HAL
open science

Survey on Computer Representations of Trees for Realistic and Efficient Rendering

Frédéric Boudon, Alexandre Meyer, Christophe Godin

To cite this version:

Frédéric Boudon, Alexandre Meyer, Christophe Godin. Survey on Computer Representations of Trees for Realistic and Efficient Rendering. [Research Report] 2301, LIRIS UMR CNRS 5205. 2006. hal-00830069

HAL Id: hal-00830069

https://inria.hal.science/hal-00830069v1

Submitted on 4 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Survey of Computer Representations of Trees for Realistic and Efficient Rendering

Frédéric Boudon Alexandre Meyer Christophe Godin

***Abstract*— This paper gives an overview of computer graphics representations of trees commonly used for the rendering of complex scene of vegetation. Looking for the right compromise between realism and efficiency has lead researchers to consider various types of geometrical plant models with different types of complexity. To achieve realist plant model, a complex structure of plant with full details is generally considered. In contrast, to promote efficiency, other approaches summarize plant geometry with few primitives allowing rapid rendering. Finally, to find a good compromise, structures with adaptive complexity are defined. Theses different types of representations and the ways to use them are presented, classified and discussed.**

The proposed classification principles rely on the type of structural details used in the plants representations. Characterization of all these methods is completed with various additional criteria including rendering primitive type, distance validity, interactive possibilities, animation ability and lighting properties.

***Index Terms*— I.3.6 [Computer Graphics]: Methodology and Techniques – Graphics data structures and data types. I.6.9 [Simulation, Modeling, and Visualization]: Visualization – Visualization techniques and methodologies. Trees, Real–time, Geometrical modeling, Multiscale Representation, Image-based rendering, Point-based rendering, Shaders.**

I. Introduction

Reproducing natural scenes as realistic as possible has always been a major goal of computer graphics. Over the last decades, computers and techniques have evolved to a point where highly complex scenes can be routinely created. Many urban visualization applications and games already achieve high level of realism. However, vegetation, which is an important part of natural scenes, is not always convincing.

Plants, and especially trees, are extremely complex objects, generally composed of numerous organs with particular arrangements in 3D space. Rendering realism therefore depends on particular characteristics of vegetal scenes:

- *Complexity of organ shapes*. At the human scale of perception, organs like flowers, leaves, fruits, trunks and branches often have complex shapes with varying degrees of symmetry, and lots of significant details.
- *Variety of organ shapes*. A complex natural scene may contain dozens to hundreds of plant species, showing an important variety of organ shapes.
- *Number of organs*. Plants and plant communities commonly contain a huge number of organs or individuals.
- *Spatial organization of organs*. The spatial distribution of plant organs obviously respects particular rules that depend on each species: branches are connected to each other, leaves are arranged along axes following various types of phyllotactic patterns and inclination angles, branches bend in particular ways and directions, etc.
- *Plant aspect drastically varies with observation distance*. At close distances, a plant usually exhibits a branching system with detailed organs. At higher distances, the same plant gets homogenized due to the aggregation of many individual details: the complexity of its branching system is less visible and may completely disappear, individual leaves in the crown cannot be distinguished from one another, and the overall impression of the crown is that of a fuzzy volume with a rather regular surface boundary.

The difficulty of modeling and rendering plants reflects this natural complexity. As a consequence, computer representations of plants need to be adapted to particular rendering situations. For close views, fine-grain details are required, while for distant views simple models are preferred to avoid excessive computation and aggregation artefacts. However, in many situations (during zooming operations or during a walk-through in a natural scene), intermediate models whose complexity can be adapted to their visual importance are required. These different types of models can coexist in the same application but requires different construction and vizualization techniques. However, most classical rendering optimization and simplification techniques that successfully apply on regular objects (see [LRC+02] for a review), visually produce non-adequate results for plants due to their disparate aspect. To face this problem, researchers have developed in the last decade a wide range of techniques and strategies to efficiently model and simplify the rendering of plants.

The goal of this paper is to survey the numerous computer representations of plants and the associated modeling and rendering strategies used to represent vegetation in computer graphics. In a previous work, Mantler *et al.* emphasized techniques used for 3D real time rendering and proposed a classification of the approaches based on rendering primitives [MTF03]. In this work, we introduce an alternative classification based on plant representation structures, and on secondary additional criteria which enables us to review a wide spectrum of rendering approaches. Note that we do consider geometrical modeling of plant organs but we do not address the problem of how plant branching structures are generated which is an important related topic (for detailed reviews see [PHHM97],

Frederic.Boudon@cirad.fr, ISA Team, INRIA Lorraine, Nancy, France.
Alexandre.Meyer@liris.cnrs.fr, LIRIS Lab, Lyon, France. This paper was begun at GRAVIR Lab.
Christophe.Godin@cirad.fr, UMR AMAP, Virtual Plants Team, INRIA-CIRAD-INRA, Montpellier, France.

[DL05]), but which would extend too much the scope of this review. Similarly, we do not address techniques of occlusion culling, which can be used to accelerate the visualization of natural scenes (a survey of this topic can be found in [COCSD02]).

To organize the various reviewed approaches, we first classify the different types of plant representations according to their level of complexity (Section II). This classification principle is then used as a guiding canvas for the survey. The different techniques used to represent plant modules and branching systems in a detailed approach are depicted in Section III, with a special focus on plant organ design. Subsequently, global strategies to render efficiently plant models are discussed in Section IV. Eventually, intermediate (multiscale) approaches that make an attempt to combine the advantage of both detailed and global approaches are revieewed in the two last sections : structural-based approaches in Section V and spatial-based approaches Section VI.

II. Classification

A. The structural criteria of plant representations

Our classification is first related to the level of details of the plant representation. Such taxonomy was already considered by Godin [God00] for discussing functional-structural models of plants. Three main axes can thus be defined (see Figure 1):

- *Detailed representations.*
  To achieve realistic representations, plants are decomposed into simples modules (leaves, internodes, *etc.*) and a geometrical model is built from their union. Such approaches aim at defining continuous geometry of branching systems and realistic geometry of leaves and barks with efficient and intuitive techniques. The possibility of representing a large amount of details make them particularly adapted for close views.
- *Global representations.*
  By contrast, at the lowest level of complexity, plants are considered as a whole. Some works attempt to represent trees with a single or few primitives to drastically simplify vegetal scene rendering. Those representations are usually adapted for distant views.
- *Multi-scale representations.*
  Intermediate approaches handle the whole range of viewpoint distances using representations with adaptive complexities (levels of details LOD). Thus, the third part of our classification is dedicated to techniques which define several scales of representations. Such multi-scale approaches can be divided into two. A first category make use of the hierarchical structures and the repetition of similar objects in the structure to setup simplification scheme, using instantiation or organs clustering for example. A second category refers to approaches that consider multi-scale organizations based on the spatial proximity of the elements. Since the tree structure is not needed, these second category of approaches has the advantage of being relatively independent of the modeling process.

In addition to these structural axes, we identified several criteria that should be taken into account when reviewing vegetation representations for rendering.

B. Rendering primitive criteria

The rendering primitive has direct impact on the result, in term of realism and computational time. In the techniques presented in this paper, we distinguish four rendering primitives:

- *Polygon.*
  Computer graphics representations are commonly based on Polygon. Nevertheless, due to the huge number of resulting primitives, the rendering of forest scene is very costly and very subject to aliasing.
- *Image-based.*
  With the cost of a single primitive, an image can represent a very complex object. Image-based rendering (IBR) has become a major technique for rendering real-world scenes acquired as images, or models with very high polygon counts. A large number of image-based rendering methods addressing different kind of problems have been proposed in the last few years [ZC04]. Despite the low cost of an image, IBR introduce many limitations around diversity, lighting, and animation because of the static nature and the relatively high memory cost of an image.
- *Volumetric approach and shader-based.*
  Trees belong to this category of objects that have no defined surfaces, since details merge with distance. An idea is to replace the indistinguishable data by a fuzzy primitive that would reproduce the same photometric behavior (*reflectance model* or *shader*) than the group of geometry it represents. The two main advantages of shader in comparison to an explicit representation of details with geometry are the gain in term of computation time and the low level of aliasing. Nevertheless, shaders can be difficult to parameterize or derive.
- *Point-based.*
  A majority of objects in naturals scenes paradoxically covers only fractions of pixels. The traditional advantage of polygon-based scan-line coherence is thus lost, while resources are wasted. An interesting recent alternative is to use point-based rendering [LW85], [PZvBG00], [RL00], [MZG01]. The idea of drawing many simples points having roughly the size of a pixel can be conveniently apply to render vegetation. The number of drawn primitives can be adapted to the distance to ensure interactive frame rate. However, to build a coarse level, points are averaged and are generally made opaque to avoid sorting and blending. This makes anti-aliasing difficult and change the visual appearance to a more opaque object.

C. Additional classification criteria

- *Off-line or realtime rendering.* Is the presented algorithm suitable for realtime rendering? If no, does

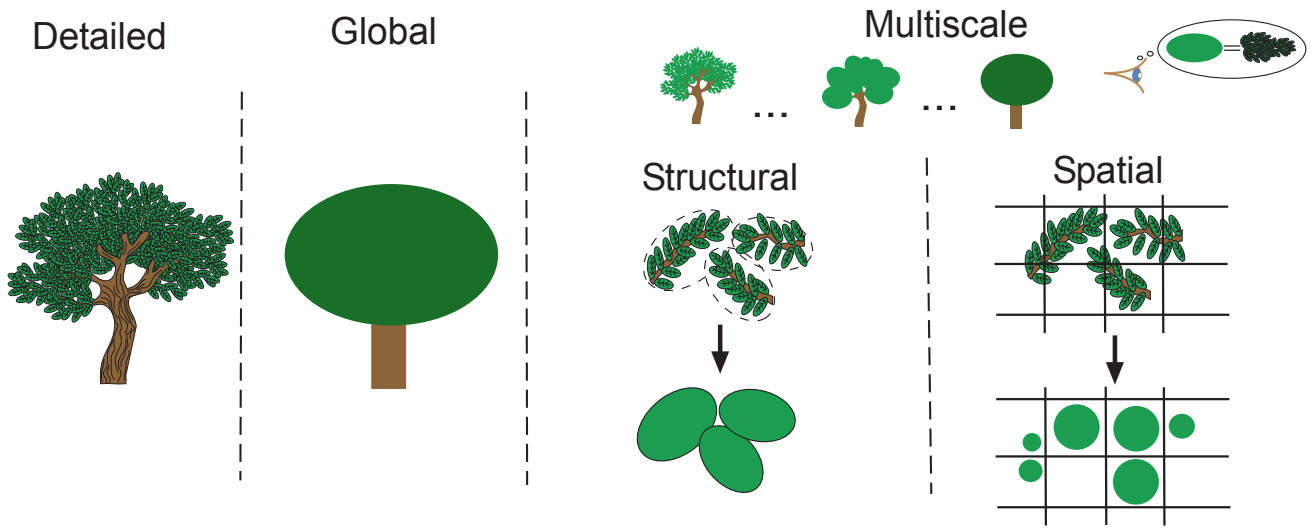

Fig. 1. Classification based on structural complexity: First, detailed representations are considered; then global representations which represent efficiently plants with few primitives and finally multiscale models with adaptive complexity. Multiscale models are divided into two classes: Models that are build using the plant modules hierarchy and models build using generic space organization rules.

the use of off-line rendering allow to improve realism ?

- *Distance validity of the technique*. What is the distance range at which the method provide accurate results?
- *Animation.* Does the method allow animation of branches, leaves? This question is mainly addressed to realtime methods.
- *Lighting characteristics.* Several points characterize a lighting:
  - Does the lighting take into account the self-shadowing (branches and leaves of the tree shadow other parts of the same tree) ?
  - In a forest, a tree is also shadowed externally by neighboring trees. Does the lighting compute shadows cast by other trees? Which shadow technique is used?
  - Can the lighting be dynamic (for realtime rendering)? If yes, what are the limitations?
- *Need of pre-computation.* Does the method build the representation as a pre-process? Or does the method generated the representation on the fly? This point induced an impact on many other aspects like the diversity allowed, the capacity of having a dynamic lighting and animating.
- *How does the method build the representation?* Does it use a specific modeling process adapted to a particular application? Or does it convert a standard modeling method's output? Or does the method simply start from a classical polygonal representation?
- *Vegetation specific or more general.* Few presented techniques are not specific to vegetation. Nevertheless, we include them in this survey because they were mainly illustrated on vegetation.

III. Detailed representations

Detailed representations aim at figuring the geometry of tree in the most realistic manner for close views. Trees are generally divided into trunk, branches and leaves which require different modeling and rendering techniques. Branches are generally built with a set of connected geometrical primitives on which textures representing barks can be applied. In this section, modeling techniques used to build the geometry of branches are first presented (Section III-A). Then, we discuss how more realistic rendering of branching systems can be achieved by using bark textures on branches (Section III-B) and detailed representations of leaves (Section III-C).

A. Branching systems

In the design of realistic trunk and branches representations, the challenge is to create continuous models from a discrete set of geometrical primitives. The main difficulty is thus to define an accurate representation of the branch junctions. Various solutions have been proposed in the literature.

1) Pioneering approaches: The simplest step in the quest for realism of branching structures is the representation of branch segments as 3D cylinders. To minimize memory cost, De Reffye *et al.* [dREF+88] use a library of symbols (such as polygonal cylinders) that are reused in different positions, with various dimensions and orientations, and which represent different instances of the same module (internode, growth unit, segment, *etc.*). Position and orientation of two successive modules can be defined relatively using turtle geometry [PL90]: to draw new organ, commands are passed to a LOGO-style turtle that process them in the continuity of the previous drawn organs. Kawaguchi [Kaw82] propose circular polyhedron to represent continuously the geometry of the slices of branching pattern (see Figure 2).

Unfortunately, these techniques do not properly capture the geometry of branch junctions, since some gaps or discon-

tinuities between elements may appear. Several subsequent solutions have been proposed to address this problem.

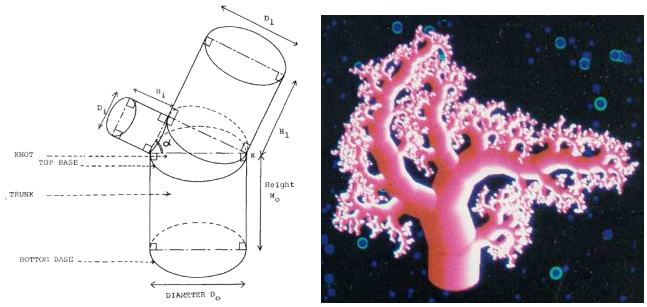

Fig. 2. Left: Branching point geometry with circular polyhedron according to Kawaguchi [Kaw82]. Right: Example of branching structure.

2) Cone–Sphere: Max [Max90] introduced the cone–sphere primitive to model branches. A cone–sphere consists of two spheres with a cone tangent to the two spheres and lying between them (see Figure 3). The outer surface of an elbow is formed by a piece of sphere resulting in a smooth junction of surfaces. But at the inner side of the elbow, the two cones intersect directly, leading to discontinuity of the tangents. A blending method is introduced to improve this based on a simple linear interpolation scheme. While cone–spheres suffice as an efficient representation of individual limbs, they offer no help in the blending and texturing at branching points. Different extensions of this approach with implicit formulation have been subsequently proposed [HB96], [Mar03], [GMW04a].

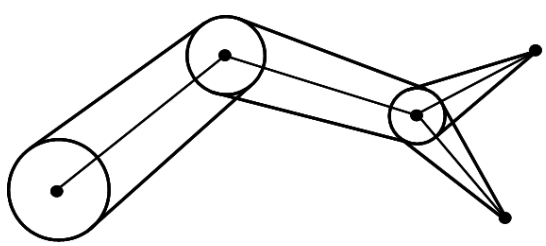

Fig. 3. Cone-Spheres used to model branches.

3) Generalized cylinders: To improve realism of branches and branching points, Bloomenthal [Blo85] introduced generalized cylinders. His approach starts from a skeleton with a position associated with each nodes. To obtain continuity of the first and second derivatives at each node, the skeleton is built from the cubic spline interpolation of each set of n edges ($n+1$ nodes) forming a branch.

The geometry of each branch is then built with a generalized cylinder, obtained by sweeping a planar generating curve, determining the branch's cross section, along the skeletal spline curve. The specific difficulty of modeling realistic branching points is handled by connecting several generalized cylinders using several parametric surfaces forming a "saddle" as shown in Figure 4.

To build the mesh of the generalized cylinder surface, a certain number of sections are evaluated along the skeleton and are connected. Even if not considered in this work, this parametric model additionally allows different degrees

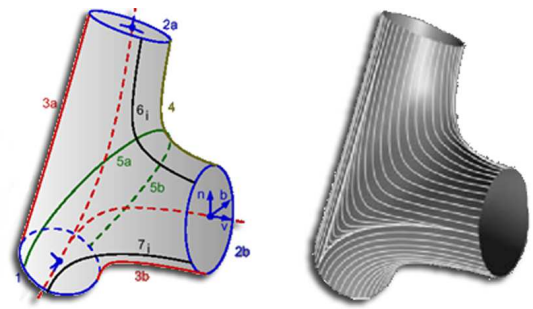

Fig. 4. Representation of the branching points with a "saddle" surface, built by crafting several parametric surfaces. On the right, texture parametrization [Blo85].

of polygonization and thus offers a basic multi-resolution scheme. Figure 5 illustrates the visual quality of the result.

Fig. 5. The mighty maple [Blo85].

4) Implicit surfaces: Bloomenthal also introduced implicit surfaces [Blo95] as a general framework to fit surfaces of branching structures. An implicit surface S is defined as the set of points $P = (x, y, z)$ at which the value of a field function $F(x, y, z)$ is equal to 0.

Implicit surfaces are an intuitive framework for modeling smooth blending of branching structures in computer graphics. As for generalized cylinders, an underlying skeleton may be used. Each skeletal primitive s_i contribute to F with a field function f_i which is decreasing with the distance to this element.

The general equation of an implicit surface is thus defined as:

$$S = \{P = (x, y, z) \in \mathbb{R}^3, F(P) = \sum_i f_i(P) = 0\} \qquad (1)$$

Each skeletal component is then represented using an implicit surface primitive, which have the inherent ability to blend smoothly with each other. In contrast to other methods, implicit surfaces process uniformly all branching structures regardless of their complexity.

However, the way each f_i varies affects the blend so that generating a smooth shape is not an easy task. A common

side effect of implicit surfaces is bulging, an unnatural-looking increase in girth where two or more branches are blending together (see Figure 6). To avoid this, different types of field functions have been proposed.

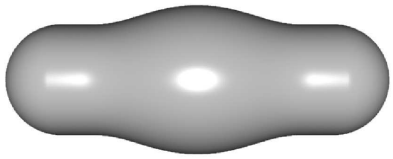

Fig. 6. Bulge when two line segments with coincident end points are blended [GMW04a].

First, Bloomenthal and Wyvill in [BW90] define *distance surfaces* to obtain branches of constant radius at the joint. Using distance surfaces, f_i are defined as decreasing C^1 continuous functions G of the distance to s_i (defined as the distance to the closest point on s_i):

$$f_i(P) = G(d(P, s_i)) \tag{2}$$

The choice of functions f_i with local support limits the influence of each skeletal component, thus providing local control of the shape and minimizing computations. Hart [HB96] generalized the distance surface method to the blending of branches of different radius. Ferley *et al.* [FCA97] proposed a general solution based on the local thickening of the skeleton near a joint using Bzier triangles as extra skeletal elements to prevent the creases on the corresponding distance surface. This solution is however not convenient to apply texture.

To avoid bulging, *convolution surfaces* were also introduced [BS91] and defined their field functions as:

$$f_i(P) = \int_{s_i} e^{-\|P-X\|^2/2} dX \tag{3}$$

Convolution surfaces based on skeletal line segments are bulge free for non-ramiform structures, but do exhibit bulging for ramiform structures. Using a polygonal skeletal structure with convolution removes bulging in ramiform structures, but cannot produce circular cross sections [GMW04a]. In general, convolution surfaces impose an increase in both computational and implementation complexity. Jin *et al.* [JFP01] use convolution surfaces with polynomial weight distributions to represent trees and other ramiform structures.

These works aim at producing representations which are globally smooth. Galbraith *et al.* [GMW04a], [GMW04b] generalize implicit representations of branching system with smooth and non-smooth blending at the joints, to take into account the branch bark ridges or branch collar of a tree. For smooth blending of branches without bulging, skeletal primitives are simply shortened to avoid interaction between fields. The non-smooth blending at the branching points are obtained using precise contact modeling [Can98] which consist of deforming implicit surfaces interpenetrating each other. In the interpenetration region, a deformation term is introduced in the field function to simulate the compression of the two surfaces due to their contact. This creates a first contact surface in the interpenetration region. At the same time, dilating fields are applied in a propagation region defined around the interpenetration region (see Figure 7). These terms create local object dilation that appears around the contact surface. These deformations yield C^1 continuous surfaces. Eventually to create scars where organs have been lost, some negative potential fields can also be used and result in an indentation into the surface.

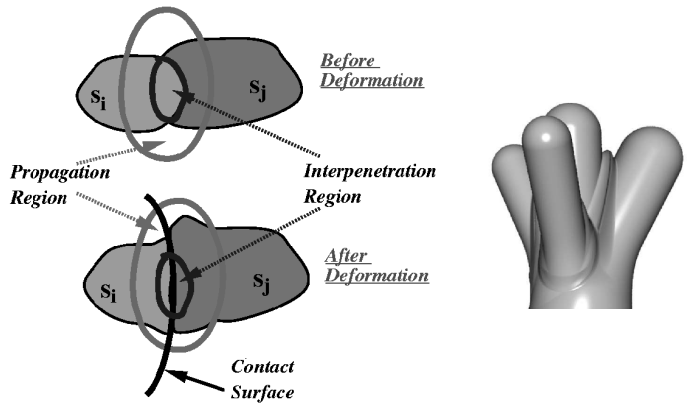

Fig. 7. On the left, precise contact modeling regions. On the right, bulging modeled with PCM on high level branching situation. From [GMW04a].

The various implicit formalisms presented in this section demonstrate the intuitiveness of such approach to model precise geometry of branches. Unfortunately implicit surfaces are computationally expensive due to complex polygonization schemes [BBB+97]. Galbraith *et al.* [GMW04a] report 60 seconds to render the poplar model shown in Figure 8 with OpenGL and one hour for ray trace.

Fig. 8. Implicit representation of a poplar tree with smooth and non smooth blending [GMW04a].

5) Subdivision surfaces: An alternative method to produce smooth surface is to use subdivision schemes that build mesh

by recursively refining an initial coarse surface. Tobler *et al.* in [TMW02] explored subdivision surfaces for the representation of branching structures and coupled them with a surface growing mechanism. Their system use generalized subdivision scheme where vertex refinement rules can be different at each subdivision step. The rule based mesh growing system is an extension of Parametric L-systems [PL90] where each parameterized symbol represent a face of the mesh. Using both of these mechanisms in combination, the authors demonstrate that a great variety of complex object can be easily modeled and compactly represented. Additionally, their modeling technique is multi-resolution. The authors suggest then to extend it to an on–the–fly generation system that can then provide realistic and interactive rendering in the spirit of methods described Section V-A. However, the creation of the initial subdivision mesh for arbitrarily complex branching structures was identified as a difficult issue. A formal approach of such a rewriting system for mesh refinement and growth has been proposed in [SPS03] and was recently used in the context of modeling leaves growth [RFL+05].

Related techniques using subdivision were used by Aitken *et al.* [AP03] in cinema industry to design and animate plants.

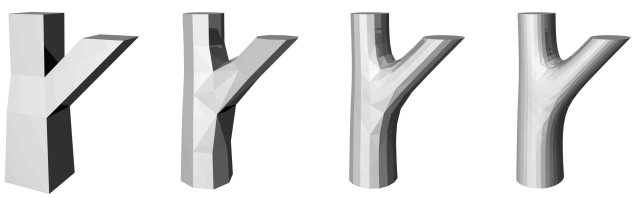

Fig. 9. Subdivision surfaces with mesh subdivision of a branching structure. From [TMW02].

B. Bark textures

Works on realistic bark textures fall into two categories : first, static dressing which add fine details to the branch geometry using alternative representations such as textures and, second, mechanical simulations which generate the bark geometry using different physically based formalisms. Related works are presented in this section.

In both cases, particular attention must be paid to surface parametrization, which is necessary to map textures on geometric models. Tiling homogeneity of textures elements on organ surface can be noticeably affected due to important diameter variations along a branch and to branching junctions, which may cause unwanted distortions and discontinuities in the bark texture.

1) Static dressing: A pioneering work in this area was done by Bloomenthal [Blo85] who use bump mapping to render fine details of the bark geometry. The bump map is computed according to an X ray scan of an actual bark plaster face. The map gives a texture's tile that is repeated along the branches. In order that the texture repeats seamlessly, the top and bottom rows must be identical, as must be the left and right columns. This is accomplished by overlapping opposite edges of the map which blurs the boundary connection. Surface parametrization is derived from generalized cylinders and ramiform surfaces at the junctions (see Section III-A.3 and Figure 4). These parametrization schemes proved to yield reasonable texture coordinates despite residual continuity and distortion problems.

Different solutions to the problem of texture continuity at the joints, based on implicit modeling, have been introduced by Hart in a series of works. In a first method [Har97], the texture of a branch is smoothly blended by interpolation with the texture of the bearing branches at the joint (see Figure 10, left). In this case, the texture is simply generated with a fractal noise function, and stretched along the skeleton axis. Hart and Baker [HB96] also introduce the use of particle flow along branches to compute texture coordinates, with special case of merging and collapsing of particle at the joints (see Figure 10, right). Volumetric textures are then applied on each particle, converted into a polygon. It can be noted that defining such texture elements for various types of bark would be particularly uneasy.

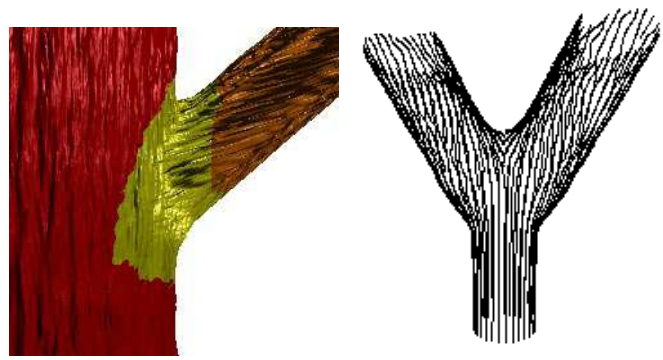

Fig. 10. Texture continuity at branching points. Left: Blending of the two textures of branches in the yellow region. From [Har97]. Right: Flow particle on branches to compute texture coordinates. From [HB96].

To improve realism, displacement mapping for bark texture are introduced by Wang *et al.* [WWT+03] on a simple branch. This technique allows to move each points of a surface in a given direction according to a height map, providing better relief impression than bump mapping. This function is implemented in recent graphic cards to achieve real time rendering. Wang *et al.* provided construction techniques of displacement maps from bark photography [WWL+03] and view-dependent visualization with shading and self-shading [WWT+03]. Maritaud [Mar03] used such displacement maps with implicit surfaces and simple parametrization of branching (See Figure 11).

2) Simulation: For the generation of realistic bark texture, most works, such as the one proposed by Federl and Prusinkiewicz [FP96], rely on mass–spring networks mapped onto the tree surface, where the springs break beyond a constraint threshold. Hirota used a bi-layered mass-spring network [HKK98], extending to bark their paper on cracks [HTK98]. Both methods provide textures with small scale cracks, but cannot represent open fractures since depth of cracks is not taken into account. Moreover, they require a huge number of springs to get interesting results, as details are only created by the simulation.

Lefebvre and Neyret [LN02] use mass–spring method with

Fig. 11. Top: Displacement maps for bark. From [WWT+03]. Bottom: Displacement maps applied on branching structure. From [Mar03]

a simplified physical based approach. They propose a method of bark generation which produces either geometry or texture and handle only fracture-based bark. Given that a tree grows mostly on its circumference, they consider independent circular strips of bark perpendicular to the axis of the tree. Each strip are modeled as a one–dimensional circular system of two alternating types of springs, one representing the material, the other representing the fracture. The fractures are initiated inside a strip when one of the material springs was elongated beyond its threshold. The crack are modeled by splitting this material spring and inserting a fracture spring. This semi-empirical model runs in interactive time, and allows automatic or influenced bark generation with parameters that are intuitive for the artist/designer.

Continuity problems at the joins and variation of diameter can be resolved accurately with simulated barks. Lefebvre and Neyret take this into account in their system : subsequent texture strips of different widths and lengths can be coherently generated and generated cracks propagate on only one particular strip of a branching pattern.

Fig. 12. Computer generated barks [LN02].

An alternative solution to mass-spring network is proposed by Federl and Prusinkiewicz [FP04] who introduced a method based on solid mechanics combined with finite element method. In this system, a bi-layered surface is considered : the top layer, representing phloem that consist of dead conductive tissues for downward circulation of the sieve, is expanded by the radial growth of the bottom layer, representing cambium inside the trunk. The surface layer is first discretized into prisms and a stress tensor is computed at each node. If the stress at a node exceeds a threshold, a fracture is initiated and extended according to the nodal stress tensor. Their method incorporates several acceleration scheme including local mesh refinement and local multi-resolution calculation of nodal stress. Accurate results require between 60 and 150 thousand elements and two hours of simulation instead of eight without acceleration schemes, showing viability of finite element method for fracture patterns. However, interactive design or rendering is difficult to achieve for a complete tree.

C. Leaves

Most of plant geometry related works suppose little variability of leaves shapes within the same tree to simplify foliage rendering. A few leaf representations are thus built with simple textured polygons and are instantiated in various part of the plant structure. For instance, Bloomenthal [Blo85] build maple leaves from digitized photographs. The resulting texture is mapped onto a tree-polygon structure (see Figure 13). The structure is hinged along dashed lines; the degree of hinging depends on the strength of an hypothetical wind.

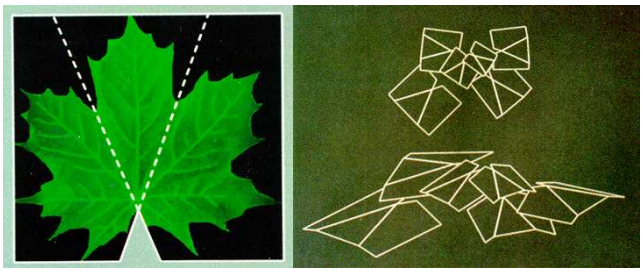

Fig. 13. Left: Digitized maple leaf textured on a tree-polygon structure. Right: Different configurations of the leaf model with various degree of hinging [Blo85].

More elaborated representations use multi-resolution approaches to display foliage of tree. These methods (presented in detail in Sections V and VI) provide realistic representations of foliage. However, individual leaves generally have poor visual quality. Several works explore modeling techniques to achieve large variety of realistic leaf mockups.

When the shape of the leaf is more complex (curled, non-planar, *etc.*), complex leaves shapes can be modeled using spline patches [PL90]. Such approach is suitable for animation: the control points are positioned at the endpoints of the structure and evolve over time.

Sweep surfaces also provide a useful technique for modeling leaves [LD99], [PMKL01]. In the simplest case, a planar generating curve, which determines the organ's cross section, is swept along a carrier path and producing an extrusion (see Figure 14). The generating curve may be closed, as is typically the case of stems, or open, as for thin leaves. Furthermore, it can be scaled according to a profile curve and may change shape while being swept.

Fig. 14. Leaves created with sweep surfaces. The surfaces are generated with L-systems by sweeping a generating open curve along an axis representing the leaf skeleton. During the sweeping, a scaling function is applied to the generating curve to follow the variation of the leaf's width [PMKL01].

These previous works are adapted to achieve high quality of non-branched leaf geometry. However, lobed leaves with branching patterns are difficult to build.

Hammel *et al.* proposed a modeling method for 2D models of leaves defined from a branching skeleton [HPW92]. A margin is build around the skeleton with an implicit contour (a one-dimensional counterpart of implicit surfaces). Branching skeleton is specified using L-systems while the implicit field functions of the contour is defined for each point according to its distance to the closest point of a skeletal element. The field function may vary along a skeletal segment in order to achieve a large variety of contour.

Mündermann *et al.* [MMP03] extend the sweep method for lobed leaves defined from a branching skeleton. Their modeling process starts with a 2D silhouette of a leaf, which can be scanned or defined interactively using a curve editor. The silhouette's skeleton is then defined interactively or computed using a 2D Voronoï diagram of the silhouette, and is represented as interconnected sticky splines [vO96] that preserve topological relationship during subsequent manipulation. The leaf surface is constructed by sweeping a cross-section between the skeleton and the silhouette. For this, the end and branching points of the skeleton are connected in a clockwise order to their closest points on the silhouette, forming segments of leaf. The segments are then subdivided into quadrilaterals (see Figure 15). So that, leaf texture can be subsequently applied.

The planar leaf model obtained in this way can be deformed in 3D space by functions that control its turning, bending and twisting along the axes of the skeleton. With such method, visually important aspects of leaf appearance can be captured and controlled interactively, allowing the design of a large variety of leaves (see Figure 15). It also gives a very good starting point for animation. It can be noted that the good visual quality of these models requires an important number of polygons that can become critical for rendering a big leafy tree, or a forest. In a next step of realism, Runions *et al.* [RFL+05] proposed recently a class of biologically-motivated algorithms for generating leaf venation patterns (See Figure 16). Their algorithms simulate the development of veins by taking into account the hormone distribution process during leaf growth.

Realistic images of leaves requires a sophisticated shapes

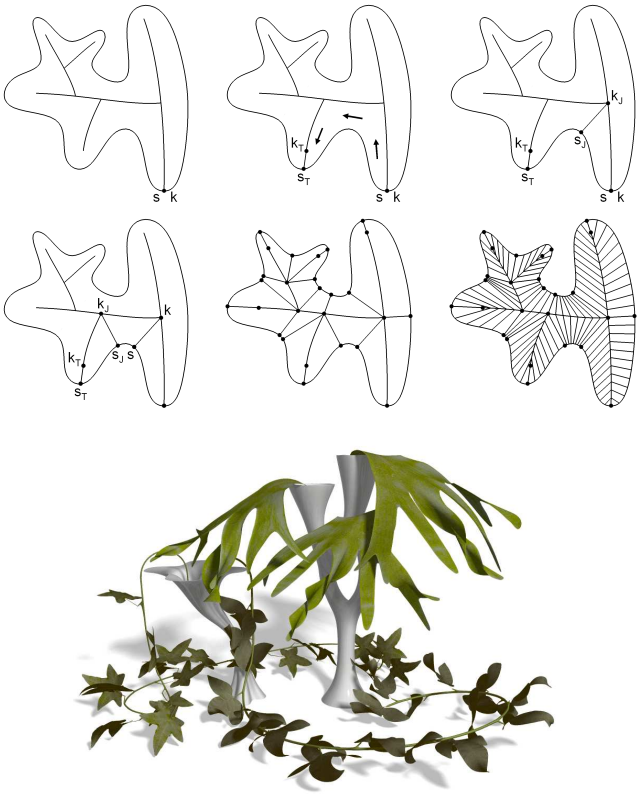

Fig. 15. Construction of a sweep surface for lobed leaves [MMP03]. Left: Starting from a silhouette and a skeleton of a leaf, end and branching points of the skeleton are examined in a clockwise manner and connected to the silhouette. The resulting leaf segment are subdivided into quadrilateral. Right: A close-up scene of leaf models individually deformed from scanned images.

Fig. 16. Left: Wang *et al.* [WWD+05] introduced a parametric model that describes leaves in terms of spatially-variant BRDFs and BTDFs and obtain realistic leaves images in realtime helped by graphics hardware. Right: Runions *et al.* [RFL+05] proposed recently a biologically-based simulation for generating leaf venation patterns.

but also accurate appearance model and lighting computation. Thus, Wang *et al.* [WWD+05] describe leaf appearance in terms of a few parametric bidirectional reflectance distribution functions (BRDF[1]) and bidirectional transmittance distribution functions (BTDF[2]). These spatially-variant BRDFs and BTDFs are compactly stored in a set of parameter maps, which can be loaded into graphics hardware for fast on-the-fly shading calculations. On real leaves, they measure the BRDF

[1]The BRDF is a function of incoming (light) direction and outgoing (view) direction which describes how much light is reflected when light makes contact with a certain material.

[2]The BDTF is a function which describes how much light is crossing a semi transparency material.

and BTDF which incorporate subsurface scattering inside leaf tissues and rough surface scattering on leaf surfaces. Their graphics hardware based algorithm extends the Pre-computed Radiance Transfer (PRT) approach to all-frequency lighting for leaves. In particular, it handles the combined illumination effects due to low frequency environment light and high-frequency sunlight by decomposing the local incident radiance of sunlight into direct and indirect components. The direct component, which contains most of the high frequencies, is evaluated on-the-fly using pre-computed light-visibility convolution data.

IV. Global representations

Many of the current interactive applications such as virtual reality environments or computer games take place in outdoor scenes. Trees and plants are an essential part of these scenes. As shown in the previous section, detailed tree model are formed by such a vast number of primitives that real-time visualization of scenes with trees seems impossible on such bases. To alleviate this difficulty, simple global tree representation, mainly based on images, have been developed to provide efficient rendering at distant views.

A. Billboard

Billboards are the most common tool for realtime rendering of forests. Thanks to their low cost, they are still considered the best choice in many recent industrial simulators. They are used in two different ways: classical billboards are small images that always face the camera (possibly with an axis constrained to be vertical) while cross billboards consist of two to four regular textured quads crossing each other. The first method shows no parallax when the camera moves, which is acceptable only in the case of objects symmetric around a vertical axis and grazing view angles. Moreover, a tree slightly behind another is likely to pop in front at a given angle. For this reason, billboard-based forests are usually sparse. The second method shows artifacts whenever one of the quads is seen at a grazing angle. In addition to the poor visual quality, billboards requires a pre-computation step of the plant images. Additionally, illumination is stored on the image leading to difficulties for dynamic lighting. For the same reason, animation of branches of leaves is not possible. Moreover, there is no simple Level-0f-Detail (LOD) scheme to gather individual billboards.

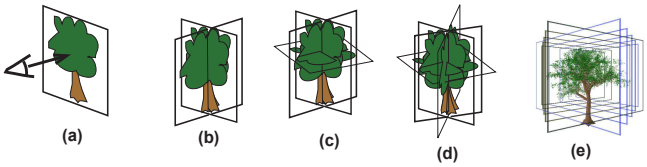

Fig. 17. (a) billboards. (b) cross billboard. (c) 3-cross billboard. (d) 4-cross billboard. (e) billboard slices to represent crown foliage [Jak00].

B. Quadric Surfaces

Early global methods which do not have parallax problems induced by image based representation such as billboards was proposed by Gardner in [Gar84]. His approach use textured quadric surfaces, like ellipsoids, bounded by an arbitrary number of planes to represent natural objects such as tree crowns. The global shape of the plant is then represented with a simple volumetric object with low cost while the irregularity of the density of foliage is still represented with procedural textures, applied to the surfaces. Various results with impressionist aspect are obtained as shown in Figure 18. However, due to its historical context, this work did not examine real time and animation rendering. Additionally, the textures seem difficult to parameterize to achieve more realistic results.

Fig. 18. Textured quadric surfaces [Gar84].

C. Basic Slicing

An other global solutions which gives some volume to the tree was proposed by Jakulin in [Jak00]. He extended the cross billboard representation to render the crown and combines it with traditional polygonal geometry rendering for the trunk and limbs of a tree. Indeed, the crown foliage is rendered using multiple parallel layers (slices) in the three orthogonal directions (see Figure 17.e). The group of slices for a specific view direction is called a slicing. During preprocessing, several sets of these slices are created from various view points. For each slicing, the primitives (*i.e.* individual leaves) are assigned to the closest slice. Each slice is then rendered to an individual texture. During rendering, the two slicings that are closest to the actual view direction are rendered simultaneously with correct transparency and blending.

The goal of this algorithm was to accommodate architectural walk-through and driving simulations. Therefore, viewing trees directly from above or below the tree is not supported. All slices are perpendicular to the ground plane, and blending between two sets provides sufficient coverage of the foliage. Notice that Section VI-A.2 presents related technique but with the benefic of levels of details and arbitrary viewpoints.

D. Extended billboard with depth and shading information

An other billboard-inspired technique is the one proposed by Qin *et al.* in [QNTN03] with the goal of fast rendering photo-realistic images of trees under various kinds of daylight. Their approach is to convert a 3D model of tree in a representation they named quasi-3D tree. A quasi-3D tree is a combination of 2D buffers which store geometrical and shading information of tree surfaces, *i.e.* their color (classic billboard), normal vectors, relative depth, and shadowing of direct sunlight and skylight.

In addition to these set of 2D buffers which are used to represent the tree perpendicularly to the ground plane as billboard enriched with depth information, they store horizontal mask images for casting shadows (see Figure 19).

The shadows of direct sunlight and skylight are processed in two steps: self-shadow and external-shadow. Self-shadow, which is the most important shading information and the most time-consuming task, is examined in preprocess for different viewing directions, colors and positions of the sun. These different directions are chosen by the user to limit the amount of data to store (see Section V-B.3 describing [MNP01] for an automated version of this sampling). External-shadows, which depend on the circumstances, are constructed during the rendering process. The shadows, including umbrae and penumbrae, are examined by employing several shadow maps (named Sun-Shadow Buffers in the article) since light sources are the sun and the sky.

Using a two-step shadowing algorithm, this method can create high quality forest scenes illuminated by both sunlight and skylight at a low computation cost (few minutes on a Pentium III at 800Mhz). It can generate both umbrae and penumbrae on a tree casted by other trees and any other objects such as buildings or clouds. Transparency, specular reflection and inter-reflection of leaves, which influence the delicate shading effects of trees, can also be simulated with verisimilitude. Nevertheless, the drawback of billboards will be visible in animations when walking around a tree, walking through or over-viewing at a forest. Thus, they suggest to compute images viewed from multiple directions similarly to techniques presented in Sections V-B.2 and V-B.3.

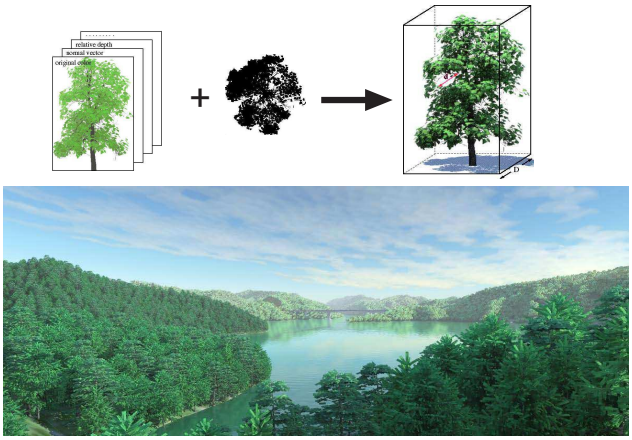

Fig. 19. Top: 2D tree buffers with geometrical and shading information + horizontal mask images for casting shadows resulting in a rendered quasi-3D tree. Bottom: a resulting landscape. From [QNTN03].

TOWARD MULTISCALE REPRESENTATION

As one can see in previous sections, accurate modeling of plants and trees requires large number of primitives. A typical outdoor scene contains many plants and trees, each of them made of hundreds of thousands of primitives. Even with modern graphics rendering algorithm (using hardware acceleration or not), the rendering time of this large number of primitives will not be acceptable. Moreover, the majority of objects in such scenes often cover only a few, or even a fraction of pixels on the screen, thus leading to use expansive anti-aliasing in the rendering. In contrast, global simplification method propose simple representation with a fixed level of details that look unrealistic at close view. To reduce the amount of rendered primitives representing a tree according to its visual importance without loosing realism, some multi-resolution or multiscale representations have to be setup.

Classical multi-resolution methods to reduce the number of polygons of an object using geometry-based simplification exist [HG97], [Hop98]. These methods, however, fail to capture the nature of plants and trees. They tend to aggregate polygons from different leaves or different branches resulting in a smaller tree with a shape visually different from the original one. Moreover, this process also alters the lighting behavior of the tree, inducing non realistic popping artefacts.

Several solutions dedicated to vegetation have been explored using various alternative graphic primitives. The structure of the tree play an important role on the possible simplification. Its multiscale hierarchy of components (generally defined by the branching order) define intuitively several levels of abstraction that can be used for simplification. Repetitive patterns of components can also be used to reduce the number of geometrical primitives to compute. The use of this type of structuring lead to a first category of multi-scale techniques presented in the next section. More generic methods build hierarchy of primitives by regrouping or merging them according to their spatial proximity. These methods are presented on Section VI.

V. MULTISCALE REPRESENTATIONS BASED ON PLANT STRUCTURE

As shown in Figure 20, tree structure is naturally hierarchical. Thus, it is natural that many techniques based their hierarchical representations (LOD) on it. This section describes this family of rendering techniques with a progression going from realtime and low memory consuming techniques to off-line and high level of realism.

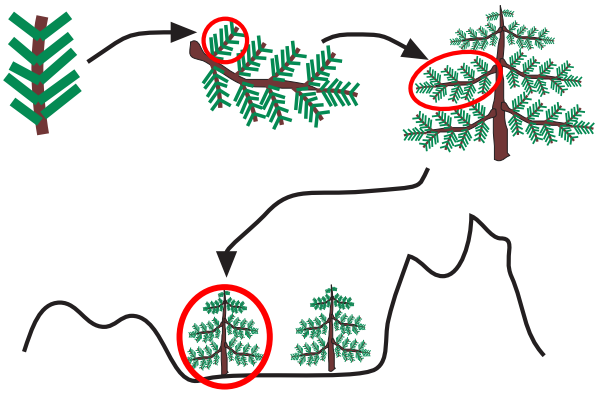

Fig. 20. Trees structure is naturally hierarchical. Trees are composed of a trunk and main branches, which group small branches together. Boughs are made of leaves or needles; and leaves, needles, boughs, branches resemble each other. These properties allow many modeling simplifications.

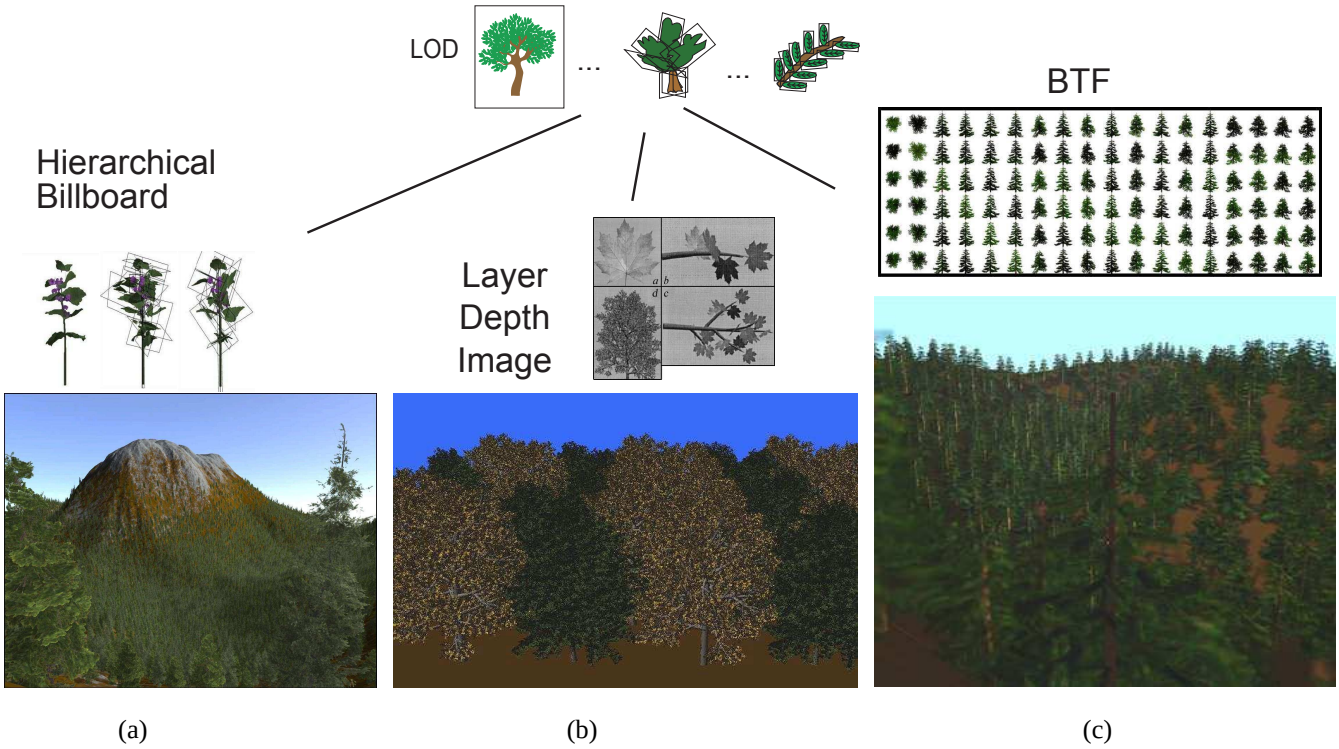

Fig. 21. The three main IBR families of LOD-scheme based on the structural nature of trees. At coarse level, the whole tree is represented by an IB primitive. At detailed level, each leaves is represented by an IB primitive. And at intermediate level, various branches are represented by IB primitives. The IB primitive can either be *(a)* billboard [BCF+05], *(b)* layered depth image [MDK99] or *(c)* either bidirectional texture [MNP01].

A. On the fly procedural generation

A natural way to build a multi-resolution representation is to include the simplification methods directly in the modeling process. Lluch *et al.* [LCV03] developed such an approach which they call procedural multi-resolution. They build procedural models based on parametric L-systems that reflect a tree visual structure at different levels of resolution. They start from a parametric string representing a tree and generate a new string with embedded multi-resolution information. The algorithm is based on a metric that quantifies the relevance of the branches of a tree. This approach allows the generation of various trees in a forest, but the data storage is still an important issue.

If the generation process is fast enough to be done in realtime, an interesting idea is then to do it on the fly only when needed, avoiding so storage issue. It allows thus the rendering of incredibly complex and detailed scenes. This idea was first apply to terrain by fractal based modeling process. Perbet *et al.* [PC01], [GPR+03] generate and animate blades of grass on-the-fly and only where the camera is looking. They use three levels of detail. Geometry and animation data related to grass are deleted once the camera is gone. Digiacomo *et al.* [GCF01] extend the idea to the animation and the interaction with trees: procedural method handles most of the trees efficiently, and physically-based method allows user interaction. Similar idea has been implemented in the commercial software [BD].

B. Image-based Representations

The good properties of an image to represent very complex objects with one single primitive can be used it in the case of trees. Various optimizations try to solve classical IBR issues (memory cost, static illumination, lack of relief, etc.). The use of the plant hierarchy to define several levels of IBR allows to bypass the issue of the fixed level of details of images. Figure 21 summarizes the three main existing approaches.

1) Hierarchical billboards or simplified textured trees: An extension to the billboard approach presented in Section IV-A is to build a simplified tree with a few dozen to a few hundred polygons approximating the foliage distribution (see Figure 21.a) [BCF+05], [CCDH05], [IDV02], [BD]. A hierarchy of LOD is built going from a simple billboard to trees with hundred polygons. Behrendt *et al.* built billboards of a tree by clustering elements of a same level in the hierarchy.

The first issue is the brutal transition from one level to another leading to popping effect. To decrease this phenomena, [IDV02] performs a per pixel transition: when the camera get further some pixels from fine polygons disappear by transparency and some others appear on a rough polygon. Thus, the number of polygons to render decreases. The second issue of this kind of approach is the static lighting encoded on images which can be tackle by storing a normal for each pixel and computing the shading during the rendering with a fragment program (per-pixel lighting). Note that this remains a coarse approximation. Simple animation of branches is possible only for near LOD. As results from [IDV02], [BCF+05] show a fly above a landscape, made of several hundred of thousand

of trees, in realtime and with dynamic lighting on NVidia GeforceFX graphics processor.

2) Hierarchical precomputed multi-Layer Z-buffers: Flat images as billboards suffer from lack of relief. IBR techniques based on depth try to improve this [CW93]. Z-buffer images are pre-computed from different fixed viewing positions in order to reproject them and produce an image from a new viewpoint. Max *et al.* in [MO95], [Max96], [MDK99] derived this idea to tree. Z-buffer images are pre-computed for twigs and branches at various levels in the hierarchical structure of a tree (see Figure 21.b), and adaptively combined depending on the position of the new viewpoint. The pre-computed images contain multiple Z levels to avoid missing pixels in the reconstruction, subpixel masks for anti-aliasing, and colors/normals for shading after reprojection. If a subobject is too close to the viewpoint, the polygons of the original model are rendered. In [MDK99], they use the graphics hardware to perform the reprojection faster. As a result, they compute scenes like the one shown in Figure 21 which count ten millions of polygons in several minutes on SGI Onyx, InfiniteReality. The pre-computed data occupy around ten megabytes of texture memory.

3) Hierarchical bidirectional texture function: An other way to limit the flatness effect of billboards is to fade nearest pre-computed views (see Figure 21.c) as described in [PCD$^+$97] for any kind of objects. The set of pre-computed views taken from various light direction is called Bidirectional Textures (BT) related to the Bidirectional Textures Function (BTF). This alpha blending between textures introduces a ghosting artifact and transparency variation (duplicated features) which show up when one moves around a tree: blending two half-faded textures is not equivalent to a single opaque texture. This can be limited by pre-computing a large set of views. Relying on even more images would correspond to bidirectional textures [SBLD03] and light-fields [LH96], [SGHS98]. But the huge amount of image data may not fit in the graphics memory.

Meyer *et al.* in [MNP01] apply this idea to trees which is well suited for instantiation and then limits the memory issue. Their data structure consists of a hierarchy of BT linked to a hierarchy of visibility cube-maps (6 shadow maps forming a cube surrounding an object) to compute the shadow. An example of hierarchy for a given tree can be a small branch plus its leaves (or needles), a larger branch, and the entire tree. A BT provides a billboard image of a shaded object (small branch, large branch or tree) for each pair of view and light directions. A BT is associated for each level of the hierarchy. The illumination reaching each tree elements is evaluated using the corresponding visibility cube-map. On SGI Onyx 2000, they achieve 7 to 20 fps fly-through of a scene with thousands trees with dynamic shading, self-shadowing and shadowing. The main drawback of their method is the amount of pre-computed data which lead to massively use instantiation and limit the diversity of trees.

C. Point and line based

In recent years, it has been shown in [MZG01], [RL00], [PZvBG00] that point-based rendering is a very efficient means for the rendering of any complex geometry, idea supported by the efficient capacity of the hardware to treat it. This approach is based on the observation that in increasingly complex scenes triangles become smaller than a single pixel. In this case, triangle based scan-line rendering wastes time in superfluous sub-pixel computation. In opposite, points merge easily. The degree of detail can thus be fluently adapted by adding or removing single points.

1) Particle Systems: The forest model of Reeves *et al.* [RB85] was probably a source of inspiration for many works there after, especially on point-based representation. They primarily emphasized the forest environment instead of concentrating on the structural detail of individual plants. In addition, they focused more on the visual results than the specific details of actual botanical data. Results are very impressive at this time (see Figure 22.b), even if computing times were about several hours.

Their stochastic modeling approach, called particle systems, builds complex pictures from sets of simple, volume-filling primitives. The particle systems processes the trees in the data base serially; it generates a complete model for each tree and then renders the model. Starting with the main trunk, the algorithm constructs the tree by recursively generating sub-branches. The data structure for the model is a tree in which each node describes a branch segment. The structure is parsed and particles are drown into the frame buffer for each branch. In addition, they describe approximate and stochastic algorithms for shading. Probability of a particle to be shadowed or highlighted depends on its position in the crown. Highlights occur where the tree's branches or leaves are exposed directly to sunlight *i.e.* close to the outer edge of the tree in the direction of the sun. For self-shadowing, the ambient and diffuse component drops off exponentially as d -the distance into the tree from the light source- increases as shown in Figure 22.a.

2) Realtime techniques: Later and as a natural evolution of Reeves model, Weber *et al.* [WP95] and Deussen *et al.* in [DCSD02] present two models to render in real-time trees with a combination of polygons and points/lines representation. They also introduce the idea that best performance can be achieved if the plants are not uniformly simplified: hidden parts can be coarser than visible parts.

To be able to render a large number of trees, the rendering algorithm can progressively reinterpret the geometry of the tree: branches meshes as lines and leaf polygons as points. The geometry (polygons, points and lines) is stored in vertex arrays on the GPU. At far distance, some individual branches and leaves will simply disappear altogether in the Weber *et al.* approach. This could lead to important visual artefacts. Thus, Deussen *et al.* merge disappearing elements in larger primitives (points or lines) to better convey the shape in distant views. To select which part of the trees has to disappear, Weber *et al.* use an automatic criteria on the size whereas Deussen *et al.* ask the user to make the selection during the modeling process.

[DCSD02] showed scenes representing several tens of million of polygons rendered at interactive frame rate, with a NVidia Geforce3 graphics processor like the one presented in

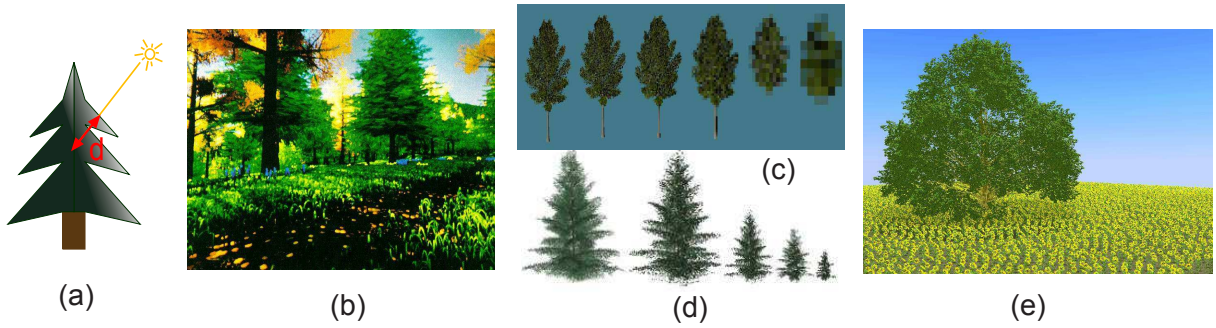

Fig. 22. Techniques where LOD are based on the structural nature of trees and where the rendering primitive is the point and line. Early [RB85] introduced *(b)* a particle-based rendering technique associated with *(a)* a stochastic shadowing where the portion of received light is function of the distance to the hull. With distance *(c)* [WP95] and *(d)(e)* [DCSD02] progressively re-interpret the tree: branches meshes as lines and leaf polygons as points.

Figure 22. The detail can also be increased automatically for high resolution images. To improve realism, these approaches allows to implement a wind effect by moving the trees elements during the rendering. And shadowing within the trees can be produced using a standard shadow map technique [RSC87].

D. Illumination based

Beside the rendering primitive type, computer graphics researchers works on illumination model. An interesting idea is to replace the indistinguishable data by a fuzzy primitive that would reproduce the same photometric behavior of the geometry it represents. This idea is used in many generic computer graphics models for surface or volume [Bli77], [Bli78], [Bli82]. In the specific case of trees, geometric details like needles or leaves are so small that they usually cannot be seen except for the nearest trees. Thus, it is a well suited case to develop shaders. Moreover, photo-realistic images can be obtained by a global illumination simulation. Thus, some works search to perform this heavy simulation to trees scenes.

1) Shaders of needles branches: Meyer and Neyret in [MN00] proposed primitives and shaders at several scales for the particular case of the pine-tree or fir-tree as shown in Figure 23. Depending on the distance, the primitive they consider is either the needle (level 1), the cone (level 2), or the bough (level 3). They computed an analytical formula representing the global reflectance and opacity at each of those levels, including the internal shadows. They use several approximation and optimization to simplify and compute the formula. The analytical properties of this set of three shaders allow to represent at various scales the cumulated effects of the smaller scales, without sampling them. In particular, it includes the internal shadows, and takes the visibility into account.

They implemented this model in a cone-tracer which reduce drastically the number of traced ray: direct rays and shadow rays. As results, images of hundred fir-trees were computed in around 10 minutes on *SGI Onyx2*, which was ten time faster than the classical ray-tracer *Rayshade*. The major issue of this approach is the difficulty to extend it to other family because of the pre-condition of strong a priori knowledge on the matter distribution which is not available for many object.

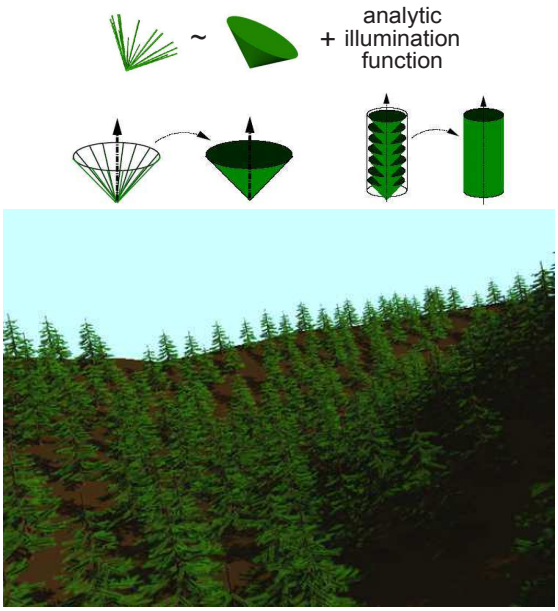

Fig. 23. [MN00] proposes three levels of analytical shaders in order to represented complex geometric structure such as branches of pine-trees. The analytical aspect allows fast computation without sampling and low aliasing in a cone-tracer renderer.

2) Global illumination: Radiosity is a global illumination methods. Each polygon has radiative properties and contributes to illumination of all other polygons of the scene. Thus, the basic resolution quadratically depends of the number of considered elements, and is particulary expensive in memory and computation time. To limit this cost, techniques build hierarchical clusters replacing a group of geometric objects with a simple shape with equivalent radiative properties [Sil95]. The exchange of energy between each pair of elements are computed at the high level of the hierarchy. This method allows to adapt the complexity of the system to resolve: the choice of the depth in the hierarchy at which the system is resolved is a compromise between precision of the solution and computation time. However, such methods are limited in scene size because of the super linear amount of memory they

require in terms of the number of input polygons.

Various techniques exists to automatically build the clusters hierarchy from polygonal representations. The main one consists of decomposing the space into an octree, recursively defined as voxels of variable sizes [Sil95]. The sizes of the voxels are defined to fit the local irregularity of the shapes. Soler *et al.* in [SSBR03] take advantage of the structuring and the self-similarity of the plants, to propose another decomposition of the space that minimize the number of cluster and allows instantiation. The clusters are used to replace the branching systems at the different levels of hierarchy in the plant structure. The same cluster, used as reference for different parts of the plant model, contains reflectance and transmittance functions simulating the energy resulting of interaction of the real geometry with lights from all directions. Such functions are costly to handle, both in term of computation times and storage. However, use of those meta-objects is especially useful in case of high degree of self-similarity in the model like in plants. With instantiation, gains on memory and computation time are important on their examples compared to classical hierarchical radiosity.

VI. Multiscale representations based on rules: spatial proximity and visibility

LOD construction based on the structure of trees has the main advantage to allow simplification on less important parts. Nevertheless, it implies strong coordination with the modeling process which can be tricky. We review in this section techniques basing LOD construction on arbitrary rules which have the main advantage to be disconnected from the modeling. Those techniques allow unstructured input like polygon soup and then build a hierarchy independent of the tree one. Rules on spatial proximity or on visibility are developed. We present techniques based on two well known data structure, octree in Section VI-A and BSP in Section VI-B. In Section VI-C, we give an overview of two techniques based on distance between objects and on visibility. As a consequence, structured-based LOD techniques are often linked to a specific modeling process whereas techniques based on rules converting any kind of input are more practical.

A. Octree and slicing

Octree data structure can be coupled with various rendering primitives as illustrated in Figure 24.

1) Ray-traced volumetric texture: Volumetric textures approach introduced by Kajiya *et al.* [KK89] to represent fur, consists of mapping a 3D layer on a surface using a 3D data set as texture pattern. This is especially adapted to a layer of continuous vegetation covering a landscape as well as many other objects like fur, organic 3D textures, etc. The key idea is to represent the 3D material by a cubic reference volume, to be mapped onto a surface (like a thick skin). The copies of the reference volume, named texels, are fitted upon the surface and are deformed in order to stick to each other. The reference volume is a 3D arrays of voxels. Each voxel encode the local geometry by specifying an opacity, a frame and the reflectance function. The rendering is done by ray-tracing, which becomes a volumetric ray-tracing when a texel is crossed. Whereas illumination function of Kajiya is specific to fur, Neyret in [Ney95], [Ney96], [Ney98] propose a generic one which allowed him to render nice images of forest as the one shown in Figure 24. Ray tracing rendering took only several minutes on SGI Indy. Notice that representing the small data by a reflectance model is related to the shader approach presented in a structural scheme Section V-D.1.

2) Real-Time volumetric texture by slicing: The ray-traced volumetric approach inspired real-time technique [MN98], [LPFH01]. To rely on hardware acceleration the volume is rendered using textured slices. This has nice properties: the parallax is perfect because the generated fragments are really 3D (i.e., at their proper depth), and the filtering is smoothly managed by the texture hardware. Few polygons (64 in [MN98]) are required for a portion of material. On the other hand, the slicing must be adapted to the viewpoint otherwise one could see between the slices at grazing angles. Moreover, the needed amount of texture memory limits the number of slices and leads thus to a repetitive aspect of the global scene.

Decaudin *et al.* in [DN04] specifically adapt it to forest. Their approach is based on repeated instances of patterns with a prism shape (called texcell): a base triangle on the ground vertically extruded. They use two different kinds of texcell. A simple one (regular texcell) where slices are parallel to the ground which gives good quality and efficiency for most location except for grazing view angle. The other one, the silhouette texcells (more expensive) are sliced parallel to the screen and used near the landscape silhouette. Their texcells store a small part of forest (tens of trees) as a 3D texture on the graphic board. They filter it non-linearly by adapting the number of slices to the distance (LOD). They also define an aperiodic mapping of the texcells in the spirit of [NC99] by creating several edge-matching patterns. Their results show complex terrains with several tens of thousand trees at interactive frame rate. In [FPF04], they present a per pixel shading approach to allow dynamic lighting. Notice, that [BCF$^+$05] use a similar approach for their far LOD.

3) Volumetric image-based representation reconstructed from photographs : Still in the grid approach family, Reche *et al.* [RMD04] tackled the challenging problem of reconstructing and rendering a tree from photographs. Thus, on each cell of a recursive grid, they estimate an opacity value and a set of textures from the calibrated pictures. On the recursive grid, the opacity of the cells are computed by an optimization process: cells opacity must match the alpha-mattes (cumulative opacity at the pixels) extracted from the photographs. In each cell, this opacity value is associated to a set of textures extracted from photographs (one texture from each view), which is used as a view-dependent texture during the rendering. Each grid cell is rendered as an alpha blended billboard attached to his center. Basically, each billboard is a view-dependent texture with transparency. Indeed, billboard is textured by the blend (multi-texturing) of the two textures corresponding to the two closest cameras. For the rendering of the whole tree, the algorithm traverse the cells back-to-front, and render the billboard for each cell.

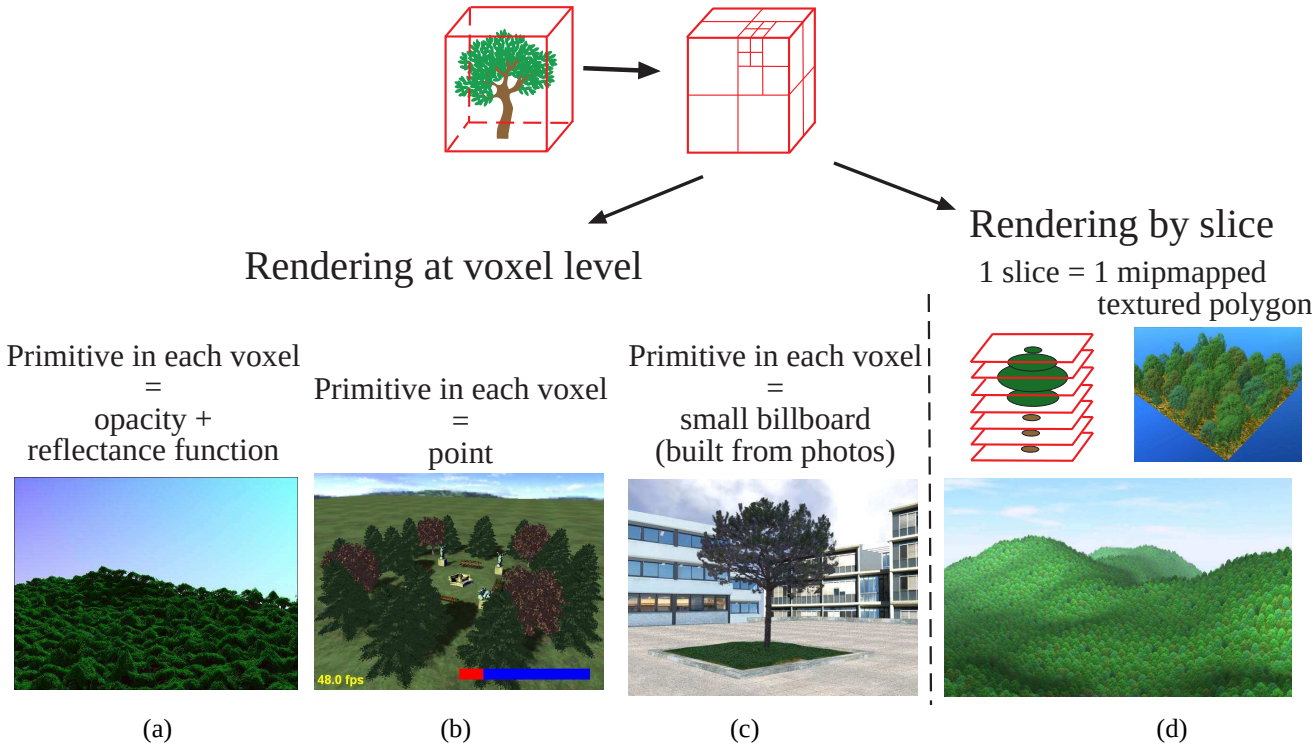

Fig. 24. Most techniques basing LOD construction on arbitrary rules use the spatial proximity to regroup primitives. Thus, octree is a well-suited data structure. On each voxel the rendering primitive differs: *(a)* shaders [Ney98], *(b)* point [DVS03] and *(c)* billboard [RMD04]. In *(d)*, by considering stack of plan in the octree, [DN04] renders a portion of forest with a stack of textured polygon.

As result, the volumetric tree structure allows interactive rendering of realistic 3D trees. Nevertheless, two limitations can be mentioned. First, the method is efficient only on trees with relatively sparse foliage since photographs of such trees typically have single pixels containing the blended projection of numerous leaves/branches and background. It is difficult to capture inside parts of dense trees. Second, it is very difficult to capture the top of the trees which is important in landscape rendering with flying movement.

4) Volumetric point-based representation: Stamminger *et al.* in [DVS03] introduced the sequential point trees data structure which is basically an octree where each cell is a point. In a point-based rendering scheme, this spatial based approach is related to the structural one presented in Section V-C. Their algorithm is not specific to trees but they mainly illustrate it with an outdoor scene including trees. It begins with a set of uniformly distributed point samples on the object (like in [SD01]), which are inserted into an octree to form a point hierarchy. Position and normal values of the children are averaged to obtain the values for the inner nodes. An inner node in the hierarchy represents the union of all its children, so the diameter monotonically increases when going up the hierarchy. The originality of the paper is that the hierarchical rendering traversal of this data structure is replaced by smart sequentialized process on the graphics processor, while the CPU remains available for other tasks. Their result shows a scene with various models including around tens of complex trees. Close views run at interactive frame rate with an ATI Radeon 9700.

B. Binary space partitioning

Marshall and Fussel in [MFI97] propose a multiresolution rendering system that compiled a given botanical description of a plant into a hierarchal volume approximation based on irregular tetrahedra. This is used in an adaptive volume refinement which creates a binary space partitioning (BSP) tree. The partitioning of the space with tetrahedra allows a more adaptive subdivision method than is possible with octrees. Their system is designed for rendering very large collections of randomly parameterized plants.

The procedural volumetric plant model used can be developed at various levels of details and memory usage. The generation of actual geometry for any subvolume can be delayed until it is needed. This drastically reduces memory consumption and initialization time, as the binary tree does not need to be built fully (idea related to Section V-A). This compilation process begins with a full tetrahedral volume as a first approximation to an object, which is then further refined recursively as needed to accommodate individual tree modules. During rendering, explicit polygon are generated for objects that are close to the viewer, while for objects hidden or further away, the bounding tetrahedra are rendered with shading properties approximating their contents as groups of microsurfaces. Result can be seen in Figure 25 on the right. To compute it, software implementation needs several minutes of computation on SPARC Center 2000E. A shading is computed on each vertices.

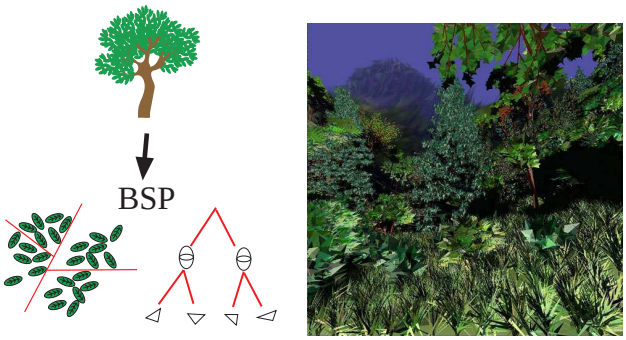

Fig. 25. Similarly to the octree approach, [MFI97] based their LOD construction on a binary space partitioning. Their rendering primitives is either tetrahedra with different size for coarse part or detailed polygons representation for he fine part.

C. Rules-based partitioning

Schemes other than grid or BSP structures were used to cluster tree elements. They are based on two rules families: distance between primitives and visibility.

1) Leaves Collapsing: Remolar *et al.* in [RCB+02] propose a simplification method that diminishes the number of polygons forming the crown of a tree while maintaining its leafy appearance. Traditional geometry simplification [RB93], [LT97], [Sch97], [CCMS97], [PH97] methods are typically not applicable, since leaves consist of many disconnected polygons. The key to their algorithm is leaf collapse. Two leaves are transformed into a single one, so that the area of the new leaf is similar to the area formed by the two initial leaves. In a preprocessing step, a multi-resolution model is created from a sequence of leave collapses. The resulting data structure is therefore created bottom-up as a binary tree, with a polygonal representation of individual leaves (the highest resolution) as the leaves, and the root nodes being the polygons required for simplified representations. An error function is used to determine which pair of leaves will be simplified to create a new one. This function takes into account the distance between two leaves (according to hausdorff distance), and their planarity.

Zhang *et al.* in [ZB03] extended this model by refining the error function with more criteria to promote collapsing of leaves of similar areas and in the first step the simplification of the smallest leaves. The multiresolution model can be traversed classically to obtain different part of the model with different resolution according to some criteria such as screen-space projection of local components [RCRB03], [ZB03].

Fig. 26. [RCB+02] builds their LOD by recursively collapse leaves.

2) Visibility-based selection: Still in the scope of reducing the number of primitives, an interesting idea is to keep only the visible ones. In that way, Guennebaud *et al.* in [GBP04] take advantage of temporal coherency in a very accurate hardware accelerated point selection algorithm. For that, their algorithm is based on a multi-pass algorithm where the main criteria is the visibility. The initialization part consists of a first rendering which gives the list of visible surfels; all hidden points have been discarded by the depth test. For each frame, this list is updated by a rendering of potentially visible surfels. Potentially visible surfels is given by any hierarchical and multi-resolution data structure: an octree [GP03] as in methods presented on Section VI-A.4. The list of visible points is efficiently managed on the GPU. Their technique is not specific to trees but works well for forest because of the high level of occlusion among trees. Globally, their algorithm increases the performance by a factor of 12 on a landscape which contains 1500 visible trees and achieves interactive frame rate (see Figure 27).

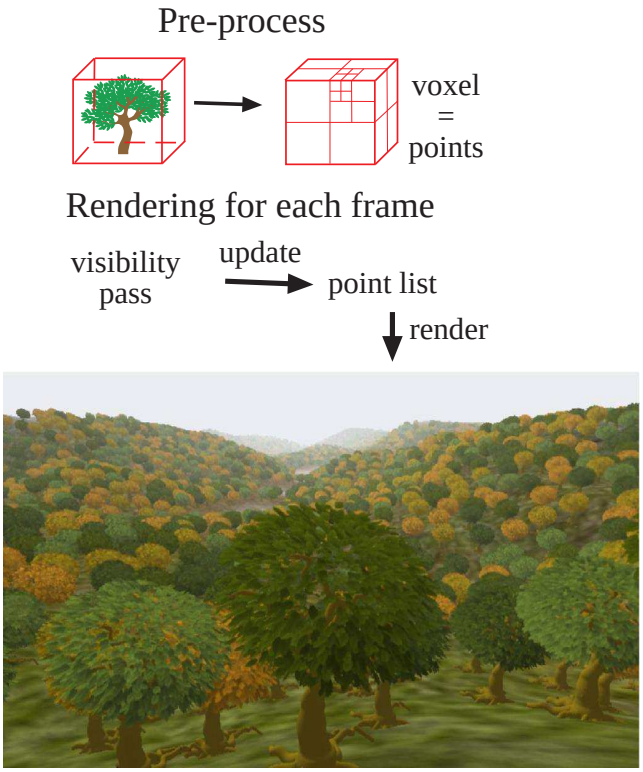

Fig. 27. [GBP04] proposes a point-based rendering technique which maintains a list of visible primitives to optimize the number of rendered primitives. To update this list they use a precomputed octree data structure.

VII. Conclusion

Tables I and II summarize the presented algorithms. Table I is dedicated to detailed techniques aiming at representing realistic trees while Table II summarizes global and multi-scale techniques focusing on efficient rendering. We made an attempt to characterize the methods according to the set of criteria presented in details in Section II, even if some

	Primitive	Perfor-mances	Realism	Lighting	Notes
Mono-scale Representations					
Branching Structure					
III-A.1 Cylinder [PL90], [dREF+88]. p. 3	Polygon	●	○	Dyn.	Simple, with artefacts.
III-A.2 Cone-Sphere [Max90]. p. 4	Implicit	●	◐	Dyn.	Simple, with artefacts.
III-A.3 Generalized Cylinder [Blo85]. p. 4	Polygon	●	●	Dyn.	Complex model.
III-A.4 Implicit Surfaces [BW90], [GMW04a]. p. 4	Implicit	◐	●	Dyn.	Complex to polygonize.
III-A.5 Subdivision Surfaces [TMW02]. p. 5	Polygon	●	●	Dyn.	Complex to define.
Bark					
III-B.1 Bump Mapping [Blo85]. p. 6	Texture	●	◐	Dyn.	With artefact on the silhouette.
III-B.1 Displacement Mapping [WWT+03]. p. 6	Texture	◐	●	Dyn., Sh., S-Sh.	Still costly to render.
III-B.1 Volumetric Texture [HB96]. p. 6	Texture	◐	●	St.	Complex to define and generate lots of polygons.
III-B.2 Polygons [FP04],[LN02]. p. 7	Polygon	◐	●	Dyn.	Generate lots of geometry.
III-B.2 Texture [LN02]. p. 6	Texture	●	◐	Dyn.	Simple, with artefact.
Leaves					
III-C Textured Polygon [Blo85]. p. 7	Polygon	●	◐	Dyn.	Simple, with artefact.
III-C Spline patches [PL90]. p. 7	Polygon	●	●	Dyn.	Classical tool for design.
III-C Sweep surfaces [PMKL01]. p. 7	Polygon	●	●	Dyn.	Limited to linear leaves. Useful for design.
III-C Implicit contour [HPW92]. p. 8	Polygon	◐	◐	Dyn.	Limited to planar leaves.
III-C Sweep surfaces on branching pattern [MMP03]. p. 8	Polygon	●	●	Dyn.	Useful for design. Generate lots of polygons.
III-C Real-Time Rendering of Plant Leaves [WWD+05]. p. 8	BRDF & BTDF	●	●	Dyn. Sh., S-Sh.	Elaborate, realistic and realtime illumination model with transparency.

TABLE I: DETAILED REPRESENTATIONS. From left to right, columns of the table indicates: the rendering primitive type of the method; rendering performance (○ several tens of minutes, ◐ few minutes and ● interactive time); visual realism (○ many artefact, ◐ few artefact for some point of views and ● realistic) and the lighting properties. The lighting properties are characterized with static (noted St.) or dynamic (Dyn.) lighting and shadowing (Sh.) and self-shadowing (S-Sh) support. The realism is a global measure of the visual quality of the different representations that take into account possible artifacts and support of complex lighting and animation. Note that animation is generally always possible with detailed models.

papers are unprecise on some points. Hardware evolution may drastically improve some others.

To summarize, techniques aiming to represent realistic trees correspond to several approaches from simulation to empirical one. A realistic tree is naturally composed of many details. This means that techniques giving realist results produce heavy representations in term of number of elements and in term of complexity of basic elements. This is emphasis in the case of landscape rendering applications. Global models were developed, usually built on image-based schemes. Various techniques taking into account complex lighting and reducing parallax issue have been proposed. Those models provide fast rendering with low memory costs but seem unsuitable for closer views. Multi-scale approaches try to bridge the gap between the two previous types of representations, by defining models with adaptive complexity. However, the definition of such progressive simplification scheme is complex on sparse model like plants. Many approaches have been explored which can be divided into structural and spatial-based. The first one consider plant structure to define efficient LOD and to use instantiation minimizing memory cost. The second divide space with arbitrary rules (octree, bsp, *etc.*) giving more general methods independent from the modeling process. Orthogonally, various rendering techniques/primitives are used. Image-based representation typically involve extensive preprocessing and offline generation of textures which may need to be adjusted for animation. Dynamic lighting can be taken into account but requires memory overhead. Polygon based approaches propose interesting techniques, but rely on geometry modification which can produce unadapted results at highest simplification rates. Point-based offer interesting support to achieve realtime rendering. Different impressive results have been obtained with plants. However, an important number of vertices is required to achieve accurate results. To limit memory consumption the idea of on the fly data generation begin to be exploited.

Globally, none of these techniques perfectly answers all problems at the same time. Nevertheless, interesting compromise between realism and efficiency has been proposed and rendering of large landscape is possible. However, some points such as animation or trees diversity stay an issue. Indeed, coarse levels which are often represented with global primitive are not well suited to animation. Moreover, to solve memory issue, the massive use of instantiation result in a lack of diversity. Structural based approaches combined with on the fly generation appear to be interesting starting points to achieve such results.

REFERENCES

[AP03] M. Aitken and M. Preston. Foliage generation and animation for "the lord of the rings: The two towers". SIGGRAPH 2003 DVD-ROM, ACM SIGGRAPH, New York, 2003. 6

[BBB+97] J. Bloomenthal, C. Bajaj, J. Blinn, M-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to implicit surfaces*. Morgan Kaufmann Publishers, San Francisco (California), 1997. 5

[BCF+05] S. Behrendt, C. Colditz, O. Franzke, J. Kopf, and O. Deussen. Realistic real-time rendering of landscapes using billboard clouds. In *Eurographics*, 2005. 11, 14, 18

[BD] Software Blueberry 3D. http://www.blueberry3d.com. 11, 18

[Bli77] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *ACM Transactions on Graphics (SIGGRAPH 1977 Conference Proceedings)*, pages 192–198, July 1977. 13

[Bli78] J. F. Blinn. Simulation of wrinkled surfaces. In *ACM Transactions on Graphics (SIGGRAPH 1978 Conference Proceedings)*, pages 286–292, August 1978. 13

[Bli82] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *ACM Transactions on Graphics (SIGGRAPH 1982 Conference Proceedings)*, pages 21–29, July 1982. 13

[Blo85] J. Bloomenthal. Modeling the mighty maple. In *ACM Transactions on Graphics (SIGGRAPH 1985 Conference Proceedings)*, volume 19, pages 305–311, San Francisco, 1985. 4, 6, 7, 17

[Blo95] J. Bloomenthal. *Skeletal Design of Natural Forms*. PhD thesis, University of Calgary, 1995. 4

	Rendering	Perfor-mances	Realism	Distance validity	Animation	Lighting	Pre-Computation	Notes
Global Representations								
IV-A Billboard, cross-billboard. p. 9	IBR	●	○	□□■■	No	St.	1 . . . 3 images per kind of trees	Fast, obvious, with artefacts.
IV-B Quadric Surfaces [Gar84]. p. 9	IBR	●	○	□□■■	No	St.	Generated textures	Fast, impressionist results.
IV-C Billboard Slices [Jak00]. p. 9	IBR	●	○	□■■■	No	St.	30 images per kind of trees	Simple solution for depth problem of billboards.
IV-D Billboards with depth and shading [QNTN03]. p. 9	IBR	◐	◐	□□■■	No	Dyn, Sh., S-Sh.	A quasi-3D tree structure per kind of trees $\simeq$ 1.5MB	Solution for lighting problem of billboards.
Structural Based Multi-scale Representations								
V-A Procedural multi-resolution [LCV03], [GPR+03], [GCF01]. p. 11	Polygon	●	◐	■■■□	Yes	Dyn.	None	Adapting generation of the model to visual need.
V-B.1 Hierarchical Billboards [BCF+05], [IDV02], [BD]. p. 11	IBR	●	●	■■■■	Only for near LOD	St.	from 10 to 100 images per kind of trees	Industrial solution.
V-B.2 Hierarchical Multi-Layer Z-Buffers [MO95], [Max96], [MDK99]. p. 12	IBR	◐	●	■■■■	Only for near LOD	Dyn.	Several dozens of images	Resolve billboards depth problem. Difficult to use hardware acceleration
V-B.3 Hierarchical Bidirectional Texture Function [MNP01]. p. 12	IBR	●	●	■■■■	Only for near LOD	Dyn., Sh., S-Sh.	Per tree: 40 MB of texture memory + 20 MB of main memory	Resolve billboards lighting and depth problems. Massive use of instantiation.
V-C.1 Particle systems [RB85]. p. 12	Point	○	◐	□■■■	Yes	Dyn., Sh., S-Sh.	5 to 10 MB of particles for a detailed tree	First work on points with trees.
V-C Realtime point rep. [WP95], [DCSD02]. p. 12	Point	●	●	□■■■	Possible	Dyn., Sh.	Geometry (polygons, points and lines) stored in vertex arrays.	Possible sampling customization according to visual importance of objects.
V-D.1 Hierarchical Shaders [MN00]. p. 13	Shader / Ray-T.	◐	●	■■□□	Possible	Dyn., Sh., S-Sh.	None	Fast in the context of ray tracing. Specific to needles branch.
V-D.2 Radiosity with instantiation [SSBR03]. p. 13	Radiosity	○	●	■■■□	Possible	Dyn., Sh., S-Sh.	26KB to 13 MB by meta-object for reflectance functions	Use of plant self-similarity for instantiation.
Spatial Based Multi-scale Representations								
VI-A.1 Volumetric Texture [Ney98]. p. 14	Volumetric & Ray-T.	◐	●	□■■■	Only global deformation	Dyn., Sh., S-Sh.	29 Mb for each kind of octree(=group of trees).	Fast in the context of ray tracing.
VI-A.2 Volumetric Texture by Slicing [DN04]. p. 14	Volumetric & IBR	●	●	□■■■	Only global deformation	St., Sh., S-Sh.	12 Mb of compressed textures per texel.	Volumetric approach using hardware applied to forest.
VI-A.3 Volumetric image-based rep. [RMD04]. p. 14	Volumetric & IBR	●	●	■■□□	No	St.	56Mb of compressed texture per tree.	Volumetric reconstruction from photographs.
VI-A.4 Volumetric point-based rep. [DVS03]. p. 15	Volumetric & Point	●	●	□■■■	No	Dyn., Sh.	Point sampling.	Linear point structure to efficiently use graphics hardware.
VI-B Hierarchical Tetrahedra [MFI97]. p. 15	Polygon	◐	◐	■■■■	No	Dyn.	Tetrahedra hierarchy: for 78.2 million polygons around 11000 tetrahedra	Adaptive multiresolution space partitioning.
VI-C.1 Leaves Simplification [RCRB03], [ZB03]. p. 16	Polygon	●	◐	■■■□	No	Dyn.	A sequence of leaves collapse: number of leaves are doubled.	Adaptation of classical mesh simplification scheme to foliage.
VI-C.2 Visibility based partitioning [GBP04]. p. 16	Point	●	●	□■■□	No	St.	Point sampling in a grid.	Optimized for moving camera.

TABLE II: GLOBAL AND MULTISCALE REPRESENTATIONS. From left to right, columns indicates: the rendering primitive type of the method; rendering performance (○ several tens of minutes, ◐ few minutes and ● interactive time); realism (○ many artefacts, ◐ few artefact for some point of views and ● realistic); distance validity with four squares representing respectively close (organs scale), medium (tree scale), far (forest scale: a tree on a dozen of pixels), very far (landscape: a tree on 2 or 3 pixels) views; the kind of animation supported; the kind of lighting supported (Dynamic/Static lighting, Shadowing and Self Shadowing) and the type and amount of pre-computed data.

[BS91] J. Bloomenthal and K. Shoemake. Convolution surfaces. In *ACM Transactions on Graphics (SIGGRAPH 1991 Conference Proceedings)*, volume 25(4), Las Vegas, July 1991. 5

[BW90] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. In *Computer Graphics (Proceedings of Symposium on Interactive 3D Graphics)*, volume 24(2), pages 109–116, 1990. 5, 17

[Can98] M.-P. Cani. Layered models with implicit surfaces. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, Jun 1998. Invited paper, published under the name Marie-Paule Cani-Gascuel. 5

[CCDH05] C. Colditz, L. Coconu, O. Deussen, and H. Hege. Real-time rendering of complex photorealistic landscapes using hybrid level-of-detail approaches. In *Real-time visualization and participation, 6th International Conference for Information Technologies in Landscape Architecture*, 2005. 11

[CCMS97] A. Ciamplaini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–24, 1997. 16

[COCSD02] D. Cohen-Or, Y. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transaction on Visualization and Computer Graphics*, 2002. 2

[CW93] Shenchang E. C. and Lance W. View interpolation for image synthesis. *Computer Graphics*, 27(Annual Conference Series):279–288, 1993. 12

[DCSD02] O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis. Interactive visualization of complex plant ecosystems. In *Proceedings of the IEEE Visualization Conference*. IEEE, October

2002. 12, 13, 18

[DL05] O. Deussen and B. Lintermann. *Digital Design of Nature. Computer Generated Plants and Organics*. Springer-Verlag, 2005. 2

[DN04] P. Decaudin and F. Neyret. Rendering forest scenes in real-time. In A. Keller H. W. Jensen, editor, *Eurographics Symposium on Rendering*, june 2004. 14, 15, 18

[dREF+88] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. In *ACM Transactions on Graphics (SIGGRAPH 1988 Conference Proceedings)*, pages 151–158, August 1988. 3, 17

[DVS03] C. Dachsbacher, C. Vogelgsang, and M. Stamminger. Sequential point trees. *ACM Trans. Graph.*, 22(3):657–662, 2003. 15, 18

[FCA97] E. Ferley, M.P. Cani, and D. Attali. Skeletal reconstruction of branching shapes. *Computer Graphics Forum*, 16(5):283–293, December 1997. Published under the name Marie-Paule Cani-Gascuel. 5

[FP96] P. Federl and P. Prusinkiewicz. A texture model for cracked surfaces, with an application to tree bark. In *Proceedings of Western Computer Graphics Symposium*, pages 23–29, 1996. 6

[FP04] P. Federl and P. Prusinkiewicz. Finite element model of fracture formation on growing surfaces. In M. Bubak, G. van Albada, P. Sloot, and J. Dongarra, editors, *Proceedings of Computational Science. ICCS 2004 (Krakow, Poland, June 6-9, 2004) Part II, Lecture Notes in Computer Science 3037*, pages 65–72, Berlin, 2004. Springer. 7, 17

[FPF04] Cohen F., Decaudin P., and Neyret F. Gpu-based lighting and shadowing of complex natural scenes. In *Siggraph'04 Poster.*, 2004. www-imagis.imag.fr/Publications/2004/CDN04/. 14

[Gar84] G. Y. Gardner. Simulation of natural scenes using textured quadric surfaces. In *ACM Transactions on Graphics (SIGGRAPH 1984 Conference Proceedings)*, pages 11–20, July 1984. 9, 18

[GBP04] G. Guennebaud, L. Barthe, and M. Paulin. Deferred Splatting. In *Computer Graphics Forum*. European Association for Computer Graphics and Blackwell Publishing, septembre 2004. (EG2004 Proceedings). 16, 18

[GCF01] T. Di Giacomo, S. Capo, and F. Faure. An interactive forest. In Marie-Paule Cani, Nadia Magnenat-Thalmann, and Daniel Thalmann, editors, *Eurographics Workshop on Computer Animation and Simulation*, pages 65–74. Springer, sept. 2001. Manchester. 11, 18

[GMW04a] C. Galbraith, P. MacMurchy, and B. Wyvill. Blobtree trees. In *Proceedings of Computer Graphics International 2004*, 2004. 4, 5, 17

[GMW04b] C. Galbraith, L. Mündermann, and B. Wyvill. Implicit visualization and inverse modeling of growing trees. In *Computer Graphics Forum (Proceedings of Eurographics'2004)*, 2004. 5

[God00] C. Godin. Representing and encoding plant architecture: a review. *Annals of Forest Science*, 57(05-juin):413–438, 2000. 2

[GP03] G. Guennebaud and M. Paulin. Efficient screen space approach for Hardware Accelerated Surfel Rendering. In *Vision, Modeling and Visualization, Munich*, pages 1–10. IEEE Signal Processing Society, 19-21 novembre 2003. 16

[GPR+03] S. Guerraz, F. Perbet, D. Raulo, F. Faure, and M.-P. Cani. A procedural approach to animate interactive natural sceneries. In *CASA03*, 2003. 11, 18

[Har97] J. C. Hart. Implicit representation of rough surfaces. *Computer Graphics Forum*, 16(2):91–99, 1997. 6

[HB96] J. C. Hart and B. Baker. Implicit modeling of tree surfaces. In *Implicit Surfaces'96*, pages 143–152, 1996. 4, 5, 6, 17

[HG97] P. S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Carnegie Mellon University, 1997. 10

[HKK98] K. Hirota, H. Kato, and T. Kanedo. A physically-based simulation model of growing tree barks. *ISPJ Journal*, 39(11), 1998. in Japanese. 6

[Hop98] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings IEEE Visualization'98*, pages 35–42, 1998. 10

[HPW92] M.-S. Hammel, P. Prusinkiewicz, and B. Wyvill. Modelling compound leaves using implicit contours. In Tosiyasu L. Kunii, editor, *Visual computing: integrating computer graphics with computer vision (Proceedings of Computer Graphics International'92*, pages 119–212, Tokyo, Japan, June 1992. 8, 17

[HTK98] K. Hirota, Y. Tanoue, and T. Kanedo. Generation of cracks patterns with a physical model. *The Visual Computer*, 1998. 6

[IDV02] Inc. Interactive Data Visualization. Speedtree product homepage. web page, 2002. http://www.idvinc.com/speedtree/. 11, 18

[Jak00] A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J.C. Torres, editors, *Proceedings of Eurographics 2000 (Short Presentations)*, August 2000. 9, 18

[JFP01] C.-L. Jin, X. ans Tai, J. Feng, and Q. Peng. Convolution surfaces for line skeletons with polynomial weight distributions. *Journal of Graphics Tools*, 6(3):1728, 2001. 5

[Kaw82] K. Kawaguchi. A morphological study of the form of nature. In *ACM Transactions on Graphics (SIGGRAPH 1982 Conference Proceedings)*, pages 223–232, July 1982. 3, 4

[KK89] J.T. Kajiya and T.L. Kay. Rendering fur with three dimensional textures. *ACM Transactions on Graphics (SIGGRAPH 1989 Conference Proceedings)*, 23(3):271–280, July 1989. 14

[LCV03] J. Lluch, E. Camahort, and R. Vivó. Procedural multiresolution for plant and tree rendering. In *Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*, pages 31–38. ACM Press, 2003. 11, 18

[LD99] B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999. 7

[LH96] M. Levoy and P. Hanrahan. Light field rendering. In *ACM Transactions on Graphics (SIGGRAPH 1996 Conference Proceedings)*, pages 31–42, August 1996. 12

[LN02] S. Lefebvre and F. Neyret. Synthesizing bark. In *13th Eurographics Workshop on Rendering*, 2002. 6, 7, 17

[LPFH01] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *ACM 2001 Symposium on Interactive 3D Graphics*, USA, Mar 2001. 14

[LRC+02] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of detail for 3D graphics*. Morgan Kaufmann, Amsterdam, 2002. 1

[LT97] K. Low and T.-S. Tan. Model simplification using vertex-clustering. In *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1997. 16

[LW85] M. Levoy and T. Whitted. The use of points as a display primitive. Technical report, Computer Science Department, University of North Carolina at Chapel Hill, 1985. 2

[Mar03] K. Maritaud. *Rendu réaliste darbres vus de près en images de synthèse*. PhD thesis, Université de Limoges, December 2003. 4, 6, 7

[Max90] N. Max. Cone–sphere. *ACM Transactions on Graphics (SIGGRAPH 1990 Conference Proceedings)*, 24(2), 1990. 4, 17

[Max96] N. Max. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *Eurographics Rendering Workshop 1996*, pages 165–174. Eurographics, June 1996. ISBN 3-211-82883-4. 12, 18

[MDK99] N. Max, O. Deussen, and B. Keating. Hierarchical image-based rendering using texture mapping hardware. In *Eurographics Workshop on Rendering '99*, 1999. 11, 12, 18

[MFI97] D. Marshall, D. Fussell, and A. T. Campbell III. Multiresolution rendering of complex botanical scenes. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 97–104. Canadian Human-Computer Communications Society, 1997. 15, 16, 18

[MMP03] L. Mündermann, P. MacMurchy, and P. Pivovarov, J. and-Prusinkiewicz. Modeling lobed leaves. In *Proceedings of Computer Graphics International*, pages 60–65, 2003. 8, 17

[MN98] A. Meyer and F. Neyret. Interactive volumetric textures. In *Eurographics Workshop on Rendering 1998*, pages 157–168, June 1998. 14

[MN00] A. Meyer and F. Neyret. Multiscale shaders for the efficient realistic rendering of pine-trees. In *Graphics Interface*, May 2000. 13, 18

[MNP01] A. Meyer, F. Neyret, and P. Poulin. Interactive rendering of trees with shading and shadows. In *Eurographics Workshop on Rendering*, Jul 2001. 10, 11, 12, 18

[MO95] N. Max and K. Ohsaki. Rendering trees from precomputed Z-buffer views. In *Eurographics Workshop on Rendering 1995*, June 1995. 12, 18

[MTF03] S. Mantler, R. F. Tobler, and A. L. Fuhrmann. The state of the art in realtime rendering of vegetation. Technical report, VRVis, http://www.vrvis.at/TR/2003/TR_VRVis_2003_027_Abstract.html, 2003. 1

[MZG01] J. van Baar M. Zwicker, H. Pfister and M. Gross. Surface splatting. In *ACM Transactions on Graphics (SIGGRAPH 2001 Conference Proceedings)*, August 2001. 2, 12

[NC99] F. Neyret and M.P. Cani. Pattern-based texturing revisited. In *ACM Transactions on Graphics (SIGGRAPH 1999 Conference Proceedings)*, pages 235–242, August 1999. 14

[Ney95] F. Neyret. A general and multiscale method for volumetric textures. In *Graphics Interface'95 Proceedings*, pages 83–91, May 1995. 14

[Ney96] F. Neyret. Synthesizing verdant landscapes using volumetric textures. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 215–224, New York City, NY, June 1996. Eurographics, Springer Wein. ISBN 3-211-82883-4. 14

[Ney98] F. Neyret. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55–70, 1998. 14, 15, 18

[PC01] F. Perbet and M.-P. Cani. Animating prairies in real-time. In *ACM Symposium on Interactive 3D Graphics*, USA, Mar 2001. 11

[PCD+97] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. Viewbased rendering: Visualizing real objects from scanned range and color data. In *Eurographics Rendering Workshop'97*, 1997. 12

[PH97] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Transactions on Graphics (SIGGRAPH 1997 Conference Proceedings)*, pages 217–223, Los angeles, 1997. ACM. 16

[PHHM97] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech, editors. *Visual models of plant development*, volume 3 of *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997. 2

[PL90] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer Verlag, New York, 1990. 3, 6, 7, 17

[PMKL01] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *ACM Transactions on Graphics (SIGGRAPH 2001 Conference Proceedings)*, Computer Graphics, pages 36–47, Los Angeles, California, 2001. ACM. 7, 8, 17

[PZvBG00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *ACM Transactions on Graphics (SIGGRAPH 2000 Conference Proceedings)*, Computer Graphics Proceedings, pages 335–342, July 2000. 2, 12

[QNTN03] X. Qin, E. Nakamae, K. Tadamura, and Y. Nagai. Fast photo-realistic rendering of trees in daylight. *Computer Graphics Forum (Proceedings of Eurographics'03)*, 22(3):243, 2003. 9, 10, 18

[RB85] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *ACM Transactions on Graphics (SIGGRAPH 1985 Conference Proceedings)*, pages 313–322, July 1985. 12, 13, 18

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In B. Falcidieno and T.L. Kunii, editors, *Geometric Modeling in Computer Graphics*, page 455465, Genova, Italy, 1993. Springer Verlag. 16

[RCB+02] I. Remolar, M. Chover, O. Belmonte, J. Ribelles, and Rebollo C. Geometric simplification of foliage. In *Eurographics2002 Short Presentation Proceedings*, pages 397–404, Saarbrcken (Germany), 2002. 16

[RCRB03] I. Remolar, M. Chover, J. Ribelles, and O. Belmonte. View-dependent multiresolution model for foliage. In *In Journal of WSCG (WSCG2003 Proceedings)*, February 2003. 16, 18

[RFL+05] A. Runions, M. Fuhrer, B. Lane, P. Federl, A. Rolland-Lagan, and P. Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings)*, 24(3):702–711, 2005. 6, 8

[RL00] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH 2000, Computer Graphics Proceedings*, pages 343–352, 2000. 2, 12

[RMD04] A. Reche, I. Martin, and G. Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings)*, 23(3), July 2004. 14, 15, 18

[RSC87] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *ACM Transactions on Graphics (SIGGRAPH 1987 Conference Proceedings)*, pages 283–291, July 1987. 13

[SBLD03] F. Suykens, K. Vom Berge, A. Lagae, and P. Dutré. Interactive rendering with bidirectional texture functions. In *Computer Graphics Forum*, volume 22, 2003. 12

[Sch97] W.J. Schroeder. A topology modifying progressive decimation algorithm. In *IEEE Visualization 97 Conference Proceedings*, volume 545, pages 205–212, 1997. 16

[SD01] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering 01)*. Eurographics, 2001. 15

[SGHS98] J. Shade, S.J. Gortler, L. He, and R. Szeliski. Layered depth images. In *ACM Transactions on Graphics (SIGGRAPH 1998 Conference Proceedings)*, pages 231–242, July 1998. 12

[Sil95] F. X. Sillion. Hierarchical solution techniques for realistic rendering. In *State of the Art Report - Graphicon'95 Confrence*, St Petersburg, Russia, 1995. 13, 14

[SPS03] C. Smith, P. Prusinkiewicz, and F. Samavati. Relational specification of subdivision algorithms. In *Applications of Graph Transformations with Industrial Relevance (AGTIVE 2003): Lecture Notes in Computer Science 3062*, pages 313–327, 2003. 6

[SSBR03] C. Soler, F. X. Sillion, F. Blaise, and P. de Reffye. An efficient instantiation algorithm for simulating radiant energy transfer in plant models. *ACM Transactions On Graphics*, 22(2), April 2003. 14, 18

[TMW02] R.F. Tobler, S. Maierhofer, and A. Wilkie. Mesh-based parametrized l-systems and generalized subdivision for generating complex geometry. *International Journal of Shape Modeling*, 8(2):173–191, Dec 2002. 6, 17

[vO96] C. W. A. M. van Overveld. Stiky splines: Definition and manipulation of spline structures with maintaned topological relations. *ACM Transactions on Graphics*, 15(1):72–98, 1996. 8

[WP95] J. Weber and J. Penn. Creation and rendering of realistic trees. In *ACM Transactions on Graphics (SIGGRAPH 1995 Conference Proceedings)*, pages 119–128, August 1995. 12, 13, 18

[WWD+05] L. Wang, W. Wang, J. Dorsey, X. Yang, B. Guo, and H.-Y. Shum. Real-time rendering of plant leaves. *ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings)*, 24(3), 2005. 8, 17

[WWL+03] X. Wang, L. Wang, L. Liu, S. Hu, and B. Guo. Interactive modeling of tree bark. In *Pacific Graphics Proceedings*, 2003. 6

[WWT+03] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. In *ACM Transactions on Graphics (SIGGRAPH 2003 Conference Proceedings)*, 2003. 6, 7, 17

[ZB03] X.P. Zhang and F. Blaise. Progressive polygon foliage simplification. In B. Hu and M. Jaeger, editors, *Plant Growth Modeling and Applications (Proceedings of PMA03)*, Beijing, China, October 2003. Springer. 16, 18

[ZC04] C. Zhang and T. Chen. A survey on image-based rendering - representation, sampling and compression. In *EURASIP Signal Processing: Image Communication*, 2004. 2