factors $f_i(\mathbf{v}_i, \mathbf{v}_j)$. In this case this is done by introducing global gaussian distributions defined by observations of models in the data base. Since the number of observations for pairs of positions p_i and p_j in world space would be way too sparse, one can define the position of the child node p_j in local coordinates of the parent node. introducing rotation angles α and γ, and a scale value s. For now, assuming identical distribution for these parameters, a set of global parameters per model can be defined as

$$\Omega = \{\mu_s, \sigma_s, \mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\} \tag{4.14}$$

with μ and σ being the mean and variance of gaussian distributions for the respective parameter $s \sim \mathcal{N}(\mu_s, \sigma_s)$, $\alpha \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$, and $\gamma \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma)$.

This defines the factor as

$$f_i(\mathbf{v}_i, \mathbf{v}_j|\Omega) = p(s_i|\Omega)p(\alpha_i|\Omega)p(\gamma_i|\Omega) \tag{4.15}$$

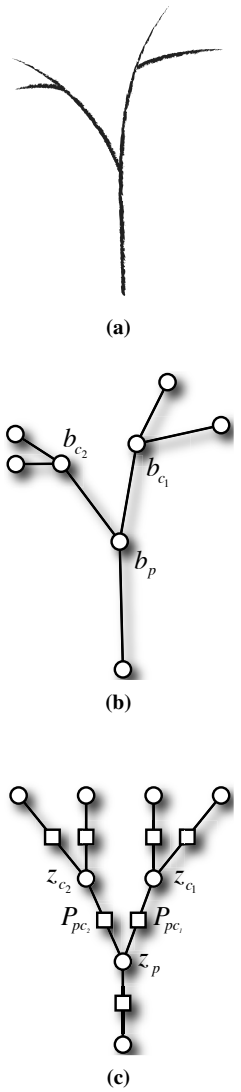


Figure 4.8: Mapping the sketch to a factor graph. (a) Tree sketch, (b) corresponding Markov tree, (c) final factor graph. Each node in the graph represents a branch segment in the sketch.

implicitly encoding the parent child relation between b_j and b_i.

4.5 *Tree Data Structure*

The tree is composed of a set of branch segments. Each branch segment b_i is represented by the position of its end point $\mathbf{p}_i$ and its direction $\mathbf{v}_i$. Given a parent segment b_p and its child segment b_c, $\mathbf{v}_c = \mathbf{p}_c - \mathbf{p}_p$. The transformation of these two vectors are modeled by a scale s and rotation matrix R_{pc} such that $\mathbf{v}_c = sR_{pc}\mathbf{v}_p$. Each branch segment b_i has its local coordinate system, the position of the end point $\mathbf{p}_i$ and the segment vector $\mathbf{v}_i$. The relation between parent-child branch segments is represented by the geometry transformation. In the local coordinate system of the parent b_p, the scaled rotation from the segment vector $\mathbf{v}_p$ of b_p to its child segment vector $\mathbf{v}_c$ is defined by a scale s and two rotation angles α and γ around the x and y axis of the parent coordinate system respectively. An alternative formulation then yields $\mathbf{v}_c = sR_y(\alpha)R_x(\gamma)\mathbf{v}_p$.

The branching shapes are typically spatially variant for the whole tree. To account for this, a Gaussian distribution is used to characterize the probability distribution of the tree parameters. The scale s of each pair of parent-child segments has the identical distribution, that is $s \sim \mathcal{N}(\mu_s, \sigma_s)$.

In physical biology the most obvious distinction in branching angles is between lateral and apical growth. The active tissue in plants is called meristem that develops either in *apical* (primary) or *lateral* buds (secondary). While the assumption of identical distribution is generally true for apical branches, it can be refined in the case of lateral branches.

For a parent segment b_p with K child segments $\{b_{c_1}, \ldots, b_{c_k}\}$, one of the segments is identified as the apical continuation of b_p, and the divergence angles and the orientation adjustment angles of all the apical segments are assumed to have identical distribution, that is $\mathcal{N}(\mu_\alpha, \sigma_\alpha)$ and $\mathcal{N}(\mu_\gamma, \sigma_\gamma)$ respectively.

The other child segments are lateral. While the divergence angle of all the lateral segments have identical distribution, that is $\mathcal{N}(\mu_\beta, \sigma_\beta)$, the distribution of orientation adjustment angles depend on their label. Based on the different orientation adjustment angle, the child segments distribute evenly around their parent segment. As shown in Fig. 4.9,

1. For branches with only one child branch ($K = 1$) one can safely assume that the child is an apical branch with

$$\alpha_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$$

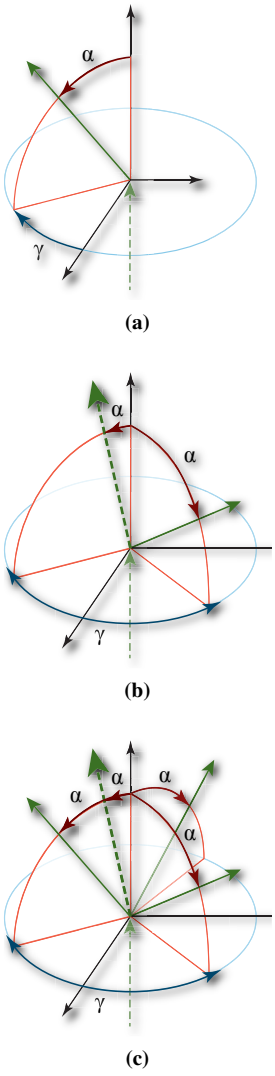


Figure 4.9: Illustration of tree parameters described in Section 4.5 for a different number of lateral branches.

and

$$\gamma_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma).$$

The distribution parameters of the child segment are $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$.

2. If $K = 2$, the rotation angles of the two child segments are

$$\alpha_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$$
$$\gamma_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma)$$

and

$$\alpha_{c_2} \sim \mathcal{N}(\mu_\beta, \sigma_\beta)$$
$$\gamma_{c_2} \sim \mathcal{N}(\mu_\gamma + \pi, \sigma_\gamma)$$

respectively. The resulting two sets of distribution parameters are $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$ and $\theta_2 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma + \pi, \sigma_\gamma\}$.

3. For more than two child branches ($K > 2$), the rotation angles of the apical segment are

$$\alpha_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$$
$$\gamma_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma).$$

For the remaining lateral child segments rotation angles are

$$\alpha_{c_i} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$$
$$\gamma_{c_i} \sim \mathcal{N}(\mu_\gamma + \frac{2\pi(i-1)}{K-1}, \sigma_\gamma)$$

with $i = 2, \ldots, K$. The corresponding parameter sets for each child segment are: $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$ for the apical and $\theta_i = \{\mu_\beta, \sigma_\beta, \mu_\gamma + \frac{2\pi(i-1)}{K-1}, \sigma_\gamma\}$ with $i = 2, \ldots, K$ for the lateral branches.

Taking this distinction between lateral and apical branching angles into account gives an extended set of global parameters and an slightly extended factor graph in the following way. Let $\Omega = \{\mu_s, \sigma_s, \mu_\alpha, \sigma_\alpha, \mu_\beta, \sigma_\beta, \mu_\gamma, \sigma_\gamma\}$ denote the global tree parameters, one can deduce the Gaussian distribution parameters θ_i of the rotation angles of each child segment b_{c_i} according to its label l_i. Reconstructing the 3D branches within the graphical modeling framework is described in the following sections.

4.5.1 Refined Markov Model and Factor Graph

The goal is to look for the optimal depth z_i of each branch b_i so that the transformation parameters have the largest probability according to the Gaussian distributions defined by the global parameter set Ω. As indicated

in Section 4.5, for a parent segment with K child branches, there are K sets of parameters θ_i with $i = 1,\ldots,K$ associated with the rotation angles. To differentiate between the type of child branches, another variable is introduced: The label for the child segment, l_i to get the distribution from Ω. This leads to a refined formulation of Eq. 4.15 with the joint probability of the two branch vectors $\mathbf{v}_i$, $\mathbf{v}_{p_i}$ and label l_i of the child segment:

$$f_i(\mathbf{v}_i,\mathbf{v}_{p_i},l_i|\Omega) \quad = \quad p(s_i|\Omega)p(\alpha_i|\theta_{l_i})p(\gamma_i|\theta_{l_i}) \tag{4.16}$$

allowing a refined factor graph, as seen in Fig. 4.10 c), introducing the label set $L = l_1,\ldots,l_N$ as an additional variable and an extended formulation of Eq. 4.13 with

$$p(Z,L|X,Y) = \prod_{i=1}^{N} f_i(\mathbf{v}_i,\mathbf{v}_j,l_i|\Omega). \tag{4.17}$$

The child segment label l_{c_i} between each pair of parent-child segments is introduced as another variable node to the factor graph. The square node f_{pc_i} between b_{c_i} and b_p represents the conditional joint probability defined in (4.16). By multiplying all factor nodes in the factor graph, one arrives at the posterior $p(Z,L|X,Y,\Omega)$ in Eq. 4.17 for the whole tree.

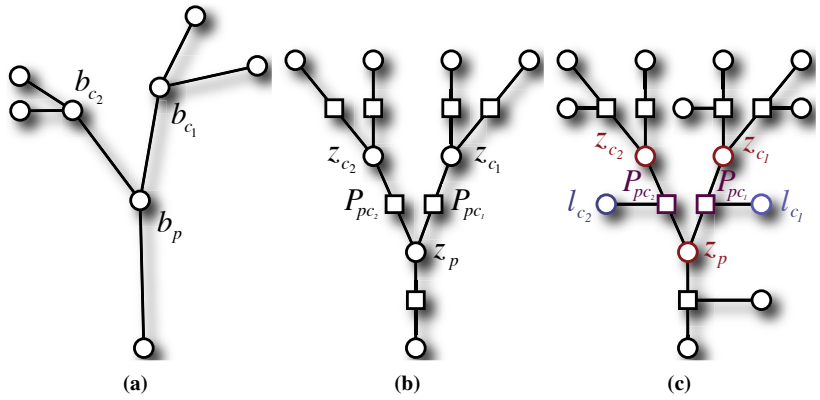


Figure 4.10: Mapping the sketch to a factor graph. (a) Markov Tree, (b) corresponding factor graph, (c) final factor graph with additional nodes for branch type label. Each node in the graph represents a branch segment in the sketch.

4.5.2 From Local to Global Coordinate Systems

The tree parameters Ω specify the rotation R_c^p between parent segment and child segment in the parent's local coordinate system while the observed (x,y) and unknown z are defined in the global coordinate system. To convert the coordinate systems from local to global:

$$\mathbf{v}_c = sR_{pc}\mathbf{v}_p = sR_p^0 R_c^p R_0^p \mathbf{v}_p, \tag{4.18}$$

where $R_{pc} = R_p^0 R_c^p R_0^p$, R_0^p is the rotation matrix from the global coordinate system to the local coordinate system of the parent segment and R_p^0 is the inverse is used.

The rotation angles α and γ of each pair of parent-child branch segments can then be deduced from the rotation matrix R_c^p.

4.6 Markov Tree Inference

Inference is the process of deducing the values of unknown variables of the graphical model in a way such that they pose the most likely configuration according to the probability distributions. By inferring the unknown variables of the branch segments, the 2D sketch is mapped to the 3D branches by relying on the global tree parameters Ω of the template selected from the database (Section 4.8). The default behavior of this system is to jointly estimate the depths of the branch segments *and* Ω, using the template values as initialization. As a result, in optimizing the tree shape based on the sketch, the final tree parameters may drift from those of the template. The user can choose to override this default behavior and ensure that Ω is preserved during the optimization. The characteristics of the final reconstructed tree would be the same as those of the template, but at the cost of a sub-optimal fit to the sketch.

Overriding the default behavior makes the optimization simpler, because Ω is unchanged throughout the optimization. This case is described first in the next section. Branch interaction (ensuring good spatial distribution and avoiding interpenetration) is accounted for in the optimization. The default system behavior is described in Section 4.6.2, where both tree shape and Ω are optimized in an EM-like (expectation-maximization) fashion (see system overview in Fig. 4.11).

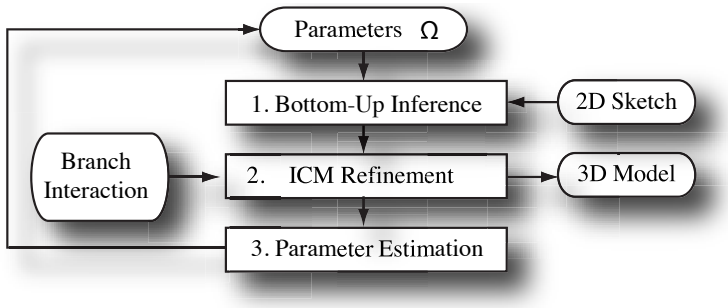


Figure 4.11: The framework of Markov tree inference.

4.6.1 Inferring Branches with Fixed Global Parameters Ω

In Eq. (4.17) the calculation of rotation angles between child and parent segment vectors depends on the rotation matrix $R^0_{p_i}$ linking local to global coordinate systems, as shown in Formula (4.18). Unfortunately, there is no closed-form solution for the objective function defined in Formula (4.17).

The global inference in Formula (4.17) can be approximated with a two-step approach. The first step is *bottom-up inference*, which computes a local solution at each generation of parent-child branches, starting from the root and ending with the terminal segments in the sketch. The second step refines the first step's results using the Iterated Conditional Mode (ICM) (see [Besag, 1986] [7] and [Bishop, 2006]).

[7] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302, 1986

BOTTOM-UP INFERENCE:

To handle the rotation chain in the tree, we use *ancestral sampling* [Bishop, 2006] to get the best sample of the unknown variables z_i and l_i of each branch b_i generation by generation, starting from the root segment. Generally, the root branch segment b_r's local coordinate system is consistent with the global coordinate system $R^0_r = I$ (identity 3×3 matrix) and $z_r = 0$. Then, we estimate the hidden variables of its descendant branch segments generation by generation. The rotation between the local and global coordinate systems of a child segment is propagated as $R^0_c = R^0_p R^p_c$.

Suppose a parent branch segment b_p has a segment vector $\mathbf{v}_p$ and local-to-global rotation R^0_p. Suppose, also, that b_p has K child branches b_{c_i} with $i = 1, \ldots, K$. For this generation, while fixing $\mathbf{v}_p$, accounting for the optimal values of z_{c_i} the label l_{c_i}, $i = 1, \ldots, K$ associated with the child branches are selected to maximize the posterior $\prod_{i=1}^{K} p(\mathbf{v}_{c_i}, \mathbf{v}_p, l_{c_i} | \Omega)$. The label l_{c_i} of all the child segments of b_p must be unique.

This can be done by analyzing all possible combinations in an exhaustive search. Testing all combinations of the labels and choosing the best solution at each generation gives the required results. The posterior in Formula (4.17) can be approximated by optimizing child branches from root to terminal nodes.

Figure 4.12 is a simple illustration of the local optimization. Here, the parent segment b_p has two child segments b_{c_1} and b_{c_2}, whose labels l_{c_1}, l_{c_2} and depths z_{c_1}, z_{c_2} are unknown (see Fig. 4.12 a)). There are two possible combinations of labels of b_{c_1} and b_{c_2} (Figure 4.12 b)). Under each hypothesis of segment labels, the corresponding parameter prior θ_1 and θ_2 for b_{c_1} and b_{c_2} are extracted from Ω as described in Section 4.5. For each child

segment, the optimal z_{c_i} to maximize the posterior $p(\mathbf{v}_{c_i}, \mathbf{v}_p, l_{c_i}|\Omega)$ given the hypothesized branch label l_{c_i} is searched for (Figure 4.12(c) shows the case for child segment b_{c_1}). Finally, the solution under the label hypothesis that maximizes $\prod_{i=1}^{2} p(\mathbf{v}_{c_i}, \mathbf{v}_p, l_{c_i}|\Omega)$ is chosen as the solution.

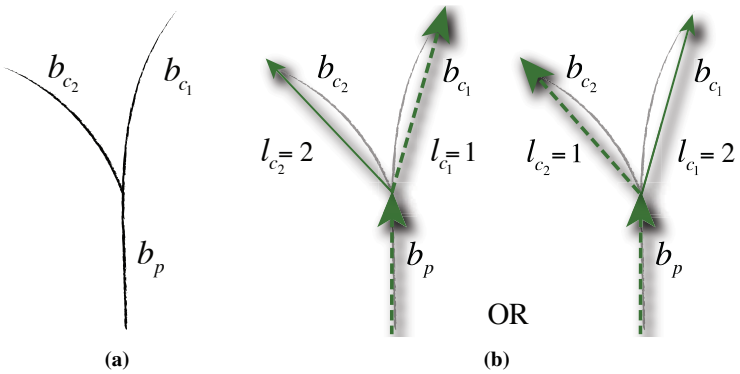


Figure 4.12: Illustration of optimization for one generation: (a) Sketch, (b) two hypotheses of labels for the child segments, and (c) given the hypothesis, search depth z (here, for b_{c_1}).

An initial 3D shape of the drawn branches is found by optimizing the unknown variables at each node from the root of the terminal nodes generation by generation. The next step is local refinement to account for branch interactions (competition between branches and avoiding interpenetration).

ICM REFINEMENT:

Since the process of bottom-up inference produces local solutions at every generation, following up with Iterated Conditional Mode (ICM) [Bishop, 2006, Besag, 1986] to refine the result is necessary. In order to avoid the complexity caused by the mapping between different branch coordinate systems, the depth z_i of each vertex is directly refined to make the divergence angle and scale of each pair of child and parent segment as consistent with the corresponding global parameters as possible.

More specifically, for a pair of parent-child segment b_i and b_{p_i}, the distributions of scale s_i with

$$s_i = |\mathbf{v}_i| / |\mathbf{v}_{p_i}| \tag{4.19}$$

and divergence angle

$$\alpha_i = \arccos\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_{p_i}}{|\mathbf{v}_i||\mathbf{v}_{p_i}|}\right) \tag{4.20}$$

are defined following $s_i \sim \mathcal{N}(\mu_s, \sigma_s)$ and $\alpha_i \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$ if $l_i = 1$, and

$\beta_i \sim \mathcal{N}(\mu_\beta, \sigma_\beta)$ otherwise (see Section 4.5). The probability considering these two items is defined as

$$p_{sa}(\mathbf{v}_i, \mathbf{v}_{p_i} | l_i, \Omega) = \prod_{i=1}^{N} p(s_i|\Omega) p(\alpha_i, l_i|\Omega). \tag{4.21}$$

During the growth of a tree, each branch competes with others to get as much space as possible. As a result, the tree branches typically distribute uniformly within the tree volume. We model this competition between branches as a probability field to affect the inference. The 3D space constrained within the crown is discretized to voxels (in experiments a grid of $N_v = 25 \times 25 \times 25$ is used), with each voxel's branch density being d_i at its center $\mathbf{p}_{v_i}$. The probability of a segment growing to the point $\mathbf{p}_i$ is

$$p_d(\mathbf{p}_i) = \prod_{i=1}^{N_v} \exp\left(\frac{-k_e d_i}{1 + \|\mathbf{p}_i - \mathbf{p}_{v_i}\|^2}\right) \tag{4.22}$$

where k_e is the *branch interaction factor*. Its default value is $k_e = 5$. By encouraging the branch to grow to the position where the density of the branches is low, the branches attempt to get as much free space around them as possible.

At each step of ICM, we update one node by

$$z^{(t+1)} - z^{(t)} = \left.\frac{\partial p_{ICM}}{\partial z_i}\right|_{\{\mathbf{v}_j\}, j=1,\ldots,N, j\neq i} \tag{4.23}$$

while fixing the other nodes to increase

$$p_{ICM} = \prod_{i=1}^{N} p_{sa}(\mathbf{v}_i, \mathbf{v}_{p_i} | l_i, \Omega) p_d(\mathbf{p}_i). \tag{4.24}$$

The refinement terminates when the changes in the nodes fall below a predefined threshold $\sum_{i=1}^{N} |z_i^{(t+1)} - z_i^{(t)}| < \varepsilon$ ($\varepsilon = 0.01$).

INTERPENETRATION BETWEEN BRANCHES

During the inference of the depths of branches, branch interpenetration is avoided if possible. Interpenetration is checked for right after the bottom up inference and affected branches are moved away from each other along the z-axis. The ICM refinement step is then applied to make the branching shape more consistent.

4.6.2 Inferring Branches Positions and Global Parameters Ω

The default behavior of the system is to jointly optimize Z, L, and the global tree parameter Ω to better fit the current sketch. An EM-like (expectation-maximization) algorithm can be introduced to simplify the inference. In the expectation step, the integral of all hidden variables (Z and L) is computed to estimate the parameters Ω, since the primary interest is to find the optimal hidden variables rather than parameter estimation, and to modify the expectation step to reflect this.

At iteration t, instead of evaluating the expectation given parameters $\Omega^{(t)}$, the optimal hidden variables $Z^{(t)}$ and $L^{(t)}$ of all the branches that maximize the posterior $p(Z^{(t)},L^{(t)}|X,Y,\Omega^{(t)})$ (*E-step*) are deduced. The maximization step uses $Z^{(t)}$ and $L^{(t)}$ to estimate $\Omega^{(t)}$ by maximizing $p(\Omega|Z^{(t)},L^{(t)},X,Y)$ (*M-step*):

$$p(\Omega|Z^{(t)},L^{(t)},X,Y) = \prod_{i=1}^{N} p(\Omega|s_i,\alpha_i,\beta_i,\gamma_i). \tag{4.25}$$

At the *E-step*, $Z^{(t)}$ and $L^{(t)}$ are computed using fixed parameters $\Omega^{(t)}$ in the same way as described in Section 4.6.1. At the *M-step*, the four parameters of the child branches are mapped to the Gaussian distribution defined by Ω, that is $(s_i,\alpha_i,\beta_i,\gamma_i) \sim \{\mathcal{N}(\mu_s,\sigma_s),\mathcal{N}(\mu_\alpha,\sigma_\alpha),\mathcal{N}(\mu_\beta,\sigma_\beta),\mathcal{N}(\mu_\gamma,\sigma_\gamma)\}$. The tree parameters are updated $(\Omega^{(t)} \rightarrow \Omega^{(t+1)})$ as the average and variance of the branch rotation angles and scale at iteration t.

During experiments fewer than 10 iterations are required to generate good results. This allows for reaction times of the system at interactive speed. The results show that the reconstructed 3D tree model is plausible even though it is a local solution. After depths of all the branch segments have been recovered, cubic B-spline curves are used to interpolate the shape of the branches and generate meshes according to Neubert *et al.* [Neubert et al., 2007] [8] (and described in detail in Chapter 5).

[8] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:88:1 – 88:10, July 2007

4.7 Branch Propagation and Leaf Population

The sketch drawn by the user is typically sparse, because humans tend to abstract and reduce the information to the necessary. The resulting 3D model associated with the sketch alone is unlikely to look compelling due to the lack of complexity. One option to overcome this oversimplification would be for the user to draw a complicated tree structure, but this would be manual-intensive. While this option is supported by the proposed system, there is a

better way: the system automatically adds complexity to the model based on what has already been drawn.This automatic feature is based on the principle of self-similarity which is widely used in plant modeling [Shlyakhter et al., 2001, Tan et al., 2007] [9] . The system randomly picks a replication block (a parent segment and its child segments) from the model, picks an open branch segment, scales, and orients the replication block, and attaches the transformed replication block to the open branch.

The growth of branches is limited by the crown—more specifically, its surface of local revolution (see [Okabe et al., 2005] for details). If the new propagated branches reach this surface, they are scaled back to touch the surface, and the propagation terminates there. Note that if the crown is not drawn by the user, the system propagates the branches by a fixed number of generations, usually fixed to five generations, which produces good results. The user can specify a different number if desired. Figure 4.13 shows branch propagation results with and without crown. By drawing the crown, the user exerts more control over the shape of the tree.

[9] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21:53–61, May 2001; and Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:87:1 – 87:8, July 2007

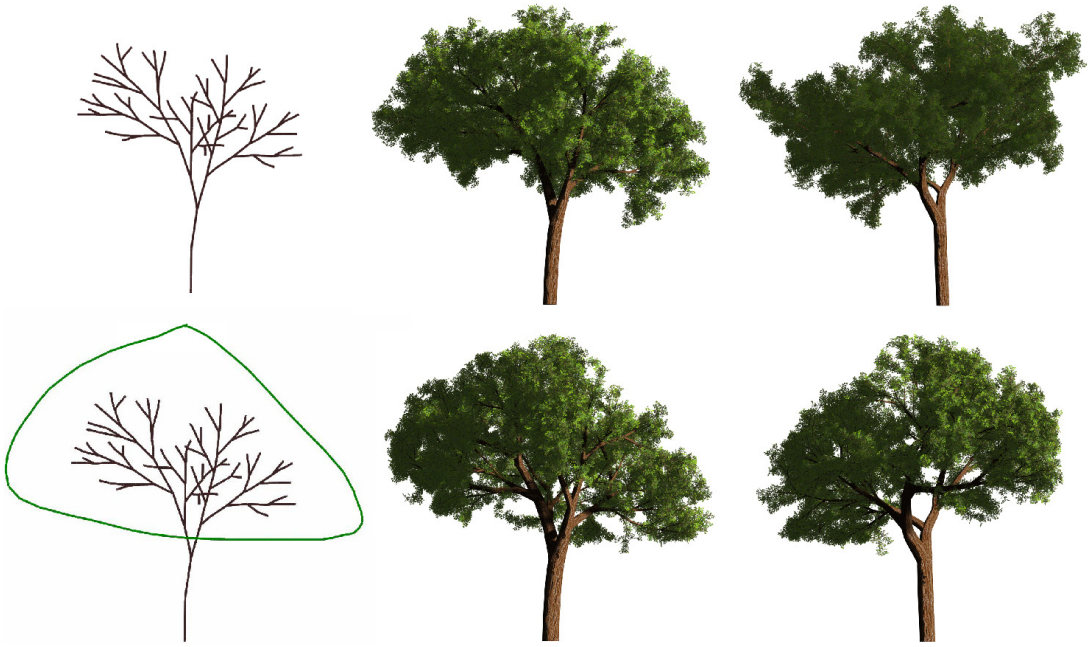


Figure 4.13: Effect of additional crown definition. Top row: without crown, bottom row: with crown. From left to right: sketch, two views of the complete tree model.

Generally, the casual user does not draw strokes with similar scales in two successive generations. This results in branch segments that are much longer than their parent segments. To make the tree appear more natural, the system introduces new lateral branches along such disproportionately long branches. By propagating a fixed number of generations, the system is able to produce a 3D tree model with reasonable complexity and realism.

The branch growing procedure is similar to the hidden branch construction algorithm described in [Tan et al., 2007], but with two important differences. First, they compute the 3D hull from multiple photos, while ours is generated from a 2D curve. Second, in their system, the branch growth is achieved by *randomly orienting* branches before adding them to the tree. In our system, the new branches inherit the branch angle parameters associated with the replication blocks, thus fixing their orientations with respect to the open branches. To make the tree appear more natural, our system also introduces new lateral branches along disproportionately long branches caused by an inaccurate sketch.

The leaves and twigs (small branches) attach to large branches in regular and predictable ways. Instead of generating the leaf geometry from a sketch, the system uses the leaf model associated with the chosen database template by default (Section 4.8). The user can override the default leaf model by selecting a different one from any other tree exemplar in the database. The leaves are replicated and placed on branches based on botanical laws governing leaf arrangement. For example, the leaves on the same twig can be directly opposite or alternate with a deviation angle.

In the following section the tree database is described, which is used to supply an appropriate set of tree parameters Ω as prior to the graphical modeling framework.

4.8 Database of Tree Templates

The chances of producing natural-looking tree models are enhanced if geometry reconstruction is guided by pregenerated natural-looking ones. Such models can be obtained through image-based modeling [Reche-Martinez et al., 2004, Tan et al., 2007] or through manual modeling [Lintermann and Deussen, 1999] [10] . The proposed tree modeling technique does not rely on the method through which the tree models in the database are generated; instead the technique adapts to whatever tree parameters are supplied and what source the parameters are coming from. In the current implementation tree models are used that were generated using the method described in [Neubert et al., 2007] as templates. There are 20 different tree exemplars in the database; chosen to represent a reasonably wide variety of trees.

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

The exemplar that best matches the sketch is chosen from the system as the template for 3D reconstruction. The selection is achieved by comparing the crown and branching shapes of the sketch with the 2D silhouettes and projected branching shapes of exemplars in the database.

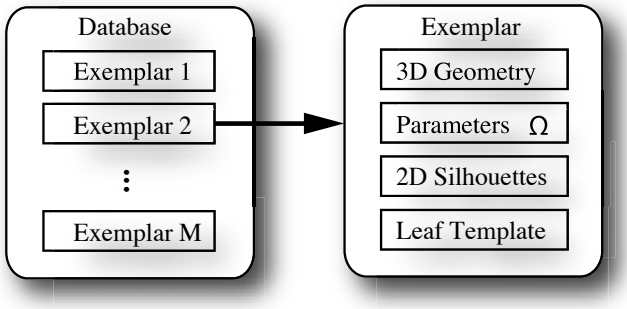


Figure 4.14: Structure of tree database.

The user may draw the crown as a single curve or series of curves (shown in Figure 4.15 as solid green curves). The system then computes the convex hull (red dashed lines) of both the crown and branch strokes except the root branch.

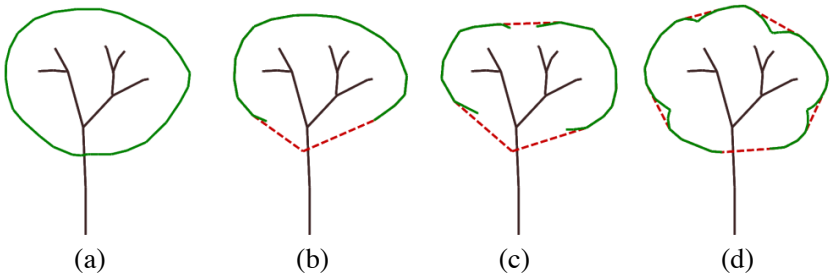


Figure 4.15: Different crown shapes drawn as user input (green solid curve): (a) complete convex crown; (b) incomplete convex crown; (c) incomplete crown with several strokes; (d) concave crown with several corners; The red dashed lines are the part added to the complete convex crown shape.

A footprint descriptor [Lamdan et al., 1988] [11] of this convex hull is used during the template selection process. It is a simple but robust method to measure the similarity of curves. First, the crown curve is normalized such that $\max(w,h) = 1$, where w,h are the width and height of the crown's bounding rectangle. Let the center of the normalized curve be C. We then compute the radial distance distribution from C to the curve, yielding a K_f dimensional vector $\mathbf{f}$ (for K_f regularly sampled directions). $\mathbf{f}$ is the footprint descriptor. Each tree exemplar is associated with a set of 2D silhouettes under different views. We compute the footprint descriptor $\mathbf{f}_E$ for each silhouette in the same manner. The similarity of the crown shape between the sketch S and the exemplar E under a view v is defined as

$$L_c(S,E|v) = -\|\mathbf{f}_S - \mathbf{f}_{E|v}\|. \tag{4.26}$$

To measure the similarity between the branching shapes of the sketch and exemplar, the average 2D length ratio and angle between each pair of parent-child segments of tree is calculated. For a parent segment in the 2D sketch,

[11] Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. Object recognition by affine invariant matching. In *Computer Vision and Pattern Recognition (Proc. CVPR '88)*, pages 335–344, 1988

it is not clear which child segment is apical or lateral. The system assumes that the child segment with the smallest 2D angle with the parent segment is apical, while the rest are assumed to be lateral. Assuming the length ratio s_{2D}, apical angle α_{2D}, and lateral angles β_{2D} of the 2D branches are Gaussian distributed, their mean and standard deviation: $x_{2D} \sim \mathcal{N}(\mu_{x,2D}, \sigma_{x,2D})$ are estimated, where $x = s, \alpha, \beta$. For each exemplar in the database, its 3D geometry is projected to 2D under different views. For each 2D projection, the Gaussian distributions of length ratio $s_{e,2D}$, apical angle $\alpha_{e,2D}$, and lateral angle $\beta_{e,2D}$ between each pair of parent-child segments are similarly estimated: $x_{e,2D} \sim \mathcal{N}(\mu_{x,e,2D}, \sigma_{x,e,2D})$, $x = s, \alpha, \beta$. The similarity of the branching shapes between the sketch and the exemplar under view v is defined as

$$L_b(S,E|v) = -\sum_{x=s,\alpha,\beta} \frac{(\mu_{x,2D} - \mu_{x,e,2D|v})^2}{\sigma_{x,2D}^2 + \sigma_{x,e,2D|v}^2}.$$

If the user draws the crown, the similarity between the sketch and exemplar (from a given view v) is the sum of similarities in crown and branching shapes:

$$L(S,E|v) = L_c(S,E|v) + L_b(S,E|v). \tag{4.27}$$

If the crown is not drawn, the similarity is set as $L(S,E|v) = L_b(S,E|v)$. The exemplar is selected for which similarity between one of its views and the sketch (given by $L(S,E|v)$) is maximized. This exemplar (with parameters Ω) will be the default template for the shape inference.

User Selection

Alternatively the user may choose to override the default by simply clicking a thumbnail to select an exemplar.

Figure 4.16 shows the results generated from the same sketch but with different exemplars. This illustrates that the system is capable of generating models that can look radically different, depending on which tree exemplar is chosen.

Note that the selected exemplar serves to provide a good initial point of the tree parameters; the final parameters inherit tree characteristics from *both* the selected exemplar and the drawn sketch. Therefore, perceptually different 3D tree models can be generated from different sketches based on the same exemplar and vice versa.

Biological Parameters

The tree exemplars serve to add realism to the output, but are not absolutely necessary. It is possible to just have preset tree parameters which are then used for all sketches. However, the resulting tree shapes may not appear

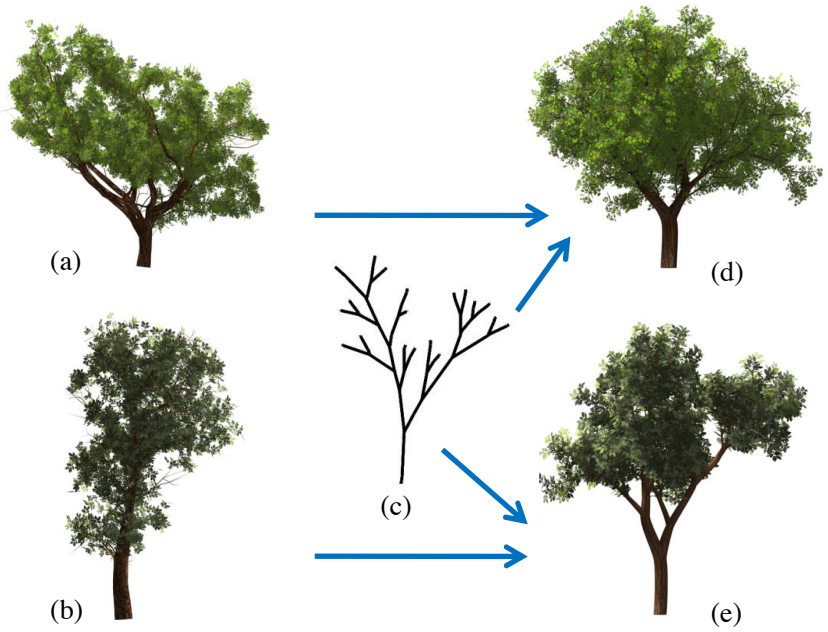


Figure 4.16: Different results generated from the same sketch, but with different manually assigned exemplars. (a)(b) are two different exemplars used to guide the inference; (c) is the input sketch; (d)(e) are corresponding output tree models.

as compelling. As with most data-driven recovery systems, the more tree exemplars that are available (with a wider variation of shapes), the better the results are expected to be.

4.9 Results and Discussions

Figure 4.17 shows a variety of results generated from sketches of different trees. The complexity of the sketch ranges from a small number of drawn strokes to a large number, with and without the optionally sketched crown. As shown in (a), a complex and reasonably realistic-looking tree can be generated from just 8 strokes. Of course, the user can better control the specific shape of the tree by drawing more branches and the crown, as can be seen in (b). Figure 4.17(c), (d) and (e) show three other tree models generated from sketches with varying numbers of strokes, with and without the crown.

Generally, the user can create visually compelling tree models at interactive rates using the proposed system (for a 2.67 GHz PC). The reconstruction of simply sketched strokes is accomplished interactively. In case of incomplete or very sparse sketches realistic tree models can be produced, as shown in Figure 4.17 (a) and (b) after few additional propagation steps. The user has the option of generating more complicated tree models with more branches, at the expense of longer propagation time. For example, the propagation of

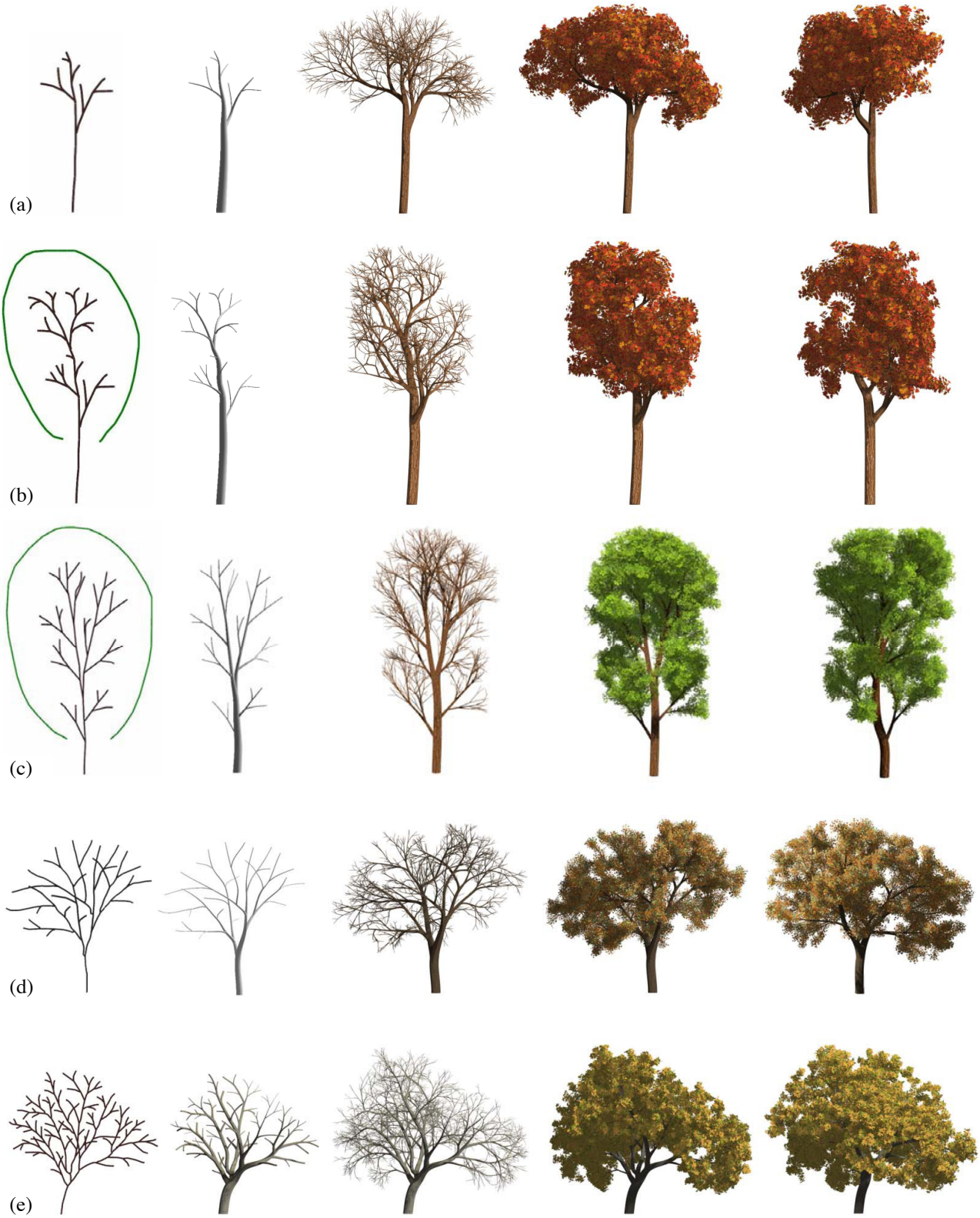


Figure 4.17: A variety of results generated from sketches of different trees. From left to right: sketch, after branch reconstruction and propagation, and two views of the complete tree model.

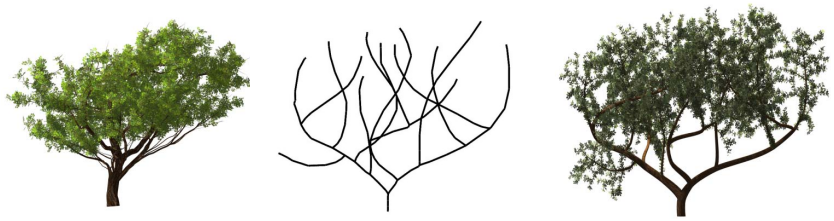


Figure 4.18: Another comparison between the exemplar and resulting tree model. From left to right: Exemplar automatically selected by our system, input sketch, and output tree model.

Figure 4.19: Failure example. From left to right: sketch, two views of the reconstructed branches.

Figure 4.17 (e) took about 50 seconds to produce 4,291 branch segments.

The system is able to generate complex and realistic-looking trees having distinct branching structures from freehand sketches and is able to cope with irregular shapes (Figure 4.18). There are significant differences between the exemplar and the output. However, it is not easy for our system to model the tree with curvy branches such as *liana* or *calyx canthus*. Figure 4.19 shows such an example. The user intended to draw a tree with a curvy main branch growing along one direction. However, the optimization objective was designed to make the branches consistent throughout the whole tree and to make them spread out. As a result, the generated 3D model does not look very natural (as seen from a side view).

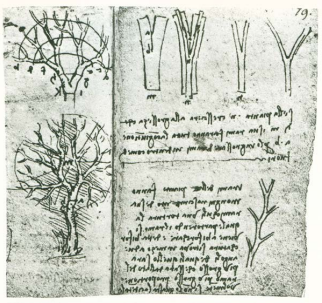
In the current implementation, the system can generate trees with a particular type of branching structure. While many tree species can easily be approximated by exploiting the self-similar property of trees, others such as palm trees or spruces are hard to create. It is possible to modify the system to accommodate spatially-varying parameter sets or branching structures, but this is a topic for future work. The cost would be a more complicated interface which presents the user with more (potentially domain-specific) options.

There are several other possible extensions to the system. It currently has no editing options—it would certainly be useful for the user to be able to interactively rotate the completed tree and edit different views whenever necessary. Editing operations include branch removal or addition and depth changes in combination with interactive and automatic updates through the system. Furthermore, in our current implementation, the propagation of thousands of branches could not be finished at interactive rates. In the future, the implementation could be optimized to achieve interactive propagation.

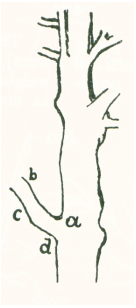
The current version of the system does not provide the option for specifying environmental effects to influence the shape of the tree model. It may, in certain cases, be desirable to be able to specify obstacles (e.g., buildings) for the tree to "grow" around. Another interesting case is to generate models of several trees in close proximity to each other, with each influencing the development of others.

5

From Graphs to Models



(a)



(b)

Figure 5.1: Leonardo da Vinci's observations about branch radii [Plate XXVII from Richter, 1970, p. 115].

There are several ways to faithfully reconstruct or generate the 3D branching graph of a tree model. This chapter will describe another important part in generating realistic models: the step from branching graphs to triangle based representations of the model. The following sections describe how the final geometric representation is generated based on the branching information given in form of a branching graph and can be described in three steps. In the first section (Sec. 5.1) allometric rules are described to deduce realistic branch radii. Section 5.2 will introduce a method to generate a water-tight meshing of the brach geometry and bifurcations. Finally, Sec. 5.4 focuses on the generation of model details including small twigs and leaves according to botanical principles.

5.1 Allometry

The first step to generate a mesh from a branching graph is to deduce the radii of the individual branches.

Leonardo da Vinci's (1453-1519) notebook [Richter, 1970] [1] is the first written record of the relationship of branching radii. He states that the sum of the area of two branches equals the area of the parent branch (see Fig. 5.1):

> All the branches at every stage of its height when put together are equal in thickness to the trunk [below them].
> [Richter, 1970, p. 205, Sec. 394f]

Da Vinci formalized the simple mathematical relation of the diameters as

$$d_0{}^2 = d_1{}^2 + d_2{}^2. \tag{5.1}$$

[1] Jean Paul Richter. *The Notebooks of Leonardo da Vinci*, volume 1. Dover Publications Inc, New York, 1970. reprinted 1888

This simple relation proofed to capture the real world surprisingly well and was later basis for the so called *pipe model* of plants [Shinozaki et al., 1964a,b] [2] that still is used to explain certain properties of plants.

pipe model of plants

[2] K. Shinozaki, K. Yoda, K. Hozumi, and T. Kira. A quantitative analysis of plant form - the pipe model theory I. Basic analysis. *Japanese Journal of Ecology*, 14: 97–104, 1964a; and K. Shinozaki, K. Yoda, K. Hozumi, and T. Kira. A quantitative analysis of plant form - the pipe model theory II. Further evidence of the theory and its application in forest ecology. *Japanese Journal of Ecology*, 14:133–139, 1964b

Theoretical and experimental biology discovered many more such relations that are described with the classical allometric equation [Niklas, 1994, pp.123][3] (*power function equation*):

$$Y_1 = \beta {Y_2}^{\alpha} \tag{5.2}$$

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

power function equation

here Y_1 is the variable of interest and Y_2 is variable measuring size, α and β are parameters describing the mathematical relationship between the two variables. Transforming Eq. 5.2 into linear form gives

$$\log Y_1 = \log \beta + \alpha \log Y_2. \tag{5.3}$$

In linear form it can be seen that α is the slope (and therefore called *scaling exponent*) and β is the value of Y_1 if $Y_2 = 1$ (*scaling coefficient*). Some of the allometric relations based on this power law are found through theoretical assumptions based on fluid simulations or material properties, others are found empirically through regression analysis.

Murray [1927][4] analyses an allometric relation closely related to Leonardo da Vinci's original observation. Murray investigates the link between circumference of branches of a ramification. Based on sampled circumference c and weight w of the branches at a branching point Murray empirically establishes the following relation

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

$$\begin{aligned} \log w &= 0.850 + 2.49 \log c \\ w &= 7.08\, c^{2.49}. \end{aligned}$$

Based on the fact that the weight w_0 of the branch before the bifurcation equals the sum of the weights of both subbranches $w_1 + w_2$ (neglecting the weight of the bifurcation itself) Murray establishes the following equation

$$\begin{aligned} w_0 &= w_1 + w_2 \\ {c_0}^{2.49} &= {c_1}^{2.49} + {c_2}^{2.49}. \end{aligned}$$

These observations are closely related to the dimensionality of fractals. Benoit Mandelbrot [1983, Chapter 17][5] mentions Murray's study in the chapter about natural branching systems as found in lungs, vasculatures, botanical trees, and river networks.

[5] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983

While traditionally the dimensionality of a line is thought to be one-dimensional, an area two-dimensional and a volume three-dimensional, fractals define a non-integer dimensionality as

$$D = \frac{\log(\text{number of self-similar pieces})}{\log(\text{magnification factor})}$$

or related to the Hausdorff dimension

Hausdorff dimension

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log 1/\varepsilon}$$

where $N(\varepsilon)$ is the number of self-similar elements of size ε that are needed to represent the original structure.

For botanical trees Mandelbrot defines, besides the fractal dimension D (which should be close to $D = 3$ to be nearly space filling), a second parameter Δ called *diameter exponent*. Tree structures that are highly self-similar as for example the lung's bronchial tree exhibit $\Delta = D \approx 3$, essentially filling the outline of the complete lung. According to Mandelbrot the best one can hope for botanical trees is that the self-similarity assumption holds for the branch tips (defining the fractal dimensionality D). Leonardo da Vinci's observation is suggesting $M = \Delta = 2$ while Murray's empirical evidence suggests $M = \Delta = 2.5$. Mandelbrot suggests a value of $M = 2 + \frac{\Delta}{D}$ with $\Delta = 2$ and $D = 3$ and explaining that D and Δ take up the integer Euclidean dimensions of solids and surfaces according to

self-similarity assumption

> The problem of energy interchange in trees can be simplified by considering the tree as a system in which as large [an area] as possible must be irrigated with the minimum production of volume while at the same time guaranteeing the evacuation of absorbed energy.

Mandelbrot finds following corollaries for the assumption that $\Delta = 2$ and $D = 3$ important, considering realistic plant models:

- Branch's leaf area is proportional to both the volume of the branch's outline and the cross-sectional area of the branch.
- The ratio between tree height h and trunk diameter d is constant: $const. = \frac{h^3}{d^2}$.
- Leaf and branch area are proportional to tree height cubed.

Using these functional proportions can greatly improve the esthetic quality of the final model.

The next important step is to make qualified assumptions about the ratio of the radii (in case of a bifurcation) of the two child branches. The intuitive assumption is to use the ratio of the branching angles to model the ratio of the radii with

$$\frac{d_1}{d_2} = \frac{\alpha_1}{\alpha_2}$$

Based on empirical evidence Murray [1926] investigates the relation between branching angles and radii of vessel systems and established following relationship

$$\cos \alpha_1 = \frac{{r_0}^4 + {r_1}^4 - {r_2}^4}{2{r_0}^2{r_1}^2} \tag{5.4}$$

$$\cos \alpha_2 = \frac{{r_0}^4 + {r_2}^4 - {r_1}^4}{2{r_0}^2{r_2}^2} \tag{5.5}$$

$$\cos (\alpha_1 + \alpha_2) = \frac{{r_0}^4 - {r_1}^4 - {r_2}^4}{2{r_1}^2{r_2}^2} \tag{5.6}$$

with parameters defined as in Fig. 5.2. Even though Eq. 5.6 are found based on vessel systems they provide an interesting basis for plant models as well.

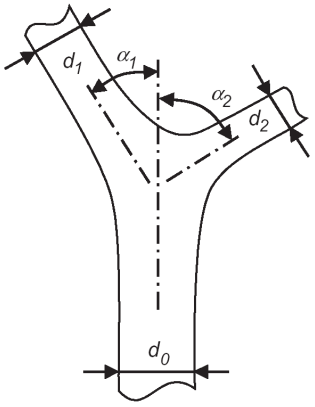


Figure 5.2: An important step generating the final model is to deduce branch radii according to botanical principles. Allometric rules allow to define the relationship between branch radii d_0, d_1, d_2 and bifurcation angles α_1 and α_2.

Holton [1994][6] makes an comparable observation for the strands modeling system introducing this relationship to computer graphics. Here the underlying assumption is that a tree is made of several strands that are not further divisible and are connecting the roots to the leaves without further branching. Counting the number of strands that run trough a branch can be used to determine the radius of the branch. Holton establishes the relation between number of strands S_i and branching angle α_i with

$$\alpha_1 = \frac{S_2}{S_0} A$$
$$\alpha_2 = A - \alpha_1$$

[6] Matthew Holton. Strands, gravity and botanical tree imagery. *Computer Graphics Forum*, 13:57–67, February 1994

with A as sum of both branching angles and S_0 number of strands of the main branch. Solving for S_2 and S_1 gives

$$\begin{aligned} S_2 &= \frac{\alpha_1 S_0}{A} \\ S_1 &= S_0 - S_2. \end{aligned}$$

The diameter of the branch d than can be found with $d = t\sqrt{S}$ with t being a plant depending scaling parameter.

5.2 Branch Geometry

All described modeling methods have in common that at one point in the modeling stage the intermediate data structure is a graph representing the branching skeleton. The branching graph G is defined as the tuple of a set of edges E and a set of nodes N representing a directed graph with no cycles

$$G = (N, E). \tag{5.7}$$

Each edge $e \in E$ is an ordered pair of vertices $e = (V \times V)$. Each node $n \in N$ is associated with the real world position v of the vertex and the radius r of the branch. This section deals with the mesh generation that allows to render the final model based on a given branch skeleton.

5.2.1 Open Uniform B-splines

Using B-splines as approximation of individual branch sections increases the realism of the model and allows for a parameterized representation of the whole branch (see Fig. 5.4).

Uniform B-splines are a weighted sum of control points P_i

$$P(u) = \sum_{i=0}^{n} P_i N_{n,i}(u) \tag{5.8}$$

and the weight functions N are recursively defined ($p = 4$ for cubic splines) as

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1, & u \in [u_i, u_{i+1}[\\ 0, & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+q} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned}$$

with local support

$$N_{i,p}(u) = 0 \quad u \notin [u_i, u_{i+p}[.$$

For cubic B-splines (p=4) the weight functions for one section of the spline $N_{i,4}$ is defined as the union of four parts b_0, b_1, b_2, and b_3 with the following properties [Salomon, 2005, pp.256][7]:

- Provide C^2 continuity at the three points where they join.
- $b_0(t)$ and its first two derivatives should be zero at $b_0(0)$.
- $b_3(t)$ and its first two derivatives should be zero at $b_3(1)$.

[7] David Salomon. *Curves and Surfaces for Computer Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005

Defining $b_i(u) = A_i u^3 + B_i u^2 + C_i u + D_i$ the above conditions result in

$$b_0(u) = \frac{1}{6}u^3 \tag{5.9}$$

$$b_1(u) = \frac{1}{6}(1 + 3u + 3u^2 - 3u^3) \tag{5.10}$$

$$b_2(u) = \frac{1}{6}(4 - 6u^2 + 3u^3) \tag{5.11}$$

$$b_3(u) = \frac{1}{6}(1 - 3u + 3u^2 - u^3) \tag{5.12}$$

and in general the Cubic B-Spline kth single segment formulation:

$$C_k(u) = \sum_{i=0}^{3} P_{k-3+i} N_{k-3+i,3,\tau}(u), u \in [0,1] \tag{5.13}$$

$$C_k(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_{k-1} \\ P_k \\ P_{k+1} \\ P_{k+2} \end{bmatrix}. \tag{5.14}$$

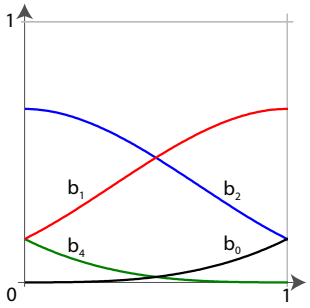


Figure 5.3: B-spline basis functions

5.2.2 *Curve Framing*

Defining the geometry around the graph structure requires to define a local coordinate system for each point on the graph or curve – a so called moving coordinate frame. This allows for defining a *generalized cylinder* with arbitrary cross-sections along the curve axis with each cross-section properly aligned with its neighbors to avoid twists. *Frenet frames* (see [Salomon,

Figure 5.4: (a) Detailed view to chiseled and smoothed branches; (b) complete tree skeleton of an oak tree

2005]) are a convenient choice, because they are defined analytically in the following way:

the first derivative $P\prime(u)$ of a curve $P(u)$ has the simple geometric meaning of a *tangent vector* of the curve

$$T(u) = \frac{P\prime(u)}{|P\prime(u)|} \tag{5.15}$$

while the second vector to define the Frenet frame is the *principal normal vector* $N(u)$ which points towards the center of curvature and is defined as

$$N(u) = \frac{K(u)}{|K(u)|} \tag{5.16}$$

with

$$K(u) = \frac{P\prime\prime(u) \cdot P\prime(u)}{|P\prime(u)|^2} P\prime(u) \tag{5.17}$$

and finally the *binormal vector* $B(u)$ orthogonal to both $N(u)$ and $T(u)$

$$B(u) = T(u) \times N(u). \tag{5.18}$$

However, from the definition one can see that in the case of N vanishing the Frenet frame is undefined and even worse whenever the curvature vector reverses direction causing twists in the orientation of consecutive frames (see Fig. 5.5).

Frenet frames

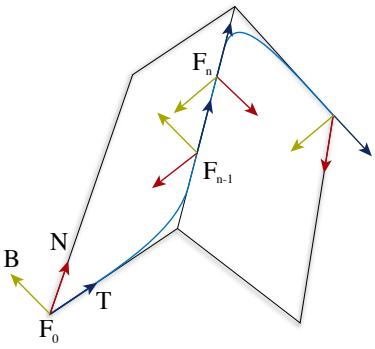


Figure 5.5: Problematic behavior of Frenet frames at saddle points: The reversed direction of normal N from frame F_{n-1} to F_n would cause twists in the resulting mesh.

A better choice therefore are *parallel transport frames* (see [Hanson et al., 1995] [8]) minimizing torsions of consecutive frames. Such an improvement is achieved by propagating an initial frame along a curve using small, local rotations, at the cost of losing the analytic properties of the frame.

[8] Andrew J Hanson, Hui Ma, and Lindley Hall. Parallel transport approach to curve framing. *Indiana University Techreport-sTR425*, pages 1–20, 1995

The initial frame at the start of the curve can be found using curvature as done for the Frenet frame. In the case of zero curvature choosing N_0 as any perpendicular vector to T_0. Given this initial frame consecutive frames are computed iteratively based on the last frame by computing P_{i+1} and T_{i+1} at the new point on the curve and rotating the old reference frame, such that T_i aligns with T_{i+1}. Rotating N_i and B_i in the same manner creates the new reference frame at P_{i+1}.

parallel transport frames

The rotation axis A is given by the normal vector of the plane defined by two consecutive tangential vectors of the curve

$$A = \frac{T_i \times T_{i+1}}{\|T_i\|\|T_{i+1}\|} \tag{5.19}$$

and the rotation angle is given by

$$\alpha = \arccos\left(\frac{T_i \cdot T_{i+1}}{\|T_i\|\|T_{i+1}\|}\right). \tag{5.20}$$

If $T_i = T_{i+1}$ the rotation angle α becomes zero, although the cross-product and accordingly the rotation axis A is undefined.

With $B_i = T_i \times A$ and $B_{i+1} = T_{i+1} \times A$, the intermediate frames can be represented as rotation matrices

$$M_i = \begin{pmatrix} T_i^x & A^x & B_i^x \\ T_i^y & A^y & B_i^y \\ T_i^z & A^z & B_i^z \end{pmatrix} \tag{5.21}$$

and

$$M_{i+1} = \begin{pmatrix} T_{i+1}^x & A^x & B_{i+1}^x \\ T_{i+1}^y & A^y & B_{i+1}^y \\ T_{i+1}^z & A^z & B_{i+1}^z \end{pmatrix} \tag{5.22}$$

where T^x is the x component of vector T.

The rotation matrix R_i between Frame i and $i+1$ becomes

$$R_i = M_i^{-1} M_{i+1} = M_i^T M_{i+1} \tag{5.23}$$

since M_i is an orthonormal rotation matrix.

The normal N_i and binormal B_i at position P_i on the curve can then be used to compute the global coordinates of points on the cross-section with local coordinates $C = (C^x, C^y)$, with $\|C\| = r_i$ the branch radius of the curve section, in the following way:

$$P_{surface} = \begin{pmatrix} C^x & C^y & 1 \end{pmatrix} \begin{pmatrix} N^x & N^y & N^z \\ B^x & B^y & B^z \\ P^x & P^y & P^z \end{pmatrix}. \tag{5.24}$$

Constructing the mesh from the cross-section points to form generalized cylinders is performed in a straight forward way. Let P_i^0 to P_i^n be n ordered points along the branch cross-section corresponding to node n_i on the curve. Then, to form the cylinder from node n_i to n_{i+1}, we generate a triangle for $j = 0$ to $n-1$ for each three points $P_i^j, P_i^{j+1}, P_{i+1}^j$ and $P_{i+1}^j, P_{i+1}^{j+1}, P_i^{j+1}$.

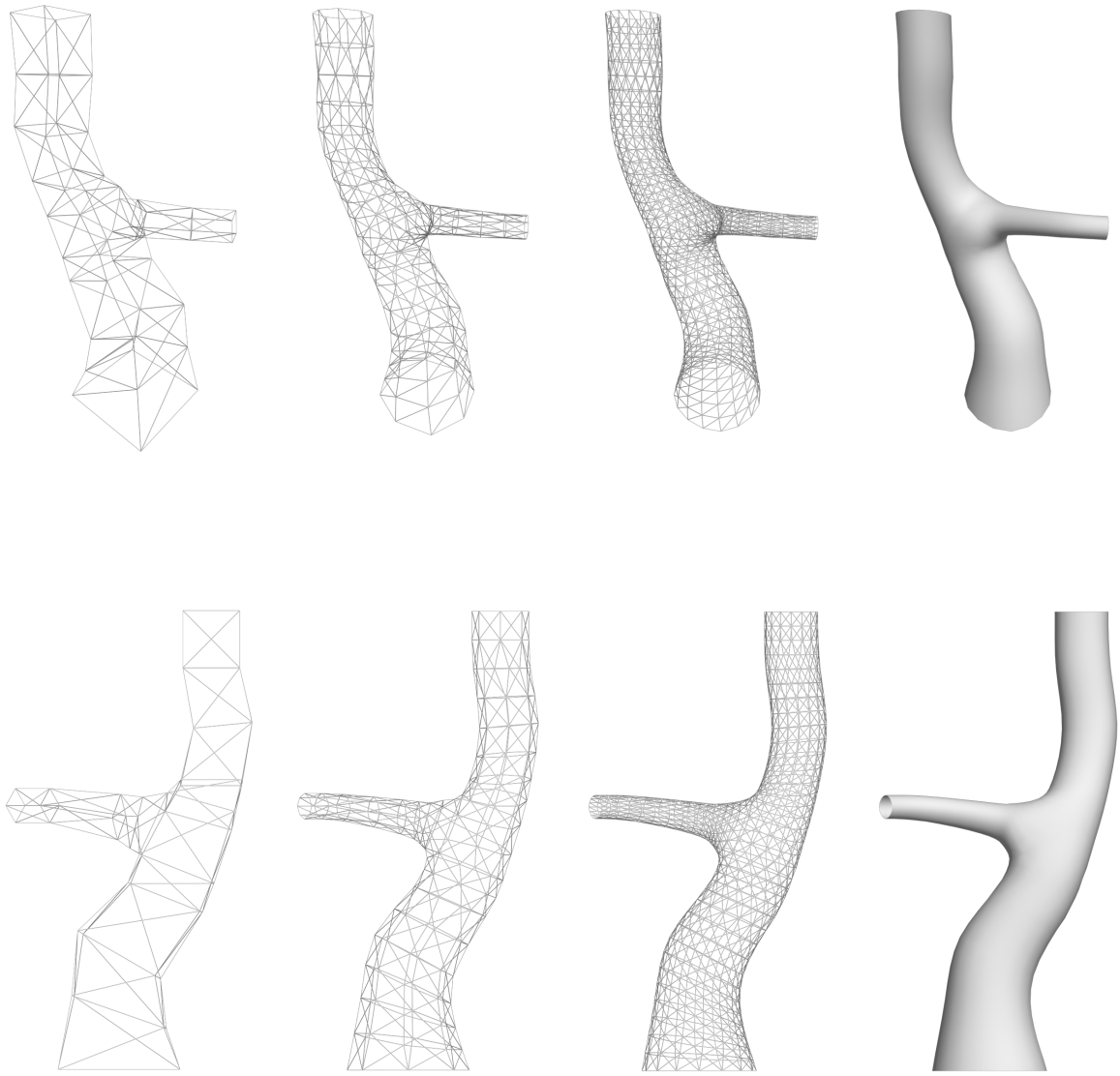
5.3 Meshing Bifurcations

Bifurcations are often modeled using generalized cylinders that intersect each other without paying attention to produce closed meshes. For most parts in a tree model this method is sufficient without introducing noticeable artifacts. However, for bifurcations of branches with large radii or bifurcations of the trunk, intersecting triangles may result in visible artifacts during rendering.

This section will provide a new method to generate closed meshes of bifurcations within a tree model, that can be refined using standard re-meshing techniques [Biermann et al., 2000] [9] to produce high quality geometric representations.

[9] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proc. of SIGGRAPH '00*, pages 113–120, New York, NY, USA, 2000. ACM Press

At bifurcation points the endpoints of different branches need to be connected. A minimal distance of the branch end points need to be guaranteed to prevent surface polygons of different branches to intersect. This minimal distances depends on the radii of the branches at the bifurcation together with their branching angles. This minimal distance can then be used to cut the branches before generalized cylinders are constructed and the void space is used to generate the mesh of the bifurcation.

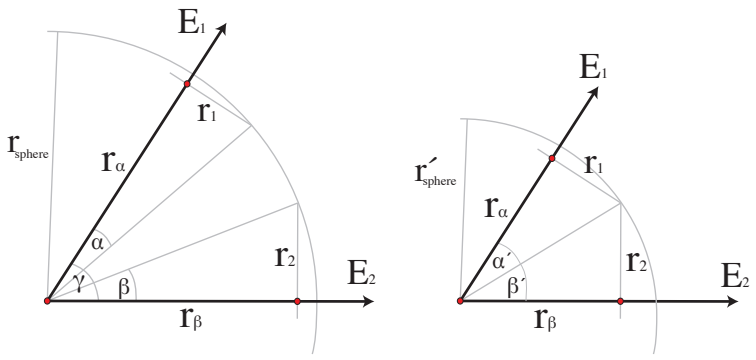


(e) (f) (g) (h)

Figure 5.6: Modeling of bifurcations. (a) Low resolution bifurcation mesh. (b) Resulting mesh after one subdivision step according to Biermann et al. [2000]. (c) Resulting mesh after two subdivision iterations. (d) Final result.

The minimal distance is determined by calculating the radius of the minimal enclosing sphere r_{sphere} of the bifurcation. This is executed by pairwise calculation of all branches E_i of the bifurcation. At this point the branching graph holds information about the angle γ between two branches and both radii r_1 and r_2. Fig. 5.7 b) shows that r_{sphere} is the optimal radius if the sum of both angles α and β equals the angle between the two branches γ.

minimal enclosing sphere



(a) $\alpha + \beta < \gamma$ resulting in r_{sphere} being larger than necessary.

(b) Optimal r_{sphere} for $\gamma = \alpha + \beta$.

Figure 5.7: Calculating the minimal radius of the enclosing sphere r_{sphere} given two edges E_1 and E_2 and their radii r_1 and r_2.

This allows for calculating angles α and β and the minimal radius of the enclosing sphere r_{sphere} given the preconditions

$$\gamma, r_1 \text{ and } r_2 \text{ known}$$
$$\gamma = \alpha + \beta$$

and

$$\sin\beta = \frac{r_2}{r_{sphere}}\ ,\ \sin\alpha = \frac{r_1}{r_{sphere}}$$

therefore

$$\begin{aligned}
\frac{r_1}{r_2} &= \frac{\sin\alpha}{\sin\beta} \\
&= \frac{\sin(\gamma-\beta)}{\sin\beta} \\
&= \frac{\sin\gamma\cos\beta - \sin\beta\cos\gamma}{\sin\beta} \\
&= \sin\gamma\frac{\cos\beta}{\sin\beta} - \cos\gamma \\
\frac{\frac{r_1}{r_2} + \cos\gamma}{\sin\gamma} &= \frac{1}{\tan\beta} \\
\frac{\sin\gamma}{\frac{r_1}{r_2} + \cos\gamma} &= \tan\beta
\end{aligned}$$

and

$$\beta = \arctan\left(\frac{\sin\gamma}{\frac{r_1}{r_2} + \cos\gamma}\right)$$
$$\alpha = \arctan\left(\frac{\sin\gamma}{\frac{r_2}{r_1} + \cos\gamma}\right). \tag{5.25}$$

The radius of the enclosing sphere r_{sphere} is

$$r_{sphere} = \frac{r_1}{\sin\alpha} = \frac{r_2}{\sin\beta}.$$

From all pairwise combinations of branches the edges of the branches need to be cut by the larges radius of all enclosing spheres. This guarantees that the generalized cylinders will be intersection free at the bifurcations.

After the branch segment's length has been reduced the two branches with the smallest radius difference are connected. This connects the branches with the smallest branching angles according to the allometric rules defined in Sec. 5.1. From the resulting four faces the face with the smallest angle to the next branch is connected to the end cross section (see Fig. 5.6 a).

This method creates so called water-tight meshes for the bifurcation and branches. This is an important property for the mesh to allow for the use of subdivision methods [Biermann et al., 2000] [10] to refine the mesh (see Fig. 5.6 b and c) and produce high resolution meshes.

[10] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proc. of SIGGRAPH '00*, pages 113–120, New York, NY, USA, 2000. ACM Press

5.4 Modeling details: Twigs and Leaves

In case of an image based approach one possibility to achieve a higher similarity to the input photographs is to add model details based on botanical motivated rules to the final tree model. Usually it is impossible to deduce information for these parts of the tree model from the photographs due to a high degree of occlusion. Length and frequency can therefore be deduced only from the density estimation of the volume model in a way that high density parts of the volume are refined with a larger number of twigs and leaves compared to sparse regions. Additionally it is possible to add another tool to change the appearance of the final model based on a small number of additional parameters that are hard to deduce automatically from the input photographs. Such parameters include length and tropism of the additional branches. These parameters can be used for different models of the same species and therefore do not limit the general power of the system.

As one of the parameters the user decides between mono-podial (branching with one main axis, observed on spruces and oak tress) and sympodial branching patterns (see Fig. 5.8 and Deussen and Lintermann [2005, p. 14]). Assuming Monochasium for the sympodial pattern, leads to a branching pattern as seen in Fig. 5.8 b), which e.g. is typical for linden trees.

Additionally the material properties can be defined that allow to influence the stiffness of the small branches. This simple parameter allows to influence

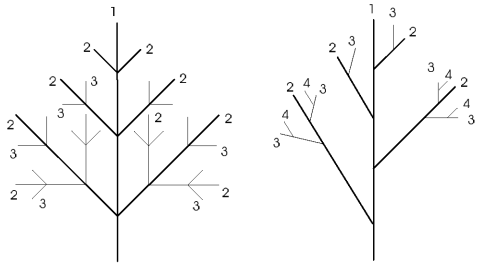


Figure 5.8: a) Mono-podial and b) sympodial branching configurations (see [Deussen and Lintermann, 2005]).

the model appearance and create in a simple manner the characteristic branching structure and tropism for several species (see Fig. 5.9).

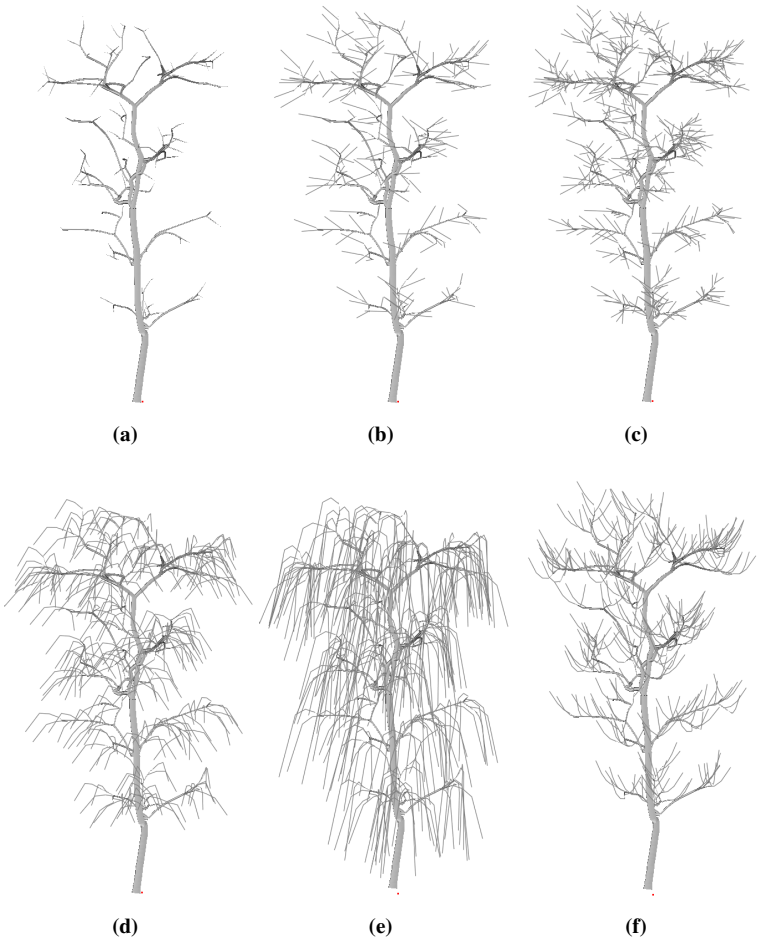


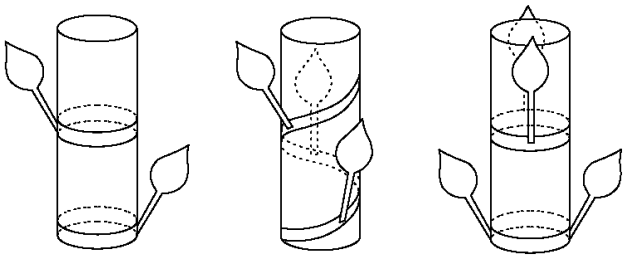
Figure 5.9: Resulting models for different parameterizations of small branches and twigs.

5.4.1 *Leaf Geometry, Position, and Orientation*

(a) (b)

Figure 5.10: a) Creation of the tiny twigs; b) leaf primitives.

After the geometry of the branches and trunk has been generated the position along the branches and orientation of leaves need to be defined following biological rules about leaf formations. *Phylotaxis* is the botanical term to describe the arrangement of leaves on a plant stem. Along an axis or branch four different arrangements (see Fig. 5.11) are identified: alternate (distichous), opposite (decussate), whorled (three or more leaves attached to a node), or in a repeating spiral (dispersed). This information is species specific and can be used if the species of the tree model is know.



(a) Distichy **(b)** Dispersion **(c)** Decussation

Figure 5.11: Different leaf formations (see Deussen and Lintermann [2005, 13f])

Using this informations makes it possible to define the point along the branch that forms the bud f as well as the main axis of the leaf a and are the basis to define the geometry that later supports the leaf texture. Modeling the positive *phototropism* of leaves, that is curvature towards the light source, is an important step to increase realism of the model's appearance. *Heliotropism* is a variation of phototropism that allows leaf lamina to respond to changes in the direction of the sun's rays. Both optimize radiation interception and increase the rate of photosynthesis. Adapting the orientation of the leaf's normal towards the sun is done by defining a point z inside the plant and defining an optimal direction $n_{opt} = \frac{a-z}{\|a-z\|}$. Rotating the leaf around its main axis to minimize the angle between n_{leaf} and n_{opt} results in a good approximation of the leaves phototropism

Phototropism

Heliotropism

$$\begin{aligned} n \bullet a &= 0 \\ n &= \operatorname{argmin}_x (x \bullet a). \end{aligned}$$

and allows to find a second vector to define the leaf's plane

$$s = n \times a.$$

(b) (c)

Figure 5.12: Different leaf orientation a) without b) with phototropism.

To create the foliage texture applied to this plane, we use photographs of natural leaves, add an alpha buffer and use a quad or a small set of triangles to support the texture (see Fig. 5.10(b)). In case of an image-based method the positions of the leaves are determined by the density values of the voxels, resulting in models where parts with high density values will contain proportionally more leaves. For trees as in Figure 3.2 the leaves appear in spatially clustered regions that we cannot represent with our course grid - therefore we refine the leaf arrangement by projecting the position of each leaf to the image planes and discarding leaves that are projected to empty or very sparse regions.

(b) (c)

Figure 5.13: Different leaf orientation a) without b) with phototropism.

Part II

Rendering of Complex Scenes

6

Level-of-detail Algorithms for Botanical Scenes and Models

Since the very beginning of computer graphics, rendering algorithms have been high consumers of computational resources and memory. Independent of the chosen rendering method, i.e. raytracing or radiosity, real-time or offline rendering, one reason for high computational costs is the scene complexity.

Various methods exist to reduce the cost of rendering. Spacial data structures reduce the number of geometry that needs to be processed. Another large family of algorithms aims to reduce the rendering costs by reducing the geometric complexity of the scene. In general these *level-of-detail* (LOD) methods aim at cutting down the rendering cost, mainly by reducing model detail or shading cost. This is basically one occurrence of the traditional tradeoff between complexity and performance, fundamental to a large class of problems in computer science.

level-of-detail

This chapter provides a brief introduction to level of detail algorithms with focus on the challenges and influence of special properties of natural scenes to those algorithms. Luebke *et al.*'s textbook [Luebke et al., 2002] [1] provides an excellent and more general overview of the huge body of work in this field.

[1] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science, 2002

Level of detail algorithms can be classified according to different inherent properties. Luebke *et al.* [Luebke et al., 2002] suggest to group LOD algorithms into *view-dependent* versus *view-independent*, and *continuous* versus *discrete* algorithms. In the context of ecosystem rendering the distinction

LOD Classification

according to Deussen and Lintermann [Deussen and Lintermann, 2004] [2] into *static* and *dynamic* LOD techniques is a better choice and we keep this notion in the following chapter.

[2] Oliver Deussen and Bernd Lintermann. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer Verlag, 2004

Static

Static algorithms precompute distinct model representations. The needed transition between different levels is achieved by blending between two representations. This requires rendering of both representations during the transition phase in combination with gradually changing the transparency. With increasing distance from the plant increasing the transparency for one model and at the same time decreasing the transparency of the next representation.

Dynamic

Dynamic representations provide a way to change the models complexity according to the projected screen size gradually without the need for blending between discrete representations. Both representations are often mixed with each other or combined with the full detail model. With increasing distance and therefore decreasing projected screen size the full detail model is replaced by its representation that allows either dynamic or static level-of-detail rendering.

The focus on geometry reduction techniques in the following section is closely related to rendering of botanical scenes.

6.1 Mesh Simplification

Most objects in todays computer graphic environments are defined in terms of triangle meshes. The early ideas with respect to level-of-detail algorithms aimed at reducing the numbers of triangles used to model an object based on local operations.

In this section, we describe the group of mesh simplification algorithms. The underlying principle here is to simplify the mesh and effectively reducing the number of polygons to describe a certain model. After briefly describing the basic algorithm of this group, we will discuss the implications of natural scenes and why these algorithms, while undoubtably useful for a large number of objects, are less effective for most natural scenes and objects.

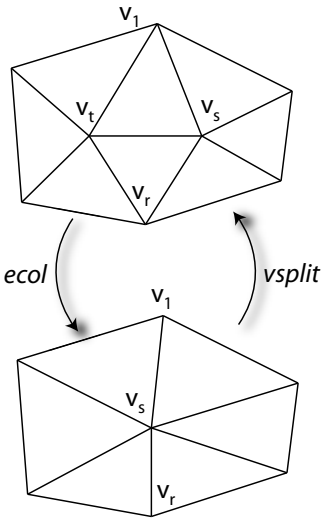


Figure 6.1: Progressive Meshes: edge collapse and vertex split operations.

Hoppe [Hoppe, 1996] [3] proposed the use of a low level local operator that reduces the mesh complexity by a small amount called *edge-collapse* (see Fig. 6.1). This operator collapses an edge and replaces the edge by a single new vertex, thus reducing the number of vertices and triangles to define the

[3] Hugues Hoppe. Progressive meshes. In *Proc. of SIGGRAPH '96*, pages 99–108, New York, NY, USA, 1996. ACM

object step by step. The inverse operator is called *vertex split*, which adds an edge and the adjacent triangles.

There are several variants [Luebke et al., 2002, pp.123] of this basic algorithm used for view-independent and view-dependent simplification.

One special property of natural models prevents these algorithm from being successfully used in this context: natural models show a lack of detailed closed meshes. There are usually two groups of geometry in a general plant model: the leaves and the trunk. The trunk and branches are often defined using generalized cylinders and make only a small portion of the overall geometric complexity. Applying the edge collapse operator to such a kind of mesh would immediately lead to a noticeable change in appearance of the underlying geometry. Leaves often consist of only two triangles supporting a leaf texture making an edge collapse without introducing strong visual differences impossible.

6.2 *Billboard Representation – Replacing Geometry Details with Texture*

One general technique to reduce the amount of geometry needed to model details has nowadays become most common, and is often overlooked that it was originally developed for this purpose. Replacing geometry details with texture can be seen as one of the oldest simplification methods in computer graphics. Instead of modeling every detail of a model with geometry, large patches are filled with a color representation of the details, basically trading memory and computational power required to store and process geometry with higher memory requirements for texture storage and texture fetches. Texture fetches are efficiently implemented in hardware and are resolution dependent since they are typically realized at a very late stage in the graphics pipeline (e.g. in the pixel shader for real time rendering on current GPUs).

Why Map Texture?

> WHY MAP TEXTURE? In the quest for more realistic imagery, one of the most frequent criticisms of early synthesized raster images was the extreme smoothness of surfaces - they showed no texture, bumps, scratches, dirt, or fingerprints. Realism demands complexity, or at least the appearance of complexity. Texture mapping is a relatively efficient means to create the appearance of complexity without the tedium of modeling and rendering every 3-D detail of a surface. (from [Heckbert, 1986] [4])

[4] Paul S Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6:56–67, November 1986

As already mentioned this technique is heavily used for plant models since naturally leaf and bark details are represented as textures to reduce the amount of geometry needed.

Figure 6.2: Billboard representations [from Behrendt et al., 2005].

This technique has been successfully extended to replace geometry with larger patches of textures that faithfully represent the details originally modeled with triangles. Billboard Clouds are one method of this family that was recently adapted to the special needs of tree models.

There are a variety of texture-based techniques, most notably impostors [Schaufler et al., 1996] and billboard clouds [Décoret et al., 2003], which have been tailored for tree rendering [Garcia et al., 2005, Lacewell et al., 2006]. These techniques generate discrete levels of detail and therefore are part of the static LOD group, which require special treatment to avoid distracting popping artifacts when adapting the detail, e.g. using the method propose by Scherzer and Wimmer [Scherzer and Wimmer, 2008]. Decaudin and Neyret [Decaudin and Neyret, 2004] use 3D textures representing parts of a dense forest and aperiodic tiling to render large scenes by volume slicing. However, this method only allows distant views, e.g. as used in flight simulators. They further extended this approach to volumetric impostors [Decaudin and Neyret, 2009], but this method shares the drawback of high memory consumption with other texture-based techniques.

Behrendt *et al.* [Behrendt et al., 2005] [5] adapt the billboard cloud approach to tree models by applying a k-means clustering to the model's geometry and then representing these clusters by one or more billboards depending on the structure of the bounding box of the point cloud (see Fig. 6.2).

[5] Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3):507–516, 2005

If the bounding box has two long sides and one short side the geometry is represented by a single billboard. If there is only one long side the authors chose a crossed billboard representation and if all sides have the same length three billboards are used. A simple heuristic is used to identify the latter case: $\sqrt{a \cdot b} > f \cdot c$ for the three sides a, b, c and $f = 0.5$.

Lacewell *et al.* [Lacewell et al., 2006] [6] choose a stochastic approach to generate the billboard representation. They define an initial billboard position and orientation based on a "seed" triangle chosen at random. The vertices of the supporting triangle T_s are perturbed within an ε distance. All remaining triangles T of the model are evaluated whether or not all vertices of the triangle are located within the ε surrounding of the billboard plane B, if so, the vertices of T are projected onto plane B. The sum of the area of all projected triangles onto B are saved and stored as B_{max} as a measure of quality for this billboard. This step is repeated several times and the billboard with the highest projected area is finally saved and the represented triangle set is removed from the mesh. The algorithm terminates if all triangles are removed from the original mesh.

[6] J. Dylan Lacewell, Dave Edwards, Peter Shirley, and William B. Thompson. Stochastic billboard clouds for interactive foliage rendering. *Journal of Graphics, GPU, and Game Tools*, 11(1):1–12, 2006

6.3 Stochastic Simplification

In this section methods are introduced that apply stochastic sampling of the model's geometry to reduce the rendering load. Stochastic simplification can be easily used with geometry representations that do not require topology information, such as point rendering methods.

The QSplat algorithm [Rusinkiewicz and Levoy, 2000] and sequential point trees [Dachsbacher et al., 2003] [7] adapt the size of prepositioned splats (rendered point primitives) to the required sampling density. Point samples can also be created on the fly, e.g. distributed randomly onto surfaces [Wand et al., 2001], stratified [Wand and Straßer, 2002], or adaptively [Stamminger and Drettakis, 2001] [8] .

All these methods have in common that—when using fewer samples—they preserve the total area by scaling the rendered remaining primitives.

Klein *et al.* [Klein et al., 2004] [9] used a stochastic simplification for polygonal scenes, however, the scene elements are only discarded, not altered, and thus this method is only suitable for coarse previews.

[7] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. *ACM Transactions on Graphics (Proc. of SIGGRAPH '03)*, 22(3):657–662, 2003

[8] Marc Stamminger and George Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques '01 (Proceedings of Eurographics Workshop on Rendering)*, pages 151–162, 2001

[9] Jan Klein, Jens Krokowski, Matthias Fischer, Michael Wand, Rolf Wanka, and Friedhelm Meyer auf der Heide. The randomized sample tree: A data structure for interactive walk-throughs in externally stored virtual environments. *Presence: Teleoper. Virtual Environ.*, 13(6):617–637, 2004

Deussen *et al.* [Deussen et al., 2002] applied stochastic simplification to rendering complex ecosystems. For reducing geometry they replaced the original triangles successively by lines and then points. Deussen and Lintermann [Deussen and Lintermann, 2004], and later Cook *et al.* [Cook et al., 2007] transfer this idea to complex geometry not restricted to point representations. They demonstrate simplification by pruning and scaling adapting not only to an object's screen size, but also to motion blur and depth of field. Their work is closely related to the algorithm described in Chapter 7.

6.3.1 Point- and Line-based Rendering

Using points as rendering primitives was already proposed by Levoy and Whitted in 1985 [Levoy and Whitted, 1985] [10] and later revised in [Rusinkiewicz and Levoy, 2000] [11] with the urging need for alternative rendering systems caused by the increasing resolution of 3D scanners and proposed as rendering primitive suitable for trees by Reeves and Blau [Reeves and Blau, 1985] [12] . Advances in 3D scanning technology made it possible to get extremely detailed models with hundreds of millions of polygons which lead to the problem of interactively displaying these objects. Rusinkiewicz and Levoy [Rusinkiewicz and Levoy, 2000] proposed the QSplat algorithm to handle the large amount of data 3D scanner produce. Compared to mesh based algorithms the authors especially designed the algorithm to reduce the amount of computation necessary per primitive, partially sacrificing the ability to exactly preserve 3D positions of any sample of the original object.

[10] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical report, University of North Carolina at Chapel Hill, 1985. TR 85-022

[11] Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. of SIGGRAPH '00*, pages 343–352. ACM Press, 2000

[12] William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 313–322, New York, NY, USA, 1985. ACM

The underlying data structure used is a hierarchy of bounding spheres (see Fig. 6.3) used for visibility culling, level-of-detail control, and rendering. Each node of the hierarchy contains the sphere center and radius, normal and color.

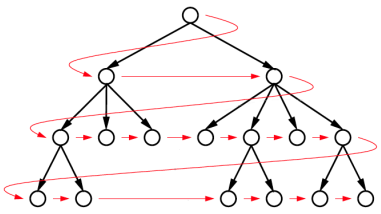


Figure 6.3: The QSplat data structure (from [Rusinkiewicz and Levoy, 2000]).

During rendering the preprocessed hierarchy is traversed recursively, skipping nodes and the respective subbranches whenever the visited node is not visible, and drawing a splat at the nodes position if either the node itself is a leaf or the benefit of recursing further is too low. The size of the drawn splat is determined by the projected size of the bounding sphere.

Weber and Penn [Weber and Penn, 1995] [13] describe a rendering system

[13] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *Proc. of SIGGRAPH '95*, pages 119–128, New York, NY, USA, 1995

for plants based on points and lines. The tree geometry is divided into different levels of visual importance, three levels for trunk up to small twigs and one level for the leaves that are rendered using points. They provide a detailed parameter list to determine the fraction of geometry that is rendered based on polygons and lines or points.

Deussen *et al.* [Deussen et al., 2002] [14] proposed a method for rendering objects with lines and points in the context of vegetation rendering (see Fig. 6.4). Each model consists of parts that are rendered using points and other parts that are naturally better approximated using line primitives. Parts that are chosen to be represented with points are sampled with $2n$ points for originally n triangles, sampling the triangles according to their area stored in a single list. During rendering only a prefix of this list depending on the projected area of the plant is drawn. This method allows to specify more important parts of the plant that are slowly reduced during rendering.

[14] Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. In *VIS '02: Proceedings of the Conference on Visualization '02*, pages 219–226, 2002

The number of points that is needed to represent the plant faithfully depends on the surface and the distance to the camera. The approximate projected area of the triangles of the original model can be calculated as

$$A'_p = \frac{1}{2}\frac{A_p}{r^2} \tag{6.1}$$

and the required number of points p as

$$p = c_p \frac{A'_p}{A'_{sp} n_p} \tag{6.2}$$

with $A'_{sp} = d'^2$ being the required point splatting area and d' the average distance between two neighboring point samples in the image plane (which is a user specified value to control the speed versus quality tradeoff).

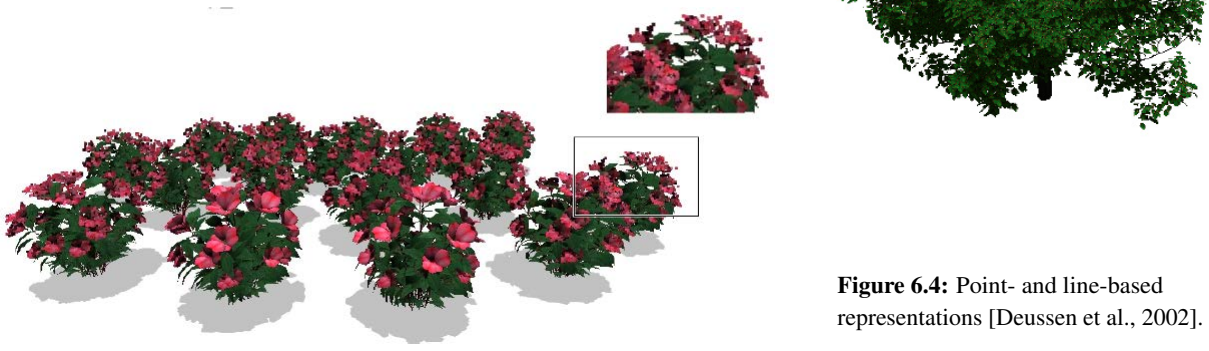


Figure 6.4: Point- and line-based representations [Deussen et al., 2002].

A general problem of using points as rendering primitives is flickering which is a special case of aliasing and mentioned as a drawback in most discussed publications.

6.3.2 Dynamic Polygonal Representations

Level of detail algorithms based on dynamic polygonal representations are described in Deussen and Lintermann [Deussen and Lintermann, 2004] [15] . Based on the fractal characteristics of plants the authors proposed a method that removes geometry from the original mesh and scales the remaining triangles to compensate the reduced geometry without changing the rendering primitives which is called "*drop and resize*". The authors apply this method to the geometry representing the leaves of the plant.

[15] Oliver Deussen and Bernd Lintermann. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer Verlag, 2004

drop and resize

Based on the overall surface area A of the plant model and the distance to the camera r, the projected overall surface A_i can be approximated with

$$A_i = \frac{1}{2}\frac{A}{r^2} \tag{6.3}$$

The scaling factor s for the remaining geometry with k leaves reduced from originally n leaves can then be calculated with

$$s(k) = w\sqrt{n/k} + (1+w). \tag{6.4}$$

For the delation rate (how fast geometry is discarded) the authors suggest a square function depending on the camera distance (following Eq. 6.3). The order in which the geometry is discarded is calculated according to positions on a horizontal plane that follows the golden section rule.

Stochastic Pruning

Later Cook *et al.* [Cook et al., 2007] [16] coined the term *Stochastic Pruning* proposing a very similar technique. The algorithm is divided into five aspects:

[16] Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. Stochastic simplification of aggregate detail. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26 (3):79, 2007

1. **Detail level:** Based on the projected area of a bounding volume determining the number of elements to exclude.
2. **Rendering priority:** Determining the order in which elements are excluded.
3. **Area preservation:** Compensating the excluded geometry by scaling the remaining elements.

4. **Contrast preservation:** Altering the colors of the elements to compensate color differences due to scaling.

5. **Smooth transition:** Fading out excluded elements to avoid popping artifacts.

The detail level λ for an object is the fraction of the elements that are included during rendering. There are many factors that influence this parameter. The authors considered size and blur and combined them to

$$\lambda = \lambda_{size}\lambda_{blur}.$$

The parameter λ_{size} depends on B, the current size of the bounding box measured in pixels. With B_0 as the size at which simplification should start and hB_0 the size at which half of the elements should be excluded λ_{size} becomes

$$\lambda_{size} = b^{log_h(1/2)}$$

with $b = B/B_0$.

According to the authors the *rendering priority order* should not be correlated to geometry position, size, normal, color or other characteristics, but instead be solely based on a random number.

Similar to [Deussen et al., 2002] the average projected area during rendering should be preserved. The total area of all N elements with average surface area a is $N \cdot a$. Rendering only a fraction of the original geometry decreases the total area to $\lambda \cdot N \cdot a$, which has to be compensated by scaling the remaining geometry by s in a way that $(\lambda \cdot N)(a \cdot s) = N \cdot a$, which translates into

$$s = \frac{1}{\lambda}.$$

Due to the aforementioned limitations of mesh- and point-based approaches, the proposed method, described in the following chapter, is based on dynamic polygonal pruning techniques. Especially the simple and efficient idea of stochastically removing geometry makes such approaches extremely simple to use and implement.

7

Realtime Rendering – Improved Model- and View-Dependent Pruning of Large Botanical Scenes

Deussen *et al.* [Deussen et al., 2002] [1] and Cook *et al.* [Cook et al., 2007] [2] present a very simple and elegant approach for stochastic simplification. However their solution does not consider any specific properties of the individual model. The suggested λ and respective scaling is solely a function of distance between viewer and model. Imagine two very different plant models: one model with very dense foliage and a large number of interior leaves and a different model with very transparent foliage (e.g. a birch). According to Cook *et al.* the applied simplification parameters would be the same.

The proposed method in this section [Neubert et al., 2011] [3] adapts the LOD parameters considering model characteristics by analyzing the rendered result. The basic assumption is that simplified models that cover the same pixels as the full detail model are of higher quality. In the case of the given examples the dense model would cover almost the same pixels than the original model even with less geometry and moderate scaling, whilst removing geometry from the birch would immediately change the number of covered pixels and thus the quality of the resulting image.

In contrast to other LOD algorithms it is hard for stochastic simplification to find a cost function based on distance error metrics (as common for progressive meshes) since the beauty and strength of the original algorithm lies in the fact that it is difficult to see the differences (which is only true for aggregate details). This makes it difficult to make a qualified comparison with these techniques.

[1] Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. In *VIS '02: Proceedings of the Conference on Visualization '02*, pages 219–226, 2002

[2] Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. Stochastic simplification of aggregate detail. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26 (3):79, 2007

[3] Boris Neubert, Soeren Pirk, Oliver Deussen, and Carsten Dachsbacher. Improved model- and view-dependent pruning of large botanical scenes. *Computer Graphics Forum*, 30(6):1708–1718, 2011

This chapter presents an optimized pruning algorithm that allows for considerable geometry reduction in large botanical scenes while maintaining high and coherent rendering quality, improving upon previous techniques by applying model specific geometry reduction functions and optimized scaling functions. Precision and Recall (PR) are introduced as a measure of quality to rendering and it is shown how PR-scores can be used to predict better scaling values. A user-study in which subjects can adjust the scaling value, shows that the predicted scaling matches the preferred ones. Lastly, the originally purely stochastic geometry prioritization for pruning is extended to account for view-optimized geometry selection, which allows to take global scene information, such as occlusion, into consideration.

7.1 Introduction

Rendering of natural scenes with vegetation as rich as in the real world has been a motivation of computer graphics research ever since. The complex visual appearance and the inhomogeneous structure of botanical objects makes real-time rendering of large scenes a challenging task that extends to this day. The obvious main reason is the tremendous amount of geometry that is needed to represent trees and plants. Storing as well as rendering such objects with full detail is beyond the capabilities even of modern graphics hardware. However, even if processing and rendering the data were possible, then the small sub-pixel details due to the complex geometry can still cause aliasing artifacts.

Many different approaches have been presented to render trees in real-time as presented in Chapter 6. Most often simple billboards or impostors [Schaufler et al., 1996] are used, or automatically generated billboard clouds [Décoret et al., 2003, Garcia et al., 2005] which are sets of billboards that better preserve occlusion and parallax effects. However, these representations are well-suited for distant objects and trees, but they are typically over simplified and close views reveal the low quality. It is also not possible to achieve coherent shading of the scene or to adapt the level of detail smoothly and without noticeable artifacts due to the planar nature of billboards [Lacewell et al., 2006].

In this section a rendering technique for complex botanical scenes based on pruning is presented. Pruning techniques (stochastically) reduce geometry by simply excluding some parts of the model, e.g. leaves, from the rendering and correcting contrast and the total rendered area by scaling the remaining leaves [Cook et al., 2007].

The proposed algorithms improves upon previous methods in several respects:

- describe a view-optimized pruning instead of purely stochastic simplification of the geometry. This allows to account for global scene information, e.g. thick and sparse forest and occlusion from neighboring trees.
- show that scaling of geometry after pruning should not be inversely proportional to the geometry reduction.
- formalizing this by introducing Precision and Recall as a measure of rendering quality. The measure does not consider pixel colors, but whether the right pixels of a rendered object are set. This is validated by conducting a user-study where subjects had to manually adjust the preferred scaling value.

7.2 Improved Scaling for Pruning Algorithms

In this section the pruning and scaling described by Deussen *et al.* [Deussen et al., 2002] and Cook *et al.* [Cook et al., 2007] is briefly recapped, and the drawbacks of these approaches are discussed. Next Precision and Recall measure for pruning and scaling are introduced, and the results of a user-study conducted for validation is presented.

7.2.1 Area Preservation and Optimal Scaling

The main objective of simplification algorithms is to preserve the overall appearance of the rendered models whilst using less geometry and thus reducing rendering cost. Deussen *et al.* [Deussen et al., 2002] as well as Cook *et al.* [Cook et al., 2007] propose a simple, and at first sight plausible rule: when the geometry is reduced down to a certain fraction then the remaining geometry is scaled such that the total area of rendered surfaces is equal to the original area.

Cook *et al.* [Cook et al., 2007] denote this scaling factor as $s = 1/\lambda$, where λ is the fraction of rendered geometry. However, the surface area that is visible after rendering the remaining geometry heavily depends on the actual rendered model. One can easily think of models where a lot of geometry can be removed and they would still cover the same projected area, i.e. the remaining geometry covers (almost) the same pixels for a certain view direction.

Stochastically pruning the geometry does not only change the area, but also—in particular when pruning strongly—the depth complexity of the rendered model. Cook *et al.* [Cook et al., 2007] account for this by calculating the expected visible area of a subset of randomly chosen elements of a model, and adapt the scaling factor accordingly.

Our results demonstrate an important and interesting fact: the largest decrease in rendering quality due to wrong scaling does not occur for strong, but for slight and moderate pruning, where the depth complexity correction has only little influence. In the next section Precision and Recall as quality measure is introduced, which does not only take the number of pixels but also their classification into correctly set and unset pixels into account.

7.2.2 Precision and Recall

A key aspect in information retrieval is *relevance* which in general is a highly subjective property. Different users may have different opinions about the relevance of a document in a certain context or query. In the context of computer graphics we are in a much better position. Considering relevance as a dichotomic property of pixels that are relevant for a particular model the idea to consider pixels that are set for the full detail model as relevant and all others as non-relevant is the straight forward idea. However, utilizing perceptional models to get a continuous notion of relevance should be the next step.

Measuring the quality of a level of detail algorithm can always be seen as asking how well is the simplified model representing the information of the full detail model. In this case measuring the *effectiveness*:

> Effectiveness is purely a measure of the ability of the system to satisfy the user in terms of the relevance of documents retrieved. (from [Rijsbergen, 1979, p.114])

The key measures for effectiveness in information retrieval are *Precision P*, *Recall R*, and *Fallout F*, that are defined according to Table 7.1 and 7.2 as

$$\begin{aligned} P &= \frac{|A \cap B|}{|B|} \\ R &= \frac{|A \cap B|}{|A|} \\ F &= \frac{|\overline{A} \cap B|}{|\overline{A}|} \end{aligned}$$

with $|\cdot|$ being the counting measure.

Defining *generality G* as

$$G = \frac{|A|}{N} \tag{7.1}$$

with N number of elements in the system as a measure of the density of relevant documents in the collection gives a relationship of Precision, Recall, and Fallout in the following way:

$$P = \frac{R \cdot G}{(R \cdot G) + F(1-G)}. \tag{7.2}$$

	relevant	non-relevant	
retrieved	$A \cap B$	$\overline{A} \cap B$	B
not retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$	$\overline{B}$
	A	$\overline{A}$	N

Table 7.1: Contingency table

true positives (tp)	$= A \cap B$
false positives (fp)	$= \overline{A} \cap B$
true negatives (tn)	$= \overline{A} \cap \overline{B}$

Table 7.2: Definition of the sets that are part of *Precision and Recall*.

Precision and *Recall* (PR) are well-known statistical classifications or measures for *exactness* and *completeness*. They are widely applied in the domain of information retrieval and are closely related to *sensitivity* and *specifity* to measure the performance of binary classification algorithms, such as support vector machines and Bayesian networks [Rijsbergen, 1979] [4] .

[4] C. J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2 edition, 1979

Several composite measures are used to avoid measuring effectiveness by means of a pair of numbers (e.g. precision and recall). The simplest example of this kind of measure is the sum of Precision and Recall as $S = P + R$ or $S_2 = P + R - 1$ or the harmonic mean

$$F = 2 \cdot \frac{P \cdot R}{P + R}.$$

In the following we will propose a composite measure suitable in the context of level of detail rendering and discuss analogies of information retrieval measures in the context of computer graphics.

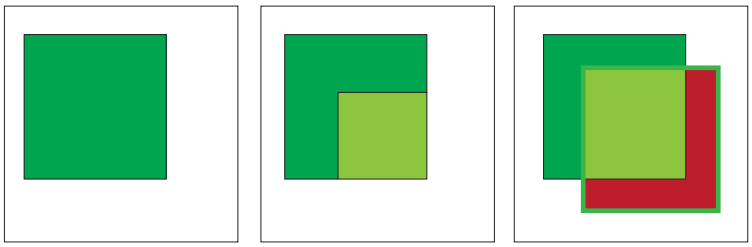


Figure 7.1: a) Dark green: pixels covered by original "model". b) Light green: pixels covered by the "model" rendered with reduced geometry and without scaling: no additional pixels are covered and thus only the Recall value is affected, while the Precision score remains 1. c) The simplified and scaled "model" covers pixels that were not covered by the original model (red, false positives). Dark green pixels are false negatives, light green ones are true positives.

Precision is defined as the ratio of correctly identified items (true positives) to both correctly and incorrectly identified items (sum of true positives and false positives). In our case, when rendering a pruned model it is the ratio of pixels that are correctly set, i.e. they would have been rendered for the full-detail model as well, and the total number of set pixels.

Recall is the quotient of correctly identified items (true positives) and all relevant items (sum of true positives and false negatives). Again translated into this scenario: the ratio of correctly set pixels and the number of correctly set pixels plus the number of pixels that should have been rendered, but which are not covered by the pruned model.

Thus, Precision and Recall (PR) are defined as (true positives tp, false positives fp, and false negatives fn):

set	pixels that are...
true positives	...correctly set, i.e. rendered for the original and for the simplified model.
false positives	...wrongly set, i.e. rendered only for the simplified model.
false negatives	...rendered for the original model, but not covered by the simplified one.

Table 7.3: Classification of pixels for Precision and Recall.

$$P = \frac{tp}{tp + fp} \tag{7.3}$$

$$R = \frac{tp}{tp + fn} \tag{7.4}$$

Table 7.3 gives an overview of the relevant pixels sets, which are shown in Fig. 7.1 for a simple example. More formally, we denote the set of all pixels covered by the original model as P_{orig}, and the set of pixels rendered for the simplified model as $P_{simplified}$, and thus get:

$$\begin{aligned} tp &= \{p | (p \in P_{simplified}) \wedge (p \in P_{orig})\} \\ fp &= \{p | (p \in P_{simplified}) \wedge (p \notin P_{orig})\} \\ fn &= \{p | (p \notin P_{simplified}) \wedge (p \in P_{orig})\}. \end{aligned} \tag{7.5}$$

Precision and Recall are reflecting how well the simplified model is representing the information—in this case the rendered pixels—compared to the original model. The PR-scores for the original model, i.e. rendering at full detail, are $P = 1.0$ and $R = 1.0$. The big advantage of PR is that not only the number of pixels is taken into account, as the preservation of the projected area does, but PR is also sensitive to whether the same pixels are covered. Thus, models rendered with reduced geometry that cover almost the same pixels as the original model will get PR-scores closer to the optimal $P = 1.0$ and $R = 1.0$.

Using PR-scores, it is now possible to define the optimal scaling value, s_{opt}, depending on the fraction of rendered geometry, denoted as λ (similar to Cook *et al.* [Cook et al., 2007]). The underlying idea of this heuristic is that one unset pixel that should have been covered is as bad as a set pixel that should not have been covered. Consequently, we choose s in a way such that we minimize the distance of the point $(P(s), R(s))$ (in the PR diagram) to the optimal PR-score at $P = 1$ and $R = 1$ proposing a combined measure:

$$s_{opt}(\lambda) = \underset{s}{\operatorname{argmin}} \sqrt{(1 - P(s,\lambda))^2 + (1 - R(s,\lambda))^2}. \tag{7.6}$$

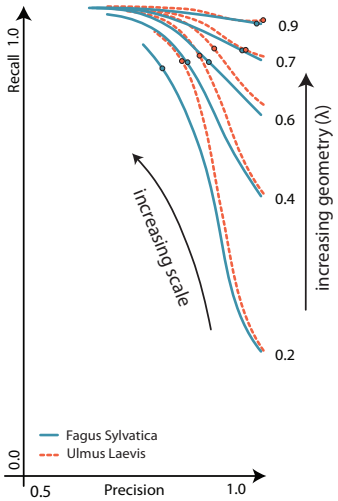


Figure 7.2: The Precision-Recall diagram for different plant models, five different geometry levels λ, and varying scaling values. An interesting case is the Ulmus model (red): it does not benefit from scaling for higher λ-values, and scaling even lowers the PR-score, i.e. the distance of the PR-coordinate to the top-right corner which represents the optimal score of $P = 1$ and $R = 1$.

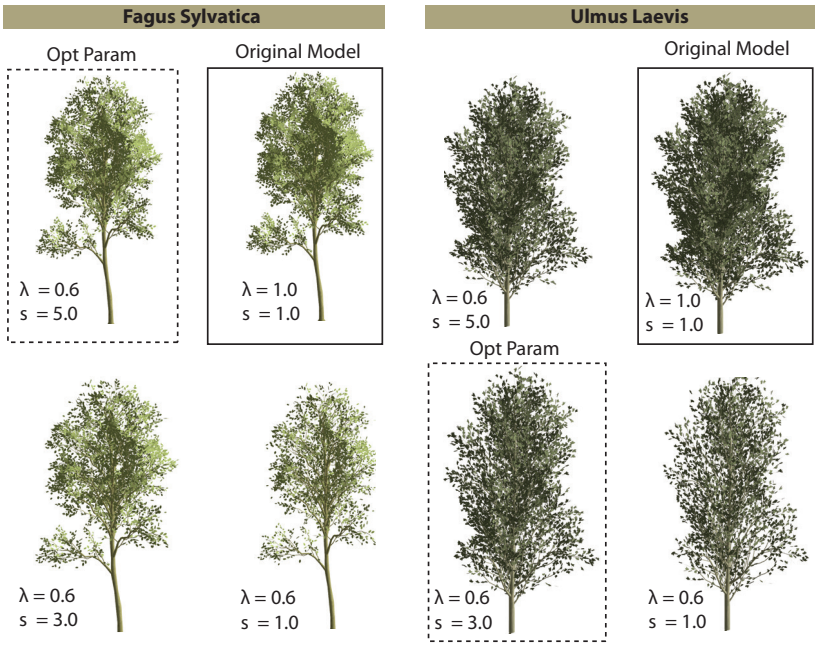


Figure 7.3: An interesting case is the Ulmus model (right): it does not benefit from scaling for higher λ-values, and scaling even lowers the PR-score, i.e. the distance of the PR-coordinate to the top-right corner which represents the optimal score of $P = 1$ and $R = 1$.

Fig. 7.2 and Fig, 7.3 show PR results for different models, values of λ, and scaling values s. In order to determine the optimal scaling, s_{opt}, for a given tree model we equidistantly sample λ in a preprocessing step and compute the respective PR-scores. During rendering, we linearly interpolate s_{opt} for non-tabulated λ-values. An interesting, but also important, property of the PR measure is that graphs for increasing λ are ordered towards the upper right corner. This reflects the intuitive assumption that using more geometry better resembles the original model.

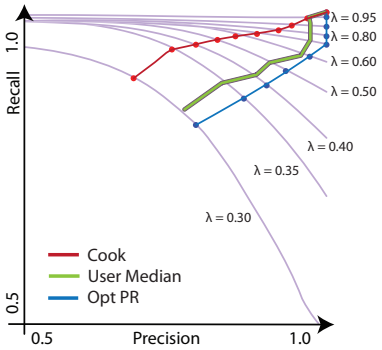


Figure 7.4: Comparison between different scale values for Picea Abies. Red: scale value according to Cook *et al.* [Cook et al., 2007] $s_{Cook} = 1/\lambda$. Green: user-preferred scale value (median). Blue: optimal scaling value found using PR-scores. Scaling does not improve the PR-scores for this model when more than 60% of the original geometry is rendered. This is also reflected in the preferred scaling values obtained from the user study.

7.2.3 Experimental Validation and User Study

The experimental validation is based on an user study with 19 subjects (both experienced and unexperienced in computer graphics) presenting an unpruned, full detail model side-by-side with a simplified version of the same model. The subjects were asked to choose the scaling value, for a given λ, such that the appearance of the reduced model resembles the full-detail model as close as possible (see Figs. 7.4 and Fig. 7.6). This procedure has been performed for ten geometry levels and five different tree models.

The study revealed that in particular for little simplification ($\lambda > 0.8$) the user-preferred scaling values were not only considerably different for every model, but also on average smaller than the scaling values computed

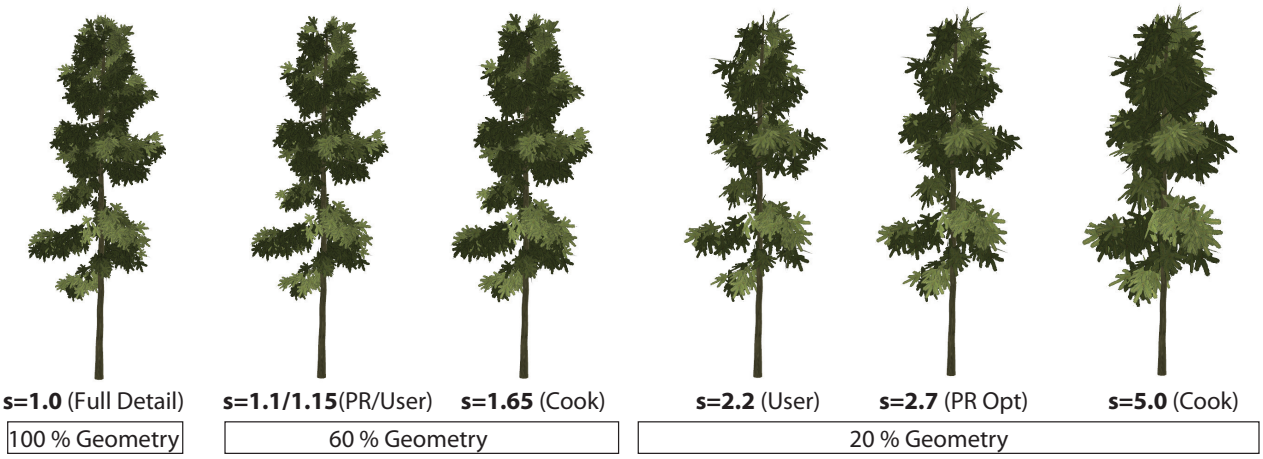
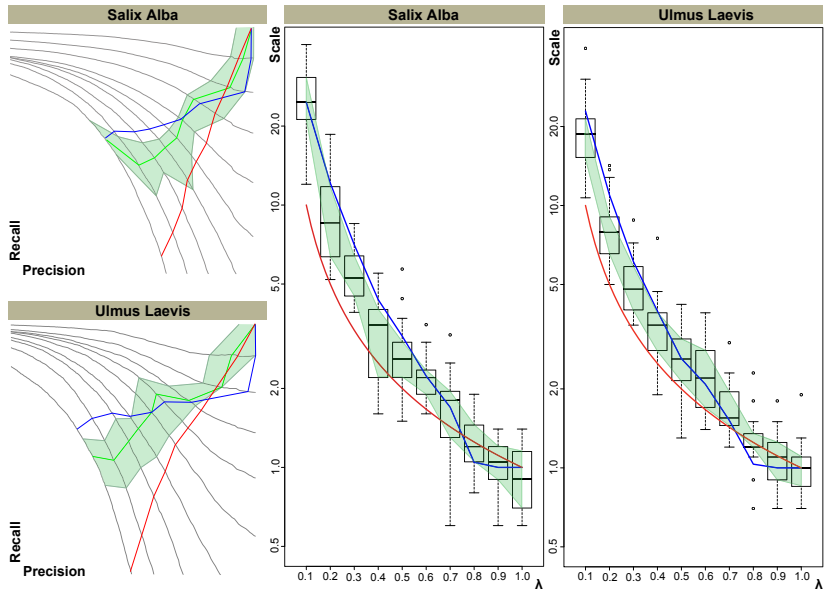


Figure 7.5: Comparison between different scale values for Picea Abies. Scaling does not improve the PR-scores for this model when more than 60% of the original geometry is rendered. The reason is that this model exhibits very dense geometry, and pruning does not immediately impact the overall appearance. This is also reflected in the preferred scaling values obtained from the user study.

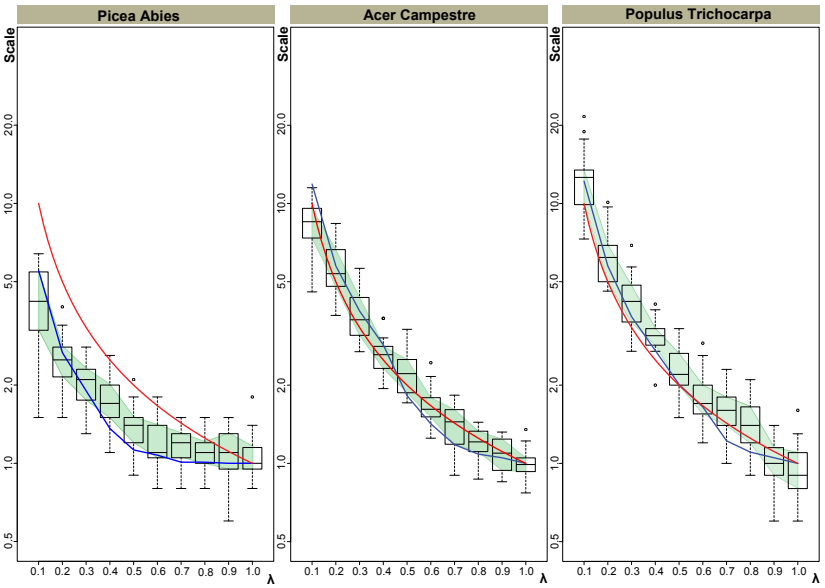
according to Cook *et al.* [Cook et al., 2007]. For smaller values of λ the standard deviation of the preferred scaling increased considerably, however, the median value was typically very close to our s_{opt}. The large standard deviation can be explained by the fact that these geometry levels are actually only used to render trees at large distances, while the model presented in the user study was rendered at full size. This obviously makes it harder to judge the appearance of the model and led to larger deviations in the preferred scale value.

Figure 7.6: This screenshot shows the implementation for the user study: the subject is asked to adjust the scaling factor (for different λ-values) for the left model, such that the rendering matches the full detail rendering on the right as close as possible.

Fig. 7.4 and Fig. 7.7 show the results of the user study. The preferred scaling values are shown in the respective PR diagram in green, next to s_{Cook} and s_{opt}. On the left in Fig. 7.7 the values are shown in a PR diagram and additionally the scaling values with respect to the different λ values on the right. For two models the user selected values are in general larger than s_{Cook}, for the Picea Abies model the user selected values are smaller; in all five cases our method faithfully predicts suitable scaling values.



(a)



(b)

Figure 7.7: Comparison between user preferred scaling values indicated in green (user median, first and third quartile), scaling according to Cook *et al.*(red), and scaling predicted by our method (blue). The λ-values of the PR-diagrams (top left) are sampled with step size 0.1 starting at $\lambda = 0.1$. The diagrams on the top right and bottom show the user selected scaling values for the same λ-levels as on the top left. While the user preferred scaling values for the Picea Abies model are significantly lower than the suggested Cook scaling, the scaling values for Salix Alba and Ulmus Laevis are higher. In all five cases the optimal scaling values predicted with our method are in close range to the user preferred values.

7.2.4 *Impact of Scaling and View Direction*

Precision and Recall are defined in image space and therefore view-dependent measures. However, the experiments indicate that for natural objects the deviations in the measure are very small. This is because such objects, e.g. trees, typically do not have a dominant view direction but rather uniformly distributed normals and vertex positions. This was also confirmed by the user study where user-selected scaling values for two different views of every model and geometry level (Fig. 7.8) are analysed. For both views, the variance and mean are very close (Fagus Sylvatica: $s(\lambda = 0.2, V_1) = 6.89$ and $s(\lambda = 0.2, V_2) = 6.75$; $s(\lambda = 0.6, V_1) = 1.94$ and $s(\lambda = 0.6, V_2) = 1.81$. The variance analysis (ANOVA) of the user-study data indicates that the hypothesis (H_0: mean is the same for both views) can be accepted with $F_{V1} = 0.07$ and $F_{V2} = 0.62$ for Fagus Sylvatica, and $F_{V1} = 2.66$ and $F_{V2} = 0.14$ for Picea Abies, well below the critical F-Value of $F_{crit}(1,36) = 4.11$ for $\alpha = 0.05$.

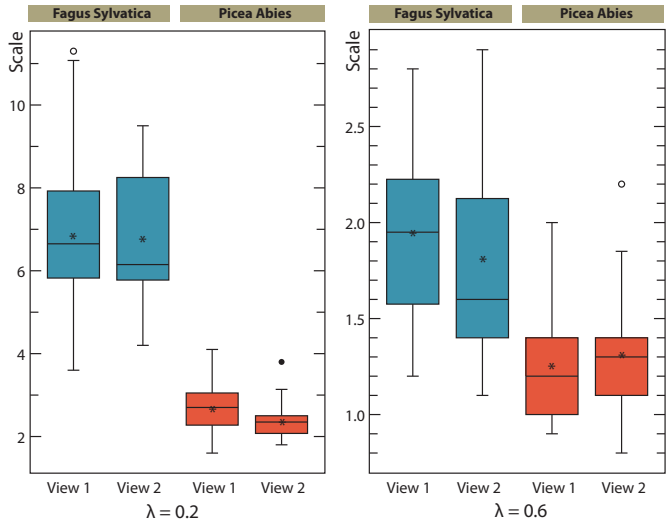


Figure 7.8: Comparison of the user selected scaling values for two different tree models and two geometry levels (left: $\lambda = 0.2$, right: $\lambda = 0.6$). For the Fagus Sylvatica model the users preferred larger scaling values (independent of the viewing direction) compared to the Picea Abies model. Besides the view independence the large difference in preferred scaling values for different plant models is significant and underlines the need for model dependent scaling. Outliers are indicated as circles, and the user average as stars.

A very important property of the PR measure, and thus for the application of this method, is that PR scores are invariant to rendering the models at different screen sizes, i.e. scaling the model without changing λ. This can be explained as follows: when reducing the size of a (pruned) model it is more likely that multiple triangles are projected onto the same pixels, and thus it is also more likely that all pixels covered by the original model and also covered by the pruned one.

7.2.5 *Detail Level Selection*

Rendering complex scenes is only possible if we reduce the level of detail for distant trees and only render with high quality when trees are close to the camera. Using the PR-score from Sect. 7.2.2, it is possible to define the quality Q of a rendering as the distance of the PR-vector to the optimal value $(1,1)$:

$$Q(\lambda,s) = 1 - \sqrt{(1-P(s,\lambda))^2 + (1-R(s,\lambda))^2}. \tag{7.7}$$

The general intention is to ensure that a tree at a certain distance, d, to the camera will be rendered at a given minimum quality. This means that the PR-vector for a model rendered with a geometry level $\lambda(d)$ has to be within a certain proximity to the top-right corner of the PR-diagram (see Fig. 7.10).

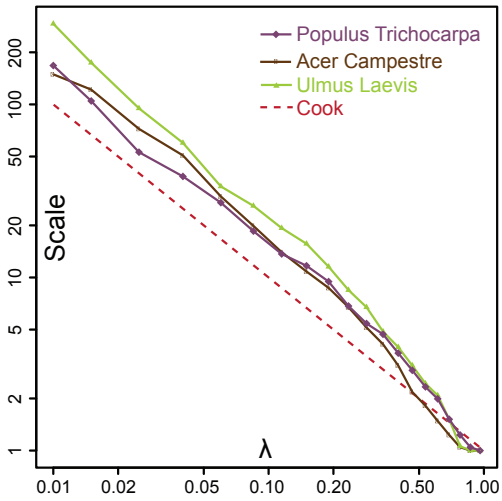


Figure 7.9: As can be seen, it is possible to reduce the geometry to a larger extend for model Populus Trichocarpa compared to model Acer Campestre, in particular for small values of λ. A log-log plot of scale vs. fraction of remaining geometry, i.e. λ. Note that the connected (s_{opt},λ) and (s_{Cook},λ) lines in the PR diagram (see Fig. 7.10) look very rough. The values plotted against the geometry level λ, however, are smooth with the expected exponential behavior. The smoothness is important to avoid popping artifacts during rendering.

That is, for rendering it is necessary to sample and store the function $\lambda(d)$ to provide this desired minimum quality for a given d. Note that determining this function takes place in a precomputation step for every tree model.

There are various options to define minimum quality, e.g. letting the user define a given maximum deviation from the optimal PR-scores and computing $\lambda(d)$ accordingly. Determining $\lambda(d)$ such that the rendering quality of the PR-optimized pruning and scaling matches their quality for the same distance d to compare the proposed method to Cook *et al.*'s. That

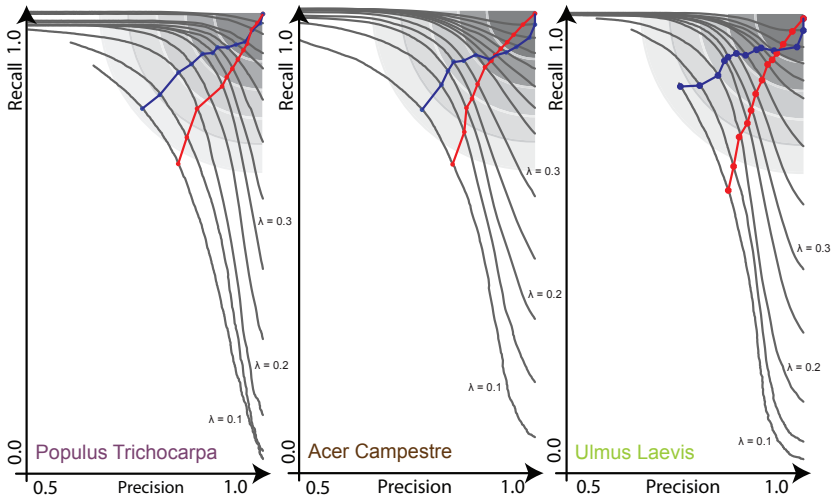


Figure 7.10: Precision-Recall graphs for three different tree models. The blue graph shows the optimal scaling values s_{opt}, the red one the standard scaling values s_{Cook} (according to Cook *et al.*). For rendering we choose λ such that we maintain a minimum quality that is required for a given viewing distance. The minimum quality requirements are indicated by the circles centered at the top-right corner of the PR diagram. As can be seen, it is possible to reduce the geometry to a larger extend for model Populus Trichocarpa compared to model Acer Campestre, in particular for small values of λ. The λ-values are sampled with 0.05 step size from $\lambda = 0.1$ to $\lambda = 0.4$ and with step size 0.1 above.

is, rendering a model with the same PR-scores, but with less geometry if possible.

This works as follows: Cook *et al.* use $\lambda_{Cook}(d) = (1-d)^2$ (with d normalized to $[0;1]$) as a simple relation of distance and geometry. Rendering the model with this pruning yields a quality $Q(\lambda, s_{Cook})$. Next, determining the smallest λ_{opt} whose rendering with the optimal scaling $s_{opt}(\lambda_{opt})$ (Sect. 7.2.2) yields equal or better quality, i.e. $Q(\lambda_{opt}, s_{opt}) \geq Q(\lambda, s_{Cook})$. This compound mapping yields a λ_{opt} and an associated s_{opt} for a given view distance d. Obviously this precomputation can only be carried out for a finite number of values. Therefore using 10 equidistant samples in $[0;0.1)$ and $[0.1;1.0)$, respectively, and linearly interpolating λ_{opt} from the stored samples.

Fig. 7.10 shows the mapping of distance to pruning for three different tree models. The plots show that model (a) and (c) can be rendered with high quality (PR-score within the second circle depicted in Fig. 7.10) even for a low value of $\lambda = 0.2$. For model (b) a higher λ-value is required even for larger viewing distances. The plots of rendering with higher λ-values reveal that model (b) and (c) suffer stronger from pruning, while model (a) preserves most of the rendered pixels of the original model.

7.3 Rendering Priority

The rendering priority reflects the order in which geometry is removed from the original model with decreasing λ. Cook *et al.* [Cook et al., 2007] tried to avoid correlation in the rendering priority between order and position, size,

surface normal, and color as much as possible, and thus prevent disturbing artifacts when rendering with reduced geometry. However, they state that in some cases the priority order might be found procedurally. In this section different ways are proposed to find the rendering priority order algorithmically in a way that ensures higher Precision and Recall values. Note that it is only the PR-scores that make it possible to compare different prioritization heuristics.

Figure 7.11: Implicit surface used for silhouette preservation.

7.3.1 Silhouette Preservation and Density Normalization

In order to optimize the rendering priority we need to determine which parts of a model are close to the boundary and will potentially be part of the silhouette, and which regions exhibit a high or low density of geometry.

To this end, it is important to define what the "boundary" of a (botanic) model is. For this, implicit surfaces that tightly enclose a model can be used. They have also been used to generate normal distributions for such models that provide more realistic and expressive illumination of foliage [Luft et al., 2007]. Implicit surface can be generated using metaballs [Blinn, 1982]: first, a set of generation points P is chosen and an influence radius r_i is assigned to every point. The center of leaf-triangles are used as generating points and an influence radius proportional to the overall plant height (5% in this case) is choosen. The contribution of a single generator point $p_i \in P$ at a point in space, q, to the global density function is defined as:

$$D_i(q) = \left(1 - \|q - p_i\|^2 / r_i^2\right)^2.$$

The sum over the contributions of all p_i yields the global density function:

$$F(q) = \sum_i D_i(q).$$

An iso-surface is then defined by a given iso-value a with $F(q) = a$, and can be triangulated using marching cubes (see inset). To extract a tree's tight hull an iso-value is chosen such that $a = 0$.

In the following different prioritization heuristics based on the global density function are discussed. Again, the quality is measured using PR-scores.

7.3.2 Varying Density

First, regions of high geometric density within a model are identified using the global density function. Triangles that are close to each other are likely

to be projected to the same location in image space. Thus removing triangles in very dense region lowers the probability of overdraw while still keeping chances high that all original pixels are covered even without scaling the remaining geometry. The evaluation of this guided geometry prioritization using the proposed PR measure, and experiments showed that the quality improves for high values of λ. For such values, there are larger variations in local density (see Fig. 7.12), as no, or little, density controlled pruning did take place. These variations obviously vanish when reducing more and more geometry prioritized in dense regions, which makes the density variation become more uniform. When this point is reached, i.e. for smaller λ, switching back to pure stochastic prioritization is beneficial. The performance increase due to density prioritized pruning depends on the variance of $F(q)$ within a model, and thus models with almost uniform density do not benefit from this strategy.

7.3.3 Orientation

As second heuristic the improvement of rendering prioritization based on the deviation between a triangle's normal and the normal on the nearest point on the implicit surface (denoted as α) is investigated. Preserving geometry facing outwards, i.e. small α, generally enforces a pixel coverage of the rendered model that is closer to that of the full detail model. Scaling triangles that resemble the models iso-surface turned out to perform well, especially for small values of λ (see Fig. 7.12).

Pruning triangles close to the implicit surface with lower probability, however, leads to inferior results. While at first sight it seems reasonable to preserve the silhouette, keeping and scaling triangles close to the surface results in many false positive pixels, in particular for strong scaling with small λ. This leads to visible artifacts and low Precision and Recall scores.

7.3.4 Combined Prioritization

Both heuristics determine "survival probabilities" for the triangles of a model. In our implementation we use an empirically found weighting to combine both of them, where the orientation heuristic has smaller impact. Using the aforementioned heuristics we compute a combined probability of keeping and removing a triangle of the model using

$$P_{Combined} = (P_{Silhouette})^2 + P_{Density} \tag{7.8}$$

normalized to $[0,1]$ and then sort the triangles for descending survival-probability (adding a small amount of randomness) to obtain a single list

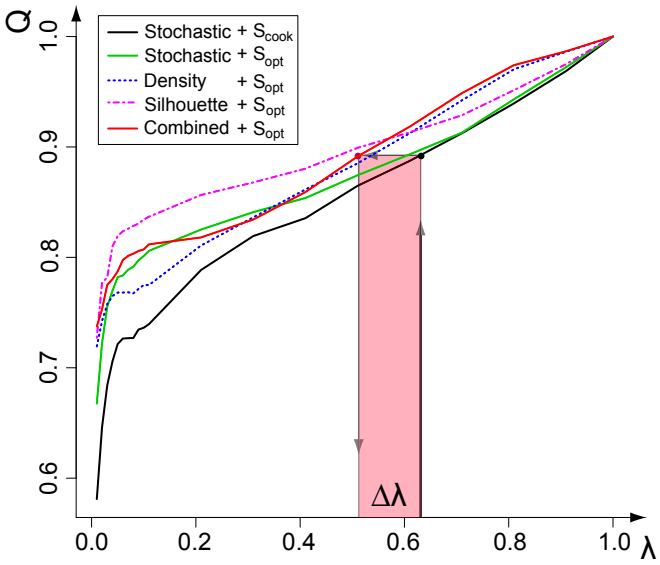


Figure 7.12: Comparison of prioritization heuristics to pure stochastic ordering for the Ulmus model. The red area shows the benefit of switching from stochastic order with s_{Cook} to the combined prioritization with s_{opt}. It is possible to maintain the quality with less geometric detail, denoted as $\Delta\lambda$.

representing the entire model. Similar to Sequential Point Trees [Dachsbacher et al., 2003], we can then render only a prefix of the list, according to λ, to render a pruned model.

7.4 View-dependent Optimization

Considering the viewing distance and thus the projected size of a model is of course one important aspect for choosing the geometry level. However, occlusion also has impact on the required detail: partially occluded trees, or trees that are completely surrounded by others, do not contribute considerably to the scene's appearance and thus should be rendered using less geometry. In this section it is shown how to determine occlusion of trees at run-time, and control the rendering accordingly.

For this the occlusion of each tree is analysed by testing the visibility of a set of sample points distributed on its iso-surface. In principle there are various possibilities to perform the visibility test, e.g. ray casting, or using some form of precomputed visibility information. To facilitate real-time rendering without precomputation, it is possible to use an image space approach relying on the depth buffer of the camera image only. Obviously this depth buffer is not available before actually rendering the geometry. However, assuming smooth camera movement it is possible to exploit frame-to-frame coherency and test the visibility of sample points using the depth buffer and transformation of the previous frame. For abrupt movements, or sample points that are projected outside the viewport, one can conservatively assume full visibility and thus render the models at possibly higher detail

than actually necessary.

From the visibility of the sample points approximate occlusion factor for each tree is deduced. The fraction f of visible to the total number of sample points is used to control λ, in a way such that $\lambda_{occl} = \lambda_{opt}(d) \cdot \max(f, 0.1)$. A hysteresis function is used to avoid popping artifacts and smooth the λ_{occl}-values over time.

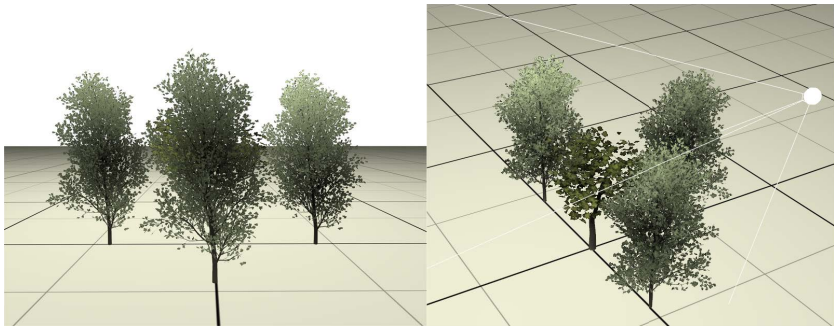


Figure 7.13: The tree hidden from the camera (white sphere) is pruned stronger than its visible neighbors.

7.5 Color Variation

The proposed method is suitable for rendering (groups of) objects that are aggregated from a large number of randomly oriented and placed geometric details. Typically one can also assume a near uniform color distribution, or large-scale gradations, for botanical objects. Cook *et al.* [Cook et al., 2007] intentionally do not correlate the rendering priority order to the color distribution or any other model characteristics. Apart from color variation Cook *et al.* propose a method to preserve color contrast during simplification. In this section the effect of the pruning algorithm on three different kinds of color variations is shown (Fig. 7.14). While large colored regions across the objects are preserved (Fig. 7.14, top), smooth color gradations show the effects similar to quantization artifacts, which is expected due to the geometry reduction (bottom). The pruning becomes most apparent for small, randomly distributed and salient details (red leaves in Fig. 7.14). Under strong pruning the fraction of covered pixels is still preserved, but the distribution becomes less random due to the smaller number of samples (bottom right). All these artifacts become less apparent if the model is rendered with a size according to the geometry level. Color variations between models are of course preserved, as the method does not prune across objects, and thus individual trees can still be identified (Fig. 7.15).

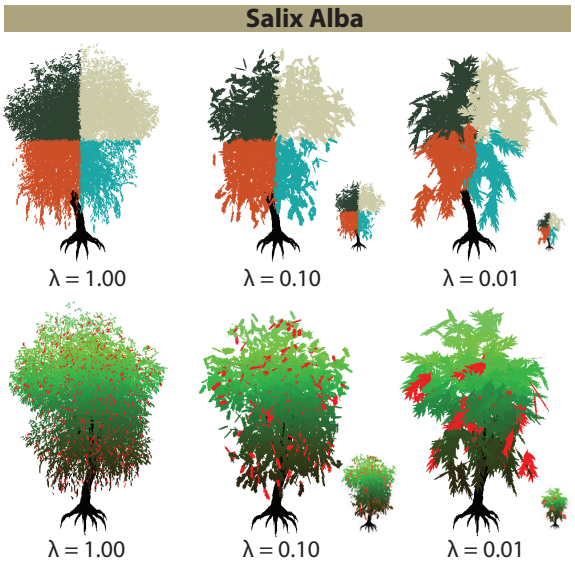


Figure 7.14: Effect of model simplification to intra model color variation: boundaries of large colored regions are maintained even for low detailed models (top row). Smooth gradations do not exhibit noticeable changes under stochastic pruning, apart from effects similar to color quantization. The arrangement of small details (e.g. the red leaves) obviously changes, but still does not cause flickering. Note that strong pruning typically occurs for distance trees.

7.6 Results and Comparison

In this section the results of the proposed method and comparisons to Cook *et al.*'s [Cook et al., 2007] method to assess rendering performance, and to billboard clouds [Décoret et al., 2003] to demonstrate the benefits of this (view-dependent) pruning over texture-based representations are presented. The method is implemented using OpenGL and all tests and measurements are performed using an Intel Core i7 at 2.8Ghz, with 4GB of memory, and a NVIDIA Geforce GTX 295 GPU.

Comparison to Billboard Clouds For rendering botanical models most real-time applications resort to a representation with relatively few textured polygons recreating the original model. For video games these models are often created manually, while billboard clouds [Décoret et al., 2003] can be used to obtain such reduced models automatically. Note that these representations do not provide a "continuous" level of detail and switching between different levels is prone to popping artifacts. The results are compared to billboard clouds by evaluating the rendering quality according to the quality measure Q (Sect. 7.2.5). It can be observed that the rendering quality varies strongly with the view direction when using billboard clouds, and significantly less with optimized pruning. Fig. 7.16 shows this comparison where the parameters of our pruning are adjusted to match the average quality of a billboard representation. Note that another applications of our PR-measure can be the billboard cloud generation itself, where it can be used to identify bad views for which the billboard representation needs to be improved.

Rendering Performance The method allows to render complex scenes

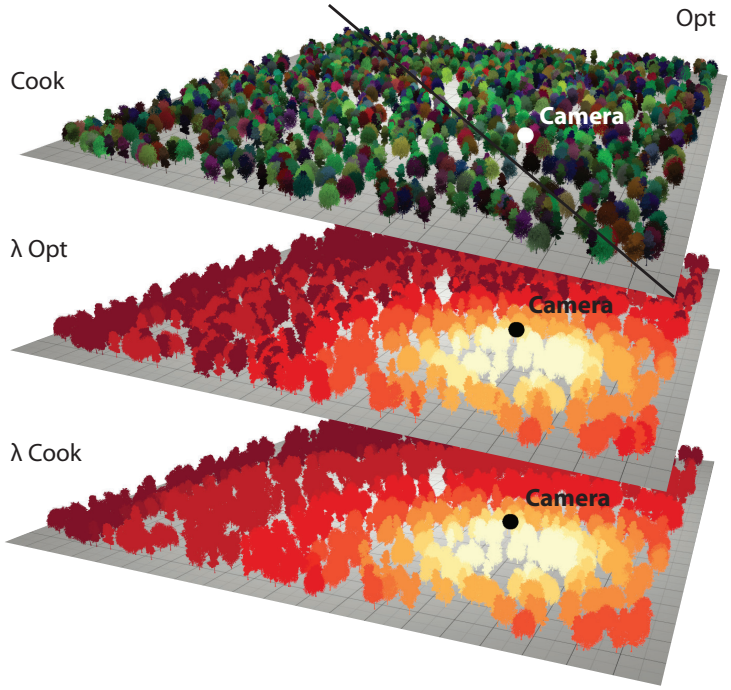


Figure 7.15: A scene with intentionally exaggerated high inter model color variance. Even for distant areas with a low amount of geometry individual tree models can be identified (top layer). Bottom two layers: Geometry distribution according to Opt and Cook.

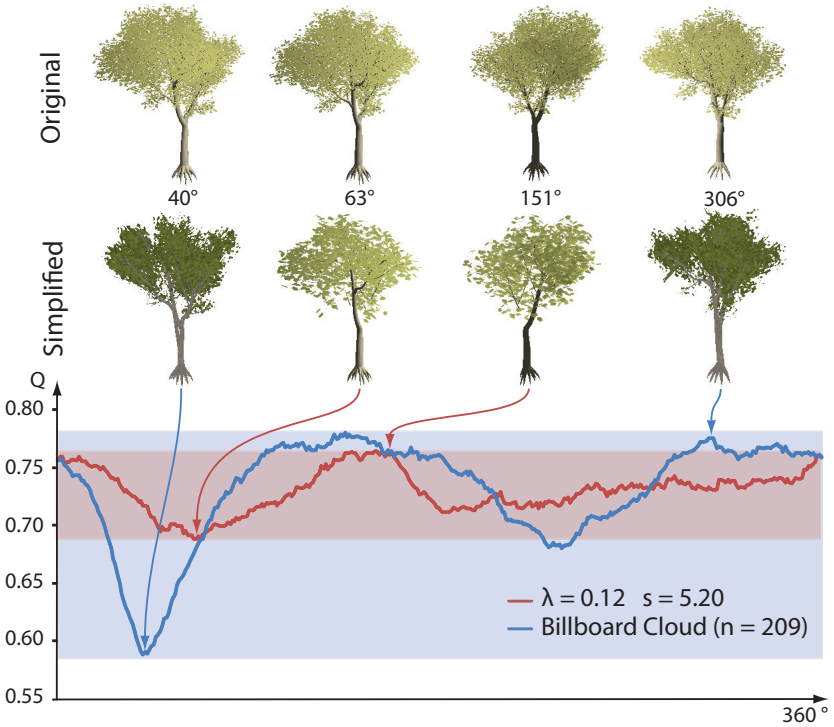


Figure 7.16: Comparison of rendering a 360° rotation of a billboard cloud model (209 billlboards) and a pruned model ($\lambda = 0.12$, $s = 5.2$). The parameters for the latter are chosen to match the average quality Q of the billboard model. Note that the variance for the billboard model is much higher.

with 5000 tree models at interactive to real-time rates, i.e. 8 to 25 frames per second at a resolution of 1600×1200 (Fig. 7.18). The full-detail geometry of the scene consists of more than 1.3 Billion vertices and renders at only 0.8 frames per second on the same hardware, i.e. far from interactive speed.

The optimized and prioritized pruning, together with the view-dependent visibility tests reduces the number of vertices per frame to about 26 million vertices. On average this yields a performance increase, compared to Cook *et al.*, of approximately 60-70% while maintaining the same quality (determined using the PR scores). Fig. 7.15 shows a complex scene with exaggerated color variation, individual tree models can still be identified even for low geometry levels (top layer). The bottom two layers visualize the color coded geometry level. While λ_{Cook} is chosen depending on the camera distance λ_{Opt} is chosen individually for each tree model according to Sec. 7.2.5. Fig. 7.17 shows a detailed evaluation of a standard camera path through a scene with 1276 trees without culling.

	Opt PR	Density	Silhouette	Combined
Avg	0.101	0.174	0.407	0.301
Min	0.043	0.022	0.308	0.014
Max	0.437	0.236	0.693	0.392

Table 7.4: Geometry reduction with different prioritizations (Fig. 7.17) compared to Cook *et al.* [Cook et al., 2007]. For example, the combined heuristic requires 30.1% less geometry on average to render at the same quality as Cook *et al.*

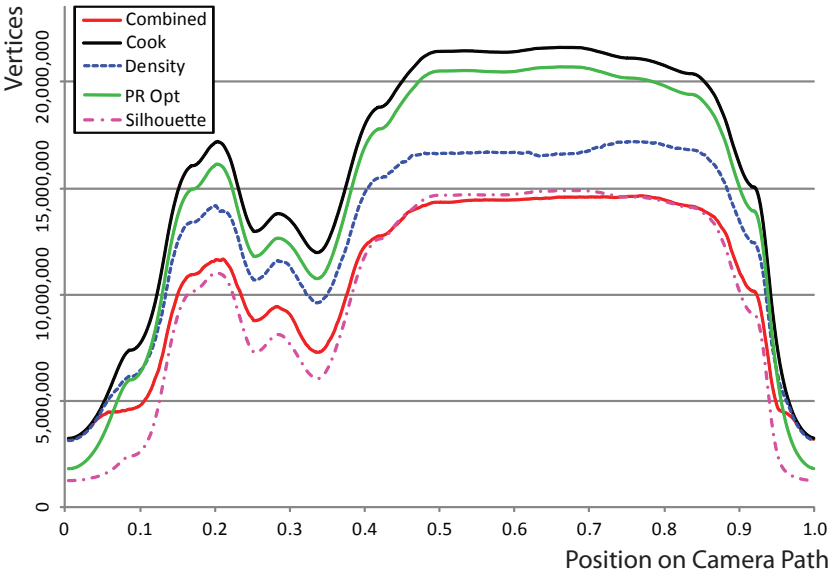


Figure 7.17: Performance evaluation of different heuristics for a camera path through a scene with 1276 tree models without culling. It can be seen that the silhouette based prioritization performs best for distant views (start and end of the camera path). While the combined prioritization performs slightly better in walk through camera positions.

7.7 Conclusions

Conventional methods to measure the quality of models of different level of detail usually take place in model space, measuring the total error between original and simplified model as the sum of squared differences. The quality

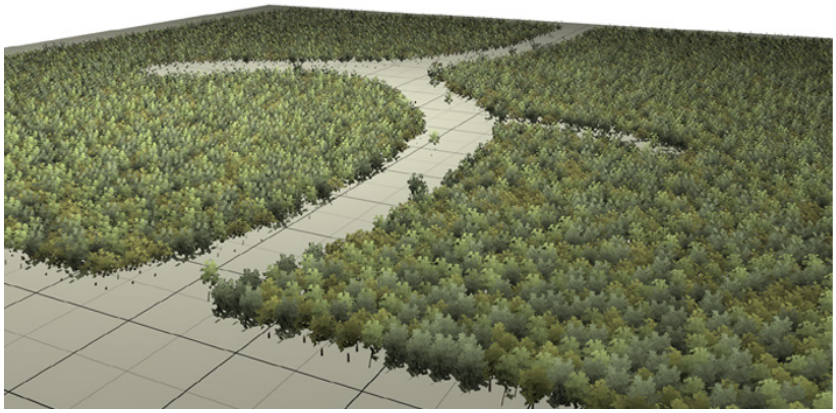


Figure 7.18: This scene consists of 5000 trees (in total 1.3 Billion vertices). With our optimized pruning we can render this scene with 15 frames per second when all 5000 trees are visible. Pruning according to Cook *et al.* [Cook et al., 2007] renders an image of the same quality at 9 frames per second.

of resulting models of algorithms in the group of *drop and resize* methods or simplified with the help of stochastic pruning is impossible to measure this way due to different fundamental intentions between these methods; stochastic pruning aims at preserving the visual appearance of the model in image space rather than preserving the geometric representation in model space. This boils down to answer the question: How well is the visual quality of an image preserved using a simplified model?

To answer this question we introduced Precision and Recall as a measure of quality for rendering complex geometry with pruning. We further improved on previous methods by applying model specific geometry reduction and optimized scaling as well as view-optimized pruning. The validity of the established quality measure was evaluated by means of a user study which indicates a considerable improvement compared to naive and purely stochastic pruning. However, this work also raises new questions. One interesting direction of future research is to consider more than just correct and incorrect pixels in PR, e.g. by accounting for deviations in the normals, measuring contrast and color differences, or to evaluate how visible differences predictors can improve the measure and whether their use amortizes.

Part III

Conclusion

8

Concluding Remarks

The topics of this thesis are the special properties of natural objects and resulting requirements specific to a variety of computer graphics research subjects. The required complexity of natural scenes is very high, which makes it difficult to achieve a convincing visual quality and accomplish near photorealism of resulting imagery. This complexity, which can be observed at different scales, has implications to both modeling and rendering of natural scenes and objects. While the lack of modern modeling software together with the need for a very efficient way to store the model information caused by limited graphics and cpu memory, lead very early to the development of mathematical descriptions for plants, the decrease of these hardware limitations lead to a change of modeling paradigm regarding natural objects. The evident next research subject was to increase the ease of use of modeling tools and decrease the tedium of manual modeling repetitive details.

The proposed modeling methods lead to improvements on this topics in various ways. Using a *guided particle simulation*, we show that a combination of image- and simulation-based methods can produce 3D models that match the trees on the input images well. By imposing image constraints in the form of captured branching patterns and density distributions, the proposed method is able to adapt the particle simulation to a given set of input images. Manually altering the input data and thus changing density or shape of the resulting model is a powerful and expressive tool for intuitively editing the model while the underlying simulation still provides a plausible result. The term *self-organizing models*, coined by follow up research results, describes this class of modeling algorithms very well.

guided particle simulation

self-organizing models

The second modeling approach is motivated by the observation that humans are able to deduce the 3D structure of complex structures from simple 2D sketches, successfully deducing the missing information with the help of common knowledge about trees. To capture this knowledge, we

introduce probabilistic graphical models based on a large data base of 3D tree models. Inferring the most plausible configuration of parameters that are missing in the input sketch, the system is able to produce high quality 3D models based on the probabilistic model. The underlying probabilistic model is not only used to infer missing informations, but is used to add additional details to sketches further increasing the modeling capabilities and expressiveness of the system. This second project shows the prospects of combining *sketch-based* and *data-driven* principles for modeling of trees.

sketch-based and data-driven modeling

Finally, the implications of complexity to *real-time rendering* are elaborated in the last part. Geometric simplification algorithms are well know in the computer graphics community for years. One large group of these algorithms reduce the geometric complexity by replacing details with textures, while another group replaces complex parts with larger patches of geometry of lower resolution. Usually the question how accurate the simplified model is, is answered using model space quality metrics based on the sum of squared differences between the original and simplified geometry. However, in the case of trees both general methods have disadvantages due to structural properties of the geometric representation of plants. Stochastic simplification algorithms solve this by randomly selecting the geometry to render and scaling the remaining geometry, following the principle of *drop and resize* of parameterized, continuous Level-of-Detail algorithms. As such, model space quality metrics render impracticable for this kind of algorithm. Image space quality measures are much closer to the idea, that what actually should be preserved is the visual representation of the model.

drop and resize

The question of how to adapt these parameters according to distance, projection size, and model is the main objective of this method. We established a quality measure based on *Precision and Recall*, well known measures from Information Retrieval, to chose model specific geometry fraction and scaling parameters. The conducted user study shows that the predicted parameter sets match values chosen by the subjects. For a standard scene with 5000 plant models and originally more than 1.3 billion polygons, we measured a significant performance increase compared to non-optimized stochastic pruning algorithms.

advanced stochastic pruning

Open question with regards to the research topic presented in this thesis include further extensions of the idea of simulation-based modeling. While the presented method is able to produce highly realistic models according to input images using a simplistic particle simulation, further research suggest to revert the direction the particles flow to form the model. While sacrificing the possibility to use image information, this allows to include other aspects

further research

into the simulation system, as defined branching angles and competition for sunlight. Based on simulation-based modeling and self-organizing tree models, a promising further research direction would therefore be a more sophisticated simulation system. Incorporating species specific simulation parameters, such as material properties and environmental factors, could lead to more expressive and biological accurate modeling system.

This interesting topic could benefit from a combination with probabilistic models for development of growth models, not only to find plausible configurations of 3D structures from 2D sketches, but also to allow for appearance transfer, and growth based editing operations.

Finally, the idea of image space quality measures has potential to be extended to non-stochastic simplification algorithms. While model space error metrics answer the question how well the geometric details during simplification are preserved–and thus keeping the focus on the model itself– they provide little insight into how much these details matter for a certain rendered image, including scene configuration and projected scene size. This aspect, how important a detail given a certain screen size is, is intuitively encoded in image space measures. However, for general models the view-independence property of image space measures no longer holds and thus results for one view cannot be as easily generalized and transferred to other views as for natural scenes.

Bibliography

Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3): 507–516, 2005.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302, 1986.

Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proc. of SIGGRAPH '00*, pages 113–120, New York, NY, USA, 2000. ACM Press.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

James F. Blinn. A generalization of algebraic surface drawing. *ACM Transaction on Graphics*, 1(3):235–256, 1982.

Billy Chen, Boris Neubert, Eyal Ofek, Oliver Deussen, and Michael F. Cohen. Integrated videos and maps for driving directions. In *Proc. of the ACM symposium on User interface software and technology*, UIST '09, pages 223–232. ACM Press, 2009.

Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based tree modeling using markov random field. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '08)*, 27:109:1–109:9, December 2008.

A. Chodorowski, U. Mattsson, M. Langille, and G. Hamarneh. Color lesion boundary detection using live wire. In *Proceedings of SPIE Medical Imaging: Image Processing vol. 5747*, pages 1589–1596, 2005.

Yung-Yu Chuang, Douglas E. Zongker, Joel Hindorff, Brian Curless, David H. Salesin, and Richard Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *Proc. of SIGGRAPH '00*, pages 121–130, New York, NY, USA, July 2000. ACM Press/Addison-Wesley Publishing Co.

Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '01)*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

Robert L. Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. *Computer Graphics (Proc. of SIGGRAPH '87)*, 21: 95–102, August 1987.

Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. Stochastic simplification of aggregate detail. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26(3):79, 2007.

Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. *ACM Transactions on Graphics (Proc. of SIGGRAPH '03)*, 22(3):657–662, 2003.

Philippe Decaudin and Fabrice Neyret. Rendering forest scenes in real-time. In *Rendering Techniques (Proc. of EGSR '04)*, pages 93–102, 2004.

Philippe Decaudin and Fabrice Neyret. Volumetric billboards. *Computer Graphics Forum*, 28(8):2079–2089, 2009.

Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2003)*, 22(3):689–696, 2003.

Oliver Deussen and Bernd Lintermann. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer Verlag, 2004.

Oliver Deussen and Bernd Lintermann. *Digital design of Nature - Computer Generated Plants and Organics*. Springer-Verlag, 2005.

Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. In *VIS '02: Proceedings of the Conference on Visualization '02*, pages 219–226, 2002.

David S. Ebert, Oliver Deussen, Ronald Fedkiw, F. Kenton Musgrave, Przemyslaw Prusinkiewicz, and Jos Stam. Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH '03: Course Notes 41, 2003.

Ismael Garcia, Mateu Sbert, and Laszlo Szirmay-Kalos. Leaf cluster impostors for tree rendering with parallax. In *Eurographics '05 Short Presentations*, pages 69–72, 2005.

Andrew J Hanson, Hui Ma, and Lindley Hall. Parallel transport approach to curve framing. *Indiana University TechreportsTR425*, pages 1–20, 1995.

Paul S Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6:56–67, November 1986.

Roger W. Hockney and James W. Eastwood. *Computer simulation using particles*. Taylor & Francis, Inc., 1988.

Matthew Holton. Strands, gravity and botanical tree imagery. *Computer Graphics Forum*, 13:57–67, February 1994.

Hugues Hoppe. Progressive meshes. In *Proc. of SIGGRAPH '96*, pages 99–108, New York, NY, USA, 1996. ACM.

Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics (Proc. of SIGGRAPH '05)*, 24(3):720–726, July 2005.

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

Jan Klein, Jens Krokowski, Matthias Fischer, Michael Wand, Rolf Wanka, and Friedhelm Meyer auf der Heide. The randomized sample tree: A data structure for interactive walk-throughs in externally stored virtual environments. *Presence: Teleoper. Virtual Environ.*, 13(6):617–637, 2004.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Johannes Kopf, Boris Neubert, Billy Chen, Michael F. Cohen, Daniel Cohen-Or, Oliver Deussen, Matt Uyttendaele, and Dani Lischinski. Deep photo: Model-based photograph enhancement and viewing. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '08)*, 27:116:1–116:10, December 2008.

J. Dylan Lacewell, Dave Edwards, Peter Shirley, and William B. Thompson. Stochastic billboard clouds for interactive foliage rendering. *Journal of Graphics, GPU, and Game Tools*, 11(1):1–12, 2006.

Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. Object recognition by affine invariant matching. In *Computer Vision and Pattern Recognition (Proc. CVPR '88)*, pages 335–344, 1988.

Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical report, University of North Carolina at Chapel Hill, 1985. TR 85-022.

Aristid Lindenmayer. Mathematical models for cellular interaction in development,parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

Bernd Lintermann and Oliver Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19:56–65, January 1999.

Hans-Andrea Loeliger. An Introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, January 2004.

David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science, 2002.

Thomas Luft, Michael Balzer, and Oliver Deussen. Expressive illumination of foliage based on implicit surfaces. In *Proc. of the Eurographics Workshop on Natural Phenomena*, pages 71–78. Eurographics Association, 2007.

Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York, 1983.

Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG '95)*, 1(2):99–108, 1995. ISSN 1077-2626.

Cecil D. Murray. The physiological principle of minimum work applied to the angle of branching of arteries. *The Journal of General Physiology*, 9(6): 835–841, 1926.

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

Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proc. of SIGGRAPH '96*, pages 397–410, New York, NY, USA, 1996. ACM.

Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:88:1 – 88:10, July 2007.

Boris Neubert, Soeren Pirk, Oliver Deussen, and Carsten Dachsbacher. Improved model- and view-dependent pruning of large botanical scenes. *Computer Graphics Forum*, 30(6):1708–1718, 2011.

Karl J. Niklas. *Plant Allometry: The Scaling of Form and Process*. The University of Chicago Press, 1994.

Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum (Proc. of Eurographics '05)*, 24(3):487–496, 2005.

Peter E. Oppenheimer. Real time design and animation of fractal plants and trees. In *Computer Graphics (Proc. of SIGGRAPH '86)*, volume 20, pages 55–64, New York, NY, USA, August 1986. ACM.

Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch, and Przemyslaw Prusinkiewicz. Self-organizing

tree models for image synthesis. *ACM Transactions on Graphics (Proc. of SIGGRAPH '09)*, 28:58:1–58:10, July 2009.

Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (Proc. of SIGGRAPH '03)*, 22:313–318, July 2003.

Sören Pirk, Till Niese, Oliver Deussen, and Boris Neubert. Capturing and animating the morphogenesis of polygonal tree models. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia '12)*, 2012a.

Sören Pirk, Ondrej Stava, Julian Kratt, Michel Abdul Massih Said, Boris Neubert, Radomír Měch, Bedrich Benes, and Oliver Deussen. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics (Proc. of SIGGRAPH '12)*, 31(4):50:1–50:10, July 2012b.

Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics (Proc. of SIGGRAPH '84)*, 18:253–259, January 1984.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

Przemyslaw Prusinkiewicz, Mark Hammel, and E. Mjolsness. Animation of plant development. In *Proc. of SIGGRAPH '93*, pages 351–360, New York, NY, USA, 1993. ACM.

Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomír Měch. L-systems: from the theory to visual models of plants. In *Proc. of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, volume 3, pages 1–12. CSIRO Publishing, 1996.

Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *Proc. of SIGGRAPH '01*, pages 289–300, New York, NY, USA, 2001. ACM Press.

Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH '06)*, 25:599–604, July 2006.

Alex Reche-Martinez, Ignacio Martin, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (Proc. of SIGGRAPH '04)*, 23(3):720–727, August 2004.

William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 313–322, New York, NY, USA, 1985. ACM.

Jean Paul Richter. *The Notebooks of Leonardo da Vinci*, volume 1. Dover Publications Inc, New York, 1970. reprinted 1888.

C. J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2 edition, 1979.

Yodthong Rodkaew, Prabhas Chongstitvatana, Suchada Siripant, and Chidchanok Lursinsap. Particle systems for plant modeling. In B. Hu and M. Jaeger, editors, *Plant Growth modeling and Applications (Proc. of PMA '03)*, pages 210–217, 2003.

Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Proc. of the Eurographics Workshop on Natural Phenomena*, pages 63–70. Eurographics Association, 2007.

Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. of SIGGRAPH '00*, pages 343–352. ACM Press, 2000.

Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 18–25, 2000.

Paolo Sabella. A rendering algorithm for visualizing 3d scalar fields. *Computer Graphics (Proc. of SIGGRAPH '88)*, 22:51–58, June 1988.

David Salomon. *Curves and Surfaces for Computer Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

Gernot Schaufler, Wolfgang Stürzlinger, and Johannes Kepler. A three dimensional image cache for virtual reality. *Computer Graphics Forum*, 15 (3):227–236, 1996.

Daniel Scherzer and Michael Wimmer. Frame sequential interpolation for discrete level-of-detail rendering. *Computer Graphics Forum (Proc. of EGSR '08)*, 27(4):1175–1181, June 2008.

K. Shinozaki, K. Yoda, K. Hozumi, and T. Kira. A quantitative analysis of plant form - the pipe model theory I. Basic analysis. *Japanese Journal of Ecology*, 14:97–104, 1964a.

K. Shinozaki, K. Yoda, K. Hozumi, and T. Kira. A quantitative analysis of plant form - the pipe model theory II. Further evidence of the theory and its

application in forest ecology. *Japanese Journal of Ecology*, 14:133–139, 1964b.

Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21:53–61, May 2001.

Mike Sips, Boris Neubert, John P. Lewis, and Pat Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum (Proc. of EuroVis '09)*, 28(3):831–838, 2009.

Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Proc. of SIGGRAPH '96*, pages 259–268, New York, NY, USA, 1996. ACM.

Marc Stamminger and George Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques '01 (Proceedings of Eurographics Workshop on Rendering)*, pages 151–162, 2001.

Josef Stoer and Roland Bulirsch. *Numerische Mathematik 2*. Springer, 1978. ISBN 3-540-08840-7. 5. Auflage.

Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Transactions on Graphics (Proc. of SIGGRAPH '04)*, 23, August 2004.

Ivan E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, Lincoln Lab, 1963.

Richard Szeliski. *Computer Vision: Algorithms and Applications (Texts in Computer Science)*. Springer-Verlag New York Inc, 1st edition, November 2010.

Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)*, 26:87:1 – 87:8, July 2007.

Stanislaw M. Ulam. Pattern of growth of figures: mathematical aspects. In G. Keps, editor, *Module, Proportion, Symmetry, Rhythm*, pages 64–74. Braziller, New York, 1966.

Michael Wand and Wolfgang Straßer. Multi-resolution rendering of complex animated scenes. *Computer Graphics Forum (Proc. of Eurographics '02)*, 21 (3), 2002.

Michael Wand, Matthias Fischer, Ingmar Peter, Friedhelm Meyer auf der Heide, and Wolfgang Straßer. The randomized z-buffer algorithm: interactive rendering of highly complex scenes. In *Proc of SIGGRAPH '01*, pages 361–370, 2001.

Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *Proc. of SIGGRAPH '95*, pages 119–128, New York, NY, USA, 1995.

Andrew Witkin and David Baraff. Physically based modeling: Principles and practice. SIGGRAPH '97: Course Notes, 1997.

Douglas E. Zongker, Dawn M. Werner, Brian Curless, and David H. Salesin. Environment matting and compositing. In *Proc. of SIGGRAPH '99*, pages 205–214, New York, NY, USA, July 1999. ACM Press/Addison Wesley Logman.