

Margarita N. Favorskaya
Lakhmi C. Jain

Handbook on Advances in Remote Sensing and Geographic Information Systems

Paradigms and Applications in Forest
Landscape Modeling

Intelligent Systems Reference Library

Volume 122

Series editors

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

Lakhmi C. Jain, University of Canberra, Canberra, Australia;
Bournemouth University, UK;
KES International, UK
e-mail: jainlc2002@yahoo.co.uk; Lakhmi.Jain@canberra.edu.au
URL: <http://www.kesinternational.org/organisation.php>

About this Series

The aim of this series is to publish a Reference Library, including novel advances and developments in all aspects of Intelligent Systems in an easily accessible and well structured form. The series includes reference works, handbooks, compendia, textbooks, well-structured monographs, dictionaries, and encyclopedias. It contains well integrated knowledge and current information in the field of Intelligent Systems. The series covers the theory, applications, and design methods of Intelligent Systems. Virtually all disciplines such as engineering, computer science, avionics, business, e-commerce, environment, healthcare, physics and life science are included.

More information about this series at <http://www.springer.com/series/8578>

Margarita N. Favorskaya
Lakhmi C. Jain

Handbook on Advances in Remote Sensing and Geographic Information Systems

Paradigms and Applications in Forest
Landscape Modeling



Springer

Margarita N. Favorskaya
Institute of Informatics and
Telecommunications
Siberian State Aerospace University
Krasnoyarsk
Russia

Lakhmi C. Jain
Faculty of Education, Science, Technology
and Mathematics
University of Canberra
Canberra
Australia

and

Faculty of Science and Technology, Data
Science Institute
Bournemouth University
Poole
UK

and

KES International
Leeds, West Yorkshire
UK

ISSN 1868-4394

ISSN 1868-4408 (electronic)

Intelligent Systems Reference Library

ISBN 978-3-319-52306-4

ISBN 978-3-319-52308-8 (eBook)

DOI 10.1007/978-3-319-52308-8

Library of Congress Control Number: 2016963337

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature

The registered company is Springer International Publishing AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The remote sensing (RS) and geographic information system (GIS), 3D forest landscape modeling using airborne and space laser and radar scanning, digital photography, and global positioning satellite systems provide novel opportunities for remote sensing monitoring and inventory of forest resources. High efficiency of laser and radar scanning in combination with centimeter spatial resolution of digital aerial photography and high accuracy for coordinate definition of trees and tree stands' morphostructural parameters by satellite geopositioning systems allow to develop effective algorithms for research of forest resources structure and dynamics, guaranteeing a real-time automatic extraction of forest inventory parameters.

Computer modeling provides the building of 3D landscape scenes based on the data of laser and radar scanning and airborne images. The innovative methods of terrain and vegetation modeling are presented. Automatic fusion of data of different types is a non-solved problem requiring a development of future efficient methods. However, static modeling scenes will not be realistic if some natural effects are not imposed. The book is divided into four parts as follows:

- Part I. Airborne LiDAR and Optical Measurements of Forest
- Part II. Individual Tree Modelling
- Part III. Landscape Scene Modelling
- Part IV. Forest Eco-system Modelling

We are grateful to the researchers for inventing the tools and paradigms which laid the foundation for research and development reported in this book.

Our thanks are due to Springer-Verlag for the opportunity to publish our book.

This book is directed to the students, researchers, practitioners, and professors interested in remote sensing and geographic information systems and applications.

Krasnoyarsk, Russia
Canberra, Australia

Margarita N. Favorskaya
Lakhmi C. Jain

Contents

1	Innovations in Remote Sensing of Forests	1
1.1	Introduction	1
1.2	Chapters Including in the Book	3
1.3	Conclusions	12
	References	12

Part I Airborne LiDAR and Optical Measurements of Forest

2	Overview of LiDAR Technologies and Equipment for Land Cover Scanning	19
2.1	Introduction	19
2.2	Development of LiDAR Technology	21
2.3	Overview of Airborne Laser Scanning	24
2.3.1	Physical Principles of Airborne Laser Scanning	24
2.3.2	Errors of LiDAR Systems	28
2.3.3	Conventional LiDAR Calibration Methods	30
2.3.4	Alternative LiDAR Calibration Methods	32
2.3.5	Equipment for Airborne Laser Scanning	33
2.4	Overview of UAV Laser Scanning	42
2.4.1	Particular Qualities of UAV Laser Scanning	42
2.4.2	Equipment for UAV Laser Scanning	43
2.5	Overview of Terrestrial Laser Scanning	47
2.5.1	Particular Qualities of Terrestrial Laser Scanning	47
2.5.2	Equipment for Terrestrial Laser Scanning	48
2.5.3	Particular Qualities of Mobile Laser Scanning	55
2.5.4	Equipment for Mobile Laser Scanning	55
2.6	Comparison of Remote Sensing Techniques for Forest Inventory	61
2.7	Conclusions	62
	References	63

3 Software Tools for Terrain and Forest Modelling	69
3.1 Introduction	69
3.2 Survey of Software Tools for Terrain Modelling	70
3.2.1 Software Tool “Terrasolid”	70
3.2.2 Cloud Platform “ArcGIS”	76
3.2.3 Software Tool “Bionatics”	80
3.2.4 Software Tool “Terragen”	83
3.2.5 Software Tool “3DNature”	84
3.2.6 Software Tool “GRASS”	85
3.2.7 Software Tool “Object Raku Technology”	89
3.3 Survey of Software Tools for Vegetation Modelling	90
3.3.1 Software Tool “Xfog”	90
3.3.2 Software Tool “SpeedTree”	92
3.3.3 Software Tool “Onyx Computing”	94
3.3.4 Software Tool “Marlin Studios”	99
3.3.5 Software Tool “E-on Software”	99
3.3.6 Additional Software Tools	101
3.4 Conclusions	108
References	108
4 Data Fusion for Evaluation of Woodland Parameters	111
4.1 Introduction	111
4.2 Related Work	113
4.3 Generalized Flowchart for Data Fusion of Airborne Laser Scanning, Imaging Spectroscopy, and Imaging Photography	120
4.4 Representation of Airborne LiDAR and Digital Photography Data	123
4.5 Method for Crown and Trunk Measurements	126
4.5.1 Shearlet Transform	128
4.5.2 Generic, Higher Order, and Probabilistic Active Contour Models	129
4.5.3 Crown Measurements	131
4.6 Experimental Results	132
4.7 Conclusions	133
References	133
Part II Individual Tree Modelling	
5 Tree Modelling in Virtual Reality Environment	141
5.1 Introduction	141
5.2 Related Work	142
5.3 Fundamentals of L-Systems	151
5.3.1 Overview of L-Systems	152
5.3.2 Overview of Parametric L-Systems	154

5.3.3	Skeleton Tree-like Structures Based on L-Systems	158
5.4	Procedural Modelling of Broad-Leaved Trees and Shrubs	160
5.4.1	Procedural Broad-Leaved Trees Modelling Based on L-Systems	162
5.4.2	Realistic Image-Based Procedural Modelling of Broad-Leaved Trees	164
5.5	Procedural Modelling of Coniferous Trees	167
5.6	Modelling Results	169
5.7	Conclusions	176
	References	176
6	Realistic Tree Modelling	181
6.1	Introduction	181
6.2	Related Work	182
6.3	Voxel Modelling of Vegetation	184
6.3.1	Voxelization	184
6.3.2	2D Analysis in Horizontal Cuts	185
6.3.3	3D Reconstruction	186
6.4	Improvement of Tree Models by Realistic Data	186
6.4.1	Space Colonization Algorithm	187
6.4.2	Graph-Based Modelling	191
6.4.3	Self-organizing Tree Modelling	192
6.4.4	Inverse Procedural Modelling	194
6.5	Experimental Results	197
6.6	Conclusions	200
	References	201

Part III Landscape Scene Modelling

7	Digital Modelling of Terrain Surface	205
7.1	Introduction	205
7.2	Related Work	207
7.3	Densification of LiDAR Point Cloud	210
7.3.1	Densification Approaches	210
7.3.2	Measurement Errors	211
7.4	Filtering of LiDAR Points	213
7.4.1	Surface-Based Filtering	213
7.4.2	Slope-Based Filtering	217
7.4.3	Morphology-Based Filtering	219
7.4.4	Segmentation-Based Filtering	221
7.4.5	Hybrid Methods	225
7.4.6	Comparison of Filtering Methods	227
7.5	Generation of Digital Terrain Model	229
7.5.1	Spatial Relationship Methods	229

7.5.2	Statistical Methods	234
7.5.3	Curve-Fitting Methods	239
7.5.4	Surfaces' Interpolation in ArcGIS Spatial Analyst	240
7.5.5	Accuracy Estimators	241
7.6	Experimental Results	243
7.7	Conclusions	246
	References	246
8	Texturing of Landscape Scenes	251
8.1	Introduction	251
8.2	Related Work	253
8.2.1	Visualization Techniques for Digital Earth Surface Model	253
8.2.2	Visualization Techniques for Tree and Plant Models	256
8.3	Fundamentals of Texture Mapping	257
8.4	Multi-resolution Texturing for Digital Earth Surface Model	261
8.4.1	Basic Techniques	262
8.4.2	Extended Techniques	263
8.5	Multi-resolution Texturing for Vegetation Models	266
8.6	Experimental Results	272
8.7	Conclusions	276
	References	277
9	Large Scene Rendering	281
9.1	Introduction	281
9.2	Related Work	282
9.2.1	Overview of Terrain Rendering Techniques	282
9.2.2	Overview of Object Rendering Techniques	284
9.2.3	Overview of Realistic Lighting	287
9.3	Large Landscape Scene Rendering	290
9.4	Large Terrain Rendering	291
9.5	Vegetation Rendering	299
9.6	Realistic Lighting	303
9.7	Shaders	306
9.8	Conclusions	317
	References	317
10	Scene Rendering Under Meteorological Impacts	321
10.1	Introduction	321
10.2	Related Work	322
10.2.1	Overview of Wind Rendering	322
10.2.2	Overview of Fog Rendering	324
10.2.3	Overview of Rain Rendering	325
10.2.4	Overview of Snow Rendering	326

10.3	Wind Rendering	328
10.4	Fog Simulation	334
10.5	Rain Simulation	340
10.6	Snow Covering Simulation	346
10.7	Natural Objects' Rendering	350
10.7.1	Rendering of the Water Surfaces	351
10.7.2	Rendering of Clouds	354
10.8	Experimental Results	356
10.9	Conclusions	359
	References	360

Part IV Forest Ecosystem Modelling

11	Lighting and Shadows Rendering in Natural Scenes	367
11.1	Introduction	367
11.2	Related Work	368
11.3	Background of Lighting	371
11.3.1	Lighting	371
11.3.2	Material Properties	381
11.3.3	Volume Rendering	384
11.4	Simulation of Lighting in Modelling Scene	385
11.4.1	Trees Lighting	386
11.4.2	Foliage Lighting	388
11.5	Simulation of Lighting Changes in Modelling Scene	390
11.6	Implementation	392
11.7	Conclusions	393
	References	393
12	Modelling of Forest Ecosystems	397
12.1	Introduction	397
12.2	Related Work	400
12.3	Forest Scene Modelling	402
12.4	Forest Ecosystems	404
12.5	Modelling of Living Conditions	405
12.5.1	Soil's Moisture and Minerals	406
12.5.2	Wind	407
12.5.3	Temperature	408
12.5.4	Solar Radiation	408
12.5.5	Forest Fire Risk Evaluation	410
12.6	Conclusions	412
	References	412

About the Authors



Dr. Margarita N. Favorskaya is a professor and head of Department of Informatics and Computer Techniques at Siberian State Aerospace University, Russian Federation.

Professor Favorskaya is a member of KES organization since 2010, the IPC member, and the chair of invited sessions of international conferences. She serves as a reviewer in international journals (*Neurocomputing*, *Knowledge Engineering and Soft Data Paradigms*, *Pattern Recognition Letters*, *Engineering Applications of Artificial Intelligence*) and an associate editor of *Intelligent Decision Technologies Journal* and *Computer and Information Science Journal*. She is the author or the co-author of 160 publications and 20 educational manuals in computer science. She co-edited 3 books for Springer recently. She supervised 8 Ph.D. candidates and presently supervising 5 Ph.D. students.

Her main research interests are digital image and videos processing, remote sensing, pattern recognition, fractal image processing, artificial intelligence, and information technologies.



Dr. Lakhmi C. Jain is with the Faculty of Education, Science, Technology, and Mathematics at the University of Canberra, Australia, and Bournemouth University, UK. He is a fellow of the Institution of Engineers Australia.

Professor Jain founded the KES International for providing a professional community the opportunities for publications, knowledge exchange, cooperation, and teaming. Involving around 10,000 researchers drawn from universities and companies worldwide, KES facilitates international cooperation and generates synergy in teaching and research. KES regularly provides networking opportunities for professional community through one of the largest conferences of its kind in the area of KES. www.kesinternational.org.

His interests focus on the artificial intelligence paradigms and their applications in complex systems, security, e-education, e-healthcare, unmanned air vehicles, and intelligent agents.

Acronyms

AABB	Axis-Aligned Bounding Boxes
ABA	Area-Based Approach
ABRDF	Apparent BRDF
AGB	Above Ground Biomass
AGL	Above Ground Level
ALS	Airborne Laser Scanning
AMDIL	AMD Intermediate Language
API	Application Programming Interface
APSRS	American Society of Photogrammetry and Remote Sensing
ASF	Adaptive Surface Filter
B	Blue
BC	Branch Chain
BLUP	Best Linear Unbiased Prediction
BMP	BitMaP
BRDFs	Bidirectional Reflectance Distribution Functions
B-Rep	Boundary representation
BSG	Branch-Structure Graph
BTDFs	Bidirectional Transmittance Distribution Functions
BTF	Bidirectional Texture Function
CAD	Computer-Aided Drawing
CAGR	Compound Annual Growth Rate
CCA	Canonical Component Analysis
CCD	Charge-Coupled Device
CERL	Construction Engineering Research Laboratory
CG	High-level shading language for programming vertex and pixel shaders
CHD	Canopy Height Distribution
CHM	Canopy Height Model
CHP	Canopy Height Profile
CHQ	Canopy Height Quantile function

CIR	Consumer InfraRed
CLOD	Continuous LOD
CPU	Central Processing Unit
CSG	Constructive Solid Geometry
CSM	Cascaded Shadow Maps
CUDA	Compute Unified Device Architecture
CU-Structural Soil	CU-Structural Soil is a mixture of crushed gravel and soil with a small amount of hydrogel to prevent the soil and stone from separating during the mixing and installation process
DBH	Diameter-at-Breast Height
DEM	Digital Evaluation Model
DES	Digital Earth Surface
DHP	Density of High Points
DLOD	Discrete LOD
DMPs	Differential Morphological Profiles
DSM	Digital Surface Model
DTM	Digital Terrain Model
DVM	Dynamic Vegetation Model
EDSS	Ecosystem Decision Support System
EM	Expectation–Maximization
ETEW	Elevation Threshold with Expanding Window
FAR	False Acceptance Ratio
FBX	FilmBoX
FFT	Fast Fourier Transform
FOV	Field Of View
FRR	False Rejection Ratio
FSA	Foliage Simplification Algorithm
FSP	Functional–Structural Plant
FTI	Feature Type Interpreter
FVVR	Free-Viewpoint Video Renderers
G	Green
GeoTIFF	Geo-referenced TIFF
GIF	Graphics Interchange Format
GLCM	Grey Level Co-occurrence Matrix
GLSL or GLslang	GL Shading Language
GMT	Greenwich Mean Time
GNSS	Global Navigation Satellite System
GPP/NPP	Gross/Net Primary Productivity
GPR	Ground-Penetrating Radar
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRASS	Geographic Resources Analysis Support System
GSD	Ground Sample Distance
GSM	Global System Mobile
HBT	Hierarchical Bidirectional Texture

HIS	Hue Saturation Intensity
HLOD	Hierarchical Levels of Detail
HLSL	High-Level Shading Language
HMLS	Handheld Mobile Laser Scanner
HSR	Hidden Surface Removal
HUO	Hierarchical Union of Organs
IBR	Image-Based Rendering
IDT	Individual Tree Detection
IDW	Inverse Distance Weighed
IFF	Interchange File Format
IMU	Inertial Measurement Unit
INS	Integrated Navigation System
IP	Imaging digital Photography
IS	Imaging Spectroscopy
ISPRS	International Society for Photogrammetry and Remote Sensing
ISR	Intelligence, Surveillance, and Reconnaissance
JPEG	Joint Photographic Experts Group
KBDI	Keetche–Byram Drought Index
kd-tree	k dimensional tree
kriging	Geostatistical interpolation techniques
Lab	Liteness and a and b for the color-opponent dimensions, based on nonlinearly compressed
LAI	Leaf Area Index
LAN	Local Area Network
LAS	Log ASCII Standard
LDA	Linear Discriminant Analysis
LDAP	Lightweight Directory Access Protocol
LDI	Layered Depth Image
LES	Large Eddy Simulation
LiDAR	Light Detection and Ranging
LiPSM	Light-space PSMs
LM	Local Maxima
LMS	LiDAR Mapping Suite
LOD	Level Of Detail
LoFS	Local Fitting Surfaces
LogPSMs	Logarithmic Perspective Shadow Maps
LPs	Lowest Points
LPC	LiDAR Point Cloud
LRU	Least Recently Used
LSA	Least-Squares Adjustment
L-systems	Lindenmayer systems
LWO	LightWave 3D Objects
MAE	Mean Absolute Error
MAV	micro-UAV

MCC	Multi-scale Curvature Classification
MCMC	Monte Carlo Markov Chain
MCWS	Marker-Controlled Watershed Segmentation
MHC	Multi-resolution Hierarchical Classification
MLFMM	Multi-Level Fast Multi-pole Method
MLS	Maximum Local Slope
MLS	Mobile Laser Scanning
MPM	Material Point Method
MRF	Markov Random Field
MS	Mean Shift
MTOM	Maximum Takeoff Mass
MUAV	mini-UAV
NBR	Nuclear, Biological, and Radiological
NDVI	Normalized Difference Vegetation Indices
NED	National Elevation Dataset
NIR	Near-InfraRed
NUAV	nano-UAV
OC	Occlusion Culling
PCA	Principal Component Analysis
PDA	Personal Digital Assistant
PDF	Probability Density Function
PGM	Portable Gray Map
PICT	Macintosh PICTure
PKI	Public Key Infrastructure
PLU	Progressive Leaves Union
PM	Progressive Morphological
PMs	Progressive Meshes
PNG	Portable Network Graphics
PRF	Pulse Repetition Frequency
PRR	Pulse Repetition Rate
PSD	The PhotoShop Data
PSM	Perspective Shadow Map
PSSM	Parallel-Split Shadow Mapping
PTD	Progressive TIN Densification
PVS	Potentially Visible Sets
QA	Quality Assurance
R	Red
RADAR	RAdio Detecting And Ranging
RAM	Random Access Memory
RANS	Reynolds-Averaged Navier-Stokes
RANSAC	RANDom SAMple Consensus
RF	Random Forest
RGBA	Red, Green, Blue, Alpha
RMSE	Root-Mean-Square Error
RMSPE	Root-Mean-Square Prediction Error

ROAM	Real-time Optimally Adapting Meshes
ROI	Regions Of Interest
RPC	Rational Polynomial Coefficient
RST	Regularized Splines with Tension
RT	Radiative Transfer
RTM	Radiative Transfer Models
SBRDF	Spatial Bidirectional Reflectance Distribution Function
SDK	Software Development Kit
SLAM	Simultaneous Localization And Mapping
SPH	Smoothed Particle Hydrodynamics
SRTM	Shuttle Radar Topography Mission
SSD	Solid-State Drive
SUSC	Segmentation Using Smoothness Constraint
SUV	Suburban Utility Vehicle
SV	Soft Voxel
SVM	Support Vector Machines
t-ADT	tree Abstract Data Type
TDM	Textured Depth Meshes
TEN	TEtrahedral Network
TGA	Truevision Targa Graphic
TIFF	Tagged Image File
TIN	Triangular Irregular Network
TLS	Terrestrial Laser Scanning
TPS	Thin Plate Spline
TRAM	Transputer Random Access Memory
TSI	Timber Species Identifier
UAS	Unmanned Aerial System
UAV	Unmanned Airborne Vehicle
VDTM	View-Dependent Texture Mapping
VHR	Very High Resolution
VI	Vegetation Indice
VLS	Vehicle-based Laser Scanning
VNS	Visual Nature Studio
Voxel	Volume element
WCS	World Construction Set

Chapter 1

Innovations in Remote Sensing of Forests

Abstract Nowadays, a remote sensing of forests is represented by three conventional types of shooting, such as the spaceborne, airborne, and terrestrial. Recently additional studies, such as the Unmanned Airborne Vehicles (UAVs) laser scanning including nano-UAV, micro-UAV, and mini-UAV (as a part of the airborne laser scanning) and the mobile laser scanning and hand-held mobile laser scanning (as a part of the terrestrial laser scanning), became available. Each type has own purposes and pros and cons. A fusion of such scaled data in a single software tool is the innovative task with a promising future. However, a process of fusion is not trivial because of different scale representation of data. Also, a wide coverage of areas leads to difficulties in the big data processing. The material of the book is mainly based on the airborne and terrestrial algorithms.

Keywords Remote sensing · Forest inventory · Airborne laser scanning · UAV laser scanning · Terrestrial laser scanning · Mobile laser scanning · Imaging spectroscopy

1.1 Introduction

Forests covering 31% of the total Earth's surface have a determining significance in the global carbon cycle and the terrestrial primary productivity [1]. The convection processes between atmosphere, vegetation and soil cannot be understood without persistent and durable observation and study of forest resources. Many difficult issues appear in this way. The current book is a contribution in remote sensing investigations including processing, modelling, and visualization based on the recent achievements in digital image processing.

The short historical review of airborne laser scanning as an excellent means of forest monitoring was represented by Hyypä et al. [2]. The stages of airborne laser scanning for forest inventory are represented in Table 1.1, while the achievements of algorithm developments are situated in Table 1.2.

Table 1.1 The stages of airborne laser scanning

Dates	Achievements in airborne laser scanning
1960–1970	Airborne laser ranging/profiling with experiments for bathymetric and hydrographic applications. The basics of biomass evaluation using laser measurements were founded at that time
1980–1990	Use of laser scanners (instead of laser profilers), the Global Positioning System (GPS) combined with the Integrated Navigation System (INS). The attempts of terrain modelling and individual tree elevation based on height and volume estimations had been started
2000–up to date	Use of various recording devices such as laser scanners, high resolution digital camera, multispectral camera, stereo camera, small-footprint scanners, terrestrial laser scanners, full-waveform digitizing LiDAR systems, and so forth. Measurements of forest growth, detection of aging, damaged, and harvested trees are based on data fusion techniques

Table 1.2 The stages of algorithms development

Dates	Achievements in algorithm development
1990–up to date	Development of the digital terrain modelling, progressive Triangular Irregular Network (TIN) densification methods, sparse TIN methods, a slope-based filtering techniques, methods of canopy height estimation, methods of individual tree estimation, geometrical models of stand heights
2000–up to date	Methods for definition of structural characteristics of forest stands (in relation to carbon content and biodiversity issues), histogram thresholding methods for separation of different tree storeys, classification/recognition of tree species, integration methods for simultaneous laser and aerial image acquisition, methods considering the impact of a whole spectrum of ecological factors including a human intervention, verification methods

The use of Light Detection And Ranging (LiDAR) in recent years contributed in study of the forest structural parameters, such as the height, shape, and foliage distribution of the canopy, and especially the individual tree [3]. Such Airborne Laser Scanning (ALS) data can be obtained from the local to regional scales. Data fusion of the Imaging Spectroscopy (IS) and the ALS facilitates a receiving of reliable information for evaluating the landcover maps, the Above-Ground Biomass (AGB), the bio-physical/chemical parameters, and the Gross/Net Primary Productivity (GPP/NPP) of forests. The excellent survey of the ALS methods and devices over 1990th–2000th is represented in study of Mallet and Bretar [3]. The IS provides spectral information about the canopy structure and the biochemical properties of canopy elements of an observed territory [4–6]. Physical nature of the IS and the ALS data are different. Thus, a fusion of these heterogeneous data is a complicated task that is solved by various approaches [7, 8].

The main goal of forest monitoring is to extract the forest variables. These methods have been divided into two categories: the area-based inventory methods (done at stand and plot level) and the tree-based inventory methods (done at individual tree level). Both categories utilize the statistical and image processing methods. However, the area-based inventory methods, forming the Digital Terrain

Model (DTM), the Digital Evaluation Model (DEM), and the Digital Surface Models (DSM), often use the regression or discriminant analysis. The image processing techniques help to extract the neighborhood information of point clouds and pixels of the DSMs more efficiently. The tree-based inventory methods imply to compute the individual physical features, such as the shape and volume of crowns, trunk diameters at the Diameter-at-Breast Height (DBH), tree heights, and so on. In this case, the image processing techniques prevail on the statistical methods. Objectively, the individual tree based inventories provide more accurate estimations. In general, the task of forest modelling includes the stages mentioned below:

- 3D LiDAR and 2D hyperspectral/multispectral/digital photography data processing in order to simulate canopy and individual tree models for evaluation of woodland parameters.
- Forest modelling on a point of view from the Earth's surface using 3D LiDAR and 2D hyperspectral/multispectral/digital photography data as a software tool of assistance for forest appraisers.
- Improvement of 3D visualization of individual trees, employing the achievements of digital image processing for forest inventory.
- Improvement of 3D visualization of landscape (forest part), using the environment approximate models that are time-based on the life-saving resources (water, light, minerals, and location balances, also weather and climate conditions) for forest monitoring.

The existing tree modelling methods can be classified into two major categories, such as the synthesis-based and the reconstruction-based. Synthesis-based methods are focused on synthesizing tree geometry from heuristics or knowledge, such as the botanic rules, shape grammar, physical simulations, or probabilistic optimization. Rule-based methods assume that the growth of trees follows specific patterns. Usually they use grammar of the L-systems or its multiple modifications to construct the trees species. In contrast, the reconstruction-based methods aim to recover 3D tree geometry that would be closed to ground-truth data received from laser scanner. As a result, very high-quality 3D models can be produced. Fusion of laser scanning data and modelling data is a cornerstone idea of this book.

1.2 Chapters Including in the Book

Chapter 2 is an introduction in airborne, UAV, and terrestrial laser scanning, including the basic physical principals, methods of registration, and equipment. According to report [9], the global airborne LiDAR market will demonstrate the Compound Annual Growth Rate (CAGR) of 12.4% from 2016 to 2022. It was found that the terrestrial lasers hold the largest share of the LiDAR market among all product types. Often the companies produce the additional equipment, providing the high-accuracy orthographic photography, thermal imaging, and multi-spectral

imaging. The laser technology was invented by Charles Townes and Arthur Schawlow in Bell Labs in 1958. Since that time, many investigations were done that was reflected in numerous publications [10, 11]. At last decades, various methods, such as LiDAR data filtering, the DTM/DEM/DSM generation, full-waveform data modelling, geometric and radiometric calibration, among others, and the special data formats of LiDAR data processing were developed. Also, three benchmark testing datasets were created for examining the algorithms and methods for urban object classification and 3D building reconstruction [12]. This chapter involves many URLs for more detailed study of numerous laser scanners and their applications. Each type of remote sensing differs of the scaled area and the achievable accuracy and precision values of measurements. The airborne and spaceborne sensors provide the spatially explicit data collected from large areas. The detection of individual trees from the ALS data is difficult and 3D segmentation techniques aim to improve the accuracy of the information derived from a tree-level analysis. Recently, the UAV laser scanning has been proposed as a tool for mapping and measuring tree metrics [13]. These systems provide a low-cost data collection with the high point densities, up to 1000 points per square meter. Objectively, the scanner and the sensors mounted on the on-board UAV platforms have higher errors and different sources of error. This means that the following investigations in the UAV technique are required. Miniaturization has led to smaller payloads of sensors, computers, communication devices, and power supplies that have allowed smaller UAVs to perform the same functions as larger UAVs. Nowadays, nano-UAV, micro-UAV, and mini-UAV are designed for military and civil purposes. The terrestrial remote sensing techniques have been deployed actively in forest inventory providing better inventory results of individual trees. However, the terrestrial techniques can be used to measure small forest areas only and also the data collected by the TLS instruments are highly affected by occlusions. Also, the Mobile Laser Scanning (MLS) and the Hand-Held Mobile Laser Scanner (HMLS) can be mentioned.

Chapter 3 contains the overview of software tools for terrain and forest modelling. Usually the extraction of the DTM is the initial step for the DEM and the DSM building in order to represent the vegetation levels. In spite of all software tools offer the similar functions, the modelling results are different due to various modelling algorithms and their program realization. Several high-top packages for terrain modelling, including the Terrasolid, the ArcGIS, the Bionatics, the Terragen, the 3DNature, the GRASS, and the Object Raku Technology, are discussed. The Terrasolid [14] develops the software products that are used world-wide for processing of the LiDAR and image data obtained from airborne and terrestrial laser scanning systems. Many components are included in the Terrasolid. Thus, the TerraScan is the main application in the Terrasolid software family for managing and processing of the LiDAR point clouds. The TerraScan Lite is a lighter version of the TerraScan for viewing laser points that are already automatically classified and for performing the processing steps that require the manual efforts. In the same way, full and lighter versions of other applications were developed, e.g., the TerraModeler and the TerraModeler Lite, the TerraPhoto and the TerraPhoto Lite,

the TerraSurvey and the TerraSurvey Lite, and so on. The ArcGIS platform provides the high quality solutions in many application, such as spatial analysis, mapping and visualization, imagery and remote sensing, real-time GIS, data management, among others [15]. Near 90 ArcGIS products are available in the ArcGIS Online, the ArcGIS for Desktop, and the ArcGIS for Server. The Bionatics [16] is a software tool to model, visualize, and manage the urban and rural territories in 3D space. The Terragen [17] is a powerful solution for rendering and animating realistic natural environments that is free for a non-commercial use. The open source/free software Geographic Resources Analysis Support System (GRASS) GIS is a public domain system with a full range of GIS capabilities [18]. The development of the GRASS is now in the hands of people, who have collaborated on this Open Source project. First software tools for vegetation modelling appeared since 1970s, when a procedural modelling as a popular theory for plants and trees' simulation developed by famous scientists, such as Lindenmayer, Oppenheimer, Weber and Penn, Prusinkiewicz, Honda, among others. The software tools, such as the Xfrog, the SpeedTree, the Onyx Computing, the Marlin Studios, the E-on Software, and additional ones, are discussed in this chapter. The Xfrog [19] is a procedural organic 3D modeller that allows to create and animate 3D trees, flowers, nature based special effects, or architectural forms. The SpeedTree 7 [20] has been released in three different versions: the SpeedTree Cinema, the SpeedTree Studio, and the SpeedTree Architect. Each version comes with the SpeedTree Modeler, the cornerstone of the SpeedTree 3D animation software experience. The Onyx Computing [21] is a software development studio specializing in the design and development of knowledge based systems for procedural modelling of vegetation.

Chapter 4 provides a technique for the airborne LiDAR, the imaging spectroscopy and/or the imaging digital photography data fusion. Data fusion promotes the direct and indirect measurements for the accurate biomass estimations. Two main approaches of forest inventory based on the ALS data prevail. In the area-based approach, the regression analysis, the nearest neighbor classification, and other classifying techniques are applied [22]. An approach of the individual tree detection meets difficulties with the accuracy segmentation of a tree and the expansive detailed modeling [23]. However, the main advantage of the individual tree detection is an obtaining of the trunk distribution series close to ground-truth. The individual tree segmentation from 3D point cloud is very difficult task due to the restricted sources of shooting, mixed tree species, and overlapping in dense stands. A literature review indicated the following steps of the generalized algorithm:

- Classification the raw LiDAR point data into the ground and above-ground points.
- Generation of the DTM.
- Normalization of the vegetation point cloud values by a subtracting the ground points from the LiDAR point cloud.

- Computation of tree heights through the elevation values of points, indicating the heights from the ground to the points.
- Individual trees segmentation, using density of 3D point cloud.
- Improvement of the tree segmentation results, using previously known information about the tree species and additional data from the hyperspectral/multispectral/digital photography shooting.

According to this generalized direction, the three-level flowchart was developed [24]. Level 1 is a level of primary data processing with additional data cleaning, interpolation, and artifacts removal. The goal of Level 2 is to build 3D canopy model and estimate canopy vegetation indexes in general. A parameterization of the individual tree with evaluation of local parameters is implemented at Level 3. The Level 3 was enforced by two innovations. A procedure for accurate matching of the LiDAR and image coordinates in the local region of interest based on a moving cluster was developed. The boundaries and textures of individual tree crowns were improved in the optical images by application of shearlets and higher-order active contour model. The obtained area measurements of the tree crowns coincide with the expert estimations, providing the accuracy 92–96%.

Chapter 5 investigates various approaches for individual tree modelling based on the mathematical models. However, the commonly used framework for a plant simulation is based on so called L-systems (Lindenmayer systems) proposed by Lindenmayer in 1968 [25]. However, all models, which were proposed in 1960–1990s, are far from the realistic models, ignoring the collisions between branches. The realistic extension of the L-systems was proposed by Lintermann and Deussen [26]. A mixture of procedural and rule-based methods is in the basis of this approach. Hereafter, the original concept of the L-systems was extended by the parametric L-systems, in which the grammar symbols are associated with the numerical parameters, the context-free and context sensitive parametric L-systems with the deterministic or stochastic production rules. The wide variety of tree-like shapes can be obtained by changing the numerical parameters and application of different rules for the branching angles. The modelling of 3D trees by an iterative reconstruction, using the evolutionary computation, image processing, and computer vision, was proposed by Zamuda and Brest [27]. An evolutionary algorithm iteratively extracted the morphological model parameters from two and more images. The vectorized auxiliary local branch parameters were used to condense and represent the matrix parameters of the branches as an alternative for the use of the L-systems. A branch structure of the coniferous trees is formed in the same manner as for the broad-leaves trees, using the special tuning rules of the L-system. The main difference deals with the foliage modelling. The foliage properties differ from species and depend on tree age as well as on the external environmental factors (e.g., water content, soil nutrients, and light availability). During experiments, a set of operators for the L-system modelling was specified, based on which the enhanced growing models of tree species were developed [28]. The experimental software tool “TreeEditor”, v. 1.04 was designed in the development environment RAD Studio 2010 using the graphical jet GLScene, the library

OpenGL, and some standard objects from the jet GLScene. It includes three main modules: the module of individual tree generation (six kinds of broad-leaved trees), the simulation module of weather conditions (imitation of wind, luminance, fog, and rain), and the module of a forest generation.

Chapter 6 presents a way for the realistic tree modelling, utilizing the procedural modelling methods and the ground-truth data of a laser scanning. Voxelization is a promising approach that evaluates a density of points dividing 3D grid cells into the high-density, the middle-density, and the zero-density structures. The analysis of their locations permits to make the hypotheses about individual trees distribution in a forest. After voxelization, 3D grid is divided in horizontal cuts in order to identify a crown shape with the possible intersections of the tree crowns. The 3D reconstruction supposes a final separation of a point cloud into the sets corresponding to the individual objects. Due to that the forest ecosystems demand a combination of the ALS and photography data, the procedural tree modelling ought to be improved by the algorithms that are considered the ground-truth ALS data. A variety of such algorithms is not wide. For these purpose, the space colonization algorithm [29], the graph-based modelling [30], the self-organizing tree modelling [31], or the inverse procedural modelling [32] are utilized. The space colonization algorithm is based on a competition for a space as the key factor, determining the branching structure of a tree. The suggestions of improvement proposed by Runions et al. [33] are concerned to the repetitively testing of a set of the attraction points for proximity to the tree nodes, using 3D Voronoi diagram with employing of 3D Delaunay triangulation routines. The irregular individual branches and the multiple levels of detail may be modelled by the regular graph, the irregular graph, or orthant neighborhood graph [34]. Based on the concept of the self-organizing, Pałubicki et al. [35] integrated a local control of the branching geometry, a competition of the buds and branches for space or light, and a regulation of this competition through an internal signaling mechanism into a model of a tree development. The main idea of the inverse procedural modeling is to combine the texture synthesis methods based on a local neighborhood comparison and the procedural modelling systems based on a local growth. During experiments, the space colonization algorithm and the inverse procedural modelling demonstrate better results in comparison with the graph-based methods, using the irregular graph structures, and the self-organizing modelling according to the shape, geometrical, and structural distance measures that were introduced by Stava et al. [36].

Chapter 7 includes the detailed description of many methods for building of the DTM that are categorized in three approaches, such as the point-wise methods, the patch-wise methods, and the global methods. The most widely used approach for the DTM construction from the LiDAR data is based on a detection of local minima in the grid cells. However, this method works well in a flat terrain with few trees and buildings [37]. Also the vegetation on slopes and the low-height vegetation may be present in a terrain. The reasonable assumption that the vegetation points are significantly higher than their neighborhoods and they can be easily filtered falls away in the steep and rough terrain but sometimes the terrain points may lie at the

same height as the vegetation points. This causes a necessity to apply the special algorithms for clustering of the LiDAR point cloud. Sithole and Vosselman [38] proposed four distinct concepts to separate the points of the objects and the bare Earth, such as the slope-based filter, the block-minimum filter, the surface-based filter, and the clustering/segmentation filter. Nowadays, the following classification of filtering is accepted: the surface-based filters, the slope-based filters [39], the morphology-based filters [40], and the segmentation/cluster-based filters [41]. For the DTMs generation, the spatial relationship methods, the statistical methods, and the curve-fitting methods are used. The spatial relationship methods are called as the deterministic interpolation techniques that generate the surfaces from the measured points based on either the extent of similarity (the inverse distance weighed method) or the degree of smoothing (radial basis functions). The geostatistical interpolation techniques (kriging) utilize the statistical properties of the measured points [42]. The minimum curvature is a popular interpolation method in the Earth science that generates a smoothest possible surface as a thin and linearly elastic plate, passing through each of the data values with a minimum amount of bending [43]. Additionally, the interpolation techniques of the ArcGIS 9 Spatial Analyst [44], involving PointInterp, Natural Neighbors, Trend method, and Topo to Raster command that had been added to the Inverse Distance Weighed, Spline, and Kriging interpolation methods provided by the ArcGIS 8.3 Spatial Analyst, are discussed.

Chapter 8 is devoted to the efficient texturing and rendering in large landscape scenes. The realistic visualization is a challenging problem due to a huge amount of information even for the individual tree and a number of these objects can achieve several hundreds and even thousands. There are two different approaches are known. The geometry-based approximations are represented by the polygonal meshes and can have the constant (non-adaptive approximations) or varying Level Of Details (LODs) (adaptive approximations). The image-based approximations employ impostors, billboards, billboard clouds, multi-layer Z-buffers, sprites with depth, the layered depth image, hierarchical bi-directional textures, and volumetric textures. The first LOD techniques were based on the discrete multi-resolution models. Such model contains a set of discrete object approximations that were generated by a repeated polygonal simplification. At run-time, only one object approximation can be selected for rendering [45]. The current LOD-based rendering approaches use the continuous multi-resolution models as the abstract data structures, including the incremental continuous multi-resolution model [46] and the hierarchical continuous multi-resolution model [47]. The finest LOD represents the full-resolution model, while the coarser LODs represent the lower resolution versions of the vertices, edges, and polygons suitable for the faster rendering. The surface parameterization specifies how points in 2D texture space are projected onto a surface in 3D object space. Two types of parameterization, such as the explicit mapping using the basic primitives and the induced mapping employing n -point correspondence, are discussed in this chapter. The modelling of terrains ought to be realistic, and the rendering of such scenes requires a low computational cost in

order to obtain the real-time application. During texturing, three challenges can appear. The first one occurs in incorrect mapping of texture, when a viewpoint is changed due to a texture represents an arbitrary subset of the model from a single viewpoint. In scenes with a complex geometry, a switching between the geometric shapes and textured shapes (or vice versa) may cause a sudden jump in the image. This is the second problem. This means that a transition between the geometry and texture ought to be smoothed. The third problem is caused by a storing of many texture samples that requires a vast amount of textures in the computer memory. The reduction of this subset can be obtained by the image warping for possible solutions. Some basic and extended techniques of the basic LOD approach are represented [48–50].

Chapter 9 presents the large scene rendering techniques, involving the terrain, object and lighting rendering methods. The LOD architecture is the basis of terrain and vegetation rendering. The terrain LOD algorithms can be categorized as the irregular meshes or triangulated irregular networks, the bin-tree hierarchies for the recursive bisection of the right triangles, the bin-tree regions as the refinement operations associated with a bin-tree structure, the tiled blocks partition a terrain into the square patches that are tessellated at the different resolutions, and the regular grids using the geometry clipmaps. The classification of the object rendering techniques is wide and includes the point-based rendering [51], the line-based rendering, the polygon-based rendering, the image-based rendering [52], and the volume-based rendering [53]. The texturing techniques are strongly connected with a category of the LOD algorithm. The multi-texturing, clipmaps, and virtual texturing are the main methods, applying in the LOD algorithms. The multi-texturing can be concern to the simple methods used to blend the multiple textures [54]. The clipmaps [55] is a common algorithm for the LOD of the textures and meshes that has been used since the 1990s. This algorithm provided an effective method of the displaying textures under the conditions of the limited GPU memory and restricted rendering possibilities of old computer assemblies. A virtual texturing is a technique for storage of the active sub-sections of the texture as small tiles in a single texture storage that then can be reassembled and rendered as if the complete texture was stored in the GPU memory. The Sparse Virtual Textures, the MegaTexture, and the Partial Resident Textures are similar implementations of the virtual textures. Also, the realistic lighting is still a challenge because the light interaction of vegetation, consisting of thousand millions of elements, is quite complex. Thus, Behrendt et al. [56] proposed a method that renders the complex geometries in a dynamic low-frequency lighting environment, using a special representation for the reflection function. A spherical harmonic lighting projects the rendering equation onto an ortho-normal basis (the Legendre polynomials) over a sphere. The shadow mapping has been used extensively in 3D virtual reality. Some perspective warping methods are introduced in literature, such as the Perspective Shadow Maps (PSMs) [57], the Light-space PSMs [58], the Logarithmic PSMs [59], the Parallel-Split Shadow Mapping [60]. At the end of this chapter, a scheme of rendering workflow of the Unity3D package with the properties of its shaders is introduced.

Chapter 10 involves the rendering techniques of the large landscape scenes under the meteorological impacts, such as wind, fog, rain, and snow, and some atmospheric phenomena. These modelling methods can be classified as the physical-based, computer-based, and hybrid approaches. The physical-based methods provide the accurate modelling with high computation cost. Many natural impacts are successfully described by the Navier-Stokes equations. Due to the complicated calculations, these methods cannot be implemented in real time that is one the important requirements in many practical applications. The opposite computer-based methods provide the simplified rendering, sometimes not realistic but fast and real-time. Here, the special solutions of computer graphics, e.g. shaders, are applied enforced by the hardware implementations. The hybrid approaches propose the intermediate decisions. Each meteorological impact is very interesting object of investigations and useful for practical applications, such as video games, serious gaming, landscape and urban scene walkthrough, various simulators, etc. Thus, a weak wind simulation is enough simple, and it can be interpreted as a synthesis of a stochastic video from a single image, sometimes known as “Tour into the Picture” system developed by Horry et al. [61]. The mid-force wind and storm wind simulation can be implemented by two approaches, such as the physically-based animations of the natural objects under physical forces [62] and the procedurally-based animations simulating the consequences of a wind influence [63]. The atmosphere always contains some concentration of sub-micron hygroscopic particles (for example, dust or smoke), which serve as the condensation nuclei of water vapor, when the air temperature is close to a dew point. This happens, when the air cools or the humidity rises. The most stable fogs are caused by an inversion of the air—ground temperature and can be of two types: the radiation fog called as a ground fog and the advection fog. The fog can be simulated using the mathematical models as well as by the low cost Perlin noise model [64]. Many researchers are addressed to the famous database of high quality renders of rain drops for various values of the illumination parameters that was produced by Garg and Nayar [65]. They presented the off-line rain rendering and streak simulation algorithm in still images and animations based a rain drop oscillation model for various atmospheric scenarios. Snow as a common natural phenomenon can be observed in three states, including:

- The snowfall that is filled the air with falling and fluttering snowflakes.
- The snow accumulation placing a white blanket over the landscape that is changing under the wind influence and the snow blanket properties.
- The snow melting under the lighting and temperature conditions, when a snow blanket or snowflakes are transferred into the water.

Also, the natural objects, such as lakes, rivers, seas, clouds, vapor trails, smoke, fire, explosions, among others, are often an integral part of landscape scene, and their realistic simulation and rendering ought to be provided in real time.

Chapter 11 studies the approaches of lighting and shadows rendering in natural scenes. The walkthrough of the natural landscapes means the providing of the best

visual quality of a scene with the imposed lighting and shadows conditions. The light sources are generally modelled as either points or directions. The lighting of a landscape scene can be interpreted as a scene with a single light source with a finite size—the Sun (a sunny day) and as a scene with the multiple light sources distributed uniformly, having a reduced lighting (a cloudy day). The objects cast the shadows, involving an umbra and penumbra. A space under umbra does not receive light from a source, while a space under penumbra obtains a light partially. The light source in a landscape scene changes its position slowly, and shadows also move and alter their shapes. The position of the light source determines a type of shadow projection, for example, an orthographic projection for infinitely far removed light source or a perspective projection, when a light source position lies outside the field of a view. The Monte Carlo ray tracing [66], shadow volume [67], and shadow mapping [68] are the traditional real-time shadow techniques for a shadow generation on a non-flat surface. A lighting of trees is difficult issue due to the light passes through the multi-layered structures of branches and foliage that have the complex reflectance properties. According to Boulanger [69], a direct lighting (the sky light) and its reflection of the ground can be estimated as a variable radiance from a low-frequency environment. The occlusions by other trees or man-made large objects are simulated by a modulating of the incoming radiance. The animations of leaves smooth the variations of an outgoing radiance. Notice that the accurate realistic shadowing of leaves and branches is too expensive for the real-time rendering.

Chapter 12 summarizes the problems of the forest ecosystems' modelling. The practical significance of forest modelling is very high in the framework of the close-to-nature forestry. At last decades, this technique has been practiced successfully in Europe. The achievements in computer graphics facilitate the process of forest monitoring, including the real walkthrough the woods and the manual measurements of many trees. The development cycles of natural forests can be used successfully in forest management planning [70]. The basic principles of close-to-nature silviculture are directed on the high-quality timber production, protecting of trees' regeneration, maintaining of a biodiversity, and ensuring of a continuous forest cover. The close-to-nature management ensures that the regeneration occurs at a sufficient rate, the large trees contribute to the self-thinning process, and natural pruning and intervention is not required. The single decision in a forest management at the tree level is whether to cut the best quality trees or not. Following this concept, all trees can be classified as “main” or “complementary”. The “main” trees in the forestry are represented in one of three categories [71], such as the stabiliser (the dominant and co-dominant trees), the sprinter because of a tree height, and the regeneration as a tree that increases the stability of the ecosystem. The “complementary” trees are the remaining trunks that are needed to support the “main” trees, provide a soil cover or protect other trees. Some models that simulate the life of trees within an ecosystem (the creation of new trees, their growth, and the elimination of the dead trees under the main factors of living conditions) are discussed [72–75].

1.3 Conclusions

This chapter provides a short description of the methods and algorithms, the software and hardware tools included in the book. All material is distributed in a logical consistency, and as the authors hope, the well structured chapters will be useful for many readers, who have interest to this huge area of investigation and implementation of the designed algorithms in various practical tasks.

References

1. Global Forest Resources Assessments. <http://www.fao.org/forest-resources-assessment/en/>. Accessed 14 Aug 2016
2. Hyppä J, Hyppä H, Litkey P, Yu X, Haggrén H, Rönnholm P, Pyysalo U, Pitkänen J, Maltamo M (2003) Algorithms and methods of airborne laser scanning for forest measurements. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXVI* (Part 8/W2):82–89
3. Mallet C, Bretar F (2009) Full-waveform topographic LiDAR: state-of-the-art. *ISPRS J Photogramm Remote Sens* 64(1):1–16
4. De Jong SM, Pebesma EJ, Lacaze B (2003) Above-ground biomass assessment of Mediterranean forests using airborne imaging spectrometry: the DAIS Peyne experiment. *Int J Remote Sens* 24(7):1505–1520
5. Ustin SL, Roberts DA, Gamon JA, Asner GP, Green RO (2004) Using imaging spectroscopy to study ecosystem processes and properties. *Bioscience* 54(6):523–534
6. Kokaly RF, Asner GP, Ollinger SV, Martin ME, Wessman CA (2009) Characterizing canopy biochemistry from imaging spectroscopy and its application to ecosystem studies. *Remote Sens Environ* 113:S78–S91
7. Morsdorf F, Nichol C, Malthus T, Woodhouse IH (2009) Assessing forest structural and physiological information content of multi-spectral LiDAR waveforms by radiative transfer modelling. *Remote Sens Environ* 113(10):2152–2163
8. Antonarakis AS, Munger JW, Moorcroft PR (2014) Imaging spectroscopy- and LiDAR-derived estimates of canopy composition and structure to improve predictions of forest carbon fluxes and ecosystem dynamics. *Geophys Res Lett* 41(7):2535–2542
9. Markets and Markets. <http://www.marketsandmarkets.com/Market-Reports/lidar-market-1261.html>. Accessed 31 July 2016
10. Hitz B, Ewin JJ, Hecht J (1998) Introduction to laser technology, 3rd edn. IEEE Press, New York
11. Silvast WT (2004) Laser fundamentals, 2nd edn. Cambridge University Press, Cambridge
12. Rottensteiner F, Sohn G, Gerke M, Wegner JD, Breitkopf U, Jung J (2014) Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS J Photogramm Remote Sens* 93:256–271
13. Wallace L, Lucieer A, Watson C, Turner D (2012) Development of a UAV–LiDAR system with application to forest inventory. *Remote Sens* 4(6):1519–1543
14. Terrasolid. <http://www.terrasolid.com/home.php>. Accessed 23 July 2016
15. Shen D, Wong DW, Camelli F, Liu Y (2013) An ArcScene plug-in for volumetric data conversion, modeling and spatial analysis. *Comput Geosci* 61:104–115
16. Bionatics. Simulation for decision. <http://www.bionatics.com/>. Accessed 19 July 2016
17. Planetside software. <http://www.planetside.co.uk>. Accessed 25 July 2016
18. GRASS GIS. Bringing advanced geospatial technologies to the world (2016) <http://grass.osgeo.org/>. Accessed 23 July 2016
19. Xfrog. <http://xfrog.com/>. Accessed 20 July 2016

20. Speedtree. <http://www.speedtree.com/>. Accessed 21 July 2016
21. Onyx Computing. <http://www.onxytree.com/>. Accessed 19 July 2016
22. Junttila V, Finley AO, Bradford JB, Kauranne T (2013) Strategies for minimizing sample size for use in airborne LiDAR-based forest inventory. *For Ecol Manage* 292:75–85
23. Yu X, Hyypä J, Vastaranta M, Holopainen M (2011) Predicting individual tree attributes from airborne laser point clouds based on random forest technique. *ISPRS J Photogramm Remote Sens* 66(1):28–37
24. Favorskaya M, Tkacheva A, Danilin IM, Medvedev EM (2015) Fusion of airborne LiDAR and digital photography data for tree crowns segmentation and measurement. In: Damiani E, Howlett RJ, Jain LC, Gallo L, De Pietro G (eds) Intelligent interactive multimedia systems and services. Springer International Publishing, Switzerland
25. Lindenmayer A (1968) Mathematical models for cellular interactions in development, Parts I and II. *J Theor Biol* 18:280–315
26. Lintermann B, Deussen O (1999) Interactive modelling of plants. *IEEE Comput Graphics Appl* 19(1):56–65
27. Zamuda A, Brest J (2014) Vectorized procedural models for animated trees reconstruction using differential evolution. *Inf Sci* 278:1–21
28. Favorskaya M, Zotin A, Chunina A (2011) Procedural modeling of broad-leaved trees under weather conditions in 3D virtual reality. In: Tsirhrintzis GA, Virvou M, Jain LC, Howlett RJ (eds) Intelligent interactive multimedia systems and services. Springer, Berlin
29. Runions A, Fuhrer M, Lane B, Federl P, Rolland-Lagan A-G, Prusinkiewicz P (2005) Modeling and visualization of leaf venation patterns. *J ACM Trans Graphics* 24(3):702–711
30. Estrada R, Tomasi C, Schmidler SC, Farsiu S (2014) Tree topology estimation. *IEEE Trans Pattern Anal Mach Intell* 37(8):1688–1701
31. Sachs T, Novoplansky A (1995) Tree form: architectural models do not suffice. *Israel J Plant Sci* 43:203–212
32. Ijiri T, Mech R, Igarashi T, Miller G (2008) An example-based procedural system for element arrangement. *Comput Graph Forum* 27(4):429–436
33. Runions A, Lane B, Prusinkiewicz P (2007) Modeling trees with a space colonization algorithm. Eurographics workshop on natural phenomena, pp 63–70
34. Germer T, Strothotte T (2008) Orthant neighborhood graphs: a decentralized approach for proximity queries in dynamic point sets. In: Braz J, Ranchordas A, Araújo HJ, Pereira JM (eds) Computer vision and computer graphics. Theory and applications, International conference VISIGRAPP 2007. Springer, Berlin
35. Palubicki W, Horel K, Longay S, Runions A, Lane B, Mech R, Prusinkiewicz P (2009) Self-organizing tree models for image synthesis. *ACM Trans Graph* 28(3):58.1–58.10
36. Stava O, Pirk S, Kratt J, Chen B, Mech R, Deussen O, Benes B (2014) Inverse procedural modeling of trees. *Comput Graph Forum* 33(6):118–131
37. Kraus K, Pfeifer N (2001) Advanced DTM generation from LiDAR data. *Int Arch Photogramm Remote Sens XXXIV(Part3/W4)*:23–30
38. Sithole G, Vosselman G (2004) Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds. *ISPRS J Photogramm Remote Sens* 59 (1–2):85–101
39. Sithole G (2001) Filtering of laser altimetry data using a slope adaptive filter. *Int Arch Photogramm Remote Sens XXXIV(Part 3/W4)*:203–210
40. Arefi H, Hahn M (2005) A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXVI(Part3/W19)*:120–125
41. Zhang J, Lin X (2013) Filtering airborne LiDAR data by embedding smoothness-constrained segmentation in progressive TIN densification. *ISPRS J Photogramm Remote Sens* 81:44–59
42. Lloyd CD, Atkinson PM (2006) Deriving ground surface digital elevation models from LiDAR data with geostatistics. *Int J Geogr Inf Sci* 20:535–563
43. Yang CS, Kao SP, Lee FB, Hung PS (2004) Twelve different interpolation methods: a case study of Surfer 8.0. XXth ISPRS Congress 35(B2):772–777

44. ArcGIS Spatial Analyst (2016) <http://www.esri.com/software/arcgis/extensions/spatialanalyst>. Accessed 25 July 2016
45. Sillion F, Drettakis G, Bodelet B (1997) Efficient impostor manipulation for real-time visualization of urban scenery. *Comput Graph Forum* 16(3):207–218
46. Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Werner Stuetzle W (1995) Multiresolution analysis of arbitrary meshes. In: 22nd annual conference on computer graphics and interactive techniques SIGGRAPH 1995, pp 173–182
47. Xia J, El-Sana J, Varshney A (1997) Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Trans Vis Comput Graph* 3(2):171–183
48. Wang H (1961) Proving theorems by pattern recognition II. *Bell Syst Tech J* 40:1–41
49. Jeschke S, Wimmer M (2002) Textured depth meshes for real-time rendering of arbitrary scenes. In: 13th eurographics workshop on rendering EGRW 2002, pp 181–190
50. Hilbert K, Brunnett G (2004) A hybrid LOD based rendering approach for dynamic scenes. In: IEEE International Conference on Computer Graphics CGIV 2004, pp 274–277
51. Reeves WT, Blau R (1985) Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Comput Graph* 19:313–322
52. Decoret X, Durand F, Sillion FX, Dorsey J (2003) Billboard clouds for extreme model simplification. *ACM Trans Graphics* 22(3):689–696
53. Decaudin P, Neyret F (2009) Volumetric billboards. *Comput Graph Forum* 28:2079–2089
54. Rocchini C, Cignoni P, Montani C, Scopigno R (1999) Multiple textures stitching and blending on 3D objects. In: 10th eurographics workshop on rendering EGWR 1999, pp 119–130
55. Tanner CC, Migdal CJ, Jones MT (1998) The Clipmap: a virtual mipmap. In: 25th annual conference on computer graphics and interactive techniques SIGGRAPH 1998, pp 151–158
56. Behrendt S, Colditz C, Franzke O, Kopf J, Deussen O (2005) Realistic real-time rendering of landscapes using billboard clouds. *Eurographics* 24(3):507–516
57. Stamminger M, Drettakis G (2002) Perspective shadow maps. In: 29th annual conference on computer graphics and interaction techniques, SIGGRAPH 2002, pp 557–562
58. Wimmer M, Scherzer D, Purgathofer W (2004) Light space perspective shadow maps. In: Keller A, Jensen HW (eds) *Rendering techniques 2004*, eurographics symposium on rendering 2004, pp 143–151
59. Lloyd DB, Govindaraju NK, Quammen C, Molnar SE, Manocha D (2008) Logarithmic perspective shadow maps. *ACM Trans Graph* 27(4):1–39
60. Zhang F, Sun H, Xu L, Lee K (2008) Hardware-accelerated parallel-split shadow maps. *Int J Image Graph* 8:223–241
61. Horry Y, Anjyo KI, Arai K (1997) Tour into the picture: using a spidery mesh interface to make animation from a single image. In: 24th annual conference on computer graphics and interactive techniques SIGGRAPH 1997, pp 225–232
62. Armstrong WW, Green MW (1985) The dynamics of articulated rigid bodies for purposes of animation. *Vis Comput* 1(4):231–240
63. Ota S, Tamura M, Fujimoto T, Muraoka K, Chiba N (2004) A hybrid method for real-time animation of trees swaying in wind fields. *Vis Comput* 20(10):613–623
64. Perlin K (1985) An image synthesizer. *Comput Graph* 19(3):287–296
65. Garg K, Nayar SK (2006) Photorealistic rendering of rain streaks. *ACM Trans Graphics* 25 (3):996–1002
66. Cook RL, Porter T, Carpenter L (1984) Distributed ray tracing. In: 11th annual conference on computer graphics and interactive techniques ACM SIGGRAPH 1984, pp 137–145
67. Crow FC (1977) Shadow algorithms for computer graphics. *Comput Graph* 11(2):242–248
68. Williams L (1978) Casting curved shadows on curved surfaces. In: 5th annual conference on computer graphics and interactive techniques ACM SIGGRAPH 1978, pp 270–274
69. Boulanger K (2005) Real-time realistic rendering of nature scenes with dynamic lighting. Ph.D. dissertation, INRIA

70. FAO, ECE, ILO Committee on Forest Technology, Management and Training, 2003. Report of the seminar on close-to-nature forestry, Zvolen, Slovakia (2016) <http://www.unece.org/fileadmin/DAM/timber/joint-committee/facts.htm>. Accessed 11 July 2016
71. Martin-Fernandez S, Garcia-Abril A (2005) Optimisation of spatial allocation of forestry activities within a forest stand. *Comput Electron Agric* 49(1):159–174
72. Deussen O, Hanrahan P, Lintermann B, Mech R, Pharr M, Prusinkiewicz P (1998) Realistic modeling and rendering of plant ecosystems. In: 25th annual conference on computer graphics and interactive techniques SIGGRAPH 1998, pp 275–286
73. Pirk S, Stava O, Kratt J, Said MAM, Neubert B, Mech R, Benes B, Deussen O (2012) Plastic trees: interactive self-adapting botanical tree models. *ACM Trans Graphics* 31(4):Article No. 50
74. Zamuda A, Brest J (2013) Environmental framework to visualize emergent artificial forest ecosystems. *Inf Sci* 220:522–540
75. Polewski P, Yao W, Heurich M, Krzystek P, Stilla U (2015) Detection of fallen trees in ALS point clouds using a normalized cut approach trained by simulation. *ISPRS J Photogramm Remote Sens* 105:252–271

Part I

Airborne LiDAR and Optical

Measurements of Forest

Chapter 2

Overview of LiDAR Technologies and Equipment for Land Cover Scanning

Abstract The land cover of the Earth's surface impacts strongly on the climate and environment. Precise monitoring of land cover plays the significant role in the management of the natural resources and planning humanity development. In this sense, a remote sensing is an indispensable tool for study and prediction the global, regional, and local ecological issues. A remote sensing implies the use of satellite, airborne, and terrestrial shootings or their combination. In this book, the airborne and terrestrial shootings are discussed as the factors that contribute greatly in the study of the natural resources. Both types of shooting are based on the laser scanning of areas, using the LiDAR technologies. The terrestrial shooting is executed by the stationary (with a tripod use) and mobile laser scanning systems (using the cars, trams, or ships). At last decades, the laser scanning based on the Unmanned Aerial Vehicles (UAVs) started to be applied. In this chapter, an overview of the LiDAR techniques and equipment for the different types of laser scanning is given.

Keywords LiDAR technologies • Airborne laser scanning • UAV-based scanning • Terrestrial laser scanning • Mobile laser scanning • Hand-held laser scanning

2.1 Introduction

The remote sensing is an efficient tool to acquire the forest monitoring in global, regional, and local scales. The airborne LiDAR systems that emerged commercially in the middle of 1990s, provide the explicit 3D laser profiling and scanning in contrast to the 2D planimetric remote sensing data. The 3D coordinates (x , y , z) of a point cloud describe the 3D topographic profile of the Earth's surface, vegetation cover, and man-made objects. The airborne LiDAR technique has been effectively used for generating the Digital Elevation Model (DEM), the Digital Terrain Model (DTM), and the Digital Surface Model (DSM). On the one hand, an airborne shooting does not

sensitive to the lighting conditions. On the other hand, the meteorological conditions, as the cloudy or foggy days, influence negatively on the shooting results.

In 1999, Ackermann [1] pointed out three directions of airborne laser scanning, such as the extended applications, the consolidation and extension of the LiDAR data processing methods, and the additional information about the Earth's surface characteristics. Further, these propositions were proved in academic and industrial spheres. The outstanding review about laser scanners and their features, classification techniques, and further development of the LiDAR data processing was provided by Yan et al. [2].

Usually the Airborne Laser Scanning (ALS) and the Terrestrial Laser Scanning (TLS) systems record the indirect signal intensity that causes the problems in comparison to the use of the direct recorded values. The transmission loss and the topographic and atmospheric effects influence on the backscatter of the emitted laser power. This leads to a noticeably heterogeneous representation of the received power. The geometric and radiometric calibrations help to solve this problem.

The active remote sensing technique delivers not only detailed information about the coordinates but also data about the reflectance characteristics of the Earth's surface in the laser wavelength. Typically, the ALS systems provide the signal in the Near-InfraRed (NIR) spectra in the wavelength interval 800–1550 nm. The ALS systems record the return amplitude of echoes with a full-waveform digitization. The emitted laser shot interacts with the surface, generating the backscatter. The received signal as a function of time contains one or more peaks that correspond to the distinct reflections of a laser beam from the opaque or penetrable (like trees or shrubs) objects. Notice that in a field of the ALS, the terms "signal intensity", "reflectance intensity", and "pulse reflectance" are often used as synonym for the return amplitude or energy of one echo. As Höfle and Pfeifer mentioned in [3], the ALS systems, recording the intensity, are the small footprint scanners that operate with a beam divergence in the range of 0.3–0.8 mrad and a flying altitude above ground up to 3500 m. Herewith, the laser footprint diameter (with a beam divergence of 0.8 mrad) would be 0.8 and 2.8 m for the flying heights of 1000 and 3500 m, respectively. The spatial resolution (a footprint area) for cameras may vary strongly that caused by the changes in the flying altitude and the topography of the scanned surface (under the assumption that a beam divergence is constant during a flight). The variations in scan geometry lead to the appearance of the under-sampling (gaps) and over-sampling (overlapping footprints) that requires a re-computation of the obtained signal according to topography and flying altitude.

The chapter is organized as follows. The development of the LiDAR technologies is discussed in Sect. 2.2. An overview of airborne, the UAV, and terrestrial laser scanning, including the basic physical principals, methods of registration, and equipment, is represented in Sects. 2.3–2.5, respectively. In spite of the UAV laser scanning is a type of the ALS, it is reasonable to discuss it in separate section because of the principal differences in navigation and equipment. Section 2.6 includes a comparison of remote sensing techniques for forest inventory. Conclusions are drawn in Sect. 2.7.

2.2 Development of LiDAR Technology

Periodically, the global airborne LiDAR market reports are becoming available at internet sites. Thus, Cary and Associates of Longmont, CO. [4] analyzed the global airborne LiDAR market from 2005 to 2008. The survey indicated that there has been an increase of 64% in the number of LiDAR systems in use, 53% in LiDAR operators, and 100% in the number of end-users in that period. Looking out to 2012, the report predicted that the sales of software and data processing services will grow from 11 to 20% range. A list of the LiDAR applications included:

- Topographic mapping [5–9].
- Flood risk assessments [10–13].
- Watershed analysis [14].
- Forestry modeling and analysis [8, 15–25].
- Habitat ecology [26–28].
- Landslide investigation [29].
- 3D building modelling [30–32].
- Road extraction [33].
- Snow depth measurement [34].

At last decades, various methods (such as LiDAR data filtering, the DTM/DEM/DSM generation, full-waveform data modelling, geometric and radiometric calibration, among others) and data formats of LiDAR data processing were developed. The American Society of Photogrammetry and Remote Sensing (APSRS) also provided a platform for the formulation of the Log ASCII Standard (LAS) data format from version 1.1–1.4 [35–39] with the guidelines and manual for manipulating of the airborne LiDAR data. Besides, the International Society for Photogrammetry and Remote Sensing (ISPRS) had offered three benchmark testing datasets for examining the algorithms and methods for urban object classification and 3D building reconstruction [32]. Also the ISPRS international conferences are organized.

The commercial topographic LiDAR sensors usually utilize the near-infrared laser with the wavelength 1064–1550 nm. The returned signal is characterized by a high separability of spectral reflectance that permits to separate different land cover materials. As a result, a continuous high resolution 1D waveform profile can be built [40, 41]. Table 2.1 provides some studies that demonstrate the use of airborne LiDAR for land cover classification in terms of the scanning configuration, data resolution, a number of feature spaces used, classification technique, and overall accuracy in 2004–2012 (some data were taken from [2] and checked using the initial references).

In report [55], the state of LiDAR market is forecasted from 2016 to 2022. The LiDAR market was studied in 2015 and valued at USD 1.39 Billion in 2016. It is expected to reach USD 3.22 Billion by 2022, at the Compound Annual Growth Rate (CAGR) of 12.4% from 2016 to 2022. It was found that the terrestrial lasers hold the largest share of the LiDAR market among all product types. Key companies,

Table 2.1 List of studies using airborne LiDAR for land cover classification

Case study	Laser scanning configuration	Data resolution	Feature spaces	Classification technique	Overall accuracy (%)
Charaniya et al. [42]	PR = 25 kHz; PS = 0.26 m; WL = 1064 nm	0.5 m	2–5	4 classes. Gaussian mixture models	66–84
Brennan and Webster [43]	BD = 0.45 mrad, FH = 950 m, PR = 40 kHz, SR = 17 Hz, WL = 1064 nm	1 m	5	7–10 classes. Object-oriented	94–98
Lodha et al. [44]	PR = 25 kHz, PS = 0.26 m, WL = 1064 nm	0.5 m	5	3–4 classes. Support vector machine	89–96
Farid et al. [45]	ALTM 1233. FH = 750 m, FP = 15 cm, SR = 28 Hz, SA = $\pm 20^\circ$, WL = 1064 nm	0.9 m	4	7 classes. Maximum likelihood	78–80
Lodha et al. [46]	PR = 25 kHz, PS = 0.26 m, WL = 1064 nm	0.5 m	5	4 classes. AdaBoost	92–96
Orka et al. [47]	Optech ALTM 3100. BD = 0.26 mrad, FH = 7500 m, FS = 75 m/s, FP = 264 m, PR = 100 kHz, SR = 70 Hz, SA = $\pm 10^\circ$, PD = 5.09 pts/m ²	0.21 m	4	3 classes. Linear discriminant analysis	68–74
Im et al. [48]	Optech ALTM 2050 LiDAR Optech HALTM 2050 LiDAR PD = 15.3 pts/m ² , PR = 50 kHz, PS = 0.4 m., WL = 1064 nm	0.25 m	8	5 classes. Object-based, decision trees	92–94
Goncalves et al. [49]	Optech's ALTM 2033 (www.optech.ca). BD = 0.3 mrad, FH = 1500 m, PR = 33 kHz, SR = 50 Hz, SA = $\pm 10^\circ$, WL = 1064 nm, PD = 4 pts/m ²	0.5 m	3	4 classes. Maximum likelihood, hierarchical classification, object-oriented	85–92
Kim et al. [50]	Optech ALTM30/70. BD = 0.31 mrad, FH = 1200 m, SA < 110°, PR = 71 kHz, PD = 5 pts/m ² , WL = 1064 nm and Optech ALTM 3100, BD = 0.31 mrad, PR = 100 kHz, SA < 10°, PD = 20 pts/m ² , WL = 1064 nm. For both systems FP = 0.372 m with leaf-on data and FP = 0.279 m with leaf-off data	0.21 m	5	15 classes. Linear discriminant analysis	83–91

(continued)

Table 2.1 (continued)

Case study	Laser scanning configuration	Data resolution	Feature spaces	Classification technique	Overall accuracy (%)
Habib et al. [51]	Leica ALS50. BD = 0.330 mrad, FH = 600 m, PR = 83 kHz, PD = 4–5 pfs/m ² , WL = 1064 nm	0.2 m	1–2	3–5 classes. Maximum likelihood	30–70
Yan et al. [52]	Leica ALS50. BD = 0.33 0mrad, FH = 600 m, PR = 83 kHz, PD = 4–5 pfs/m ² , WL = 1064 nm	0.2 m	1–2	5 classes. Geometric and radiometric corrections	43–70
Buján et al. [53]	Optech ALTM 3025. BD = 0.2 mrad, FH = 1300 m, PR = 25 kHz, PD = 4 pfs/m ²	0.5 m	5	6 classes. Object-based, decision trees	93–97
Shaker and El-Ashmawy [54]	Leica ALS50 BD = 0.33 mrad, FH = 600 m, WL = 1064 nm	0.2 m	4	4 classes. Principle component analysis	70–73

BD is a beam divergence, FH is a flying height, FP is a footprint, FS is a flying speed, PD is a point density, PR is a pulse rate, PS is a point spacing, SR is a scan rate, SA is a swath (scan) angle, WL is a wavelength

operating in the market, are the Quantum Spatial (Aerometric) (U.S.), the Faro Technology (U.S.), the Geodigital (Canada), the Leica Geosystems AG (Switzerland), the Optech, Inc. (Canada), the Riegl Laser Measurement Systems GmbH (Austria), the Topcon Positioning Systems, Inc. (U.S.), the Trimble Navigation Limited (U.S.), the Sick AG (Germany), the Velodyne (U.S.), the BLOM (Norway), the Michael Baker International (U.S.), and the YellowScan (France).

2.3 Overview of Airborne Laser Scanning

A typical LiDAR system consists of a laser scanner and Global Positioning System/Inertial Navigation System (GPS/INS) navigation components. The laser scanner mounted on the platform produces a wide swath over, which the distances to the mapped surface are measured. The distances from the sensor to the mapped surface are computed by the time delay between the laser pulse transmission and detection. The laser beam direction (an encoder angle), at which the laser is scanned, is measured. The onboard GPS/INS components provide the position and orientation of the platform's movement. Then all types of information are used in the post-processing in order to calculate the coordinates of the point cloud. The laser technology was invented by Charles Townes and Arthur Schawlow in Bell Labs in 1958. Since that time, many investigations were done that was reflected in numerous publications, e.g., the books [56–58].

The physical principles of airborne laser scanning are discussed in Sect. 2.3.1. Section 2.3.2 explains the sources of the main errors, appearing during a laser scanning. Sections 2.3.3–2.3.4 provide the conventional and alternative LiDAR calibration methods, respectively. An overview of the equipment for airborne laser scanning is represented in Sect. 2.3.5.

2.3.1 Physical Principles of Airborne Laser Scanning

The basic measuring principle of laser scanning (airborne, using the UAV, or terrestrial) is the same but with the own features of application. Jelalian [59] proposed the radar range equation under the assumption that the receiver field of view matches the beam divergence and emitter and detector have the same distance to the target. Equation 2.1 comprises the sensor, target, and atmospheric parameters, where P_r is a received signal power, P_t is a transmitted signal power, D_r is a receiver aperture diameter, R is a range from sensor to target, β is a laser beam width, η_{sys} is a system transmission factor, η_{atm} is an atmospheric transmission factor, σ is a target cross-section.

$$P_r = \frac{P_t D_r^2}{4\pi R^4 \beta_t^2} \eta_{sys} \eta_{atm} \sigma \quad (2.1)$$

The effective target cross-section (backscattering cross-section) σ contains all target characteristics and is defined by Eq. 2.2, where Ω is a scattering solid angle of the target, ρ is a target reflectance, A_s is a target area.

$$\sigma = \frac{4\pi}{\Omega} \rho A_s \quad (2.2)$$

The direction of the reflection is determined by the angle of incidence α . This is an angle between the laser beam and the target area. It is defined as an angle enclosed by a surface normal and a laser shot direction. The reflectance is an averaged over the total target area portion of reflected to incident radiation from the target area in the laser wavelength. The specular or diffuse reflections influence differently on the direction and strength of the backscattering cross-section. The target size is the effective area illuminated by the laser beam, i.e., the size of the orthogonal-to-ray projected area of the scatterer.

Jelalian [59] introduced the simplification of Eq. 2.2 under the following assumptions:

- The entire footprint is reflected on one surface (extended target) and the target area A_s is circular. Hence, the entire footprint is defined by the laser beam width β_t and the range R .
- The target has a solid angle of π steradians ($\Omega = 2\pi$ for scattering into a half sphere).
- The surface has the Lambertian scattering characteristics. If an incident angle is greater than zero ($\alpha > 0^\circ$), then a target cross-section σ has a proportionality of $\cos \alpha$ [60, 61].

As a result, the parameters A_s and σ can be calculated, using Eqs. 2.3–2.4.

$$A_s = \frac{\pi R^2 \beta_t^2}{4} \quad (2.3)$$

$$\sigma = \pi \rho R^2 \beta_t^2 \cos \alpha \quad (2.4)$$

A substitution of Eqs. 2.3–2.4 in Eq. 2.1 leads to an inverse range square dependency of the received signal power, independent of the laser beam width (Eq. 2.5).

$$P_r = \frac{P_t D_r^2 \rho}{4R^2} \eta_{sys} \eta_{atm} \cos \alpha \quad (2.5)$$

Jelalian [59] show that the areas of the non-extended diffuse targets have different range dependencies. Thus, the point targets with an area smaller than the

footprint (e.g., a leaf) are range independent. The linear targets areas (e.g., a wire) are linear range dependent. As a consequence, the received power reflected from the non-extended targets underlies an inverse range-dependent function with higher power ($1/R^4$, $1/R^3$). Often the ALS systems do not record the emitted power or even the emitted waveform. Hence, the optical transmission efficiency of all optical components in the ALS system as well as the system transmission factor η_{sys} are assumed to be constant for a certain ALS system but vary for different systems and over time. The aperture diameter D is also set to be a constant factor.

The system-independent atmospheric effect defined by η_{atm} stands for the average atmospheric conditions at the time of a flight. A wavelength range strongly influences on the loss of energy primarily due to scattering and absorption of the laser photons in the atmosphere between the scanner and target, even if the atmospheric conditions are constant. For a wavelength of $1.06\text{ }\mu\text{m}$, the effect of scattering considerably exceeds a contribution of absorption [62]. For horizontal propagation, the attenuation a can range from 0.2 dB/km for extremely clear conditions to 3.9 dB/km for haze conditions [59]. For vertical propagation, the atmospheric transmittance typically increases with higher altitudes (the vertical propagation results in lower average attenuation coefficients in comparison to the horizontal propagation). For flying heights of 1000 , 2000 , and 3000 m above ground and above sea level, the average vertical attenuations are 0.22 , 0.17 , and 0.14 dB/km , respectively (these values were obtained for mid-latitude summer and rural aerosol conditions with a visibility of 25 km). Equation 2.6 helps to evaluate an influence of the atmospheric factors, where a is an atmospheric attenuation coefficient, dB/km , R is a range from sensor to target, m , the factor $10,000$ originates from a given in dB per km .

$$\eta_{atm} = 10^{-2Ra/10000} \quad (2.6)$$

From practical experience, the noise in the intensity measurement is around 10% . On the one hand, a value of the atmospheric transmission factor η_{atm} can be neglected due to very clear atmospheric conditions and small ranges. On the other hand, the model covers the large range differences. This leads to consider the factor η_{atm} . Due to the lack of meteorological data (e.g., temperature, water vapor, or aerosol concentration) and the high spatial variability of these parameters, an approximated value for parameter a has to be chosen as the average atmospheric conditions of a flight. Equation 2.5 can be rewritten, where factor C represents the sensor parameters (e.g., P_t , D , system losses) that are assumed to be constant within a flight, using the same ALS system settings.

$$P_r(R) = \alpha \frac{\rho}{R^2} 10^{-Ra/10000} \cos \alpha \cdot C \quad (2.7)$$

The left side of Eq. 2.7 is converted into a voltage, amplified in the ALS system, and finally transformed into a digital form through an unknown proprietary function. Assuming a linearity in this transformation, it is further possible to compare

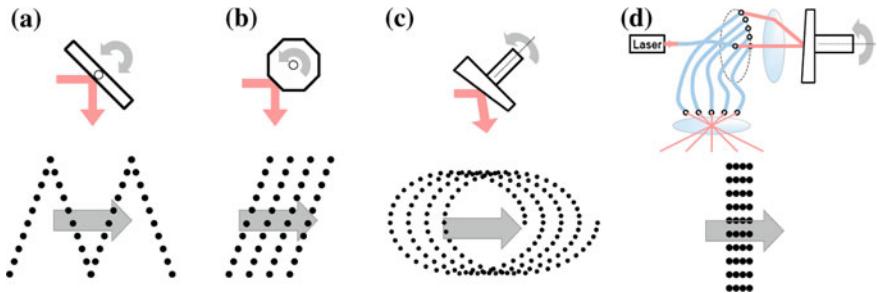


Fig. 2.1 Scanning mechanisms and ground patterns: **a** oscillating mirror and linear pattern (Z-shaped), **b** rotating polygon and linear pattern (parallel), **c** nutating mirror (palmer scan) and elliptical pattern, **d** fiber switch and linear pattern (along the *flight lines*)

the intensity measurements. The peaks of the Gaussian pulse waveforms can be compared (not more than 50% of a range width). Having the target areas with a defined reflectance ρ measured by a spectrometer on the ground, one can estimate a direct determination of parameter C . The normalization of the intensity to an average range and the estimation of C offer a reasonable approximation of surface reflectance under the mentioned above assumptions.

The LiDAR scanning pattern on the ground is affected by the scanning device, as well as the flight path, the flight speed, and the terrain topography. The types of scanning devices used in the commercial LiDAR systems are the following: an oscillating mirror, a rotating polygon, a nutating mirror, and a fiber switch. Their schematic illustrations are depicted in Fig. 2.1.

The oscillating mirror and rotating polygon yield the linear scanning patterns: the “zigzag” patterns (Fig. 2.1a) and the parallel lines on the ground (Fig. 2.1b), respectively. The non-linear scanning pattern is produced by the nutating mirror, providing an elliptical path of the laser beam (Fig. 2.1c). The advantage of the nutating mirror is that the ground is scanned twice from different directions (forward and backward). This means that the occluded areas in the forward scanning can be captured in the backward scanning. The fiber scanner consists of two arrays of glass fibers, transmitting and receiving. The number of glass fibers determines the number of laser footprints across the flying direction (Fig. 2.1d). The fiber scanner has an important advantage with respect to maintaining the stability of measuring laser beam directions because each glass fiber has a fixed beam direction. According to the reviews of the commercial LiDAR systems [63], the oscillating mirror and rotating polygon are the most popular.

The coordinates of the LiDAR footprints are the result of combining the derived measurements from each of its system components as well as the boresight and lever-arm parameters relating to such components. The relationship between the system measurements and the parameters is embodied in the LiDAR georeferencing equation [64], considering four coordinate systems, such as a mapping frame (ground coordinate system), an Inertial Measurement Unit (IMU) body frame, a laser unit frame, and a laser beam frame.

2.3.2 Errors of LiDAR Systems

The random and systematic errors are the main types of errors in the LiDAR system. The random errors are the errors of the system's measurements (the position and orientation of the integrated GPS/INS, encoder angles, and laser ranges) and provide a frequency distribution, usually normal distribution. The systematic errors are caused by the physical laws, can be predicted, and may lead to the biases in the boresight angles and lever-arm offsets of the system components and biases in the encoder angles and laser ranges of the system measurements.

The impact of random errors on the obtained point cloud can be studied by two ways. The first approach is based on a simulation procedure, while the second approach uses the law of error propagation. The simulation procedure defines the impact of the imposed noise in the system measurements. Suppose that an initial surface and trajectory are given. Then a noise is added to the system measurements, and a surface is reconstructed through the LiDAR georeferencing equation. The differences between the initial and reconstructed coordinates of footprints determine the random error influence. The impact of a noise in the system measurements are mentioned below [65]:

- The position noise leads to similar noise in the derived point cloud. The effect is independent of the system flight altitude and scan angle.
- The orientation noise (platform attitude or encoder angles) affects the horizontal coordinates more than the vertical coordinates within the nominal scan angle ranges. The effect is dependent on the system flight altitude and scan angle.
- The noise in laser range mainly affects the vertical component of the derived coordinates (especially in the nadir region). The effect is independent of the system flight altitude but influences on the system's scan angle.

The second approach studies the stochastic characteristics of dependent variables based on the law of error propagation applied to the LiDAR georeferencing equation [66]. For the LiDAR footprint, the law of error propagation is used to estimate the precision of the derived coordinates based on the precision of the system measurements. Using this approach, one can choose the parameters, providing the best precision for the given system and flight configuration, especially for flat and horizontal solid surfaces. Such expected precision of the measurements can be estimated based on the LiDAR system manufacturing parameters. However, the manufacturing parameters do not consider vegetation and atmospheric effects.

The systematic biases in the system measurements (the encoder angles and laser ranges) and system parameters (the boresight angles and lever-arm offset relating the system components) lead to the systematic errors in the derived point cloud. Alike the random errors, the impact of systematic biases can be derived through mathematical analysis of the LiDAR georeferencing equation or using a simulation process. The impact of the various systematic biases of the LiDAR system on the ground may depend on the flight direction (forward and backward), the flight altitude, and the encoder angle (Table 2.2).

Table 2.2 Impact of biases in the parameters and measurements of the LiDAR system with a linear scanner on the derived point cloud [65]

Type of bias	Flight altitude	Flight direction (forward/backward)	Encoder angle (across flight direction)
Biases in the lever-arm components	Effect is independent of the flight altitude	Planimetric effect is dependent on the flight direction Vertical effect is independent of the flight direction	Effect is independent of the encoder angle
Bias in the boresight pitch angle	Effect is dependent on the flight altitude	Planimetric effect along the flight direction is dependent on the flight direction	Effect is independent of the encoder angle
Bias in the boresight roll angle	Planimetric effect across the flight direction is dependent on the flight altitude Vertical effect is independent of the flight altitude	Planimetric effect across the flight direction and vertical effect are dependent on the flight direction	Planimetric effect across the flight direction is independent of the encoder angle Vertical effect is dependent on the encoder angle
Bias in the boresight yaw angle	Effect is independent of the flight altitude	Planimetric effect along the flight direction is independent of the flight direction	Planimetric effect along the flight direction is dependent on the encoder angle
Bias in the laser range measurements	Effect is independent of the flight altitude	Planimetric effect across the flight direction and vertical effect are independent of the flight direction	Planimetric effect (D_x) across the flight direction and vertical effect (D_z) are dependent on the encoder angle (D_x more than D_z)
Bias in the encoder angle scale factor	Effect is dependent on the flight altitude	Planimetric effect across the flight direction and vertical effect are independent of the flight direction	Planimetric effect across the flight direction and vertical effect are dependent on the encoder angle

For a simulation process, a surface (usually flat) and a trajectory ought to be given. Then the biases are added to the system parameters and measurements in order to generate a bias-contaminated point cloud. The obtained point cloud is compared with the true surface in order to estimate the impact of the systematic errors. A compatibility of the overlapping strips is evaluated using two strips in forward and backward directions.

2.3.3 *Conventional LiDAR Calibration Methods*

The outcome of the LiDAR system is a point cloud, including 3D coordinates and, sometimes, the intensity values. The LiDAR georeferencing equation is a function of the boresight angles, lever-arm offset, and system measurements. Usually the calibration of the LiDAR system is required because of the biases of the system parameters. The Quality Assurance (QA) procedure helps to improve a quality of point cloud coordinates. The conventional calibration procedures require a full access to the raw measurements from the GPS/INS components and laser scanner as well as the use of control data. Notice that the vertical accuracy of LiDAR data is relatively high (5–30 cm) compared to other methodologies, while the horizontal accuracy is about 20–50 cm [67]. The quality of the LiDAR data is affected by many factors:

- Accuracy of the integrated GPS/INS position and orientation.
- Validity of the system parameters.
- Point density.
- Flight altitude and speed.
- Amount and density of vegetation and buildings.

The geometric and radiometric calibrations are differed. Referring to the guidelines in part 2 and part 3 of the VDI/VDE 2634 [68, 69], the accuracy of 3D optical measuring systems based on area scanning shall be evaluated by checking the equipment at regular intervals. This can be achieved by the length measurements and measurements in the same way of typical measurement objects. Sometimes, the measurements of sphere spacing error are reasonable. The precision of 3D laser scanning systems includes the errors in distance and angle measurements, and depends from the algorithm for fitting the spheres/targets in the point cloud. The influence of these errors is difficult to determine separately. The detailed information for the TLS systems one can find in [70].

Höfle and Pfeifer [3] introduced two different methods for correcting the laser scanning intensity data as the data-driven correction, when the predefined homogeneous areas are used to estimate empirically the best parameters for a given global correction function accounting all range-dependent influences, and the model-driven correction that improves the intensity values based on the physical principle of radar systems. For the data-driven correction, the parameters are

evaluated, using the Least-Squares Adjustment (LSA) method. For both approaches, a set of laser points with their corresponding plane positions (x, y, z) and intensities I is required for reconstruction of a laser shot vector (direction and length) that is used for calculating the range and angle of an incidence.

For the data-driven correction, it is assumed that the recorded intensity is proportional to the ground reflectance and related to the flying height via empirical monotonic functions. It is considered that all physical effects are compensated, and the parameters calculated once can be used for further flights (with the same system settings and comparable atmospheric conditions). The general mathematical model has a view of Eq. 2.8, where I is an intensity, R is a range, I^{1000} is an intensity that is observed with $R = 1000$ m, $f(R)$ is an empirical function that can be represented by Eqs. 2.9 and 2.10 or other ones.

$$I(R) = I^{1000}f(R) \quad f(R) < f(R + \Delta R) \quad \forall \Delta R > 0 \quad f(1000) = 1 \quad (2.8)$$

$$f(R) = \frac{1}{k_1R + (1 - 1000k_1)} \quad (2.9)$$

$$f(R) = \frac{1}{k_1R + k_2R^2(1 - 1000k_1 - 1000^2k_2)} \quad (2.10)$$

The parameters k_1 and k_2 are the unknown coefficients that are calculated from the series of experiments. Thus, if the functional relationship is quadratic, at least three notably different ranges are required. The measurements ought to be such that one echo is returned. This requirement is introduced in order to overcome an over-parameterization.

The model-driven correction is carried out under the following propositions:

- The reflectors are assumed to be the Lambertian reflectors.
- The surface slope can be estimated from a neighborhood of points.
- The atmospheric conditions are known and constant.
- The transmitted laser power is assumed to be constant.
- The incoming power is linear respect to the recorded intensity values.

The first two assumptions are fulfilled over an open terrain, while in the forests multiple reflections are the norm and it is impossible to obtain the measures close to physical properties. If all propositions are fulfilled, then the result is a value directly proportional to ρ and is impressed by Eq. 2.11, where $\rho_{\text{diffuse}}(R_s, \alpha)$ is a value proportional to ρ normalized on standard range R_s , I is a recorded intensity, R is a recorded range, α is an angle of incident defined as angle between the surface normal and the incoming laser shot ray.

$$I(R_s) = \rho_{\text{diffuse}}(R_s, \alpha)\alpha \frac{R^2}{R_s^2} 10^{2Ra/10000} \frac{1}{\cos \alpha} \quad (2.11)$$

The coefficient a in Eq. 2.11 considers an average value for the atmosphere between airplane and ground. However, the atmospheric conditions during a flight may be complex. More sophisticated meteorological model ought to suppose that the coefficient a is a function of a range R and the absolute elevation z because the concentration of the scattering particles decreases with an altitude.

2.3.4 Alternative LiDAR Calibration Methods

The elimination of systematic errors provides a full potential accuracy of laser scanners. Bang [65] and Habib et al. [71] investigated the system parameters (including the mounting parameters) and the systematic errors in a laser scanner (the encoder angle scale factor and the laser range bias), using the point cloud coordinates of the overlapping strips. This leads to appearance of two alternative calibration methods that determined the biases in the system parameters and overcame the limitations of well-known calibration procedures of raw LiDAR data. The “simplified method” is applied in the case of the parallel overlapping strips, when the systematic biases are estimated using the identified discrepancies between the conjugate primitives in the overlapping LiDAR strips. The “quasi-rigorous method” deals with the non-parallel strips over a rugged terrain. This method requires the time-tagged LiDAR point cloud and the navigation data (trajectory position).

The simplified method is functioned under the following assumptions:

- Linear scanning systems are considered.
- Variations in the object space elevations are much smaller than the flight altitude.
- The flight lines are parallel.
- The platform trajectory is straight.
- Values of the roll and pitch angles are small enough to be ignored.
- Values of the boresight angles are assumed to be very small.

In contrast to the simplified method, the quasi-rigorous method deals with the non-parallel strips, heading variations and varying terrain elevations. It can be achieved by the use of the time-tagged point cloud and the trajectory position data. This method was developed under the following assumptions:

- Linear scanning systems are considered.
- Values of the roll and pitch angles are very small, and can be ignored.
- Values of the boresight angles are small, and can be ignored.

Unlike the simplified method, the quasi-rigorous method does not require the straight flight lines because the firing point position and heading are estimated by the trajectory position data and the time-tagged points.

The mathematical derivation for both methods may be found in the researches of Bang [65] and Habib et al. [71].

2.3.5 Equipment for Airborne Laser Scanning

The commercial civil companies, leaders in manufacturing of equipment for the ALS, are the Teledyne Optech, the Applanix POS AV, the TopoSys Harrier Model Range, the Riegl, the Leica, among others. Consider the products of the Teledyne Optech that has a wide family of laser scanners and digital cameras for the ALS systems.

Over the last 30 years, the Optech has proven that airborne surveying offers many advantages for acquiring spatially-located data across large areas [72]. From an aircraft's vantage point, the sensors capture the wide areas of terrain, such as cities, forests and farmlands, or long corridors such as power lines, railways, and rivers. Airborne sensors also have the speed and flexibility to survey many square kilometers in a single flight, even over rough or inaccessible terrain. The spheres of application are depicted in Fig. 2.2. The available Optech LiDAR systems are described in Table 2.3.

The Optech has a full line of airborne camera systems (from high resolution 80 megapixel RGB, to multispectral and thermal cameras) for a variety of applications in addition to being optimized for the LiDAR integration [73]. All airborne camera systems include a kinematic mounting for flexible configuration and a choice of interchangeable lenses to fit the specific flight requirements. The available Optech cameras are described in Table 2.4.

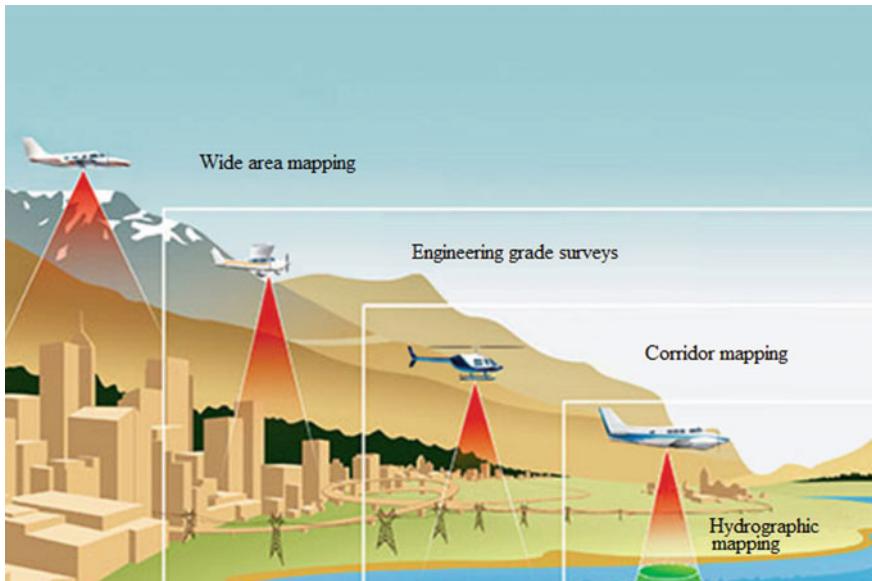


Fig. 2.2 Spheres of application in a schematic view (from [72])

Table 2.3 The Optech LiDAR systems from [72]

Title and assignment	Advantages	Applications	View
The Optech Eclipse, autonomous LiDAR and imagery data collection	<p>Fully autonomous airborne collection system for low-cost platforms</p> <p>Completely integrated active and passive imaging system</p> <p>Class 1 eye-safe laser for freedom of operation in environments</p> <p>Minimum of 4 points/m² from 1000 m (3300 ft) the above ground level (AGL)</p> <p>Up to 7 returns per shot for improved vertical density</p> <p>Rotating polygon scanner for parallel scan lines</p>	<p>Small-area surveying</p> <p>Power line mapping</p> <p>Pipeline monitoring</p> <p>Construction site monitoring</p> <p>Mine monitoring</p> <p>Stockpile surveying</p> <p>Archaeology and cultural heritage documentation</p>	 The Optech Eclipse System
The Optech Galaxy	<p>550 kHz effective pulse repetition frequency (PRF)</p> <p>Up to 8 returns per pulse</p> <p>Fully programmable scanner provides a high point density at lesser field of view (FOV)</p> <p>Minimum target separation distance less than 0.7 m</p> <p>Wide dynamic range</p>	<p>Wide-area mapping</p> <p>Power line and transportation corridor</p> <p>Natural resource management</p> <p>Engineering and infrastructure modeling</p> <p>Urban mapping</p> <p>Defense and security</p>	 The Optech Galaxy

(continued)

Table 2.3 (continued)

Title and assignment	Advantages	Applications	View
Galaxy's new PulseTRAK and SwathTRAK technologies (patents pending) make surveying simpler than ever before by providing unique innovative feature sets that maximize productivity, increase information content and reduce overhead costs. The Optech Galaxy may be installed in a tactical UAV integrated in a helicopter pod for power line surveying or gyro-stabilized with an orthometric camera for a wide-area mapping	Real-time XYZI point clouds in LAS format Low power requirements and compact form factor Tight integration with the Teledyne Optech's modular line of digital cameras (RGB, NIR, thermal, and multispectral)	The Optech Galaxy LiDAR sensor with optional GSM 4000	
The Optech Titan	900 kHz of high-density mapping capability Independent wavelengths, featuring "green" and NIR channels for topographic, shallow water bathymetry, vegetative mapping, and environmental modeling High-precision timing electronics, narrow laser pulse widths and beam divergences Fully programmable scanner 29 MP high-resolution, fully electronic QA camera Optional embedded 80 MP orthometric camera with a forward motion compensation Real-time XYZI point display	Topographic mapping Land cover classification Seamless shallow water bathymetry Environmental modeling Forest inventory and vegetative classification Natural resource management Disaster response	

(continued)

Table 2.3 (continued)

Title and assignment	Advantages	Applications	View
The Optech Orion The Optech Orion Airborne Laser Terrain Mapper is an ultra-compact total mapping solution with high data precision and accuracy. Modular components, direct upgrade options, and plug-and-fly passive imaging sensors deliver the scalable and flexible configurations. Three models of the Orion include: <ul style="list-style-type: none"> - The Orion C is a low-altitude system for corridor and engineering mapping applications, such as power line, pipeline, and urban infrastructure surveying. The industry-leading ranging precision is of <10 mm. - The Orion M is a low- to mid-altitude system provides a wider operational envelope enabling a wider range of applications including, urban modeling, engineering and transportation, and small topographic surveys. - The Orion H is a high-performance sensor used for the low-altitude corridor projects and/or high-altitude mountain surveys. 	Compact form factor and low power requirements Exceptional small-target detection capability Real-time data monitoring with in-air LiDAR point cloud display and coverage maps Output XYZI point clouds in the LAS format in real-time Passive imaging capability with Optech's modular line of digital cameras (RGB, NIR, thermal, and multispectral) Flexible multi-sensor mounts for both helicopter and aircraft installations The consistent data quality and accuracy with production-focused workflow software with automated calibration capabilities	Corridor and asset Defense and security Natural resources Surveying and mapping (urban mapping)	 The Optech Orion system
		 The Optech Orion with integrated camera in sled mount	
		 The Optech Orion mounted in GSM	(continued)

Table 2.3 (continued)

Title and assignment	Advantages	Applications	View
A multi-laser sensor, the Optech Pegasus includes a fully embedded, high-resolution orthometric camera and delivers extremely dense and information-rich datasets. The Optech Pegasus provides the highest point density and data accuracy at the lowest cost per square mile/km	<p>Multi-laser design obtains twice the efficiency and data density of single-laser systems</p> <p>Fully-embedded, medium-format digital camera provides simultaneously-collected, high-resolution imagery</p>	<p>Natural resources (agriculture, environmental, forestry)</p> <p>Intelligent digitizer automatically sub-samples collection rates above 100 kHz</p> <p>Detailed pulse amplitude and cross-section information</p> <p>Records only signals of interest (above user threshold)</p> <p>Real-time data display</p> <p>The SSD storage for unrestricted flight altitudes</p> <p>Timestamps waveforms with GPS time for synchronization with other altimeter data</p> <p>Accurate and precise datasets in the LAS 1.3 formats</p>	 <p>The Optech Pegasus system</p>
The Optech Waveform Recorder	<p>The Optech Waveform Recorder provides waveform capture capability for all ALTM models. Benefits include true ground detection, increased return density, improved target separation distances, and improved classification results. This standalone configuration also enables the Optech Waveform Recorder to be moved from system to system as the need arises, giving surveyors the on-demand capability they need</p>	<p>The Optech Waveform Recorder</p>	 <p>The Optech Waveform Recorder</p>

Table 2.4 The Optech cameras (high-accuracy orthographic photography, thermal imaging, multispectral imaging) from [73]

Title and assignment	Advantages	Applications	View
The Optech CS-10000 Aerial Digital Camera	<p>Field-replaceable shutter Interchangeable lenses</p> <p>Kinematic mounting for precise positioning The global system mobile (GSM) and sled mounts for configuration with the LiDAR sensors</p> <p>Batch processing provides the efficient imaging workflows</p> <p>80-Mpixel sensor with superior Ground Sample Distance (GSD) capacity</p> <p>Standalone or integrated with LiDAR, the Optech CS-10000 is a complete corridor and small-area mapping solution. With 10,320 pixels across by 7760 pixels along the flight line, the Optech CS-10000 delivers a breath-taking imagery for detailed feature visualization and accurate engineering applications. Based on the patented technologies and advanced camera engineering, the Optech CS-10000 is optimized for both standalone imaging and the LiDAR integration</p>	<p>Corridor and asset (asset management, utilities)</p> <p>Defense and security</p> <p>Engineering (rail)</p> <p>Natural resources</p> <p>Surveying and mapping (digital photogrammetry, disaster management, urban mapping)</p>	
The Optech CS-6500 Aerial Digital Camera	<p>Electronic shutter for extremely fast frame rates</p> <p>Interchangeable lenses to fit expanding business needs</p> <p>Metric quality data results deliver accuracy</p> <p>The Optech CS-6500 is a stable, scalable, and versatile camera in a small and reliable form factor. Benefits include the LiDAR</p>	<p>Corridor and asset (asset management, pipelines, utilities)</p> <p>Defense and security (tactical surveillance and Intel)</p> <p>Engineering (rail)</p>	

(continued)

Table 2.4 (continued)

Title and assignment	Advantages	Applications	View
The Optech CS-LW640 Long-wave Thermal Camera	Designed for integration with the LiDAR and other cameras A field of view of 6500 pixels across by 4300 pixels along the flight line delivers more efficient survey flights. The Optech CS-6500 is seamlessly integrated with other Optech LiDARs and cameras to create a unique airborne imaging solution	Surveying and mapping (urban mapping)	 The Optech CC-R camera controller
The Optech CS-LW640 with other Optech cameras	Long-wave infrared sensing expands possible mapping solutions Ruggedized construction for durability and custom mounting Kinematic mounting for precise alignment with the LiDAR and RGB cameras Geometric calibration and fast pre-processing	Corridor and asset (utilities) Defense and security Natural resources (environmental, forestry) Surveying and mapping (urban mapping)	 The Optech CS-LW640 thermal camera
The Optech Orion C LiDAR sensor platform	Creates a unique airborne imaging solution. The Optech CSLW640 is based on an uncooled microbolometer sensor with the resolution of 640×480 , and shares the core features of all Optech cameras. Integrated with the Optech Orion C LiDAR sensor platform, it is a proven and powerful tool for mapping thermal features in 3D thermal mapping	Surveying and mapping (urban mapping)	 The Optech CC-R camera controller
The Optech CS-MW640 Mid-wave Thermal Camera	Mid-wave infrared sensing expands possible mapping solutions Kinematic mounting for precise alignment with the LiDAR and RGB cameras Geometric calibration and fast pre-processing Ruggedized construction leverages extensible CS-Series architecture	Corridor and asset (utilities) Defense and security Natural resources (environmental, forestry) Surveying and mapping (urban mapping)	 The Optech CS-MW640 thermal camera

(continued)

Table 2.4 (continued)

Title and assignment	Advantages	Applications	View
The Optech CS-MS1920 Aerial Digital Camera The Optech CS-MS1920 multispectral camera with other Optech cameras creates a unique airborne imaging solution. The Optech CS-MS1920 is based on a patented high-definition 3-Charge-Coupled Device (CCD) camera design with color-separating optics that works together with large-format progressive-scan CCD sensors to maximize image resolution, dynamic range, and FOV. The Optech CS-MS1920 camera is the preferred choice for specialized applications requiring true multispectral capabilities, and delivers the ultimate in digital imaging quality. The one-inch HD sensor format provides the large pixel and sensing area needed for wide coverage and high dynamic range. Advanced features such as exposure control and white balance maximize the usability. Video preview capability provides a convenient progressive scan real-time display	3, 4, and 5 spectral band configurations Ruggedized design and kinematic mounts for precise mounting Geometric calibration for superior image quality Designed for integration with the LiDAR and cameras 3 high-definition CCD with high dynamic range and 400–1000 nm sensitivity Prism beam splitter architecture with patented compensating optics Georeferenced solutions with batch pre-processing Custom filters and polarization option	Corridor and asset Defense and security Natural resources Surveying and mapping (disaster management, wide area mapping)	 The Optech CC-R camera controller
The Optech Orion C LiDAR sensor platform for 3D thermal mapping			 The Optech CS-MS1920 multispectral camera

(continued)

Table 2.4 (continued)

Title and assignment	Advantages	Applications	View
The Optech D-8900 Aerial Digital Camera With a footprint of 8900 pixels across by 6700 pixels along the flight line, the Optech D-8900 aerial digital camera is tightly integrated with the Optech Pegasus ALTM, and delivers accurate and precise imagery. A wide range of lens and filter options supports easy reconfiguration for specialized projects and applications	Extended acquisition windows True metric performance Adaptable and flexible with interchangeable lenses Robust controller for operational flexibility Improved workflow Improved time to delivery Simplified service and maintenance	Corridor and asset (asset management, utilities) Engineering (rail) Surveying and mapping (disaster management, urban mapping)	 The Optech D-8900 camera

The Optech D-8900 camera controller and tablet

Notice that a fusion of the LiDAR data and an imagery of any type is not a trivial task due to different nature of laser and video shooting [74, 75]. The detailed information about equipment for the ALS systems one can find in the corresponding sites, e.g., [76–79].

2.4 Overview of UAV Laser Scanning

Initially, the UAVs were introduced for the remote area investigation and mapping initially for military purpose. However, with the development of low-cost and light-weight sensors, the UAV systems became capable of carrying GPS, the Inertial Measurement Unit (IMU), and camera to serve a variety of civil applications, such as vegetation control [80], forest inventory [24], changing rivers [81], coastal flooding analysis [10], ice surface observation [82], forest fire monitoring [83], automatic detection of heat losses in buildings [84], etc. Nagai et al. [85] presented the UAV LiDAR system and direct georeferencing technique to process the LiDAR data for digital 3D modelling. Lin et al. [86] created a small helicopter system (4.5 kg) that is able to lift up a payload of 7 kg, including GPS/IMU, LiDAR, CCD camera, and thermal camera. This system had a wide scale mapping applications, such as the tree height estimation, the pole detection, the road extraction, and the DTM generation. Lin et al. [87] combined the data collected from the UAV imaging system and mobile mapping system for land cover classification in the urban environment. However, such issues as system calibration, sensor modelling, data fusion with onboard camera, and on-line data processing do not solved fully yet.

The particular qualities of UAV laser scanning are represented in Sect. 2.4.1, while the equipment for UAV laser scanning is performed in Sect. 2.4.2.

2.4.1 Particular Qualities of UAV Laser Scanning

The UAV is navigated by a small onboard GNSS/INS module. The main components of the own navigation module include the gyroscopes for measurement of angular velocity, the air pressure sensor, the magnetometer, and the accelerometer. The mission planning (flight path, flying height, velocity, and aims) is prepared using the ground station before a flight. The mission is executed fully automated but a wireless communication allows to track the actual position of the UAV and adapt the flight plan if it is possible. A pseudo-automatic or manual mode, e.g., for landing, is possible in the case of a signal loss or other unexpected problems. Flight-attitude data are logged and either transmitted in real time to the ground station or downloaded after the flight.

Because of limited payload and place, the restricted number of devices can be mounted on the UAV. Usually the UAV is equipped with light-weight cameras that

deliver the images in high quality and resolution after the mission. However, the camera's parameters can be unstable during a flight that causes a necessity in hardware or software stabilization. For vegetation analyses, the small and light-weight models of multispectral/hyperspectral sensors, the NIR, and/or Consumer InfraRed (CIR) sensors, laser scanners, spectrometers, and laser distance rangers and radar sounder are maintained [88].

The georeferencing can be done in several ways:

- Direct georeferencing (the direct measurement of camera position and orientation for each image) can be done with flight attitude data from the GNSS/IMU module [89].
- Georeferencing using the ground control. For dense forest, the extensive field work is necessary for accurate measurements of ground control.
- Combination of the direct georeferencing and ground control for accurate evaluation of the exterior orientation parameters.

The topics of the UAV autonomous control [90], the GPS-denied navigation [91], and the path planning [92] are discussed widely in areas including robotics, mechatronics, computer vision, and computer science. However, at present a lot of the existing algorithms in literature have not been realized onboard of the UAV in real time. This is a crucial problem, especially for indoor navigation. The developments in the Unmanned Aerial System (UAS)-based remote sensing, focusing on various types of the UASs, are actively discussed in literature [24, 93, 94].

2.4.2 *Equipment for UAV Laser Scanning*

One of the greatest advantages of the UAV laser scanning is its high flexibility and the relatively low operational costs. The miniaturization of the sensors and the increasing reliability of the navigation systems make the UAVs an indispensable instrument for many applications. Consider the UAV equipment manufactured by one of the leader companies in the world (Table 2.5).

Miniaturization has led to smaller payloads of sensors, computers, communication devices, and power supplies that have allowed smaller UAVs to perform the same functions as larger UAVs. Nowadays, nano-UAV (NUAV), micro-UAV (MAV), and mini-UAV (MUAV) are designed for military and civil purposes. Mini-UAVs are classified as man-portable, air-launched, and multi-mission devices [96]. Any UAV with a wingspan less than 2 m but greater than 30 cm is considered the MUAV. The MUAVs are suitable for military applications, such as for Intelligence, Surveillance, and Reconnaissance (ISR) missions, Nuclear, Biological, and Radiological (NBR) detection, communications relay, wiretapping, radar interference, and operations in cities and high-density population areas. The MAV is any UAV that has a wingspan of 30 cm or smaller. The MAVs are useful for battlefield reconnaissance, air monitoring, the NBR detection, target identification,

Table 2.5 The UAV equipment from [95]

Title and assignment	Advantages	Applications	View
The RIEGL VUX-1UAV survey-grade unmanned laser scanner The RIEGL VUX-1UAV is a very lightweight and compact laser scanner, meeting the challenges of emerging survey solutions. Specifications include: – 10 mm survey grade accuracy – 5 mm precision – Operating flight altitude up to more than 1000 ft – Field of view up to 330° – Scan speed up to 200 scans/s – Eye safe operation at Laser Class 1 – Main dimensions: 227 mm × 180 mm × 125 mm – Weight: 3.5 kg (without cooling fan device)/3.75 kg (with cooling fan device)	Regular point pattern, perfectly parallel scan lines Cutting edge the Riegl technology, providing echo signal digitization, online waveform processing, and multiple-time-around processing Multiple target capability, practically unlimited number of targets echoes Electrical interfaces for the GPS data string and Sync Pulse LAN-TCP/IP interface Scan data storage on internal 240 GByte SSD memory	Agriculture and forestry Archaeology and cultural heritage documentation Corridor mapping; power line, railway track, and pipeline inspection Topography in open-east mining Construction-site monitoring Surveying of urban environments Resource management	
The RIEGL VQ-480-U lightweight airborne unmanned laser scanners The V-Line Airborne Laser Scanner RIEGL VQ-480-U provides high speed data acquisition, using a narrow infrared laser beam and a fast line scanning mechanism. A high-accuracy laser ranging is based on the Riegl's unique echo digitization and online waveform processing. The scanning mechanism is based on a fast rotating multi-facet polygonal mirror. Specifications include:	High-accuracy ranging based on echo digitization and online waveform processing High laser repetition rate, fast data acquisition Multiple target capability, unlimited number of targets Perfectly linear scan lines Compact, rugged, and very lightweight design Electrical interfaces for the GPS data string and Sync Pulse	Topography and mining Corridor mapping Power line inspection Archaeology and cultural heritage Target classification	

(continued)

Table 2.5 (continued)

Title and assignment	Advantages	Applications	View
<ul style="list-style-type: none"> - 50–550 kHz laser pulse repetition rate - 25 mm accuracy - 25 mm precision - 10 m minimum range - Field of view 60° - Eye safe operation at Laser Class 1 - Main dimensions: 347.5 mm × 183 mm - Weight: 7.5 kg (without optional mounting frame; weight mounting frame: approx. 1 kg) 	<p>Mechanical interface for the IMU mounting Integrated TCP/IP Ethernet interface</p>	<p>Precision agriculture Topography in open-cast mining Terrain and canyon mapping Archaeology and cultural heritage documentation Surveying of urban environments Construction-site monitoring Corridor mapping (power line, railway track, and pipeline inspection)</p>	 <p>The RIEGL RiCOPTER with VUX-SYS</p> <p>The RIEGL VUX-SYS is a complete miniaturized airborne laser scanning system solution of low weight and compact size for flexible use in the UAS/UAV/RPAS, helicopter, gyrocopter and ultra-light aircraft installations. The system consists of the RIEGL VUX-UAV laser scanner, an IMU/GNSS system, a control unit and up to 4 optional cameras. The excellent measurement performance of the RIEGL VUX-UAV in combination with a precise fiber optic gyroscope and the GPS/GLONSASS receiver integrated grade measurement accuracy</p>
		<p>The RIEGL RiCOPTER with VUX-SYS</p> <p>The RIEGL RiCOPTER remotely piloted aircraft system equipped with the RIEGL VUX-SYS complete miniaturized, lightweight ALS System</p> <p>The RIEGL VUX-UAV lightweight airborne laser scanner fully integrated (providing 230° FOV, an effective measurement rate up to 350,000 measurements/sec, and 10 mm accuracy) Fibre-optic gyroscope and the GPS/GLONSASS receiver integrated Remote control via low-bandwidth data link Operates up to 2 digital cameras</p>	(continued)

Table 2.5 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL RiCOPTER remotely piloted aircraft system The robust and reliable airborne scanner carrying platform provides full mechanical and electrical integration of sensor system components into aircraft fuselage. The extremely lightweight carbon fiber main frame, foldable propeller carrier arms, and shock-absorbing undercarriage enable stable flight, safe landings and handy transportation. The flight characteristics of the X-8 array octocopter are smooth and stable in hovering positions as well as in demanding flight maneuvers under challenging conditions. For surveying missions, the RIEGL RiCOPTER is equipped with the RIEGL VUX-SYS for RiCOPTER, comprising the RIEGL VUX-1UAV LiDAR sensor, a IMU/GNSS unit, a control unit, and up to four high-resolution cameras with overall maximum payload of 16 kg (sensors and power supply)	High performance X-8 array foldable octocopter Payload weight 16 kg (sensors and power supply) Maximum Take-off Mass (MTOM) < 2.5 kg Flight endurance with maximum load 30 min Operating flight altitude AGL of >500 ft (operational limits for civil unmanned aircraft according to national regulations) Cushioned landing legs and shock-absorbing undercarriage for stable flights and safe landings Foldable propeller carrier arms, integrated carrying handle, and special box for transportation	Precision agriculture Topography in open-cast mining Terrain and canyon mapping Archaeology and cultural heritage documentation Surveying of urban environments Construction-site monitoring Corridor mapping (power line, railway track, and pipeline inspection)	 The RIEGL RiCOPTER remotely piloted aircraft system

and communications relay. The MAVs also can be used to reconnoiter building interiors.

A swarm has been defined as “modeled flight that is biologically inspired by the flights of flocking birds and swarming insects” [97]. The studies of grouping UAVs began until the early 1990s. The main idea was that a swarm of the UAVs can be performed like a network of assets and complete missions that have been reserved for larger UAVs or manned aircraft. It is evident that the use of multiple mini-UAVs or micro-UAVs rather than a single large one increases the efficiency greatly. Also, a swarm of inexpensive mini-UAVs and micro-UAVs possesses a redundancy advantage: if one member of the swarm is lost in action, the rest part of the swarm continues to carry out the mission.

2.5 Overview of Terrestrial Laser Scanning

The spheres of the TLS measurements are restricted by the non-large areas of surveillance. Due to the dense point clouds, the TLS is successfully applied in biomass estimations for forestry needs [98]. Fixed-position (mounted on a tripod) terrestrial laser scanners offer a high potential for 3D mapping of smaller areas with high detail. The TLS is the most suitable technique for detailed modelling of individual trees and canopies, providing the basic tree parameters, such as the number and position of trees, Diameter-at-Breast Height (DBH), and tree height. However, this technique, as well as the use of the hand-held scanners, cannot be applied for each tree in the forest. The appearance of the mobile laser scanning systems extended the scale possibilities of such investigations. Another interesting application of the TLS is a measurement of rill erosion in natural volumes, such as the banks of rivers, the slopes, mountains, etc. [99].

The particular qualities of the TLS are discussed in Sect. 2.5.1, while the equipment for this type of laser scanning is represented in Sect. 2.5.2. Similar to the mentioned above sections, the particular qualities of a mobile laser scanning are considered in Sects. 2.5.3 and 2.5.4 provides the equipment for a mobile laser scanning.

2.5.1 Particular Qualities of Terrestrial Laser Scanning

The TLS allows to scan surfaces from different angles, producing a rather comprehensive point cloud with sub-centimeter resolution, as compared to the resolutions of a few points per square meter that is typical for the Global Positioning System (GPS) based surveys. The high sampling rate of 4–40 kHz and the possibility to register several overlapping point clouds enable to promote the rapid survey across hundreds of meters. Resolution, range, and sampling rate of the TLS

permit the seamless integration of the collected data, creating the highly accurate DEMs.

The principle of TLS is the following: a highly collimated laser beam scans over a predefined solid angle in a regular scanning pattern and measures the time of flight of the laser signal. The scanning range of the mid-range terrestrial system allows to conduct the distance measurements in interval 2–800 m. Maas et al. [100] analyzed the main tree parameters, such as reliability and precision of the DBH, tree height, and trunk profile obtained by use of the TLS system. Five different study areas with varying scan parameters were used during the experiments. It was found that:

- An accuracy of tree detection was 97.5%.
- The DBH measurement accuracy determined with callipers ranged from 1.48 to 3.25 cm.
- The Root-Mean Square Error (RMSE) of the tree height measurements for different species was 4.55 m and remained 2.07 m, even with the removal of two outliers.

However, a quality of the TLS depends on the amount of object occlusion and the external environmental factors, such as wind or relative humidity (fog or light rain). The TLS in natural forest environments meets different levels of occlusion among the various vegetation components. The amount of occlusion depends on the width of the light beam, the point cloud density, and the use of the first or last return. Also, the resulting point clouds can be altered by the presence of a wind if the scanning time exceeds a few seconds. In the case of multiple scans from different viewpoints, a geometric scan registration adds another level of complexity to data pre-processing, reducing the adverse effects of object occlusion due to oversampling of areas. Côté et al. [101] proposed an architectural model of trees based on the following assumptions:

- The accounting of even small structural elements that can be resolved by the laser scanner (individual needles on a conifer).
- The lack of a priori information about leaf or shoot/needle inclination.
- The sophisticated distinguishing of wood and foliage from point cloud data.

In cases when the TLS data set is used, the 3D modelling of tree architecture is highly dependent on the scan acquisition parameters and data quality. Such constraints limit the efforts to model 3D tree architecture, especially for the coniferous species because they can be scanned already with the needles.

2.5.2 Equipment for Terrestrial Laser Scanning

Consider the equipment for the TLS manufactured by one of the leader companies in the world (Table 2.6).

Table 2.6 The terrestrial equipment from [102]

Title and assignment	Advantages	Applications	View
The RIEGL VZ-400i 3D laser scanning system The RIEGL VZ-400i is 3D laser scanning system that combines an innovative new processing architecture, internet connectivity, and a suite of MEMS sensors with Riegl's latest laser scanning engine technology. The real-time data flow is enabled through dual processing platforms. The first platform includes a dedicated processing system for data acquisition, waveform processing, and system operations. The second processing platform enables the real-time data registration, georeferencing, filtering, and analysis to be executed simultaneously. The RIEGL VZ-400i harnesses this power by streaming it in real time via the integrated 3G/4G/LTE modem, WiFi, Bluetooth, and Ethernet communications hardware. With its integrated gyroscope, accelerometer, compass and barometer, the RIEGL VZ-400i's 1200 kHz pulse repetition rate can be fully utilized in nearly any environment and orientation. The system enables an incredible range of flexibility by providing support for numerous external peripherals and accessories via its integrated USB ports and stable mounting points	Processing architecture for data acquisition and simultaneous geo-referencing, filtering, and analysis in real time Cloud connectivity via WiFi and 4G LTE High laser pulse repetition rate of up to 1.2 MHz Eye safe operation at Laser Class 1 Wide field of view $100^\circ \times 360^\circ$ High speed data acquisition up to 500,000 measurements/s Range up to 800 m, accuracy 5 mm Multiple target capability for an unlimited number of target echoes Optional full waveform data output	Architecture and facade measurements As-built surveying Archaeology and cultural heritage City modeling Tunnel surveying – Civil engineering Forestry Monitoring Research Topography	 The RIEGL VZ-400i 3D laser scanning system

(continued)

Table 2.6 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VZ-400 3D laser scanner with online waveform processing This system provides high speed, non-contact data acquisition, using a narrow infrared laser beam and a fast scanning mechanism. High-accuracy laser ranging is based upon the Riegl's unique echo digitization and online waveform processing that allows achieving superior measurement capability even under adverse atmospheric conditions and the evaluation of multiple target echoes. The line scanning mechanism is based upon a fast rotating multi-facet polygonal mirror. The RIEGL VZ-400 is a very compact and lightweight surveying instrument	Up to 122,000 measurements/s 5 mm accuracy and 3 mm repeatability 600 m range 1.5 m minimum range The Field of view 100° vertical/360° horizontal Eye safe operation at Laser Class 1 Main dimensions: 180 mm × 203 mm × 308 mm Integrated GPS receiver with antenna Various interfaces (LAN, WLAN, USB 2.0) Weight 9.6 kg	Street and railway mapping City modeling Mapping of coastal lines Power lines Civil engineering	 The RIEGL VZ-400 3D Laser scanner with online waveform processing

(continued)

Table 2.6 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VZ-1000 3D terrestrial laser scanners with online waveform processing This system provides high speed, non-contact data acquisition for ranges more than 1400 m, using a narrow infrared laser beam and a fast scanning mechanism. High-accuracy laser ranging is based upon the Riegl's unique echo digitization and online waveform processing that allows achieving superior measurement capability even under adverse atmospheric conditions and the evaluation of multiple target echoes. The line scanning mechanism is based upon a fast rotating multi-facet polygonal mirror. The RIEGL VZ-1000 is a very compact and lightweight surveying instrument	122,000 measurements/s, 300 kHz laser pulse repetition rate 8 mm accuracy and 5 mm precision 1400+ m range 2.5 m minimum range The Field of view 100° vertical/360° horizontal Eye safe operation at Laser Class 1 Main dimensions: 200 mm × 203 mm × 308 mm Integrated GPS receiver with antenna Various interfaces (LAN, WLAN, USB 2.0) Weight 9.8 kg	Topography and mining Archaeology and cultural heritage As-built surveying Monitoring	 The RIEGL VZ-1000 3D terrestrial laser scanners with online waveform processing (continued)

Table 2.6 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VZ-2000 long-range high speed 3D laser scanners The V-Line 3D Laser Scanner RIEGL VZ-2000 is characterized by an extremely high measurement rate offering high accuracy data acquisition up to 400,000 measurements/s and up to 240 scan lines/s. The scanner further offers exceptional long range measurement performance of more than 2000 m to natural surfaces, while still maintaining completely eye safe operation (Laser Class 1). The Riegl's unique V-Line technology based on echo digitization, online waveform processing, and multiple-time-around processing is the key to enabling such high speed, long range, high accuracy measurements even in poor visibility and demanding multi-target situations caused by dust, haze, rain, snow, etc.	Very long range up to more than 2,000 m 8 mm accuracy and 5 mm precision Very high effective measurement rate up to 400,000 measurements/s Eye safe operation in Laser Class 1 Wide FOV $100^\circ \times 360^\circ$ Main dimensions: 196 mm \times 203 mm \times 308 mm Integrated L1 GPS receiver with antenna Various interfaces (LAN, WLAN, USB 2.0) Weight 9.9 kg	Surveying in open-pit mining Measurement of bulk materials Civil engineering City modeling Mapping of construction sites and construction-site monitoring Archaeology and cultural heritage Monitoring Topography and mining	 The RIEGL VZ-2000 long-range high speed 3d laser scanners

(continued)

Table 2.6 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VZ-4000 3D terrestrial laser scanners with online waveform processing This 3D RIEGL VZ-Line Laser Scanner offers superior and unrivaled long range measurement performance of up to 4000 m reflectorless while still maintaining completely eye safe operation (Laser Class 1). The Riegl's unique V-Line technology is based on echo digitization and online waveform processing and is the key to enabling such extreme long range measurements. The RIEGL VZ-4000 operates even in poor visibility and demanding multi-target situations caused by dust, haze, rain, snow, etc. that are frequently found in difficult environments such as mine sites	Very long range up to 4000 m 15 mm accuracy and 10 mm precision 4000 m range 5 m minimum range Eye safe operation in Laser Class 1 Wide FOV $60^\circ \times 360^\circ$ High speed data acquisition up to 222,000 measurements/s Main dimensions: 248 mm \times 226 mm \times 450 mm Built-in calibrated digital camera Integrated L1 GPS receiver with antenna Built-in SSD data storage media Weight 14.5 kg	Topography and mining Monitoring Civil engineering Archaeology and cultural heritage	 The RIEGL VZ-4000 3D terrestrial laser scanners with online waveform processing

(continued)

Table 2.6 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VZ-6000 3D ultra long-range terrestrial laser scanners with online waveform processing The high speed, high resolution terrestrial 3D Laser Scanner RIEGL VZ-6000 offers an extremely long measurement range of more than 6000 m for topographic applications. Being the Laser Class 3B companion to the RIEGL VZ-4000, it is, due to its laser wavelength, exceptionally well suited for measuring snowy and icy terrain in glacier mapping and monitoring applications in mountainous regions. The RIEGL VZ-6000 is compatible with the well-proven Riegl software package RISCAN PRO for terrestrial laser scanning, the Riegl's interface library RiVLib, as well as the workflow-optimizing software packages RiMONITOR and RiMINING	Ultra long range up to 6000 m Up to 222,000 measurements/s 15 mm accuracy and 10 mm precision 6000 m range 5 m minimum range The FOV 60° vertical/360° horizontal Main dimensions: 248 mm × 226 mm × 450 mm Integrated L1 GPS receiver with antenna Built-in calibrated digital camera Built-in SSD data storage media Weight approx. 14.5 kg	Glacier and snowfield mapping Topography and mining Monitoring Civil engineering Archaeology and cultural heritage	 The RIEGL VZ-6000 3D ultra long-range terrestrial laser scanners with online waveform processing

A description of some other specifications for the TLS instruments, such as FARO LS 880HE80, Leica HDS6100, and Sick LMS151 as well as their application in experiments, one can find in [103].

2.5.3 *Particular Qualities of Mobile Laser Scanning*

The Mobile Laser Scanning (MLS) is a new technology at that the objects are continuously mapped by laser distance measurement from driving vehicles (such as a multi-van pickup, the Suburban Utility Vehicle (SUV), or even a ship or railway train), and transformed into 3D point cloud, using the GPS/IMU data. A comparison to the laser mapping technologies, like the ALS and the TLS, shows that the MLS combines components of both ALS and TLS. Angle of view, high point density and genuine 3D data are comparable with the TLS. Georeferencing capability and efficiency (in the sense of speedy mapping of large areas) are comparable characteristics of the ALS. Just like the ALS and the TLS, the MLS allows to combine the laser scanners with the RGB cameras in order to collect image data simultaneously with the laser data and to georeference them. A mobile mapping is expected to provide ease of mobilization and low costs in comparison to an airborne laser scanning. Especially, the MLS is attractive for projects, involving small areas and specific tasks. The current fields of the civil MLS application are the following:

- Inventory data collection for urban planning.
- Inventory-taking for the GIS applications (road inventory, road condition, tree register, lighting register, traffic sign register, railways inventory).
- Basic information for navigation (abnormal load transports).
- Basic information for modelling (volume control, facade documentation).

One of the most important operations for railways is the maintenance of infrastructures and mapping its surroundings for actual information. These operations need to be fast, accurate, and reliable for planning. The MLS technology is the most effective way of capturing the high resolution data. Due to the high volume of captured data, a post processing is very time consuming and labor intensive. In order to process the data effectively, the automatic solutions are required. The example of railways extraction is depicted in Fig. 2.3.

2.5.4 *Equipment for Mobile Laser Scanning*

Typical mobile LiDAR system consists of one or more laser scanners and digital cameras mounted on a vehicle (car, tram, or others). In the outdoor spaces, the trajectory of a vehicle is determined using the GPS and the high-grade IMU. Often

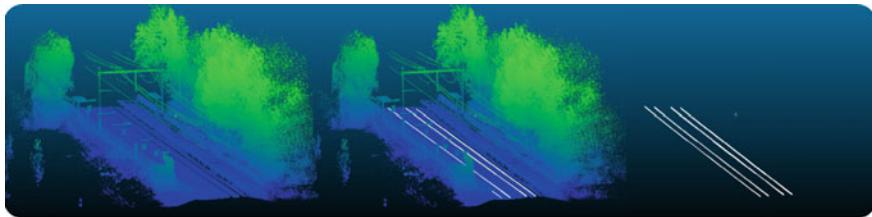


Fig. 2.3 Extraction of railways from LiDAR data from [104]

a wheel rotation sensor is added to obtain the odometry data. However, this type of system cannot be directly used for indoor applications because the GPS is not available indoors.

The MLS systems have been commercially available for several years and can achieve an accuracy of a few tens of mm in range 400 m [105]. Consider the equipment for the MLS manufactured by one of the leader companies in the world (Table 2.7).

One can mention other mobile scanners, e.g., the Trimble MX8 [107]. This device is a premium mobile spatial imaging system capturing fully synchronized, high-quality georeferenced point clouds and high-resolution imagery. The vehicle-mounted system is designed for surveyors, engineers, and geospatial professionals conducting as-built modeling, inventory, inspection, encroachment analysis, and asset management for roadways, bridges, railway, utilities and other infrastructure management. The main features are the following:

- Performance 360° mobile dual laser scanners collecting over one million points per second.
- High-frequency digital cameras at set orientations, capturing the high resolution panorama and surface imagery.
- The POS LV positioning and orientation system, delivering extremely fast position updates (up to 200 Hz) and high accuracy results, even when the Global Navigation Satellite System (GNSS) signals are interrupted.
- Rigidly mounted and fully calibrated pod with a wide navigation and sensor base for easy installation on a variety of vehicle types.
- Trimble Trident software to extract survey, the GIS and construction deliverables.

The given above classification may be extended by additional types of laser scanning, such as the Hand-Held Mobile Laser Scanning (HMLS) or a use of the Ground-Penetrating Radar (GPR). The HMLS has the outstanding ability to collect the topographic data in complex terrains, combining the inherent scale and reliability of laser techniques with the flexibility of on-foot surveying and the typical for scanning systems density of point cloud. At present, the HMLS has a maximum laser range of 30 m and utilizes a structured light measurement approach with ranges of up to 1–5 m. The automated 3D scene reconstruction from the raw data is

Table 2.7 The mobile equipment from [106]

Title and assignment	Advantages	Applications	View
The RIEGL VMX-1HA high performance dual scanner mobile mapping system. The RIEGL VMX-1HA is a high speed, high performance dual scanner mobile mapping system that provides dense, accurate, and feature-rich data at highway speeds. With 2 million measurements and 500 scan lines per second, this turnkey solution is ideally suited for survey-grade mobile mapping applications. This powerful technology comprises two RIEGL VUX-1HA High Accuracy LiDAR sensors and a high performance INS/GNSS unit. Optional camera systems complement the LiDAR data with precisely georeferenced images. Seamless Riegl workflow for the MLS data acquisition, processing, and adjustment is provided by the Riegl's proven software suite	<p>High laser pulse repetition rate of up to 2 MHz 500 scan lines per second Range 420 m Eye safe operation at Laser Class 1 The FOV 360° Multiple target capability Optional integration of up to 6 cameras Aerodynamically-shaped protective cover</p>	<p>Transportation infrastructure mapping Rail mapping Road surface measurements City modeling Rapid capture of construction sites and bulk material Surveying in open-pit mining GIS mapping and asset management As-built surveying</p>	 <p>The RIEGL VMX-1HA high performance dual scanner mobile mapping system</p>
The RIEGL VMX-1HA-RAIL	<p>Easy to mount lifting frame for fast installation by crane Additional mechanical interface for quick installation on various rail cars Main cable extensions for safe remote operation Optical distance measurement indicator optimized for railways</p> <p>With the RIEGL VMX-1HA-RAIL, the Riegl offers a proven, fully integrated high speed mobile laser scanning system for the demanding field of railway applications. It provides acquisition of a 360° field of view recording trackage, overhead wiring, rail heads and the complete periphery. The corresponding Riegl software packages offer comfortable features in data acquisition and processing and provide direct interfacing to third-party software packages to get main outcomes like clearance analysis, collision detection with train passage simulations, axis based measurements and surface monitoring, e.g., tunnel analysis</p>	<p>Mapping of rail tracks and rail infrastructure Rapid safe data capture with minimal disruption to network schedules Clash detection Axis based measurements of rail infrastructure Clearance analysis</p>	 <p>The RIEGL VMX-1HA-RAIL</p>

(continued)

Table 2.7 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VMQ-1HA high speed single scanner mobile mapping system The RIEGL VMQ-1HA is a compact, economically priced high speed single scanner mapping system, well suited for a variety of mobile mapping applications. The system consists of a measuring head, including one RIEGL VUX-1HA high accuracy LiDAR sensor, a compact control unit for system operation, and a special roof mount for convenient mounting. The optional integration of up to four cameras allows simultaneous acquisition of imagery to complement the captured LiDAR data. Seamless Riegel workflow for the MLS data acquisition, processing, and adjustment is provided by the Riegel's proven software suite	High laser pulse repetition rate of up to 1 MHz 250 scan lines per second Range 420 m Eye safe operation at Laser Class 1 The FOV 360° Multiple target capability Optional integration of up to 4 cameras	Transportation infrastructure mapping Road surface measurements City modeling Rapid capture of construction sites and bulk material As-built surveying GIS mapping and asset management Surveying in open-pit mining	
The RIEGL VUX-1HA compact rugged laser scanner The VUX-1HA high accuracy mobile scanning system is a very lightweight, compact and rugged laser scanner, that is easily mountable to whatasoever type of moving platform. It is perfectly suited for challenging survey missions based on cars, trains, robots, etc. This powerful technology comprises two RIEGL VUX-1HA high accuracy LiDAR sensors and a high performance INS/GNSS unit. The FOV of 360°, a very high laser pulse repetition rate of more than 1 MHz as well as an accuracy of 5 mm allow for excellent measurement results in applications like tunnel profile measuring, indoor and outdoor laser mapping, and railway applications	5 mm survey-grade accuracy Scan speed up to 250 scans/s Measurement rate up to 1,000,000 measurements/s The FOV of 360° Compact (227 × 180 × 125 mm), lightweight (3.5 kg), and rugged Electrical interfaces for the GPS data string and Sync Pulse LAN-TCP/IP interface Scan data storage on internal 240 GByte SSD memory	Indoor and outdoor laser mobile mapping Tunnel profile measurements Railway applications like clearance analysis	

(continued)

Table 2.7 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VMX-450 compact mobile laser scanners The RIEGL VMX-450 mobile laser scanning system offers extremely high measurement rates providing dense, accurate, and feature-rich data even at high driving speeds. The roof-carrier mounted measuring head integrates two RIEGL VQ-450 laser scanners as well as inertial measurement and GNSS equipment, housed under an aerodynamically-shaped protective cover. A camera platform ensures a setup of up to six digital cameras	Roadways mapping Rail corridors Waterways, ports, and harbors Monitoring of urban and vacant areas	 The RIEGL VMX-450 compact mobile laser scanners	
The RIEGL VMX-450-RAIL railway mapping mobile laser scanners With the RIEGL VMX-450-RAIL, the Rieg offers a proven, fully integrated high speed mobile laser scanning system for the demanding field of railway applications. It provides acquisition of a 360° FOV recording trackage, overhead wiring, rail heads and the complete periphery. The fully-calibrated measuring head with optional camera system and open interfaces to various external sensors is combined with a lifting frame for crane installation and a mounting frame interface to different rail cars for quick and user-friendly system setup. The corresponding Rieg software packages offer comfortable features in data acquisition and processing and provide direct interfacing to third-party software packages for main applications	Mapping of rail tracks and rail infrastructure Rapid safe data capture with minimal disruption to network schedules Clash detection Axis based measurements of rail infrastructure Clearance analysis	 The RIEGL VMX-450-RAIL railway mapping mobile laser scanners	(continued)

Table 2.7 (continued)

Title and assignment	Advantages	Applications	View
The RIEGL VQ-450 mobile laser scanners with online waveform processing The V-Line "Full Circle" laser scanner RIEGL VQ-450 is a very high speed, non-contact profile measuring system using a narrow infrared laser beam and a fast line scanning mechanism, enabling full 360 degree beam deflection without any gaps. It is characterized by a Laser Pulse Repetition Rate (PRR) of up to 550 kHz and a scanning rate of up to 200 lines per second. Multi-target capability based on echo digitization and online waveform analysis offers superior measurement capabilities even under adverse atmospheric conditions	Very high laser PRR up to 550 kHz Very high scanning rate up to 200 lines/s Very long range up to 800 m High-accuracy ranging Multiple target capability, unlimited number of targets Perfectly linear scan lines Compact, rugged, and lightweight design Integrated LAN-TCP/IP interface	Street and railway mapping City modeling Mapping of coastal lines – Power lines Civil engineering	
The IMU/GNSS unit, fully integrated to support the RIEGL VZ-400(i), VZ-1000 and VZ-2000 scanners for mobile data acquisition The fully integrated, accurate, and compact the RIEGL VMZ hybrid mobile laser mapping system enables combined static and kinematic data acquisition, using a single RIEGL VZ-400, VZ-1000 or VZ-2000 laser scanner. This leads to lower mobilization costs and a high return on investment. Flexible setup, easy mounting, and a user-friendly workflow mobilize the Riegl 3D terrestrial laser scanner for applications like mapping of transportation infrastructure, city modeling, mine surveying, bulk measurements, etc.	The IMU/GNSS unit, fully integrated to support the RIEGL VZ-400(i), VZ-1000 and VZ-2000 scanners for mobile data acquisition Quick switch from mobile to terrestrial applications, and vice versa Image data acquisition with a calibrated and the GPS synchronized NIKON DSLR camera Additionally, panoramic camera systems such as POINT GREY Ladybug are available Single power supply for VZ scanner and the IMU GNSS unit from a standard car battery Easy system operation with single laptop running RiACQUIRE	City modeling Surveying in open-pit mining Measurement of bulk materials Road surface scans Shore surveying and marine applications Civil engineering Topography Monitoring Facade modeling As-built surveying Architecture Archaeology	

carried out using the sophisticated Simultaneous Localization And Mapping (SLAM) algorithm that simultaneously computes the full trajectory of scanning and point cloud of surface measurements. The HMLS system works better in the indoor scenes with the static surfaces in comparison to the outdoor environments with highly irregular and covered with vegetation surfaces that cause more challenges during reconstruction [108]. Ryding et al. [109] demonstrated the efficiency of the HMLS system application in comparison to the conventional tripod TLS system. They show that the HMLS approach provides better survey coverage time per surveyor because the end-user can easily direct the scanner toward the points of interest and capture only the required data from the whole survey area.

The GPR is an established noninvasive (i.e., nondestructive) inspection method that has been used worldwide for more than thirty years to locate subsurface objects such as pipes, utilities, and other engineering and environmental targets. Bassuk et al. [110] proposed the methodology for locating of tree roots, using the GPR. The GPR employs the electromagnetic waves that reflect, refract and/or diffract from the boundary in a predictable manner in the boundaries between objects with different electro-magnetic properties of materials [111]. The benefits of a mapping tree roots based on the GPR measurements are the following:

- The method is capable of scanning root systems of large trees under field conditions in a relatively short time.
- The method is completely noninvasive and does not disturb the soils or damage the trees examined.
- The method allows the repeated measurements that reveal a long-term root system development.
- The method allows observation of root distribution beneath hard surfaces (e.g., concrete, asphalt, bricks, pavers, roads, buildings).
- The accuracy is sufficient to detect the structural roots with diameters till 1 cm.

Differences between the radar output and associated root counts in sample zones were found to be normally distributed in the CU-Structural Soil (CU-Structural Soil is a mixture of crushed gravel and soil with a small amount of hydrogel to prevent the soil and stone from separating during the mixing and installation process).

2.6 Comparison of Remote Sensing Techniques for Forest Inventory

The remote sensing data are successfully applied within the forestry industry, particularly, in computation of forest inventory metrics. The airborne and spaceborne sensors provide the spatially explicit data collected from large areas. It is well-known that the spatial extent and resolution are inversely related that makes the precision and accuracy are suboptimal for many tasks [25, 112].

The ALS data of 3D canopy structure are captured at heights between 500 and 5000 m. The forest properties, such as biomass and the Leaf Area Index (LAI) at the stand level, are evaluated using these ALS data as the input data for corresponding methods [113, 114]. The detection of individual trees from the ALS data is difficult with the reported detection rates varying significantly (between 40 and 96%) [115–117]. The 3D segmentation techniques may improve the accuracy of the information derived from a tree-level analysis [115, 118]. However, the required accuracy of tree segmentation is still not sufficient to estimate a number of forest metrics. More, the bias toward large dominant trees can cause significant overestimation of final inventory values such as timber yield [119]. This means the necessity of acquisition of numerous statistical data from large forest areas with the following processing and evaluation with real inventory data [25].

The terrestrial remote sensing techniques have been deployed actively in forest inventory [100]. During the TLS, the woody components of the canopy are often visible that allows to estimate the key tree-level inventory metrics, such as the DBH, shape of trunk, and crown length, objectively and with high precision. However, the terrestrial techniques can be used to measure small forest areas only and also the data collected by the TLS instruments are highly affected by occlusions. Therefore, the multiple viewing points are required in order to avoid a downward bias in trunk detection. The MLS systems overcome the small area restriction of the TLS by deploying the laser scanner onboard a moving vehicle or using the hand-held scanner. These systems are promising in deriving of individual tree-level parameters [120] but require further investigation.

Recently, the UAV laser scanning has been proposed as a tool for mapping and measuring tree metrics [24]. These systems provide a low-cost data collection with point densities up to 1000 points per square meter. Jaakkola et al. [121] provided a pilot study, showing that the underestimation of tree height presents in both the ALS and the TLS, while it is significantly reduced within the UAV data. This effect may be explained by the increased point density of the point clouds in comparison to the ALS and the TLS. (Objectively, the scanner and the sensors mounted on the on-board UAV platforms have higher errors and different sources of error.) Wallace et al. [122, 123] verified the precision of forest metrics from the UAV data scanning in a eucalypt plantation forest. The results of the percent canopy cover analysis proved that the UAV scanning data can provide the fast, low-cost, and objective information in this scope.

2.7 Conclusions

Three main types of laser technique, such as the airborne, the UAV, and terrestrial laser scanning, were considered, beginning from the physical basics and ending the commercial laser scanning systems for civil applications. Each type of laser scanning has the own applications in dependence on the mission, the scale of a studying area, and financial possibilities of the end-users. At present, one can choose the

laser scanner that has a high suitability for decision of the given practical task. Also, it was shown that a forest inventory is one the most popular topic among other applications that apply the laser scanning technologies.

References

1. Ackermann F (1999) Airborne laser scanning—present status and future expectations. *ISPRS J Photogramm Remote Sens* 54(2–3):64–67
2. Yan WY, Shaker A, El-Ashmawy N (2015) Urban land cover classification using airborne LiDAR data: a review. *Remote Sens Environ* 158:295–310
3. Höfle B, Pfeifer N (2007) Correction of laser scanning intensity data: data and model-driven approaches. *ISPRS J Photogramm Remote Sens* 62(6):415–433
4. LiDAR News. <http://blog.lidarnews.com/global-airborne-lidar-market-report/>. Accessed 31 July 2016
5. Webster TL, Forbes DL, Dickie S, Shreenan R (2004) Using topographic LiDAR to map flood risk from storm-surge events for Charlottetown, Prince Edward Island, Canada. *Can J Remote Sens* 30(1):64–76
6. Casas A, Benito G, Thorndycraft V, Rico M (2006) The topographic data source of digital terrain models as a key element in the accuracy of hydraulic flood modelling. *Earth Surf Proc Land* 31(4):444–456
7. Mallet C, Bretar F (2009) Full-waveform topographic LiDAR: state-of-the-art. *ISPRS J Photogramm Remote Sens* 64(1):1–16
8. Hancock S, Lewis P, Foster M, Disney M, Muller JP (2012) Measuring forests with dual wavelength LiDAR: a simulation study over topography. *Agric For Meteorol* 161:123–133
9. Renslow MS (2012) Manual of airborne topographic LiDAR. American Society for Photogrammetry and Remote Sensing
10. Webster TL, Forbes DL, MacKinnon E, Roberts D (2006) Flood-risk mapping for storm-surge events and sea-level rise using LidAR for southeast New Brunswick. *Can J Remote Sens* 32(2):194–211
11. Mason DC, Horritt MS, Hunter NM, Bates PD (2007) Use of fused airborne scanning laser altimetry and digital map data for urban flood modelling. *Hydrol Process* 21(11):1436–1447
12. Tsubaki R, Fujita I (2010) Unstructured grid generation using LiDAR data for urban flood inundation modelling. *Hydrol Process* 24(11):1404–1420
13. Arrighi C, Brugioni M, Castelli F, Franceschini S, Mazzanti B (2013) Urban microscale flood risk estimation with parsimonious hydraulic modelling and census data. *Nat Hazards Earth Syst Sci* 13:1375–1391
14. Garraway K, Hopkinson C, Jamieson R (2011) Surface moisture and vegetation influences on LiDAR intensity data in an agricultural watershed. *Can J Remote Sens* 37(3):275–284
15. Lim K, Treitz P, Wulder M, St-Onge B, Flood M (2003) LiDAR remote sensing of forest structure. *Prog Phys Geogr* 27(1):88–106
16. Chasmer L, Hopkinson C, Smith B, Treitz P (2006) Examining the influence of changing laser pulse repetition frequencies on conifer forest canopy returns. *Photogramm Eng Remote Sens* 72(12):1359–1367
17. Cuesta J, Chazette P, Allouis T, Flamant PH, Durrieu S, Sanak J, Genau P, Guyon D, Loustau D, Flamant C (2010) Observing the forest canopy with a new ultra-violet compact airborne LiDAR. *Sensors* 10(8):7386–7403
18. Hyppä J, Hyppä H, Leckie D, Gougeon F, Yu X, Maltamo M (2008) Review of methods of small-footprint airborne laser scanning for extracting forest inventory data in boreal forests. *Int J Remote Sens* 29(5):1339–1366

19. Lang MW, McCarty GW (2009) LiDAR intensity for improved detection of inundation below the forest canopy. *Wetlands* 29(4):1166–1178
20. Morsdorf F, Nichol C, Malthus T, Woodhouse IH (2009) Assessing forest structural and physiological information content of multi-spectral LiDAR waveforms by radiative transfer modelling. *Remote Sens Environ* 113(10):2152–2163
21. Korpela I, Orka HO, Hyppä J, Heikkinen V, Tokola T (2010) Range and AGC normalization in airborne discrete-return LiDAR intensity data for forest canopies. *ISPRS J Photogramm Remote Sens* 65(4):369–379
22. Allouis T, Durrieu S, Chazette P, Bailly JS, Cuesta J, Véga C, Flamant P, Couteron P (2011) Potential of an ultraviolet, medium-footprint LiDR prototype for retrieving forest structure. *ISPRS J Photogramm Remote Sens* 66(6):S92–S102
23. Gatzilis D (2011) Dynamic range-based intensity normalization for airborne, discrete return LiDAR data of forest canopies. *Photogramm Eng Remote Sens* 77(3):251–259
24. Wallace L, Lucieer A, Watson C, Turner D (2012) Development of a UAV–LiDAR system with application to forest inventory. *Remote Sens* 4(6):1519–1543
25. Wulder MA, White JC, Nelson RF, Næsset E, Ørka HO, Coops NC, Hilker T, Bater CW, Gobakken T (2012) LiDAR sampling for large-area forest characterization: a review. *Remote Sens Environ* 121:196–209
26. Bradbury RB, Hill RA, Mason DC, Hinsley SA, Wilson JD, Balzter H, Anderson GQ, Whittingham MJ, Davenport JI, Bellamy PE (2005) Modelling relationships between birds and vegetation structure using airborne LiDAR data: a review with case studies from agricultural and woodland environments. *IBIS* 147(3):443–452
27. Hartfield KA, Landau KI, Van Leeuwen WJ (2011) Fusion of high resolution aerial multispectral and LiDAR data: land cover in the context of urban mosquito habitat. *Remote Sens* 3(11):2364–2383
28. Desikan P, Karunakaran K, Gokulnath G (2013) Design of an aquatic park and salvation of endangered aquatic species in its natural habitat. *APCBEE Procedia* 5:197–202
29. Jaboyedoff M, Oppikofer T, Abellán A, Derron MH, Loyer A, Metzger R, Pedrazzini A (2012) Use of LIDAR in landslide investigations: a review. *Nat Hazards* 61(1):5–28
30. Haala N, Kada M (2010) An update on automatic 3D building reconstruction. *ISPRS J Photogramm Remote Sens* 65(6):570–580
31. Wang R (2013) 3D building modeling using images and LiDAR: A review. *Int J Image Data Fusion* 4(4):273–292
32. Rottensteiner F, Sohn G, Gerke M, Wegner JD, Breitkopf U, Jung J (2014) Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS J Photogramm Remote Sens* 93:256–271
33. Quackenbush LJ, Im J, Zuo Y (2013) Road extraction: a review of LiDAR-focused studies. In: Wang G, Weng Q (eds) *Remote sensing of natural resources*. CRC Press, Boca Raton
34. Deems JS, Painter TH, Finnegan DC (2013) LiDAR measurement of snow depth: a review. *J Glaciol* 59(215):467–479
35. ASPRS LIDAR data exchange format standard, version 1.0. http://www.asprs.org/wp-content/uploads/2010/12/asprs_las_format_v10.pdf. Accessed 31 July 2016
36. LAS specification, version 1.1. http://www.asprs.org/wp-content/uploads/2010/12/asprs_las_format_v11.pdf. Accessed 31 July 2016
37. LAS specification, version 1.2. http://www.asprs.org/wp-content/uploads/2010/12/asprs_las_format_v12.pdf. Accessed 31 July 2016
38. LAS specification, version 1.3-R11. http://www.asprs.org/wp-content/uploads/2010/12/LAS_1_3_r11.pdf. Accessed 31 July 2016
39. LAS specification, version 1.4-R13. http://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf. Accessed 31 July 2016
40. Kaasalainen S, Hyppä J, Litkey P, Hyppä H, Ahokas E, Kukko A, Kaartinen H (2007) Radiometric calibration of ALS intensity. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXVI, Part 3/W52*:201–205

41. Wagner W (2010) Radiometric calibration of small-footprint full-waveform airborne laser scanner measurements: basic physical concepts. *ISPRS J Photogramm Remote Sens* 65 (6):505–513
42. Charaniya A, Manduchi R, Lodha S (2004) Supervised parametric classification of aerial LiDAR data. In: IEEE 2004 conference on computer vision and pattern recognition workshop CVPRW 2004, vol 3, pp 1–8
43. Brennan R, Webster T (2006) Object-oriented land cover classification of LiDAR derived surfaces. *Can J Remote Sens* 32(2):162–172
44. Lodha SK, Kreps EJ, Helmbold DP, Fitzpatrick D (2006) Aerial LiDAR data classification using support vector machines (SVM). 3rd Int Symposium on 3D Data Processing, Visualization, and Transmission, pp 567–574
45. Farid A, Rautenkranz D, Goodrich DC, Marsh SE, Sorooshian S (2006) Riparian vegetation classification from airborne laser scanning data with an emphasis on cottonwood trees. *Can J Remote Sens* 32(1):15–18
46. Lodha S, Fitzpatrick D, Helmbold D (2007) Aerial LiDAR data classification using AdaBoost. In: 6th international conference on 3-D digital imaging and modeling 3DIM 2007, pp 435–442
47. Orka HO, Næsset E, Bollandsas OM (2007) Utilizing airborne laser intensity for tree species classification. In: ISPRS workshop on laser scanning 2007 and SilviLaser 2007, vol XXXVI, Part 3/W52, pp 300–304
48. Im J, Jensen JR, Hodgson ME (2008) Object-based land cover classification using high-posting-density LiDAR data. *GIScience Remote Sens* 45(2):209–228
49. Goncalves G, Seco L, Reyes F, Miranda D (2008) Land cover classification of rural areas using LiDAR data: a comparative study in the context of fire risk. In: 8th international conference on LiDAR applications in forest assessment and inventory SilviLaser 2008, pp 427–436
50. Kim S, McGaughey RJ, Andersen HE, Schreuder G (2009) Tree species differentiation using intensity data derived from leaf-on and leaf-off airborne laser scanner data. *Remote Sens Environ* 113(8):1575–1586
51. Habib A, Kersting A, Shaker A, Yan WY (2011) Geometric calibration and radiometric correction of LiDAR data and their impact on the quality of derived products. *Sensors* 11 (9):9069–9097
52. Yan WY, Shaker A, Habib A, Kersting AP (2012) Improving classification accuracy of airborne LiDAR intensity data by geometric calibration and radiometric correction. *ISPRS Jof Photogramm and Remote Sens* 67:35–44
53. Buján S, González-Ferreiro E, Reyes-Bueno F, Barreiro-Fernández L, Crecente R, Miranda D (2012) Land use classification from LiDAR data and ortho-images in a rural area. *Photogram Rec* 27:401–422
54. Shaker A, El-Ashmawy N (2012) Land cover information extraction using LiDAR data. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXIX-B7:167–172*
55. Markets and Markets. <http://www.marketsandmarkets.com/Market-Reports/lidar-market-1261.html>. Accessed 31 July 2016
56. Hitz B, Ewin JJ, Hecht J (1998) Introduction to laser technology, 3rd edn. IEEE Press, New York
57. Weber MJ (1999) Handbook of laser wavelength. CRC Press, Boca Raton
58. Sifvast WT (2004) Laser fundamentals, 2nd edn. Cambridge University Press, Cambridge
59. Jelalian AV (1992) Laser radar systems. ArtechHouse, Boston
60. Rees WG (2001) Physical principles of remote sensing, 2nd edn. Cambridge University Press, Cambridge
61. Jutzi B, Still U (2006) Range determination with waveform recording laser systems using a Wiener Filter. *ISPRS J Photogramm Remote Sens* 61(2):95–107
62. Kim II, McArthur B, Korevaar E (2001) Comparison of laser beam propagation at 785 nm and 1550 nm in fog and haze for optical wireless communications. Proc SPIE 4214, Optical Wireless Communications III, 26. doi:[10.1117/12.417512](https://doi.org/10.1117/12.417512)

63. Lemmens M (2009) Airborne LiDAR sensors. *GIM Int* 23(2):16–19
64. Shan J, Toth CK (2009) Topographic laser ranging and scanning. CRC Press, Boca Raton
65. Bang KI (2010) Alternative methodologies for LiDAR system calibration. PhD dissertation, Calgary, Alberta
66. Mikhail EM, Ackerman F (1976) Observations and least squares. University Press of America, Lanham
67. McGlone JC, Mikhail EM, Bethel J, Mullen R (2004) Manual of photogrammetry, 5th edn. American Society for Photogrammetry and Remote Sensing, Bethesda
68. VDI. The Association of German Engineers. http://www.vdi.eu/guidelines/vdivde_2634_blaett_2-optische_3_d_messsysteme_bildgebende_systeme_mit_flaechenhafter_antastung/. Accessed 7 Aug 2016
69. VDI. The Association of German Engineers. http://www.vdi.eu/guidelines/vdivde_2634_blaett_3-optische_3_d_messsysteme_bildgebende_systeme_mit_flaechenhafter_antastung/. Accessed 7 Aug 2016
70. Cuartero A, Armesto J, Rodríguez PG, Arias P (2010) Error analysis of terrestrial laser scanning data by means of spherical statistics and 3D graphs. *Sensors* 10(11):10128–10145
71. Habib A, Bang KI, Kersting AP, Chow J (2010) Alternative Methodologies for LiDAR System Calibration. *Remote Sens* 2(3):874–907
72. Teledyne Optech. Airborne Survey. <http://www.teledyneoptech.com/index.php/products/airborne-survey/>. Accessed 3 Aug 2016
73. Teledyne Optech. <http://www.teledyneoptech.com/index.php/products/airborne-survey/camera-systems/>. Accessed 3 Aug 2016
74. Wang H, Glennie C (2015) Fusion of waveform LiDAR data and hyperspectral imagery for land cover classification. *ISPRS J Photogram Remote Sens* 108:1–11
75. Favorskaya M, Tkacheva A, Danilin IM, Medvedev EM (2015) Fusion of airborne LiDAR and digital photography data for tree crowns segmentation and measurement. In: Damiani E, Howlett RJ, Jain LC, Gallo L, De Pietro G (eds) Intelligent interactive multimedia systems and services. Springer International Publishing, Switzerland
76. Applanix. POS AV. <http://www.applanix.com/products/airborne/posav.html>. Accessed 3 Aug 2016
77. TopoSys Harrier Model Range. <http://www.gim-international.com/content/news/toposys-harrier-model-range>. Accessed 3 Aug 2016
78. RIEGL USA. <http://products.rieglusa.com/item/airborne-scanners/lms-q680i-long-range-laser-scanner/item-1001/>. Accessed 3 Aug 2016
79. Leica. LiDAR Sensors. <http://leica-geosystems.com/products/airborne-systems/lidar-sensors>. Accessed 3 Aug 2016
80. Ax M, Thamke S, Kuhnert L, Kuhnert KD (2013) UAV based laser measurement for vegetation control at high-voltage transmission lines. *Adv Mater Res* 614–615:1147–1152
81. Geerling GW (2008) Changing rivers: analysing fluvial landscape dynamics using remote sensing. PhD thesis, Centre for Sustainable Management of Resources, Radboud University of Nijmegen, The Netherlands
82. Crocker RI, Maslanik JA, Adler JJ, Palo SE, Herzfeld UC, Emery WJ (2012) A sensor package for ice surface observations using small unmanned aircraft systems. *IEEE Trans Geosci Remote Sens* 50(4):1033–1047
83. Merino L, Caballero F, Martínez-de-Dios JR, Maza I, Ollero A (2012) An unmanned aircraft system for automatic forest fire monitoring and measurement. *J Intell Robot Syst* 65(1): 533–548
84. Martínez-De Dios JR, Ollero A (2006) Automatic detection of windows thermal heat losses in buildings using UAVs. In: 2006 world automation congress, pp 1–6
85. Nagai M, Chen T, Shibasaki R, Kumagai H, Ahmed A (2009) UAV-borne 3-D mapping system by multisensor integration. *IEEE Trans Geosci Remote Sens* 47(3):701–708
86. Lin Y, Hyypa J, Jaakkola A (2011) Mini-UAV-borne LiDAR for fine-scale mapping. *IEEE Geosci Remote Sens Lett* 8(3):426–430

87. Lin Y, Hyypä J, Rosnell T, Jaakkola A, Honkavaara E (2013) Development of a UAV–MMS-collaborative aerial-to-ground remote sensing system—a preparatory field validation. *IEEE J Sel Top Appl Earth Obs Remote Sens* 6(4):1893–1898
88. NORUT. <http://norut.no/>. Accessed 7 Aug 2016
89. Pfeifer N, Glira P, Briese C (2012) Direct georeferencing with onboard navigation components of light-weight UAV platforms. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXIX, Part B7*:487–492
90. Pounds P, Mahony R, Corke P (2010) Modelling and control of a large quadrotor robot. *Control Eng Pract* 18(7):691–699
91. Cesetti A, Frontoni E, Mancini A, Zingaretti P, Longhi S (2009) Vision-based autonomous navigation and landing of an unmanned aerial vehicle using natural landmarks. In: 17th mediterranean conference on control and automation MED'2009, pp 910–915
92. Mittal S, Deb K (2007) Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In: IEEE congress on evolutionary computation, pp 3195–3202
93. Lisein J, Linchaint J, Lejeune P, Bouché P, Vermeulen C (2013) Aerial surveys using an unmanned aerial system (UAS): comparison of different methods for estimating the surface area of sampling strips. *J Trop Conserv Sci* 6(4):506–520
94. Whitehead K, Hugenholtz CH (2014) Remote sensing of the environment with small unmanned aircraft systems (UASs), part 1: a review of progress and challenges. *J Unmanned Veh Syst* 2(3):69–85
95. RIEGL USA. Unmanned scanners. <http://products.rieglusa.com/category/all-categories-unmanned-scanners>. Accessed 3 Aug 2016
96. Gu K, Leet J, Alon A, Singh M (2012) E/ME 103 final report. <http://pickar.caltech.edu/e103/papers/Micro%20UAVs.pdf>
97. Calise AJ, Preston D (2008) Swarming/flocking and collision avoidance for mass airdrop of autonomous guided parafoils. *J Guidance, Control Dyn* 31(4):1123–1132
98. Kankare V, Holopainen M, Vastaranta M, Puttonen E, Yu X, Hyppä J, Vaaja M, Hyppä H, Alho P (2013) Individual tree biomass estimation using terrestrial laser scanning. *ISPRS J Photogramm Remote Sens* 75:64–75
99. Vinci A, Brigante R, Todisco F, Mannocchi F, Radicioni F (2015) Measuring rill erosion by laser scanning. *Catena* 124:97–108
100. Maas H-G, Bienert A, Scheller S, Keane E (2008) Automatic forest inventory parameter determination from terrestrial laser scanner data. *Int J Remote Sens* 29(5):1579–1593
101. Côté J-F, Fournier RA, Egli R (2011) An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR. *Environ Model Softw* 26(6):761–777
102. RIEGL USA. Terrestrial Scanners. <http://products.rieglusa.com/category/terrestrial-scanners>. Accessed 3 Aug 2016
103. Kaasalainen S, Jaakkola A, Kaasalainen M, Krooks A, Kukko A (2011) Analysis of incidence angle and distance effects on terrestrial laser scanner intensity: search for correction methods. *Remote Sens* 3(10):2207–2221
104. Mobile Laser Scanning Applications. <http://www.geosignum.nl/index.php/product-services/geosolutions/mobile-laser-scanning>. Accessed 3 Aug 2016
105. Haala N, Peter M, Kremer J, Hunter G (2008) Mobile LiDAR mapping for 3D point cloud collection in urban areas—a performance test. *Int Arch Photogramm Remote Sens Spat Inform Sci* 37:1119–1127
106. RIEGL USA. Mobile scanners. <http://products.rieglusa.com/category/mobile-scanners>. Accessed 3 Aug 2016
107. Trimble MX8. <http://www.trimble.com/Imaging/Trimble-MX8.aspx?tab=Overview>. Accessed 3 Aug 2016
108. James MR, Quinton JN (2014) Ultra-rapid topographic surveying for complex environments: the hand-held mobile laser scanner (HMLS). *Earth Surf Process Land* 39(1):138–142
109. Ryding J, Williams E, Smith MJ, Eichhorn MP (2015) Assessing handheld mobile laser scanners for forest surveys. *Remote Sens* 7(1):1095–1111

110. Bassuk N, Grabsky J, Mucciardi A, Raffel G (2011) Ground-penetrating radar accurately locates tree roots in two soil media under pavement. *Arboric Urban Forest* 37(4):160–166
111. Daniels DJ (2004) Surface-penetrating radar, 2nd edn. The Institute of Electrical Engineers, UK
112. Xie Y, Sha Z, Yu M (2008) Remote sensing imagery in vegetation mapping: a review. *J Plant Ecol* 1(1):9–23
113. Andersen H-E, Strunk J, Temesgen H (2011) Using airborne light detection and ranging as a sampling tool for estimating forest biomass resources in the upper Tanana Valley of interior Alaska. *West J Appl Forest* 26(4):157–164
114. Staahl G, Holm S, Gregoire T, Gobakken T, Næsset E, Nelson R (2011) Model-based inference for biomass estimation in a LiDAR sample survey in Hedmark County, Norway. *Can J Forest Res* 41(1):96–107
115. Reitberger J, Schnör C, Krzystek P, Stilla U (2009) 3D segmentation of single trees exploiting full waveform LiDAR data. *ISPRS J Photogramm Remote Sens* 64(6):561–574
116. Holopainen M, Mäkinen A, Rasimäki J, Hyppä J, Hyppä H, Kaartinen H, Viitala R, Vastaranta M, Kangas A (2010) Effect of tree-level airborne laser-scanning measurement accuracy on the timing and expected value of harvest decisions. *Eur J Forest Res* 129(5):899–907
117. Kaartinen H, Hyppä J, Yu X, Vastaranta M, Hyppä H, Kukko A, Holopainen M, Heipke C, Hirschmugl M, Morsdorf F, Næsset E, Pitkänen J, Popescu S, Solberg S, Wolf BM, Wu J-C (2012) An international comparison of individual tree detection and extraction using airborne laser scanning. *Remote Sens* 4(4):950–974
118. Yao W, Krzystek P, Heurich M (2012) Tree species classification and estimation of stem volume and DBH based on single tree extraction by exploiting airborne full-waveform LiDAR data. *Remote Sens Environ* 123:368–380
119. Vastaranta M, Holopainen M, Yu X, Hyppä J, Mäkinen A, Rasimäki J, Melkas T, Kaartinen H, Hyppä H (2011) Effects of individual tree detection error sources on forest management planning calculations. *Remote Sens* 3(8):1614–1626
120. Lin Y, Hyppä J, Kukko A, Jaakkola A, Kaartinen H (2012) Tree height growth measurement with single-scan airborne, static terrestrial and mobile laser scanning. *Sensors* 12(9):12798–12813
121. Jaakkola A, Hyppä J, Kukko A, Yu X, Kaartinen H, Lehtomäki M, Lin Y (2010) A low-cost multi-sensoral mobile mapping system and its feasibility for tree measurements. *ISPRS J Photogramm Remote Sens* 65(6):514–522
122. Wallace L, Watson C, Lucieer A (2014) Detecting pruning of individual stems using airborne laser scanning data captured from an unmanned aerial vehicle. *Int J Appl Earth Obs Geoinf* 30:76–85
123. Wallace L, Lucieer A, Watson CS (2014) Evaluating tree detection and segmentation routines on very high resolution UAV LiDAR data. *IEEE Trans Geosci Remote Sens* 52(12):7619–7628

Chapter 3

Software Tools for Terrain and Forest Modelling

Abstract Huge amount of data obtained from airborne, terrestrial, and mobile laser scanners require the suitable software tools for storing and analyzing the data in order to gain the required results. The development of software tools for the LiDAR data processing that began since 1980s is not trivial due to the high requirements for hardware, effective algorithms, real-time rendering of realistic scenes, containing millions of primary elements. Also, a forest scene simulation based on a procedural modelling evolves successfully as a demanded direction in design of virtual worlds, computer games, and movies. Among the great variety of software tools, some packages and platforms, such as Terrasolid, ArcGIS, Xfog, SpeedTree, forge ahead. All packages are developed persistently, with appearance new versions periodically. However, a choice of a software tool's version depends strongly from a solving task and the resources of organization. Inside the commercial products, free software with the limited functionality is supported as a way to understand their necessity in the future projects of the end-users.

Keywords Remote sensing · Lidar data · Software tool · Terrain modelling · Forest modelling · Procedural modelling

3.1 Introduction

The LiDAR technologies that are applied in the Airborne Laser Scanning (ALS) and the Terrestrial Laser Scanning (TLS) can provide the point clouds with a point density of up to thousands of points per square meter. For example, an Optechs Lynx Mobile Mapper captures a total of 144 million points of five blocks in 20 min [1]. In addition to the huge quantity of point clouds, the multiple high-spatial-resolution cameras provide a significantly large quantity of image data. For example, a Trimble MX8 system, integrating two RIEGL VQ-250 laser scanners and four CCD cameras, can collect a total of 35 GB in 20 min [2]. In most cases, the existing LiDAR processing algorithms and software tools ought to divide the substantial amounts of the LiDAR points into a number of data blocks [3] or

rasterize the LiDAR data for a series of post-processing procedures [4]. Because of a huge amount of information, the software tools require an open, shared, and inter operated environment, distributing a data processing among multiple shared-servers. The cloud computing is a promising choice for processing of massive remote sensing data.

Section 3.2 provides a short survey of software tools for terrain modelling, while a survey of software tools for vegetation modelling is represented in Sect. 3.3. Section 3.4 gives the concluding remarks.

3.2 Survey of Software Tools for Terrain Modelling

The representation of software tools as the software tools for terrain modelling and vegetation modelling is conditionally. Several popular packages for terrain modelling, including the Terrasolid, the ArcGIS, the Bionatics, the Terragen, the 3DNature, the GRASS, and the Object Raku Technology, are discussed in Sects. 3.2.1–3.2.7, respectively.

3.2.1 *Software Tool “Terrasolid”*

The Terrasolid [5] develops the software products that are used world-wide for processing of the LiDAR and image data obtained from airborne and terrestrial laser scanning systems. Most of the Terrasolid applications are built on top of Bentley software. The compatibility of the Terrasolid software versions with Bentley products is listed in [6]. The Terrasolid offers full and light versions of its software products mentioned in Table 3.1. These software products can be combined in packages tailored for specific working requirements.

The TerraScan is the main application in the Terrasolid software family for managing and processing of the LiDAR point clouds. It offers import and project structuring tools for handling the large amount of points of a laser scanning campaign as well as the corresponding trajectory information. Various classification routines enable the automatic filtering of the point cloud. The results from the automatic classification can be refined using half-automatic and manual classification tools in combination with versatile 3D point cloud visualization options.

Most of the automatic classification routines can be combined in macros for a batch processing. In combination with the TerraSlave, a macros can be executed outside the TerraScan and the MicroStation, even on other workstations over a Local Area Network (LAN). The TerraScan supports several import and export formats, including the Log ASCII Standard (LAS) format, the TerraScan Binary, and the TerraScan Fast Binary formats as well as ASCII formats that can be defined according to the end-users’ needs.

Table 3.1 Full and light version of the Terrasolid

Component	Assignment	Functions
The full versions of the Terrasolid applications	Processing of the LiDAR data, images and survey data	Process raw laser points, images, trajectories and survey data with a software package of the TerraScan combined with additional Terrasolid applications, such as the TerraMatch, the TerraModeler, the TerraPhoto and the TerraSurvey Speed-up a work by distributed and advanced batch, processing with additional seats of the TerraSlave
The light versions of the Terrasolid applications	Using the LiDAR data and/or images for manual work or under field conditions	Visualization of classified LiDAR data and/or images, and perform time-consuming manual processing steps with the TerraScan Lite and the TerraPhoto Lite Work under field conditions with the TerraModeler Lite and the TerraSurvey Lite
The Terrasolid visualization software	Mono/stereo visualization of very large point clouds	Visualization of very large point clouds in mono and stereo mode with the TerraStereo that includes sophisticated methods for point cloud display and makes use of high-end graphics hardware in order to speed up dynamic and stereo visualizations. The TerraStereo runs as stand-alone application
The Terrasolid infrastructure applications	Designing and updating the infrastructure objects (the applications do not rely on laser or image data but are smoothly integrated with the MicroStation and the other Terrasolid software products)	Design roads with the TerraStreet or its light version the TerraStreet Lite, improve its capabilities by combining it with the TerraModeler Design and maintain pipeline networks with the TerraPipe, the TerraPipeNet, or the TerraGas Use the TerraHeat for designing pipe networks of a district heating system and the TerraBore for mapping soil layers (in Finnish language only)

The point cloud management, processing, and visualization are only one part of the TerraScan capabilities. In addition, the software provides tools for creating 3D vector data based on the laser points. There is also an opportunity to produce 3D vector models of buildings (up to the LOD2) automatically over large areas. The

toolsets for checking and modifying the building models manually enable the creation of more accurate and higher-quality building models. For the application field of power line processing, the TerraScan contains tools for the automatic vectorization of power line wires, manual placement of tower models as well as labeling, reporting, and danger object analysis tools.

The TerraScan Lite is a lighter version of the TerraScan for viewing laser points, which are already automatically classified and for performing the processing steps that require the manual efforts. It replaced the previous the TerraScan Viewer represented in 2012, when more functionality was added to the software. The typical possibilities of the TerraScan Lite are the following: a visualization of laser data, a manual point classification, an automatic classification of model or contour keypoints, and a model extraction as well as manual check and modification of building vector models that have been created automatically with the TerraScan building vectorization tool. The functionality of the TerraScan Lite includes:

- Loading and saving laser data in various formats.
- Loading trajectories.
- Setting up projects.
- Manual classification tools.
- Export options such as export lattice models or raster files.
- Tools for drawing 3D vector data based on laser points.
- Various commands for processing loaded points like transform points, cut overlap or color extraction from images in the TerraPhoto (Lite).
- Adjusting laser points to the geoid models. (That the Earth does not have a geometrically perfect shape is well established, and the geoid is used to describe the unique and irregular shape of the Earth. The traditional, orthometric height is the height above an imaginary surface called the geoid that is determined by the Earth's gravity and approximated by mean sea level.)
- Convert the geoid models.
- Set polygon elevations.
- Read building vector models from text files generated with the TerraScan building a vectorization macro function.

With the TerraModeler one can create, edit, and utilize surface models. The TerraModeler creates the surface models using the Triangular Irregular Network (TIN) from various sources, such as the LiDAR points stored in binary files or loaded in the TerraScan, XYZ ASCII files, and graphical design elements. The software offers versatile visualization options, including the colored shaded surfaces, contour lines, grids, colored triangle nets, elevation texts, slope directions, and textured surfaces (in combination with the TerraPhoto). Additional functionality includes the production of contour lines and lattice models in batch processing, modification of the TIN, creation of profiles, calculation of volumes, calculation of elevation or volume differences between two surface models, several labeling options as well as other tools for design purposes. Completed with various export options, the TerraModeler is a versatile tool for many types of design and modeling tasks.

The lighter version of the TerraModeler, the TerraModeler Lite, has a reduced functionality and is optimized for designers, surveyors, and other users, whose primary aim is to create the TINs, display models, and calculate volumes from a small amount of survey data. It replaced the former TerraModeler Field in 2012. The functionality of the TerraModeler Lite includes:

- Creation of the surface models from several sources, e.g., the laser points, the breakline elements (A breakline is a line in the TIN or terrain dataset that represents a distinct interruption in the slope of a surface. The triangles in the TIN or terrain dataset may not cross a breakline. The z values along a breakline can be constant or variable.), xyz text files.
- Display of surfaces using different display methods, such as contour lines, grids, triangles, elevation texts, or slope directions.
- Modification of surfaces by excluding long triangles, thinning, modifying elevations, inserting breaklines, inserting or removing elevation points, manipulating elevations inside specified areas.
- Creation of new surfaces by copying, subtracting, or merging existing surface models.
- Visual analysis of surface models by displaying elevations, elevation differences between surfaces, slope gradients, and directions.
- Drawing vector elements based on the surface elevations.
- Drawing labels for slopes and areas.
- Manipulation of vector elements by changing their elevation, thinning or inserting vertices, copy elements, place elements relative to an alignment element.
- Creation of rule files for a breakline handling in surface triangulation.
- Creation of setting files for contour lines and peaks/pits in preparation of batch contour line or lattice file production in the TerraModeler.
- Display of profiles based on surfaces, draw profiles in the DGN-file.
- Calculation of quantities and intersections between two surfaces.

The TerraPhoto is specifically developed for processing images captured together with laser data during a survey mission. This software enables the production of rectified images and ortho mosaics based on ground model that has been extracted from the laser data. The positioning of the source images can be refined using the tie points for image-to-image adjustment, while the ground control point can be involved for improving the absolute accuracy of the image block. With additional functionality, such as color adjustment options, the creation of selection shapes for several object types or areas (e.g., buildings, water), the inclusion of vector models for true-ortho photo production, the TerraPhoto lets create the ortho photos of good positional and color-coordinated quality. Besides starting a camera calibration from scratch, the TerraPhoto is able to convert the calibration files from several mobile and airborne systems into its own camera file format. It reads numerous image formats, such as ECW, GeoTIFF, TIF, BMP, CIT, COT, RLE, PIC, PCX, GIF, JPG2000, and PMG.

The TerraPhoto offers many tools for visualizing laser and image data together, partly in combination with tools from the TerraScan and the TerraModeler. This includes the opportunity of draping ortho photos on a ground model or on building roofs, the creation of wall textures from horizontally-looking cameras of mobile or airborne systems as well as the display of rendered views. With the TerraPhoto one can also create the fly-through animations from laser and image data in an easy and intuitive way. Finally, the extraction of color values from images or ortho photos to laser points can be performed with the TerraPhoto and the TerraScan.

The TerraPhoto Lite is a light version of the TerraPhoto that was designed for doing the extensive manual work for preparing the production of high-quality ortho photo and other visualization tasks. It replaced the previous TerraPhoto Viewer in 2012, when more functionality was added to the software. Typical uses for the TerraPhoto Lite are the display of raster data as background information, e.g., for mapping tasks, rendering of 3D scenes with large image volumes, production of fly-through animations as well as improving the color balance between images and creating seamlines for the TerraPhoto ortho photo production. The functionality of the TerraPhoto Lite includes:

- Create and edit the mission files.
- Import Lynx survey and Pictometry survey data.
- Create and modify camera definitions.
- Convert camera system calibrations into the TerraPhoto camera definition.
- Manage trajectories.
- Load and edit image lists, edit image information.
- Define color corrections.
- Define color points.
- Create and edit selection shapes and seamlines.
- Draw image footprints, draw projections into the design file.
- Create camera views.
- Manage raster references.
- Create rendered views and fly-through animations.
- Fix normals of 3D building models.

The TerraMatch is a sophisticated tool for improving the accuracy and quality of the raw laser point cloud. It compares laser data from overlapping flight or drive paths and calculates correction values for the misalignment angles as well as the xyz location errors. The comparison and correction value calculation can be either based on the surface matching or different types of tie lines. A tie line matching comprises the points or lines on horizontal, vertical or sloped surfaces that can be used for matching the flight/drive paths to each other but also known point or line locations that enable the adjustment of the laser point cloud to control the measurements. Especially in a terrestrial and mobile laser scanning, the mismatch between overlapping drive paths usually changes over time during a surveying campaign, depending on the quality of the GPS signal on different locations. This requires a

fluctuating correction solution that can be achieved with the control measurements and the TerraMatch tie lines.

The TerraMatch is also used for misalignment angle calibration that should be checked and possibly improved at the beginning of data processing. For this purpose, data are collected on a specific calibration side that offers a point cloud dedicated for the calibration task to the software. If the system includes several laser scanners, then the calibration involves a scanner-to-scanner matching. The TerraMatch ought to be accompanied by the TerraScan.

The TerraSurvey handles the data from a field surveying with the total stations and GPS. It reads-in data from text files, recognizing automatically a number of survey data formats, and creates a survey drawing as 3D design file. For processing the survey data in the TerraSurvey, each survey point is assigned a feature code that identifies the surveyed object, for example, a tree, a manhole cover, an elevation point or points along a road center line. The graphical display for each object is defined by one or more drawing rules. Several options for displaying the data in list or table views help the end-user to do modifications or fix errors in the data. The TerraSurvey is a useful addition to the TerraModeler for creating the TINs from the survey elements. An optional sub-code for each survey element is used by the TerraModeler for recognizing automatically the modelling information for this element and determines, whether it is included into the terrain model as a random point, breakline or not at all.

If the TerraSurvey is used in a combination with the TerraPhoto and the TerraScan, then the absolute accuracy of image positions in an image list is improved by use of the ground control points from the TerraSurvey. Also, one can assign the feature codes to vectorized objects, such as power line wires or road break lines that have been vectorized or digitized in the TerraScan. All together the TerraSurvey is a drawing tool for feature coded survey data and, in combination with the TerraScan and the TerraModeler, it enables the use of such survey data for the terrain modelling and other mapping tasks. The TerraSurvey Lite is a light mapping application with the restricted functions.

The TerraStereo is a stand-alone application for visualizing very large point clouds. It uses the high-performance graphics boards for rendering huge amounts of points fast and in high quality. The software enables:

- Visualize up to 50 billion points, which are organized in the TerraScan project.
- Perform visual analysis and quality check in both, small and large scale displays.
- Navigate through the point cloud freely or based on the TerraScan trajectory files.
- Create animations.
- View laser data in stereo mode and create stereo screen captures.
- Easily change between different display channels and point rendering modes, such as elevation, intensity and class coloring, color by RGB values, shaded surface display, point density coloring, the TerraZ rendering method for masking foreground objects, different quality levels for point rendering.

- Combine different visualization channels in order to improve the perceptibility of objects in the point cloud.
- Switch point classes on/off.
- Apply an elevation exaggeration to the point cloud for specific analysis tasks.
- Measure distances within the point cloud.
- Digitize the line vector elements and export them to the TerraSurveyor.

The TerraBore is a software for the registering soil layer data into a database, editing the data, and visualizing the data in a map or cross-section view. With the TerraModeler, this data can be turned into a 3D model of soil layers.

3.2.2 *Cloud Platform “ArcGIS”*

The ArcGIS platform provides the high quality solutions in many application, such as spatial analysis, mapping and visualization, apps, content, imagery and remote sensing, real-time GIS, community engagement, 3D, data management, the Computer-Aided Drawing (CAD), among others [7, 8]. Near 90 ArcGIS products are available in the ArcGIS Online, the ArcGIS for Desktop, and the ArcGIS for Server.

The ArcGIS Online is a complete, cloud-based mapping platform. It includes the following possibilities:

- Apps for everyone. The ArcGIS apps get work done with the greatest ease and maximum benefit, with little to no configuration required, with or without an Internet connection.
- Ready-to-use maps. The ArcGIS includes a Living Atlas of the World, comprised of authoritative maps and data on thousands of topics.
- Visualization. With the ArcGIS, one can visualize large amounts of data and convey information.
- Analytics. Spatial analytics allows to identify and quantify the implications, consequences, and impact of decisions.
- Administration. The ArcGIS includes everything to control and manage people and content, from assigning custom roles and privileges to managing licenses and content to viewing the system’s health status.
- The ArcGIS solutions involves the templates to configure the ArcGIS applications, deploy web apps, and implement best practices. It is available for a range of industries, such as local and state government, emergency management, utilities, telecommunications, military, and intelligence.
- Secure and trusted. The ArcGIS Online is designed as a secure system with the controlled access.
- Tools for developers. These tools help to develop the projects in the API, SDK, or app builder and deploy across any device.

The ArcGIS for Desktop includes two main applications for mapping, editing, and analysis: the ArcMap and the ArcGIS Pro. The ArcMap is the leading, traditional GIS authoring, editing, and geo-processing application that allows to create the high-quality maps, edit, and manage the spatial data, and perform the full spectrum of analyses needed to turn the raw data into the valuable information. The ArcGIS Pro is a 64-bit application that has a contextual interface and project-centric workflow. The key features are the following:

- Conduct spatial analysis. Many tools for performing spatial analysis are included in ArcGIS for Desktop. These tools allow to turn data into the actionable information and to automate many of the GIS tasks.
- Manage data more efficiently. With support for more than 70 data formats, one can easily integrate all types of data for visualization and analysis. An extensive set of geographic, tabular, and metadata management, creation, and organization tools are available.
- Explore a World of content. The ArcGIS Online is a part of the ArcGIS for Desktop license. The ArcGIS Online browses the world's most extensive online geographic resource. The content may be combined in any way and displayed on a map.
- Automate advanced workflows. Advanced editing and coordinate geometry tools simplify the data design, input, and cleanup.
- Easily create maps. The high-quality maps can be produced without the hassles associated with complex design software. With the ArcGIS for Desktop, one can take advantage of a large library of symbols and simple wizards and pre-defined map templates.
- Start geocoding. There is a wide range of applications, for which geocoding can be used. With geocoded addresses, one can display the address locations and see patterns within the information.
- Access advanced imagery. There are many ways to work with image data (raster data) in the ArcGIS for Desktop. One can use it as a background (base map) to analyze other data layers, apply different types of specifications to the image dataset, or use it as part of the analysis.
- Provide the clients what they need. The sharing and collaboration tools of the ArcGIS for Desktop help to publish the projects via simple, focused apps.

The ArcGIS for Server includes the following capabilities:

- Reach of GIS capabilities using client apps. The ArcGIS for Server contains a range of client apps that do everything from providing a dashboard summarizing critical business information to bringing maps into mainstream business intelligence software and helping field crews collect data. Client apps include operations dashboard for the ArcGIS, the ArcGIS Maps for Office, the Esri Maps for IBM Cognos, the Esri Maps for SharePoint, the Collector for ArcGIS, and the Explorer for ArcGIS.

- Centrally management of GIS assets. The ArcGIS for Server includes an integrated map viewer, map-making tools, and application templates to quickly and easily create and deploy web applications.
- Power enterprise GIS with web services. The ArcGIS for Server and its extensions can expose the complete range of the ArcGIS functionality, e.g., management of web services for mapping, geocoding, the GIS analysis, web editing, network analysis, database access, and geodata management.
- Support for real-time GIS. The ArcGIS for Server (with the ArcGIS GeoEvent Extension for Server) enables to connect to any sensor, such as GPS, mobile, and social media. One can process and filter real-time data, making it possible to monitor assets, update maps and data, and send notifications via e-mail, text, or instant messages.
- Share imagery. The ArcGIS for Server enables to publish and share imagery data. The ArcGIS Image Extension for Server enhances the capabilities of the ArcGIS for Server to manage, produce, and exploit large numbers of imagery and rasters.
- Access spatial data in databases. The ArcGIS for Server works with the spatial data stored in relational database management systems, such as the IBM DB2 and the IBM Informix Dynamic Server, the Microsoft SQL Server, the Microsoft SQL Server Express, and the Microsoft SQL Azure, the Netezza, the Oracle, and the PostgreSQL. Also, the ArcGIS for Server includes the geo-database, the common data storage and management framework for the ArcGIS.
- Integration with the existing IT infrastructure. The ArcGIS for Server fits the IT infrastructure and security requirements. It supports all deployment types: on-premises, cloud, virtual, or hybrid. One can add the ArcGIS Online to deployment or use the ArcGIS for Server independently. The ArcGIS for Server supports the Windows Active Directory, the Lightweight Directory Access Protocol (LDAP), and the Public Key Infrastructure (PKI) security. Also, it scales to increase capacity or accommodate demand.

The ArcGIS Spatial Analyst is one of the main components of the ArcGIS. The ArcGIS Spatial Analyst has the following capabilities:

- Create query maps and analyze cell-based raster data.
- Perform integrated raster/vector analysis.
- Derive new information from existing data.
- Query information across multiple data layers.
- Integrate cell-based raster data with traditional vector data sources.

The ArcGIS Spatial Analyst is integrated into the ArcGIS interface using all benefits of the ArcGIS Geostatistical Analyst and the ArcGIS 3D Analyst. The basic objects for terrain modelling are presented in Table 3.2. They are classified as surface-based, volume-based, and hybrid objects (Boundary REPresentation (B-Rep), Constructive Solid Geometry (CSG), TEtrahedral Network (TEN), Soft Voxel (SV), Triangulated Irregular Network (TIN)).

Table 3.2 Basic objects for terrain modelling

Approach	Object	Symbol	Approach	Object	Symbol
Surface-based	Grid		Volume-based	3D array	
	Shape			Octree	
	Facet			CSG	
	B-Rep			TEN	
				GIS SV	
				Tri-Prism	
Hybrid	TIN Octree			Pyramid	
	TIN CSG			3D Voronoi	
				Geocellular	
				Voxel	

A surface representation in its simplest form is done by storing the x , y , and z values (points' representation). The contours or isolines are used to define a common characteristic along a line, joining the locations of equal value to each other. If a contour line presents the heights, then the connected points of equal elevation show usually a mean of sea level on a map. The TIN is a vector data structure used to store and display surface models. The TIN partitions a geographic space using a set of irregularly spaced data points, each of which has the x , y , and z values. These points are connected by the edges, forming the contiguous, non-overlapping triangles and creating a continuous surface that represents a terrain. The grid is a spatial data structure that defines space as an array of cells of equal size and is arranged in rows and columns. In the case of a surface representation, each cell contains an attribute value that represents a change in z value. The examples of surface representation are depicted in Fig. 3.1. A flow-chart of data processing in the ArcGIS Spatial Analyst is depicted in Fig. 3.2.

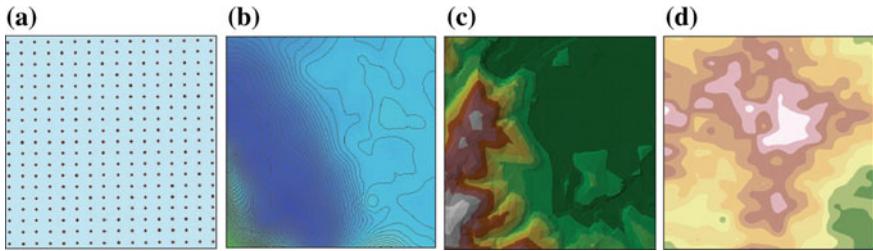


Fig. 3.1 Surface representation in ArcGIS spatial analyst: **a** points, **b** contours, **c** the TINs, **d** grids

3.2.3 Software Tool “Bionatics”

The Bionatics [9] is a software tool to model, visualize, and manage the urban and rural territories in 3D space. The Bionatics based on the procedural 3D plants modeling technology delivers a complete suite of professional solutions for the needs of architecture, landscape, 3D animation, video game, and virtual reality. The Bionatics provides highly innovative and efficient software technologies to simulate, visualize, and manage large territories in 3D space. This software tool involves the innovative technologies, such as the LandSIM3D, the Blueberry3D, the natFX, the EASYnat, and the REALnat.

The LandSIM3D allows an automatic 3D modelling of the territory using data obtained from the GIS, digital terrain elevation, and aerial images. It provides an easy way to model rapidly cities and landscape in 3D from 2D maps and vector data. The LandSIM3D is applied for urban planners and landscape architects, civil engineering (railway, road, and energy), regions, natural parks, and preserved areas, forestry management, mines and quarries. The LandSIM3D is able to build the realistic real-time 3D digital models providing the following possibilities (version 1.5 of the LandSIM3D):

- Ergonomic workspace mixing 2D and 3D views available through an interactive mode. Interactive edition and modification of vector data.
- Symbolic drawing capabilities offered onto the terrain to better explain a site or a project.
- 3D plant modelling workshop allowing the alignment, pruning, and fine tuning of the plants.
- Road and river simulation with a width parameter along the path.
- Measurement tool in 2D/3D views.
- Some real-time special effects, such as natural fog with fine tuning options.

Version 2.0 of the LandSIM3D facilitates the real-time loading and display of large territories, storing the results in common 3D formats like 3DS or the Google SketchUp. The LandSIM3D v2.0 streams the 3D models on the fly and displays them in 3D view near by the camera in addition to the full procedural display of the rest of the territory. Such technology allows a very realistic visualization of the

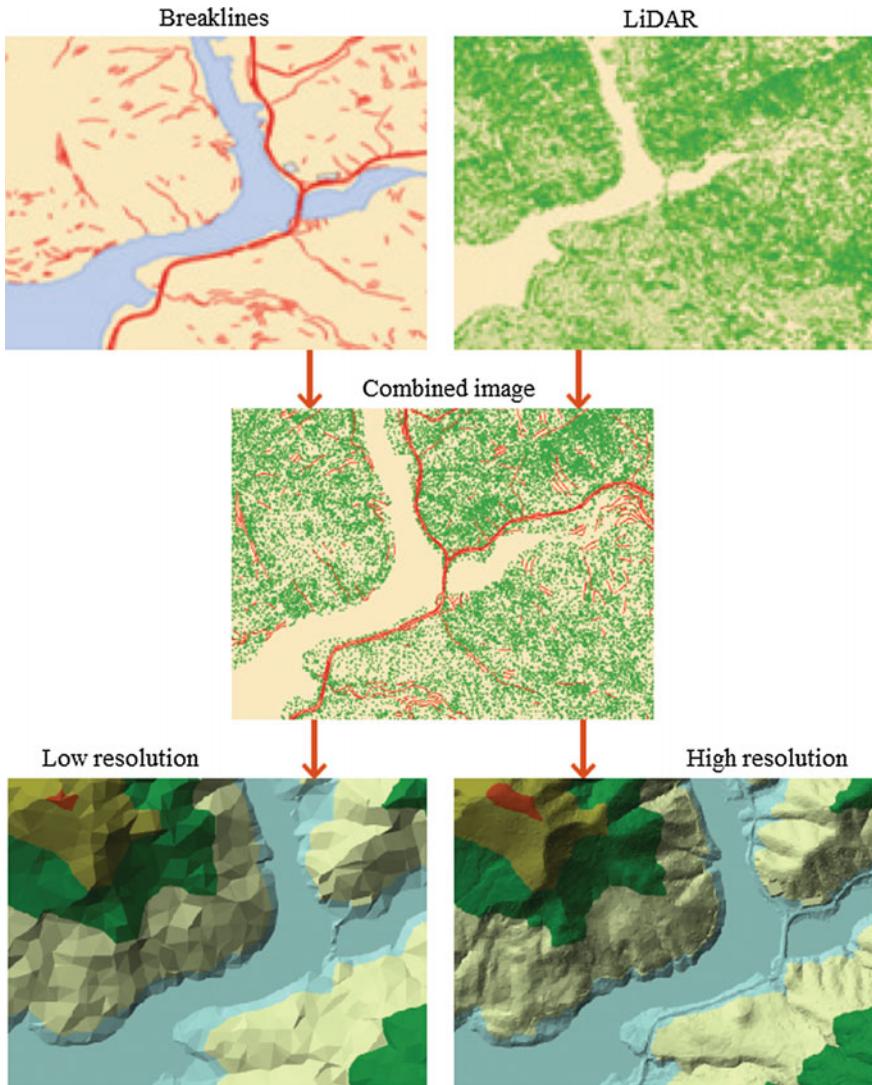


Fig. 3.2 Flow-chart of data processing in ArcGIS Spatial Analyst

territory while maintaining the interactivity in the display of the scene. Many expected functionalities were added to the new version, such as:

- Import/Export with the Google SketchUp format.
- Automatic generation and mapping of the roof textures from the aerial imagery.
- New procedural roof model with two slopes and editable orientation.
- Interactive visualization of a territory or a project at different period of time.
- Acceleration of data processing during the importation phases.

- New LandSIM3D viewer allowing the interactive comparison of project's alternatives.
- Multiple selection and edition of vector and object properties.
- New procedural buildings.
- Shadow simulation.

The Blueberry3D technology [10] is a real-time 3D visualization by offering highly detailed simulation capabilities on very large scale territories thanks to its procedural approach. The Blueberry3D is a procedural terrain middleware solution developed for Defense Simulation Industry, allowing rapid creation and visualization of vast real-time 3D databases with high level of detail, bridging the gap between high resolution 3D content and real-time display performance. The Blueberry3D technology allows to compute the triangles on the fly from that a vectorial description. This innovative approach significantly reduces the number of polygons to be processed by the graphic board and the memory size needed to store the scene. The Blueberry3D consists of a standalone Terrain Editor to entirely build the database from the GIS & elevation data, imagery and the OpenFlight objects and the RT Integration Kit SDK for the real-time application development. Since v4, it also includes the Tactical Terrain SDK that brings same the Hi-Res Terrain and content to the CGFs or physics engines.

Instead of use a limited set of discrete LODs, the Blueberry3D employs the mathematical fractals to model the ground surface and vegetation. Details are not stored but during a zooming, fractals automatically generate the additional details. This technique supports the unlimited LODs. During computing details (e.g., trees) at run time, a memory usage is minimized. A “deep-freeze” function supports a photorealistic rendering in a scene from any perspective. This function iterates the fractals further to create the finer details and to compute the soft shadows. Fractals adjust the ground level between the sample points in a height raster, smoothing a transition between terrain classes (e.g., forests, meadows, and lakes).

This software tool uses the input textures (ortho photos) and can generate its own textures for a LOD management. During rendering, the linear features like roads, rivers, and fences, are automatically converted to smooth curves. The Blueberry3D includes several pre-defined trees and shrubs that are described by the parameters of average height, trunk diameter, leaf density, etc. The end-user can design the own vegetation. Also, this software tool supports the time-of-day effects and atmospheric effects (wind, waves, reflections, haze, fog), shadows, etc. The objects, including military ones, can be added as a dynamic content in the virtual worlds.

The Blueberry3D supports ARC/INFO BIL, ASCII GRID and binary GRID, the ArcView Shape, the USGS DOQ and the SDTS DEM, the NGA DTED, the Portable Gray Map (PGM), the Windows Bitmap (BMP), the Tagged Image File (TIFF), the GeoTIFF, JPEC, GIF, PNG, and TARGA input formats. The database is stored implicitly the mathematical formulae that are used to compute a geometry during a run-time. Also, the end-users can define the camera paths through 3D worlds to create movies.

The natFX's version 5 includes the following functions:

- Display of real-time proxies in the 3DS Max viewport. The natFX proxy mode uses a simplified drawing of a plant and speed up interactive work in a viewport. During the rendering, the selected representation of a plant is transferred to the rendering engine, thus limiting memory usage.
- Vegetation loading alert. While working with 3DS Max, the natFX sums up the polygon count of the plants inserted into the scene. Then each proxy representation is colored from green to red, depending on its own polygon count.
- The natFX material library. It allows the user to edit, modify, or customize the bark, leaf, flowers, and fruit materials of any plant.
- Edition/modification of multiple plants in the 3DS Max scene. The natFX offers a capability to select a bunch of plants within the scene and increment or decrement their ages, set the season, change their LODs from one click.

3.2.4 Software Tool “Terragen”

The Terragen [11] is a powerful solution for rendering and animating realistic natural environments that is free for a non-commercial use. The Terragen is able to produce very realistic images of landscapes with plausible terrain depicting a surface coloration, detailing, shadows, clouds, atmospherics, lighting, water, and caustic effects due to the procedural modelling and ray tracing techniques. The Terragen has no capability to include objects (vegetation, in particular) within a terrain. Typically, the Terragen scene is exported to an animation/rendering packages like 3DS Max.

The Terragen is a scenery generator, including the sculpting tools and random terrain generation. The Terragen supports the landscapes with mountains, valleys, water, sunlight, clouds, shadows, haze, etc. up to 4097×4097 pixels in a real-world scaling. Any view point can be chosen, and the altitude of any point on the terrain can be determined in meters. Also, a fractal terrain can be generated. The Terragen's rendering engine has the automatic level-of-detail adjustment. Two terrains can be combined by use of difference methods. The surface of a landscape can be divided into different components (for example, grass, rock) in order to create a hierarchical surface color map. The Terragen can render a water surfaces with ripples, waves, and soft reflections. A cloud generator and a sunlight penetration system provide the atmospheric effects.

The Terragen can import and export the raw height field information in 8-bit grayscale. The compiled files can be exported as BMP files, the LightWave 3D Objects (LWO) files, 8-bit binary files, among others. Plug-ins are available to facilitate the integration of the Terragen scenery into the LightWave scenes and animations.

The Terragen 3 is available in Creative and Professional editions, both of which have optional animation features. A free non-commercial version is also available. The Terragen 3 Creative allows to access to the powerful core Terragen 3

technology at an entry level price. The Terragen 3 Creative includes the improved Global Illumination, new noise-free Depth of Field, a Content Library, enhanced 3D preview with textured model display mode, and improved performance.

The Terragen 3 Professional is the flagship product in the Terragen 3 lineup, with the most complete feature set and a full complement of professionally-oriented tools. Building on the core Terragen 3 features in the Terragen 3 Creative, the Terragen 3 Professional adds the render layers and elements, a Linux render node, spherical camera, FilmBoX (FBX) import/export, accelerated cloud rendering options, and so on. The Terragen 3 Professional is a powerful, production-ready application, able to integrate quickly and easily into most visual effects pipelines.

3.2.5 *Software Tool “3DNature”*

The 3DNature [12] is a continuation of the software tools, originally formed in the late 1980s by a video production company Questar Productions and called as the World Construction Sets (WCSs). The Visual Nature Studio (VNS) 1 was the first fully GIS-landscape visualization tool represented in 2001. The VNS integrated the high-speed terrain grid, support for unlimited projections, datums and spheroids with dynamic re-projection, and import of geo-referenced (GeoTIFF) images with an expanded list of the supported file formats. The Search Queries allowed for the instantaneous binding of Components (like Roads) to a large number of vector paths based on the specified criteria. The Extruded Walls made it easy to build the cities rapidly from outlines. The Dynamic ReImport helped the VNS keep up with the external data that might change frequently. The VNS immediately made the task of creating visualizations from diverse GIS data significantly faster and easier. A specialized version of the VNS known as the Ecosystem Decision Support System (EDSS) project was created under contract to be an automated visualization back-end for a sophisticated growth modelling tool for a US Army base.

In 2002, the 3DNature unveiled the WCS 6 with the DEM Paint, a completely reworked renderer, true volumetric clouds, an expanded content library of pre-made components, Path/Vector transfer, Joystick Control, Real-time OpenGL foliage preview, better 3D object support, cleaner reflections, High Dynamic Range output, and Post Processing. The innovative Scene Express add-on and the NatureView Express real-time Viewer for the WCS and the VNS, bringing a complete GIS-based authoring system for high-detail, realistic 3D landscapes in interactive real time, appeared in 2003. In 2004, the VNS Forestry Edition and the Forestry Wizard were presented with a capability of the GIS forestry visualization. In 2005, the 3D Nature Foliage Library Components for SketchUp were introduced. It brought the vast 3D Nature foliage library for SketchUp in a form of ready-to-use foliage SketchUp components. The Scene Express 2 provided an export support for FBX, GIS, and VNS. In 2008, the Visual Nature Studio version 3 was announced.

The VNS can convert the input contour lines, survey points, and breaklines into grids. The DEM Editor allows to edit the DEM values numerically. The VNS

permits to interpolate the higher-density elevation data by use of the Delaunay triangulation or spline operators. The VNS includes the foliage objects, natural textures, city-like textures (to simulate urban areas), pre-built components, and projects. The VNS can import and combine data in a variety of projections, datums, and coordinate systems. The VNS supports ArcINFO, ASCII DEMs, ADF, E00, ASCII arrays, DTED, GTOPO30, BIL, SDTS, MICRODEM Binary Array, and United Kingdom Ordnance Survey NTF DNM formats. This software tool supports DXF, JPEG, PNG, TIFF, BMP, ECW, GeoTIFF, AVI, and QuickTime formats. Also, it can render the stereo pairs. The VNS can export the ArcView Shape files in 2D and 3D spaces with the attributes and provide a support for external programs, such as the LightWave 3D, the Inspire 3D, and the 3D Studio Max. The VNS renders the trees created in the Onyx Tree Professional, LSYSTEMS, and other programs. The VNS enables to drag the images, merge the DEMs of different resolution, and visualize the transparent water and volumetric atmospheres. The Scenario manager has tools for managing multiple variations of a project.

The 3D Nature is a tool that expedites the creation and population of forestry visualization projects. The Forestry Edition works in conjunction with the Visual Nature Studio 2 and later versions. The Forestry Edition for the VNS 2 and 3 directly connects the forest stand and polygon attributes. The Forestry Edition understands the point, polygon and raster input data, and is available for the VNS.

Using the 3DNature, Visualization for Professionals [13], several scenarios can be implemented, for example: the present conditions, 20 years hence with no management action, 5 years post-treatment, and 20 years post-treatment, in a view of panoramic still images.

3.2.6 Software Tool “GRASS”

The open source/free software Geographic Resources Analysis Support System (GRASS) GIS is a public domain system with a full range of GIS capabilities. The GRASS [14] was originally developed in 1985–1995 at the US Army Construction Engineering Research Laboratory (CERL) in Champaign, Illinois to support land management at military installations [15]. After the CERL stopped its development in 1995, Baylor University continued the GRASS’ development and made available the version 4.2 in 1997. The development of the GRASS is now in the hands of people, who have collaborated on this Open Source project. Since 1 May, 2016, the stable release of the GRASS GIS 7.0.4 is available.

The open source nature of the GRASS allows it to be customized according to each user’s needs [16]. The GRASS flexibility resides in the accessibility of its code that is written in ANSI C, avoiding the necessity of learning a different programming or GIS-specific macro language to write new modules. It allows to modify or add modules to official distribution, for example, *r.sun* (computes a radiation from a geo referenced Digital Elevation Model (DEM)) [17] or *r.terraflow* (computation of

a flow routing and flow accumulation in massive grids) [18]. The main GRASS GIS capabilities are mentioned below:

- *Raster analysis:*
 - Automatic raster line and area to vector conversion.
 - Buffering of line structures.
 - Cell and profile data query.
 - Colortable modifications.
 - Conversion to vector and point data format.
 - Correlation/covariance analysis.
 - Expert system analysis.
 - Map algebra (map calculator).
 - Interpolation for missing values.
 - Neighborhood matrix analysis.
 - Raster overlay with or without weight.
 - Reclassification of cell labels.
 - Resampling (resolution).
 - Rescaling of cell values.
 - Statistical cell analysis.
 - Surface generation from vector lines.
- *3D-Raster (voxel) analysis:*
 - 3D data import and export.
 - 3D masks.
 - 3D map algebra.
 - 3D interpolation (IDW, Regularised Splines with Tension).
 - 3D Visualization (isosurfaces).
 - Interface to Paraview and POVray visualization tools.
- *Vector analysis:*
 - Contour generation from raster surfaces (IDW, Splines algorithm).
 - Conversion to raster and point data format.
 - Digitizing (scanned raster image) with mouse.
 - Reclassification of vector labels.
 - Superpositioning of vector layers.
- *Point data analysis:*
 - Delaunay triangulation.
 - Surface interpolation from spot heights.
 - Thiessen polygons.
 - Topographic analysis (curvature, slope, aspect), LiDAR.
- *Image processing:*
 - Support for aerial and UAV images, satellite data (optical, radar, thermal).
 - Canonical Component Analysis (CCA).

- Color composite generation.
 - Edge detection.
 - Frequency filtering (Fourier, convolution matrices).
 - Fourier and inverse Fourier transformation.
 - Histogram stretching.
 - Hue Saturation Intensity (HSI) transformation to Red Green Blue (RGB).
 - Image rectification (affine and polynomial transformations on raster and vector targets).
 - Ortho photo rectification.
 - Principal Component Analysis (PCA).
 - Radiometric corrections (Fourier).
 - Resampling.
 - Resolution enhancement (with RGB/HSI), RGB to HSI transformation.
 - Texture oriented classification (sequential maximum a posteriori classification).
 - Shape detection.
 - Supervised classification (training areas, maximum likelihood classification).
 - Unsupervised classification (minimum distance clustering, maximum likelihood classification).
- *DTM-Analysis:*
 - Contour generation.
 - Cost/path analysis.
 - Slope/aspect analysis.
 - Surface generation from spot heights or contours.
 - *Geocoding:* Geocoding of raster and vector maps, including LiDAR point clouds.
 - *Visualization:*
 - 3D surfaces with 3D query (NVIZ).
 - Color assignments.
 - Histogram presentation.
 - Map overlay.
 - Point data maps.
 - Raster maps.
 - Vector maps, Zoom/unzoom—function.
 - *Map creation:*
 - Image maps.
 - Postscript maps.
 - HTML maps.
 - *SQL-support:* Database interfaces (DBF, SQLite, PostgreSQL, mySQL, ODBC).
 - *Geostatistics:*

- Interface to “R” (a statistical analysis environment).
- Matlab.
- *Temporal framework:*
 - Support for time series analysis to manage.
 - Process and analyze (big) spatio-temporal environmental data. It supports querying, map calculation, aggregation, statistics and gap filling for raster, vector and raster3D data.
 - A temporal topology builder is available to build spatio-temporal topology connections between map objects for 1D, 3D and 4D extents.
- *Additional functions:*
 - Erosion modelling.
 - Landscape structure analysis.
 - Solution transport.
 - Watershed analysis.

Each GRASS data type (raster, vector, and site (site data means the point data with three spatial dimensions and values)) as well as 3D grid format may be used for visualization in a single 3D space. There are four object types with various ways of representation, such as surfaces, vector sets, site objects, and volumes:

- A surface requires at least one raster data set to display a topography data or other raster data sets to represent attributes of color, transparency, masking, and shininess. These data sets have been derived from vector (e.g., contour) or scattered point data using tools within the GIS.
- 3D vector sets are not supported directly. In order to display 2D vector sets in 3D space, they ought to be associated with one or more surfaces. Then 2D vector sets are draped over any of these surfaces.
- Point data is represented by 3D solid glyphs. Attributes from the database may be used to define the color, size, and shape of the glyphs. 2D site data must be associated with one or more surfaces, and 3D site data may be associated with a surface (e.g., sampling sites measured as depth below terrain).
- 3D grid data are represented by isosurfaces or cross sections at user-defined intervals. Color of the isosurfaces may be determined by threshold value or by draping color representing a second 3D grid data set over the isosurfaces.

A number of the GRASS applications is growing rapidly due to the typical advantage of Open Source/Free Software projects, where applications and development are tightly entwined. Thus, the applications are organized in the following categories: environmental monitoring, geology, mathematical models, LiDAR, solar radiation, climate and atmospheric models, landslides risk mapping, wildfires risk mapping, glaciers, geomorphology, traffic and acoustic pollution, archeology, forest management, avalanches risk mapping, flood forecasting and management, GPS planning, wildlife management, GRASS and WebGIS, and GRASS and Personal Digital Assistants (PDAs).

The GRASS modules for multi-resolution analysis with wavelets [19] provides the GRASS of an efficient way (from both a mathematical and a computational point of view) to represent the same phenomena at different resolutions and filter out the signal features. Some applications to geomorphologic problems are presented. Spatio-temporal monitoring of evolving topography using LiDAR, real-time kinematic GPS, and sonar data [20] present the spatial interpolation and topographic analysis of different types of high resolution data, such as the LIDAR, the GPS, and sonar data. Managing and processing of the LiDAR data within the GRASS [21] describes a procedure for the automatic detection and removal of features over a high resolution Digital Surface Model (DSM) from the LiDAR data.

Two basic methods have to be distinguished, when transforming 3D point data to volumetric data:

- Direct conversion of 3D points to their 3D voxel representation restricted to existing data values. 3D data can be converted directly using the corresponding GRASS modules. If no 3D region yet exists, then it may be defined as the 2D region definitions specified at the GRASS and extended by the third spatial dimension with a user defined voxel resolution. In the case of soil data modelling, the negative z values are defined. In the case of meteorological or other above-surface applications, only positive z values are used. All voxels, not being related to a sites list entry, receive a NULL (no data) value during conversion.
- Rendering of a full volume through 3D interpolation including the missing values estimation. The missing values are interpolated using, for example, the Regularized Splines with Tension (RST) algorithm [22, 23].

One can find how to build the groundwater flow models using the GRASS tool in [24].

3.2.7 *Software Tool “Object Raku Technology”*

The Object Raku Technology [25] is a collection of software tools for analysis of the LiDAR data. It includes Feature Type Interpreter (FTI) for accurate and automatic information extraction of the LiDAR data, forestry analysis (Timber Species Identifier (TSI) for accurate identification of timber species from the LiDAR data), Geo modelling, and Urban Terrain Visualization.

The Object Raku Technology’s FTI builds on the foundation of the Sextant Geospatial software suite. It helps to identify buildings and vegetation in dense data sets that is a crucial step to realize the full value of the LiDAR collected. The FTI focuses on solving the problem of feature extraction with the highest level of speed and accuracy possible, including the size, location, and direction of objects. The source data is presented in a familiar ortho-like visualization format and so is intuitively easier to interpret than a mass of points. Geo-analysts can leverage the well-established image analysis techniques in addition to conventional point-cloud

approaches. The FTI processes data, representing in the LAS and GeoTIFF formats (of the first return, last return, intensity, and color imagery). The LiDAR data can be classified as bare Earth, buildings, vegetation (low, medium, and high), potential water areas, and roads. The FTI is configured to run on Windows 7 (32 or 64-bit) and will function well with most video cards produced after 2010. 4 GB RAM is recommended and Microsoft DirectX 9c or higher is required.

The Object Raku Technology's TSI is a system of automated software components that analyzes the LiDAR data to determine the location and species of individual trees. The TSI is a system designed to readily incorporate any tree species. The TSI code was written to enable an almost limitless species catalog, including Douglas fir, western red cedar, western hemlock, Amabilis fir (balsam), yellow cedar (cypress), red alder, big leaf maple, sitka spruce, and lodgepole pine. The TSI begins by segmenting the forest canopy into the individual trees. For a given point cloud area, the segmentation process produces an area shape-file for each tree, including a height attribute. The TSI can validate the timber species with accuracy measured to within 10% of available cruise and scale information and is 84% for second growth (328 harvested blocks) and 72% for old growth (209 harvested blocks). The TSI analysis module, called the Block Design Tool, is built on the ESRI ArcGIS engine and allows the forester to outline proposed harvest areas and determine potential profitability based on harvest cost and revenue parameters.

The Geo-specific Modelling involves a wide array of the GIS and 3D modelling technologies, including the Object Raku's Sextant software suite. The Urban Terrain Visualization builds the precise 3D building models that can be applied in the fields of city planning for simulations, microclimate investigations, in telecommunications for transmitter placement, and military operations in urban terrain.

3.3 Survey of Software Tools for Vegetation Modelling

First software tools for vegetation modelling appeared since 1970s, when a procedural modelling as a popular theory for plants and trees' simulation developed by famous scientists, such as Lindenmayer [26], Oppenheimer [27], Weber and Penn [28], Prusinkiewicz [29], Honda [30]. The software tools, such as the Xfrog, the SpeedTree, the Onyx Computing, the Marlin Studios, the E-on Software, and additional ones, are discussed in Sects. 3.3.1–3.3.6, respectively.

3.3.1 Software Tool “Xfrog”

The Xfrog [31] is a procedural organic 3D modeller that allows to create and animate 3D trees, flowers, nature based special effects, or architectural forms. The

Xfrog is available as a Mac OS, Windows, and Linux plug-in for Maya, Cinema 4D and as a standalone application. The Xfrog is an acronym called as X-windows based Finite Recursive Object Generator. The XfrogPlants are realistic, highly detailed, fully textured 3D trees, shrubs, and flowers, carefully modelled with the Xfrog procedural software. Near 600 species are available in libraries or as individual species. The average size of 3D models is about 200,000–1,500,000 polygons.

One can create the trees or other plants that grow respect the Sun and gravity, create the leaves and branches that move in the wind, the biological reactions taking place over time, the architectural models that change over time, etc. Near 1000 models of trees, flowers, and other types of vegetation are released in 20 XfrogPlants libraries. The Xfrog is not related to fractals or to strict mathematical approaches, such as the L-systems. The Xfrog is a unique approach to represent natural processes in computer graphics.

The Xfrog software involves the Xfrog for Cinema 4D (organic modelling plug-in for Maxon Cinema 4D), the Xfrog for Maya (organic modeling plug-in for Autodesk Maya), the Xfrog for Windows (standalone organic modelling software for Windows), and the XfrogTune (polygon reduction tool for Xfrog models). The Xfrog 5 includes many objects, for example, branching, phyllotaxis (to recreate the distributions typical for flowers and blossoms), and tropism. All parameters in the Xfrog 5 can be animated and all Xfrog 5 Objects can be combined with Objects in Maya.

All connections between the Xfrog objects are established inside the Xfrog 5 Visor. Thus, the object Branch creates the branching structures. One Branch on its own looks in its default shape like a conical horn, similar to a trunk. This shape is achieved through the multiplication of the cross-section splines that are connected to form a hull. Each cross section is scaled down sequentially along the length of the branch, ending in a point. A linking of several Branch objects together creates a tree structure with several branching levels. Each object represents one branching level. Examples of the Xfrog object and structures as well as examples of the generated trees are depicted in Figs. 3.3 and 3.4, respectively.

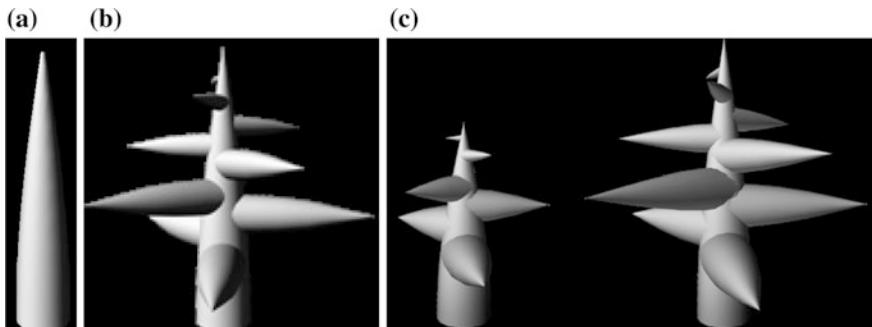


Fig. 3.3 Examples of Xfrog object and structures: **a** trunk, **b** tree structure, **c** tree growth

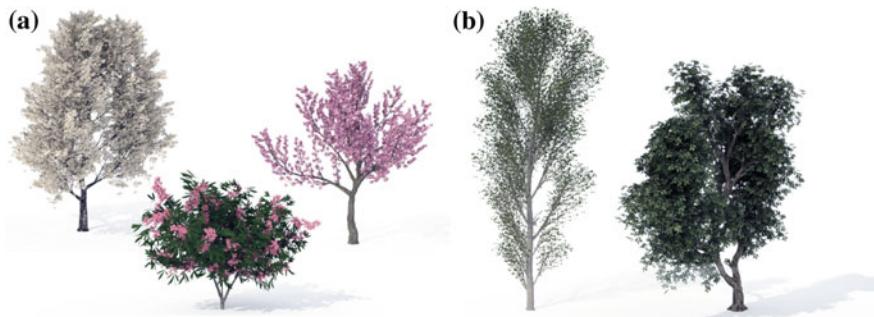


Fig. 3.4 Examples of generated trees: **a** blossoming trees, **b** European trees

Unfortunately, this software tool suffers from a less-than-intuitive interface. The plants created with the Xfog are impressively detailed with the wind simulation or the growth of a plant.

3.3.2 Software Tool “SpeedTree”

The SpeedTree 7 [32] has been released in three different versions: the SpeedTree Cinema, the SpeedTree Studio, and the SpeedTree Architect (as the SpeedTree for Games features). Each version comes with the SpeedTree Modeler, the cornerstone of the SpeedTree 3D animation software experience.

The SpeedTree for Games v7 is a middleware solution that has set the industry standard for modelling and real-time rendering of trees and plants for games, including the SpeedTree Modeler, complete the Tree Model Library, the SpeedTree Compiler, and full-source Software Development Kit (SDK).

The SpeedTree Modeler permits to integrate the trees and life-like vegetation into the game environments. It has the following features:

- Hand drawing of the branch structures directly in the viewport, using the tablet device or mouse.
- Possibility to break a branch or delete individual branches and leaves entirely.
- Using the forces to curl, twist, or gnarl any tree part.
- Growing of the trees around or inside arbitrary meshes.
- Scale resolution of the polygonal details without affecting the shape of a model.
- Testing of the real-time physical interactions, wind, and the LOD in the Modeler.
- Editing of individual tree parts without affecting on the rest of the tree. The node-editing approach lets to tune or delete any branch, frond, or leaf.
- World building with a tree placement export directly from the SpeedTree Modeler.
- Weld branches have not the lighting seams or branch intersections.

The Tree Model Library includes broadleaf trees (55 types), conifers (23 types), palms and cacti (19 types), shrubs and flowers (42 types), marine (10 types), and miscellaneous and fantasy (27 types). Examples of trees modelling are depicted in Fig. 3.5.

The SpeedTree Compile was developed to quickly combine the efficient real-time tree models, texture atlases, and 360 degree normal-mapped billboards. It has the following possibilities:

- The SpeedTree Compiler has a wizard mode that quickly gets the real-time trees, with simpler options.
- Efficient binary file format. The Compiler computes and renders not only billboards and texture maps but also efficient binary tree models.
- Perfect billboards even under a dynamic lighting. The Compiler outputs all the texture layers needed to pull it off.
- Fusion the trees with texture atlases that reduces a number of texture look-ups by combining textures from multiple trees into a shared set of atlases.

Functions of the SpeedTree SDK are mentioned below:

- Vast forest rendering in real-time mode and with high forest density.
- Improved rendering supports a shadow mapping, translucency, ambient occlusion, per-pixel lighting, alpha-to-coverage, etc.
- The LOD system yields seamless transitions.
- Boundless terrain and grass engines.
- Wind-blown trees look very realistic.

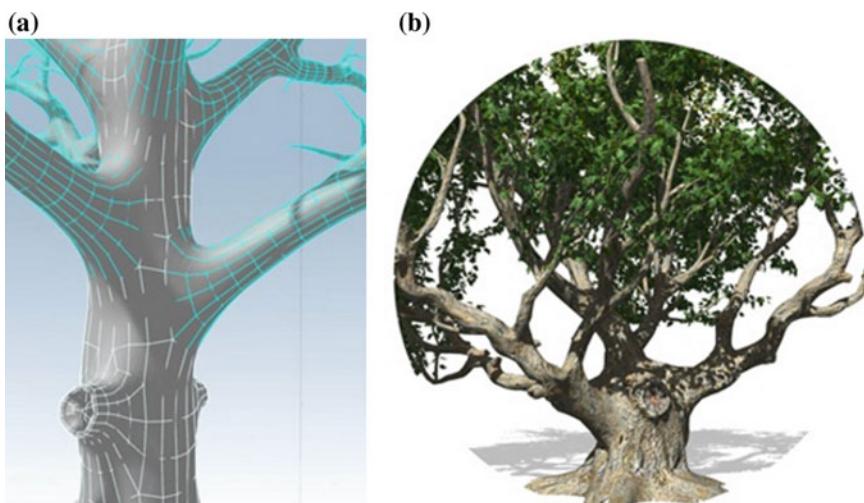


Fig. 3.5 Examples of trees modelling: **a** pipeline tree, **b** cropped trees (from [33])



Fig. 3.6 Forest scene generated by the SpeedTree SDK (from [34])

With the version 7 of the SpeedTree SDK, the hues can be varied per-instance and per-vertex to achieve more naturalist color and texture effects. The size and hue variation provides the appearance of many unique trees with just a single tree model. A forest scene generated by the SpeedTree SDK under image-based ambient lighting is depicted in Fig. 3.6.

Generators are collections of like nodes that comprise the tree model (each individual tree part, such as a branch or leaf card, is called a node). With a typical tree model, there will be a generator for the trunk that generates one node (the trunk itself), a generator for the main branches that may have many individual nodes, and a generator for the leaves that grow off these branches, which have even more nodes. There are several types of generators and each one creates its own type of nodes. The types of generators are listed below in Table 3.3.

The SpeedTree Cinema is a powerful 3D animation tool that is used to create plants and trees for feature films. The SpeedTree Studio is devoted to vegetation modelling, animation, and rendering features even available for modest production budgets.

3.3.3 Software Tool “Onyx Computing”

The Onyx Computing [35] is a software development studio specializing in the design and development of knowledge based systems for procedural modelling of vegetation. This company was founded and incorporated in 1992. The OnyxGARDEN Suite version 2211 for Windows is a cost saving collection of essential tools for a serious computer graphics artist. The package contains six

Table 3.3 Types of generators in the SpeedTree 7

Type of generator	Description
Tree generator 	Every tree model has exactly the one Tree Generator. This is the only generator that does not have a “Generation” property group (it is always at the root of the hierarchy). The Tree Generator also houses global tree settings, such as for level of detail, leaf collision, and degradation
Spline generator 	The Spine Generator is responsible for generating the spine nodes. Both branches and fronds are the part of the Spine Generator. The icon will change, depending on the enabled geometry types. The four states are the following ones: a spine (no branches or fronds), the branches, the fronds, or both (branches and fronds)
Hand drawn generators 	The Hand Drawn Generators are signified with a drawing hand overlay in the lower right corner of the icon. Each branch level has a target hand drawn generator that has a check (\checkmark) next to the hand overlay
Leaf generator 	The Leaf Generator is responsible for generating leaf nodes. Leaves can be of two types: leaf cards (billboards) or leaf meshes (arbitrary mesh objects)
Zone generator 	The Zone Generator serves as a parent for other objects to grow of, similar to the Tree Generator. The Zone Generators are either of the disc or mesh variety
Proxy generator 	The Proxy Generator creates stand-in objects called proxies that usually get attached to the Zone Generator. The position, scale, and rotation of proxy nodes can be exported as a tree location file in SWA format

stand-alone plant creators/generators/editors with corresponding libraries of more than 460 ready to use the broadleaf and coniferous trees, shrubs, palms, bamboos, flowers, grasses, and grass covers. These stand-alone modellers are compatible with virtual rendering software. They export out the plants in six most popular file formats: 3DS, C4D, LWO, OBJ, DXF, FAC. The Onyx Computing works under Windows and Macintosh operating systems. The milestones of the Onyx Computing development are pointed in Table 3.4.

The short description of the Onyx Libraries is located in Table 3.5.

The main key features of the Onyx modellers are the following:

- Wide variety of 3D, biologically faithful trees, bushes, flowers, and grasses.
- Fast procedural modelling.
- Numerous parameters for modelling control and detail.
- Spread grass over imported 3D terrain.
- Fast, non RAM-dependent rendering.
- Render with light and shadow.
- Wind sequence export.

Table 3.4 Milestones of the Onyx Computing development

Year	Description
1992	TREE, the first developed tree parametric modeller, was released. It cuts down the time spent on modelling dramatically and adds natural beauty to trees generated with the software. The scenes with those trees begin to look as if taken directly from the nature
1993	TREE Professional gets additional generator specifically designed for modelling conifer trees. This broadleaf-conifer duo covers even wider variety of tree species and pushes standards of quality and performance higher up
1994	TREE Professional gets additional generator – a parametric modeller for palms
1996	TREE STORM, the plug-in for trees on the wind, was released
1997	TREE Professional for Windows platform was released
1998	TREE STORM plug-in for 3DS Max was released
1999	TREE Classic was released. It is a junior package to TREE Professional made for those, who prefer working with images rather than 3D models, and for those that use TREE Storm plug-in
2000	Sun light was incorporated in TREE's proprietary rendering engine. The algorithms capture all these natural effects with unmatched fidelity
2001	BAMBOO, the first dedicated parametric modeller for bamboos, was released. The generator has captured the plant's structural beauty and the elasticity, endurance, and tenacity, for which bamboos have earned a special place in Chinese and Japanese cultures
2002	The OnyxTREE Suite of parametric modellers for the Mac OSX platform was released
2003	New version of the OnyxTREE STORM plug-in for the ElectricImage was released
2005	The OnyxFLOWER, the first dedicated parametric modeller for flowers, was released. The OnyxTREE Suite was renamed to the OnyxGARDEN Suite. The new name reflected the fact that the suite of modellers included the plant families and species other than trees for building virtual gardens and landscapes
2008	The OnyxGRASS, the first dedicated parametric modeller for grasses, was released. The modeller created a wide variety of decorative grasses, wild grasses, and grass covers. Notice that the OnyxGRASS is the result of in-depth research of botanical literature, photographic material, and numerous field studies
2008	A proprietary ONX 3D object file format was released. At this time, the OnyxGRASS was the only modeller that supports ONX file. The Onyx2Max, the ONX import plug-in for 3DS Max, was released
2009	The TREE STORM 2010 and the Onyx2 Max 2010 plug-ins for 3DS Max 2010 were released
2010	The OnyxGRASS 2 was released. The major features were 3D terrain OBJ import, spreading grass over that surface, and export it out
2011	The TREE STORM 2012 and the Onyx2 Max 2012 plug-ins for 3DS Max 2012 were released. Also, the OnyxTREE BROADLEAF 7030 and the OnyxPALM 7030 were released. These modellers export wind 3D tree sequence export to LightWave, Cinema, Modo, Maya, etc.
2012	The TREE STORM 2013 and the Onyx2Max 2013 plug-ins for 3DS Max 2013 were released

Table 3.5 Short description of Onyx libraries

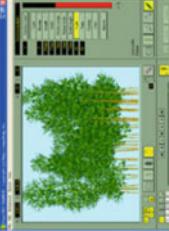
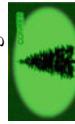
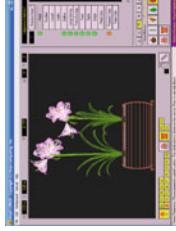
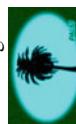
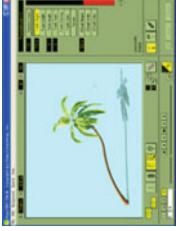
Title of libraries	Short description	Main screen
The OnyxBAMBOO Creator/generator/editor + bamboo libraries 	The OnyxBAMBOO is a dedicated procedural creator and modeller of 3D bamboo plants. It creates and models a single bamboo, many bamboos planted on a defined area, and continuous bamboo growth over time. The modeller comes with the BAMBOO Library of pre-modelled plants ready to use or edit	
The OnyxTREE BROADLEAF Creator/generator/editor + broadleaf libraries 	The OnyxTREE BROADLEAF is a dedicated procedural creator and modeller of 3D broadleaf trees, shrubs, and bushes. Besides 3D, it exports lit single polygon panel trees with shadow. The modeller comes with the extensive BROADLEAF Library with over 220 ready-to-use presets of trees, bushes and other plants.	
The OnyxTREE CONIFER Creator/generator/editor + conifer libraries 	The OnyxTREE CONIFER is a dedicated procedural creator and modeller of 3D CONIFER trees and bushes. Besides 3D, it exports lit single polygon panel trees with shadow. The modeller comes with the extensive CONIFER Library with over 50 of ready-to-use and editable presets of trees, bushes and other plants	
The OnyxFLOWER Creator/generator/editor + flower libraries	The OnyxFLOWER is a dedicated parametric modeller for flowers. With hundreds of parameters carefully structured	(continued)

Table 3.5 (continued)

Title of libraries	Short description	Main screen
 The OnyxGRASS Creator/generator/editor + grass libraries	around the principal flower elements, the modeller gives the user unprecedented control and enables easy creation of a wide variety of flower specie	
 The OnyxPALM Creator/generator/editor + palm libraries	The OnyxGRASS is the first ever dedicated parametric modeller for grasses. The modeller creates a wide variety of decorative grasses, wild grasses, and grass covers and it comes with the library of ready-to-use grasses and grass covers. This modeller has a well structured, intuitive interface, easy to understand, and especially so by all those who are visually inclined	

- Save 3D model in formats 3DS, C4D, DXF, FAC, LWO, OBJ as well as in plugin ONX.
- Exports numbered growth sequences.
- Exports grass image as BMP and TGA.

The wind is modelled as a light breeze, moderate breeze, and moderate gale, respectively.

3.3.4 Software Tool “Marlin Studios”

The Marlin Studios [36] includes the man-made and natural objects, the TreesFarm particularly, in the landscape scenes. This software tool can produce over 200 trees and foliage textures that may be applied to the billboards within the animation packages and real-time simulation. Like most billboard textures, the trees have preset lighting, generally with the Sun high in the sky. This causes a necessity to bring out the contrast in the leaves. The shadow-casting versions of the tree images are also included in the package. Each tree and plant comes in three resolutions and includes the bump maps and an alpha map (in order to create a transparency). The TreeFarm includes:

- The 3D TreeFarm Deciduous. It involves two libraries of 76 easy to use basic 3D tree models with 25 leaf variations and colors and 9 different optional bark and leaf maps per tree, such as translucency and diffusion. Based on these libraries, 1450 pre-textured 3D tree models were constructed.
- The 3D TreeFarm Palms. It includes 82 tree models with different leaf types and colors.
- The 2D Virtual Trees & Foliage. It contains the library of alpha map textures with 66 trees, 136 plants, and flowers. This library provides 202 tree and foliage textures, all provided in alpha map texture form with defined transparency. It includes 20 panoramic backgrounds and 60 background photos.
- The 2D Tropical Trees & Foliage. It comprises 50 trees and plenty of foliage alpha maps. This library provides 116 tropical tree and foliage textures, all of them have an alpha channel transparency. 2D alpha map textures can be used as an alternative or enhancement to 3D models.

Besides trees' models, the Marlin Studios includes 3D/2D animated people models, 32 3D people models and 76 2D people, 3D cars and trucks with drivers.

3.3.5 Software Tool “E-on Software”

Since its inception in 1997, the E-on Software [37] has pioneered the concept of Digital Nature and enjoyed the widespread adoption of its solutions across a variety of industries and user communities. The Digital Nature Products include solution for CG

Professionals, products for 3D Artists, and products for 3D Enthusiasts. Each product contains the Vue, the PlantFactory, the Ozone, the Carbon Scatter, and the LumenRT.

The Vue is a premiere solution for seamlessly integrated Digital Nature environments.

The PlantFactory is a revolutionary 3D vegetation modelling, animation, and rendering software, dedicated to the high-level shading language for programming vertex and pixel shaders (CG), SFX, Architecture, and Gaming communities. The main features of the PlantFactory are the following:

- Various types of vegetation, from simple grass to super elaborate hero trees.
- Vegetation can be painted, assembling simple building blocks or completely graphing all plant properties.
- Procedural geometry and materials of unlimited detail are used.
- All plant can be animated, using precise wind and breeze algorithms.
- 3D vegetation may be exported as standard OBJ, 3DS, C4D, LWO, ABC (alembic), or animated FBX.
- Stills and animations can be rendered.
- Plant library may be expanded through the TPF Nursery.

The PlantFactory versions involve the products, representing in Table 3.6.

As a next generation vegetation platform, the PlantFactory incorporates four foundational elements:

- Multi-Platform/Multi-Discipline operates across all CG platforms including real time, streaming, and offline rendering systems and caters to the needs of CG, SFX, Landscaping, GeoDesign, Architecture, and Gaming communities.
- Botanical Coherency allows the creation of more botanically accurate plant appearances and behaviors.
- Precise Control help to create the plants of any desired shape, appearance, and behavior using simple building blocks.
- Massive Populations creates the landscapes, containing millions of plants.

The plants creating with the PlantFactory technology can be exported to any 3D Application, using multiple export formats such as FBX, 3DS, OBJ, C4D, LWO, etc. Plants are exported fully rigged and textured. Breeze or wind animated plants can be exported, using a fully rigged mesh or as a cloud of animated vertices.

The Ozone is a simulation of hyper-realistic 3D atmospheres. The Ozone 2015 plug-in lets create and render hyper-realistic skies and atmospheres in 3DS Max, Maya, Softimage, LightWave, and Cinema4D. The Ozone 2015 implements the cutting-edge technologies developed by the software for simulation and rendering of atmospheric effects. Its atmospheres provide an accurately simulated environment that affects all elements of scenes and behaves according to nature's rules. The Ozone 2015 is built on the same algorithms and integration technology as the Vue xStream but is limited to atmospheres and skies. With four atmospheric models – spectral, volumetric, standard, and environment mapping, over 100 preset atmospheres and 100 preset cloud shapes, the Ozone 2015 is the most complete, realistic and productive solution for creating skies and atmospheres that come alive.

Table 3.6 The PlantFactory versions

Emblem	Short description
	<i>Beautiful 3D Plants</i> Easily produce high-quality, photo-realistic 3D plants: hand-draw from scratch, assemble pre-made components or customize existing models Export of static 3D meshes for use in other applications or loading of plants directly in the VUE and the Carbon Scatter
	<i>Animated Plants for CG Professionals</i> Wind & animation effects and control every detail of plant designs, using the advanced procedural modeling graph Export of fully animated plant meshes to any 3D application or loading of plants directly in the VUE, the Carbon Scatter, and the LumenRT
	<i>The Solution for Production Studios</i> Fully animated 3D plants in concert with production workflow With expanded collaboration and compositing capabilities, advanced artistic control and dependable productivity, the PlantFactory Producer is the perfect solution for demanding production studios
	<i>Plants for VUE Artists</i> Designed for the VUE 2014 + Esprit, Studio or Complete users, this special version of the PlantFactory integrates seamlessly with a product and represents the perfect combination of functionality
	<i>Plants' Export to Any 3D Application</i> Plant species extracted from the PlantFactory Nursery can be exported as static meshes for use in other 3D applications

The Carbon Scatter 2015 can create the wind-swept forests or animated crowds and render them directly inside 3DS Max, Maya, and Cinema4D. The Carbon Scatter plug-ins integrate the EcoSystem algorithms directly inside leading CG applications, allowing to populate the scenes with millions of native objects and render them.

The LumenRT is the Communication Platform for Architecture and GeoDesign. The LumenRT provides an interactive 3D medium to create and explore the GeoDesign projects. As a result, the LumenRT helps the projects designed in true harmony with their environment prevail.

3.3.6 Additional Software Tools

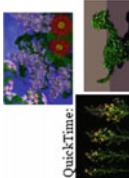
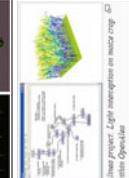
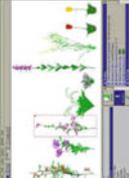
Additional both commercial and non-commercial implementations of terrain and vegetation visualization program [38] are depicted in Table 3.7 (plant generation software packages) and Table 3.8 (plant software: only available as a modeller plug-in).

Table 3.7 Plant generation software packages

Tool	Description	Pros	Cons	Image
The Lenné3D http://lenne3d.com/	Very powerful plant software, once available as plug-ins, is now built into their the Biosphere 3D software	Leverages the Xfrog. The results of several years of academic research into real-time rendering of large number of plants	Rather large file sizes	
The ngPlant http://ngplant.sourceforge.net/	Standalone application for creating plants. Looks very much like a simpler version of the Xfrog, with a more narrow focus	Free and open source. Seems fairly powerful for a free program	Lots of complex controls, could take awhile to learn	
The Unity Tree Creator http://docs.unity3d.com/Manual/class-Tree.html	A tree authoring tool is part of the Unity game engine	Good looking trees, optimized for real-time rendering	Tightly integrated with the game engine, unsure if useful beyond the Unity	
The Virtual Gardening and the Virtual Bonsai http://www.jfp.co.jp/garden_e/ and http://www.jfp.co.jp/bonsai_dl/	Standalone application from JFP, Inc., for modelling gardens and bonsai trees	Beautiful output, real-time interaction (OpenGL)	Only available in Japanese	
The Arbaro http://arbaro.sourceforge.net/	Tree modelling tool, free and open-source, written in Java	Free (gratis and libre), Writes OBJ files	Look like just a few types of plant possible	
The Ivy Generator http://graphics.uni-konstanz.de/~lutf/ivy-generator/	The Ivy grows from a single root following different forces and constraints	Free and cross-platform	Minimal UI	

(continued)

Table 3.7 (continued)

Tool	Description	Pros	Cons	Image
The Algorithmic Botany: ViLAB (Linux) and L-Studio (Windows) http://algorithmicbotany.org/	Dr. Prusinkiewicz's full package, was available as shareware. Latest website has no information about availability	Lots of features Most major papers in the field came from research with this package		 
The OpenAlea http://openalea.gforge.inria.fr/doku.php	Research software	Looks powerful	Very low level, focus is on biological research	
The Treemagik http://www.aliencodec.com/product_treemagik.php	Tree modeler for Windows, from Aliencodec	Low-polygon output with texture per branch Many output formats Cheap	Limited range of output, no real species Really weird nearly-illegible UI Highly restricted use of output	
The Plant-Lite https://www.thegamecreators.com/?m=view_product&id=2088	Plant modeler does rocks, mushrooms, grass, weeds, ferns, lilies, flowers, bushes, twigs, tropicals	Same as the Treemagik: Low-polygon output with texture per branch Many output formats Cheap	Same as the Treemagik: Limited range of output, no real species Really weird nearly-illegible UI Highly restricted use of output	
The Plant Studio http://www.kunzfernheit.com/PlantStudio/	Herbaceous plant modeler for Win32. Use to be inexpensive, now free	Does gardens and small flowering plants very nicely Great deal of parameter control in real botanical terms Free	Uses a dated, built-in software renderer instead of a 3D API 3D output is limited to POV and DXF	

(continued)

Table 3.7 (continued)

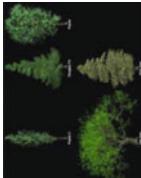
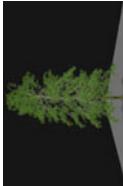
Tool	Description	Pros	Cons	Image
The Forester Arboratum http://www.dartnall19.co.uk/	Tree modeler for Windows. Generates detailed trees in POV-Ray, X, and OBJ formats		Output is generally 2-20 + Mb files Only 5 example trees Doesn't seem to do non-tree plants	
The Intel Smoke Library (see Procedural Trees and Procedural Fire in a Virtual World) https://software.intel.com/en-us/articles/procedural-trees-and-procedural-fire-in-a-virtual-world/	Open-source library for procedural simulation of fire and smoke that curiously also does a procedural creation of tree			

Table 3.8 Plant software: only available as a modeler plug-in

Tool	Description	Pros	Cons	Image
The Sapling https://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Curve/Sapling_Tree	The Blender Plug-in (free)	Looks like rather basic branching geometry but capable of a few common tree types		
The Branches http://www.branches.ws/	The 3DS MAX Plugin (free with registration)	This is not a tool for generating procedural trees but a modeling tool for creating trees by hand intended for games		
The Laubwerk Plants Kits http://www.laubwerk.com/	The 3DS MAX Plugin	Quality of the trees looks good There is a free version that includes two species	It is just 10 species of tree, although each comes in 3 shapes, 3 ages and four 4 seasons, for a total of 36 variations	
The TreesDesigner http://www.polas.net/trees/index.php	The LightWave Plugin with the separate Leaves Generator	Where the branches/trunk split, it is quite smooth, better than the intersecting cylinders of nearly all plant software Reportedly, there are no restrictions on use of the output	The branch angles look rather unnatural, and it appears to only do tree-type plants	

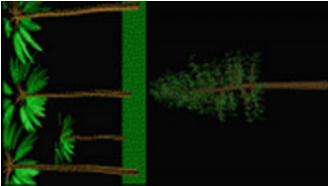
(continued)

Table 3.8 (continued)

Tool	Description	Pros	Cons	Image
The DPIT Nature Spirit http://www.fluidsimulation.de/navie/index.php	Plug-in for the Cinema 4D		Really odd, lumpy, unnatural-looking trees	
The Maya Paint Effects http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=7635770#Paint	The Maya plug-in (included with the standard Maya product)	One can create a jungle of trees and plants in a matter of minutes and paint a grass onto the ground		
The Digital Landscapes http://www.jp.co.jp/landscape_e/index.htm	Softimage plug-in (NT/Irix) from JFP, Inc. Based on the research of Dr. Norishige Chiba of the Department of Computer Science at the Faculty of Engineering at Iwate University	Beautiful output, impressive control of plant distribution	Only available as a plug-in	

(continued)

Table 3.8 (continued)

Tool	Description	Pros	Cons	Image
The Tree Druid http://www.zenstar.com/ treedruid_sa.htm	Plug-in for 3D modeller Carrara, also available as a standalone app	Includes 36 trees	An old program by this name had very crude, unrealistic plants	

The software tools represented in Tables 3.7–3.8 can be recommended as a prototype budget step in vegetation modelling.

3.4 Conclusions

This chapter presents a short survey on commercial and non-commercial software tools for terrain and vegetation modelling. The main functionality of all packages is similar, however not all of them can provide the realistic and real-time rendering in large natural scenes. These software tools are the result of the developers' efforts in a deep fusion of the innovative algorithms, high-performance program implementation, and modern hardware. The further improvement and extension are forecasted.

References

- Conforti D, Zampa F (2011) Lynx mobile mapper for surveying city centres and highways. ISPRS—international archives of the photogrammetry, remote sensing and spatial information sciences, XXXVIII-5/W16, pp 219–222
- Trimble MX8. <http://www.trimble.com/Imaging/Trimble-MX8.aspx>. Accessed 24 July 2016
- Pu S, Rutzinger M, Vosselman G, Elberink SO (2011) recognizing basic structures from mobile laser scanning data for road inventory studies. ISPRS J Photogramm Remote Sens 66 (6):S28–S39
- Mongus D, Zalik B (2012) Parameter-free ground filtering of LiDAR data for automatic DTM generation. ISPRS J Photogramm Remote Sens 67:1–12
- Terrasolid. <http://www.terrasolid.com/home.php>. Accessed 23 July 2016
- Terrasolid. https://www.terrasolid.com/ssl/download_software.php. Accessed 23 July 2016
- Shen D, Wong DW, Camelli F, LiuY (2013) An ArcScene plug-in for volumetric data conversion, modeling and spatial analysis. Comput Geosci 61:104–115
- Bunting P, Clewley D, Lucas RM, Gillingham S (2014) The remote sensing and GIS software library (RSGISLib). Comput Geosci 62:216–226
- Bionatics. Simulation for Decision. <http://www.bionatics.com/>. Accessed 19 July 2016
- Blueberry3D Version 4—the Hi-Res terrain consistency. <http://www.blueberry3d.com/Site/product/blueberry3d.php>. Accessed 19 July 2016
- Planetside software. <http://www.planetside.co.uk>. Accessed 25 July 2016
- DNature. <http://www.3dnature.com>. Accessed 25 July 2016
- DNature. Visualization for Professionals. <http://www.3dnature.com/index.php/products/forestry-wizard-forestry-edition/bighorn-national-forest-case-study/>. Accessed 25 July 2016
- GRASS GIS. Bringing advanced geospatial technologies to the world (2016) <http://grass.osgeo.org/>. Accessed 23 July 2016
- Neteler M, Mitasova H (2004) Open source GIS: a GRASS GIS approach. The Kluwer international series in engineering and computer science, 2nd ed. Kluwer Academic Publishers, Boston
- Petrasova A, Harmon B, Petras V, Mitasova H (2015) Tangible modeling with open source GIS. Springer, Heidelberg
- Suri M, Hofierka J (2004) A new GIS-based solar radiation model and its application to photovoltaic assessments. Trans GIS 8(2):175–190

18. Arge L, Chase JS, Halpin P, Toma L, Vitter JS, Urban D, Wickremesinghe R (2003) Efficient flow computation on massive grid terrain datasets. *GeoInformatica* 7(4):283–313
19. Zatelli P, Antonello A (2002) New GRASS modules for multiresolution analysis with wavelets. In: Ciolfi M, Zatelli P (eds) Open source free software GIS—GRASS users conference 2002. <http://www.ing.unitn.it/~grass>
20. Mitasova H, Drake T, Harmon R, Hofierka J, Mcninch J (2002) Spatio-temporal monitoring of evolving topography using LiDAR, RTKS and sonar data. In: Ciolfi M, Zatelli P (Eds) Open source free software GIS—GRASS users conference 2002. <http://www.ing.unitn.it/~grass>
21. Brovelli MA, Cannata M, Longoni UM (2002) Managing and processing LiDAR data within GRASS. In: Ciolfi M, Zatelli P (eds) Open source free software GIS—GRASS users conference 2002. <http://www.ing.unitn.it/~grass>
22. Mitásová H, Mitás L (1993) Interpolation by regularized spline with tension: I. Theory. *Implement Math Geol* 25(6):641–655
23. Mitásová H, Hofierka J (1993) Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis. *Math Geol* 25(6):657–667
24. Carrera-Hernandez JJ, Gaskin SJ (2006) The groundwater modeling tool for GRASS (GMTG): open source groundwater flow modeling. *Comput Geosci* 32(3):339–351
25. Object Raku Technology. <http://www.objectraku.com/>. Accessed 22 July 2016
26. Lindenmayer A (1971) Developmental systems without cellular interaction, their languages and grammars, parts I and II. *J Theor Biol* 30:455–484
27. Oppenheimer PE (1986) Real time design and animation of fractal plants and trees. 13th annual conference on computer graphics and interactive techniques, pp 55–64
28. Weber J, Penn J (1995) Creation and rendering of realistic trees. In: 22nd annual conference on computer graphics and interactive techniques, pp 119–28
29. Prusinkiewicz P, Lindenmayer A (1990) The algorithmic beauty of plants. Springer, New York
30. Honda H (1971) Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *J Theor Biol* 31:331–338
31. Xfrog. <http://xfrog.com/>. Accessed 20 July 2016
32. Speedtree. <http://www.speedtree.com/>. Accessed 21 July 2016
33. Speedtree 7 for games (2016) <http://www.speedtree.com/tree-modeler.php>. Accessed 21 July 2016
34. SpeedTree SDK (2016) <http://www.speedtree.com/middleware-integration-sdk.php>. Accessed 21 July 2016
35. Onyx Computing. <http://www.onxtree.com/>. Accessed 19 July 2016
36. Marlin Studios. <http://www.marlinstudios.com>. Accessed 25 July 2016
37. e-on software. <http://www.e-onsoftware.com>. Accessed 26 July 2016
38. VTP. <http://www.vterrain.org/Plants/plantsw.html>. Accessed 26 July 2016

Chapter 4

Data Fusion for Evaluation of Woodland Parameters

Abstract The airborne shooting provides different types of information including the Light Detection And Ranging (LiDAR) data, hyperspectral/multispectral/digital photography shooting data, and additional information about parameters of a shooting. The proposed generalized flowchart for data fusion of the airborne laser scanning, imaging spectroscopy, and imaging photography involves three levels of processing, i.e., a preprocessing level, a level of canopy modelling and evaluation, and a level of segmentation of individual trees, including the measurements and computation of indirect parameters. Data fusion promotes the accurate direct and indirect measurements. However, difficulties of image stitching because of the parallax effect lead to the distortions between the ground truth 3D LiDAR coordinates and 2D coordinates in imagery. The proposed method for fusion of the LiDAR and digital photography data provides an accurate segmentation of individual tree crowns in order to receive the reliable biomass measurements. The boundaries and textures were improved in optical images by application of shearlets and higher-order active contour model. This permits to evaluate tree crowns in a plane more accurately. The obtained area measurements of the tree crowns are promising and coincide with the expert estimations, providing the accuracy 92–96%.

Keywords Airborne laser scanning • LiDAR • Digital photography data • Data fusion • Woodland parameters • Higher-order active contour • Shearlets • Forest inventory

4.1 Introduction

Remote sensing inventory of forest becomes the conventional technique with various purposes. The temporal changes of biomass have the ecological and economical significance as well as the spatially-explicit mapping of a forest biomass. The most methods are based on rich empirical datasets and cannot provide good analytical models. The attempts to predict the Canopy Height Distributions (CHDs)

and the Canopy Height Quantile functions (CHQs) deserve a special attention. In previous researches, the Canopy Height Profile (CHP) was estimated using the scanning LiDAR images of canopies by echo recovery [1–3].

Due to a possibility to cover large territories, the Airborne Laser Scanning (ALS) remains the most promising approach for forest inventory and monitoring. However, this is very expensive and meteorological-dependence procedure with some statistical inaccuracies, among which the following factors ought to be mentioned:

- A laser pulse can strike or miss the tops of the trees.
- A laser pulse may penetrate into a tree canopy prior to being reflected.
- It is unclear, whether a laser pulse has struck the ground due to an unknown Earth's relief.

At the same time, small footprint LiDAR data, estimating biomass of individual trees, have yielded better results in monitoring of small territories [4].

Two classes of remote sensing systems are available. The passive systems detect the radiation subsequently from an external source of energy, while the active systems generate and direct the energy toward a target and detect the radiation. The LiDAR is an active system that emits and detects the reflected discrete pulses of a laser light. Unlike the RADio Detecting And Ranging (RADAR), the LiDAR data depends strongly from the meteorological impacts, such as cloud, rain, or snowfall [5]. A possibility of the LiDAR pulses to penetrate tree crowns and other vegetations makes the application of this device very popular and indispensable in the remote sensing of a landscape. The discrete-return LiDAR data are stored as a set of coordinate triples (x , y , z or latitude, longitude, and elevation, respectively). The target objects are computed from the time difference between the laser pulse being emitted and returned, the angle of a pulse, and the absolute location of the sensor on or above the Earth's surface. Sometimes, the auxiliary variables, such as the intensity of the returned pulses and scan-angle, are utilized [6, 7].

Also, the remote sensing of forest and agricultural crops can be implemented by the Unmanned Airborne Vehicles (UAVs) that have the potential preferences against the airborne shooting, especially during the season changes. The UAV scanning can be carried out with a cloud cover sky. Moreover, the UAV images with a higher resolution can be delivered to the end-user in real time. The supplemented techniques of airborne, UAV, and terrestrial scanning provide rich information about forest, and the obtained data can be used in classification and verification techniques. Usually the UAVs are equipped by the on-board thermal and narrowband multispectral imaging sensors, budgetary variant of which require the geometric and radiometric calibrations [8–10].

The rest of this chapter is organized as follows. Related work is discussed in Sect. 4.2. The generalized flowchart for data fusion is represented in Sect. 4.3. Section 4.4 describes a representation of airborne LiDAR and digital photography

data. Section 4.5 presents a method for crown and trunk measurements based on accurate segmentation, using the shearlet transform and high order contour model. Experimental results are situated in Sect. 4.6 concluded by Sect. 4.7.

4.2 Related Work

In literature, two main approaches of forest inventory based on the ALS data prevail. In the Area-Based Approach (ABA), the regression analysis, the nearest neighbor classification, and the Random Forests (RFs) are applied [11–13]. An approach of the Individual Tree Detection (ITD) meets difficulties with the accuracy segmentation of a tree and the expansive detailed modelling [14]. The main advantage of the ITD is an obtaining of the trunk distribution series close to ground-truth. In most ITD methods, maximums in the Canopy Height Model (CHM) are used for individual tree segmentation with following estimation of tree parameters. As mentioned in [15], the Terrestrial Laser Scanning (TLS), the ALS, and the Vehicle-based Laser Scanning (VLS) can be used for biomass estimation and dimensions measurement at the ITD level.

Zhao et al. [16] suggested the scale-invariant biomass models as the linear and nonlinear functional models. These models are based on a height distribution of the canopy and individual trees. The canopy is simulated as a height distribution, and the individual trees are modelled by distributions of the tree height, diameter, crown base height, and crown projection area in a framework of the tree species. Such models ought to be scale-invariant. Let $p_1(h)$ and $p_2(h)$ be two CHDs for two contiguous regions with areas of a and b . Zhao et al. proposed a linear functional model of the combined regions predictor $p_{12}(h)$ in a view of Eq. 4.1, where h is a height.

$$f[p_{12}(h)] = f\left[\frac{a \cdot f[p_1(h)] + b \cdot f[p_2(h)]}{a + b}\right] \quad (4.1)$$

Functions $f[p(h)]$ in numerator of Eq. 4.1 can be represented as an inner product in a functional space due to linear functional of $p(h)$:

$$f[p(h)] = \langle p(h), K(h) \rangle. \quad (4.2)$$

The most common inner product in Eq. 4.2 may be rewritten in a view of a scale-invariant Above-Ground Biomass (AGB) model, providing by Eq. 4.3, where M is the AGB per unit area, $K(h)$ is an unknown function that parameterizes the model $f_k[\cdot]$.

$$M = f_k[p(h)] = \langle p(h), K(h) \rangle = \int_{h \geq 0} p(h) \cdot K(h) dh \quad (4.3)$$

The functional form of $K(h)$ is determined from a training dataset. A function or curve $p(h)$, $h \geq 0$, is defined from the obtained LiDAR data, and its functional values are available at a discrete set of points.

In order to achieve the real scale-invariant model, Zhao et al. [16] introduced a non-linear functional with the CHQs. A quantile function $Q(q)$, $0 \leq q \leq 1$, is an inverting function of $F(h)$

$$F(h) = \int_0^h p(h')dh',$$

that is $Q(q) = F^{-1}(q)$. Then Eq. 4.3 may be rewritten as Eq. 4.4, where h_{\max} is a maximum height.

$$M = K(h_{\max}) - \int_0^1 q \cdot \frac{dK(Q(q))}{dq} dq \quad (4.4)$$

The experiments show that good measurements of the biophysical variables, such as the height, crown width, crown base height, and component biomass, are provided by the LiDAR scanning with resolution less than 5.0 m.

Hollaus et al. [17] studied one of the evident approaches for determination of a tree height in the mountains as a data subtraction in the Digital Surface Model (DSM) and the Digital Terrain Model (DTM). The received results were evaluated by empirical function of Keylwerth [18], Eq. 4.5, where h is a tree height in meters, a_0 is a variable coefficient, a_1 is a constant coefficient, dbh is a tree diameter at breast height in centimeters.

$$h = e^{a_0 + \frac{a_1}{dbh}} + 1.3 \quad (4.5)$$

The expected standard deviations of the estimated heights range between 1.3 and 1.9 m.

The 3D forest canopy as a processing result of the TLS data including the individual trees' simulation was developed by Côté et al. [19]. Here, 3D forest canopy is considered as the spatial heterogeneous system in aspects of the standing trees as the canopy, branches, and leaves within crown or needles within conifer shoots and temporal dynamic system in scale from seconds/minutes (wind simulation), days (meteorological impacts), and years (season impacts).

Numerous approaches for a canopy reconstruction are based on the required accuracy and a level of detail according to the investigation purposes. Some tasks require a simulation at local scales in order to characterize the geometry of plants [20]. Other applications estimate a tree structure down to a leaf cluster level [21]. A tree representation includes the decomposition of the tree components, their topological connectivity, and geometric description of shapes. This process can be

organized relatively easily for the terrestrial observations. To receive the good similarity result for the airborne surveillance is very difficult but this is a high demanded task in a forest inventory. The ground-truth simulation and visualization are the result of a fusion of various data, knowledge rules, and facts. Such statement requires the huge amount of forest observations and a building of the realistic seasonal vegetation models.

Maltamo et al. [22] introduced the main individual tree quality attributes and the Root-Mean-Squared Errors (RMSEs) for a trunk volume, a proportion of sawlogs, the Diameter-at-Breast Height (DBH), a height, a crown height, and a height of the lowest dead branch. The bias and the RMSE are calculated in relative units and percents by Eqs. 4.6–4.7, where n is a number of measurements, y_i is an obtained value of the parameter, \hat{y}_i is a predicted value of the parameter, \bar{y} is a mean of values [23].

$$BIAS = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)}{n} \quad BIAS\% = 100 \cdot \frac{BIAS}{\bar{y}} \quad (4.6)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad RMSE\% = 100 \cdot \frac{RMSE}{\bar{y}} \quad (4.7)$$

The individual tree segmentation from 3D point cloud is very difficult task due to the restricted sources of shooting, mixed tree species, and overlapping in dense stands. A literature review indicated the following steps of the generalized algorithm:

- Classification the raw LiDAR point data into the ground and above-ground points.
- Generation of the DTM.
- Normalization of the vegetation point cloud values by a subtracting the ground points from the LiDAR point cloud.
- Computation of tree heights through the elevation values of points, indicating the heights from the ground to the points.
- Individual trees segmentation, using density of 3D point cloud.
- Improvement of the tree segmentation results, using previously known information about the tree species and additional data from the hyperspectral/multispectral/digital photography shooting.

The numerous techniques are adapted for segmentation of the individual trees using the LiDAR data. One can mention the region growth from the highest laser pulses of the CHM [24, 25], the watershed segmentation [26], the local maximum extraction [27], the normalized cut [25], among others. Better results are obtained using a fusion of the LiDAR and optical data, for example, the adaptive multiscale filtering [28] or Gaussian filtering [29].

The original graph-based segmentation method that detects the individual tree crowns, using the airborne LiDAR data, was proposed by Strîmbu and Strîmbu [30]. They prepared a hierarchical LiDAR data structure from the horizontal layers of vegetation and created the topological structure of the forest. These authors declared that their method has a wide range of procedures, including the crown segmentation, calculation of an average drop in levels between any two parent-child patches, and indication of the tree species composition in a forest.

The ALS point cloud can be regarded as a multimodal distribution, where the local maximums both in density and height correspond to the crown apices. Ferraz et al. [31] investigated the ability of the Mean Shift (MS) algorithm to extract the local maximums in the point cloud. Due to the complexity of the forest stands, a single kernel bandwidth is unsuitable, and Ferraz et al. proposed to apply a bottom-up iterative method based on an adaptive MS algorithm that sequentially finds the individual vegetation features.

The MS algorithm has been primarily applied to the image segmentation [32]. In study of Ferraz et al. [31], it was used for segmenting of 3D point cloud. Also a method for estimating the Probability Density Function (PDF) of a random variable X that is distributed in a d -dimensional space R^d (kernel density estimation) helped to estimate the possible displacement. Each point X_i contributes to the PDF based on its distance from the center of the volume, where the data are distributed. The estimated PDF is computed by Eq. 4.8, where n is the number of samples of a random variable, K is a chosen kernel function, and h (called the bandwidth) is a smoothing parameter that determines the contribution of each sample. K is a non-linear function of the distance from the data points to X .

$$\hat{f}_{h,K}(X) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{X - X_i}{h}\right) \quad (4.8)$$

Ferraz et al. defined a radially symmetric kernel that satisfies

$$K(X) = c_{k,d} \times k(\|X\|^2),$$

where $c_{k,d}$ is a normalization constant, which makes K integrate to one, and k is called the kernel profile. The algorithm tries to determine the local maxima of the density function $\hat{f}_{h,K}(X)$, which correspond to the zeros of the gradient $\nabla \hat{f}_{h,K}(X) = 0$.

Ferraz et al. [31] analyzed preliminary the biophysical characteristics of stands in a modelling area, covering 9 km² that is located near the city of Águeda in northwest Portugal (40° 36'N, 8° 25'W). As a result, the vertical structure of a stand was described by seven height classes (0–0.6 m, 0.6–1 m, 1–2 m, 2–4 m, 4–8 m, 8–16 m, and >16 m) that could be aggregated in situ to better represent the vegetation strata. This method provides the genuine 3D segments corresponding to the individual vegetation features of the main forest layers: a ground vegetation, an understory, and an overstory.

The segmentation of coniferous forests was developed by Li et al. [33]. This method works well if a relative spacing between trees exists, and under the assumption that a horizontal spacing at the tree tops is larger than a horizontal spacing at the tree bottoms. To overcome the challenge of particularly overlapping trees, a classifying procedure of the points from the highest to the lowest was proposed. Points with spacing larger than a fixed or adaptive threshold are excluded from the target tree, while points with spacing smaller than the threshold are classified according to horizontal profile of the coniferous tree shape. The authors consider that their algorithm is flexible, and the classification rules can be modified, depending on the characteristics of other trees.

Lu et al. [34] developed a method that is based on two properties from the LiDAR point cloud: the intensity of pulses in each point and the topological relationship between points. First, the points of a tree trunk are extracted and labeled. Second, the points of small branches and withered leaves are detected. Third, the remaining points are allocated due to a topological relationship. The authors concluded the algorithm that was applied to segment the trees in a deciduous broadleaf forest at the Shavers Creek Watershed in Pennsylvania detected 84% of the tree (recall), 97% of the segmented trees were correct (precision), and the overall F-score was 90%.

One would like to point a growing number of the Very High Resolution (VHR) multispectral satellite images applicable for estimation of a tree height, a trunk diameter, a density, a basal area, and a trunk volume or biomass [35–38]. A sub-metric spatial resolution 0.5–1 m in panchromatic band or 2–4 m in multispectral bands provides a computation of various spectral and Haralick's texture features of forests [39]. At present, a texture analysis of the VHR satellite images is often based on wavelet or the Fourier transformation, variogram, or the Grey Level Co-occurrence Matrix (GLCM) method. Beguet et al. [39] used two first order texture features (mean and variance) and eight second order GLCM (energy, entropy, correlation, inverse difference, inertia, cluster shade, cluster prominence, and Haralick's correlation) texture features. Panchromatic image and images of four multispectral bands [Blue (B), Green (G), Red (R), Near-InfraRed (NIR)] were the initial material. Various combinations of GLCM parameter values under different parameters (radius of the moving window, displacement, orientation, and quantification level number) were tested. The experiments show that it would be possible to identify several classes of crown diameters ranging from 1 m or less to about 11 m.

The Local Maxima (LM) filtering and the Marker-Controlled Watershed Segmentation (MCWS) were applied by Zhang et al. in research [40]. The main idea consists in detection of individual trees from the ALS data by integrating the LM and the MCWS low-level image processing techniques into a high-level probabilistic model. The proposed model assigns each pixel to vegetation, background, or center of a tree. The issue of crown segmentation with symmetric and asymmetric tree crowns, including their overlapping, is considered in details. However, these authors approximated a projection of the tree crowns by circles only that can be considered as very rough approximation.

Kalliovirta and Tokola [41] proposed the models for the DBH and the age models in a form of Eqs. 4.9–4.10, where d_{DBH} is a diameter at breast height, mm, h is a height, dm, d_{CRM} is a maximum of crown diameter, dm, α is a stand variable from database or aerial photograph, ε is an error.

$$\begin{aligned}\sqrt{d_{DBH}} &= f(\sqrt{h}) + \varepsilon \\ \sqrt{d_{DBH}} &= f(\sqrt{d_{CRM}}) + \varepsilon \\ \sqrt{d_{DBH}} &= f(\sqrt{h}, \sqrt{d_{CRM}}) + \varepsilon \\ \sqrt{d_{DBH}} &= f(\sqrt{h}, \sqrt{d_{CRM}}, \alpha) + \varepsilon\end{aligned}\tag{4.9}$$

$$\begin{aligned}\ln(age) &= f(\ln(h)) + \varepsilon \\ \ln(age) &= f(\ln(d_{CRM})) + \varepsilon \\ \ln(age) &= f(\ln(h), \ln(d_{CRM})) + \varepsilon\end{aligned}\tag{4.10}$$

In general, models providing by Eqs. 4.9–4.10 can be rewritten as Eqs. 4.11–4.12, where $\text{var}(\varepsilon)$ is a variance of error ε .

$$d_{DBH} = f(\cdot)^2 + \text{var}(\varepsilon)\tag{4.11}$$

$$age = \exp[f(\cdot)] * \left(1 + \frac{1}{2} \text{var}(\varepsilon)\right)\tag{4.12}$$

For verification models (Eqs. 4.11–4.12), data from Finnish national forest inventory sample plots located throughout the country were used. The young trees with height <3 m were not included in the results of forest inventory. However, an inventory of sapling stand is another matter.

These models were specified by Kankare et al. [42] with empirical coefficients suitable for investigation of Evo territory, Finland. The model of the DBH has a view of Eq. 4.13.

$$\sqrt{d_{DBH}} = f(\sqrt{h}, \sqrt{d_{CRM}}) = -7.04657 + 1.3058\sqrt{h} + 0.5462\sqrt{d_{CRM}}\tag{4.13}$$

The sloping terrain can introduce a systematic error in computation of a tree height because the distance from the highest crown return to its projection on the DTM is used [43, 44]. Vega et al. [45] proposed the original approach, including three steps:

- The introduction of a new point cloud normalization approach to improve both crown identification and characterization.
- The use of absolute maxima instead of local maxima to locate tree apices.

- The introduction of a multi-scale and multi-criteria framework to optimize the efficiency of tree detection.

The simplified theoretical model one can find in the research [46].

The backbone of forest monitoring is the periodic observations of tree foliage in the forest canopy layer and its evaluation. Biomass can reduce under some stress factors and natural/human damages. Brandtberg [47] investigated a distribution of the LiDAR point cloud for individual trees under leaf-off and leaf-on conditions. Brandtberg discovered that the R-distribution of small footprint leaf-off LiDAR data is typically extremely positively skewed, while the corresponding leaf-on LiDAR data is negatively skewed, due to the differing likelihood of returned laser strength. Also Brandtberg concluded that a transparency is a salient characteristic of a leaf-off deciduous tree, while it is non-salient for a green (“leaf-on”) coniferous tree.

Polewski et al. [48] investigated very important part of forest ecosystems—the downed dead wood. Their method for detecting individual tree trunks from the unstructured ALS point cloud is suitable for both discrete and full waveform data. The output data is a list of subsets of the original points that correspond to individual detected fallen trees. It is assumed that the reference data in a form of accurate fallen trunk positions, lengths, and diameters are available. First, the trunk segments in the point cloud are detected. Second, these segments are merged to produce the individual fallen trees.

Based on crown condition variables, the Leaf Area Index (LAI) can be computed as a factor of a foliage density. The LAI is widely used in agricultural and ecological studies as a necessary parameter in agricultural, ecological, climatic, and hydrological models. The satellite data cannot provide the reliable LAI values. It is possible to speak about the LAI estimation at different spatiotemporal scales during airborne, UAV, and terrestrial LiDAR scanning. The special task deals with a crop monitoring, using multispectral optical shooting.

For the LAI estimation, various measures are proposed based on direct methods as the ground-based approaches [49, 50] and indirect methods as the airborne approaches [51]. The direct methods are time consuming and labor-intensive, while the indirect methods are costly and meteorological dependently. However, the LAI remote sensing inverse models are promising. This direction is developed and can be classified into three categories:

- The empirical-based model is based on statistical equations between the Vegetation Indices (VIs) and the in situ LAI for large territories.
- The physical process model or the Radiative Transfer (RT) uses the theory of geometrical optical and radiative transfer considering observation angles, the canopy structure parameters, and background effects.
- The hybrid model is a result of combining the empirical and physical approaches. It provides the integrated LAI inverse model.

The main modifications of the LAI, such as total LAI, projective LAI, silhouette LAI, effective LAI, and true LAI, are discussed in details by Zheng and Moskal in review [52].

Literature review demonstrates various approaches for evaluation of the woodland parameters. The generalized flowchart for data fusion in order to provide the direct and indirect measurements is described in the following Sect. 4.3.

4.3 Generalized Flowchart for Data Fusion of Airborne Laser Scanning, Imaging Spectroscopy, and Imaging Photography

A set of devices is chosen according to the solving task and availability of equipment for airborne shooting. The ALS data is used always due to a necessity to obtain the ground-truth coordinates of impenetrable and penetrable surfaces. The Imaging Spectroscopy (IS) and/or the Imaging digital Photography (IP) data (as grey-scaled or colored texturing images) play the significant role in classification/recognition/identification stage. In spite that Fig. 4.1 depicts all three types of data, usually the IS or the IP data are collected and analyzed in conjunction with the ALS data [53, 54].

A generalized flowchart includes three levels of data processing. Level 1 is a level of primary data processing with additional data cleaning, interpolation, and artifacts removal. As one can see from Fig. 4.1, the IS and the IP data are two-dimensional, while the ALS data are three-dimensional. This directly creates the problems with a data fusion. The IS/IP includes an imposed grid, and the ALS data are also leaded to an imposed grid with the same sizes in a current layer of vertical direction OZ. Therefore, the IS and the IP data bring information about surface only, while the ALS data involve information layer by layer in a vertical direction, and it is possible to build 3D surface and terrain models dealing with 3D digital canopy model creation at Level 2.

The goal of Level 2 is to build 3D canopy model and estimate canopy vegetation indexes, in general. The input data of flowchart provide the evaluation of geometric parameters of canopy and individual trees. It is possible to speak about the statistical and physical approaches in data fusion [55]. Statistical approach analyzes the evident measurements, e.g., reflectance and/or height values. It is conventionally based on regression models and recognition procedures of imaging. Physical approach cannot apply directly to fuse the ALS and the IS data. It uses the Radiative Transfer Models (RTMs) in order to estimate pigments, nitrogen, and water content [56]. Physical models for the LiDAR data appeared recently, and now they are applied for small footprint ALS data only. Extraction of individual trees from airborne data is very complicated task and cannot be solved without the optical information in a hope to recognize a type of trees and use the corresponding generated model of a tree. In such manner, a transmission to detailed scales becomes possible.

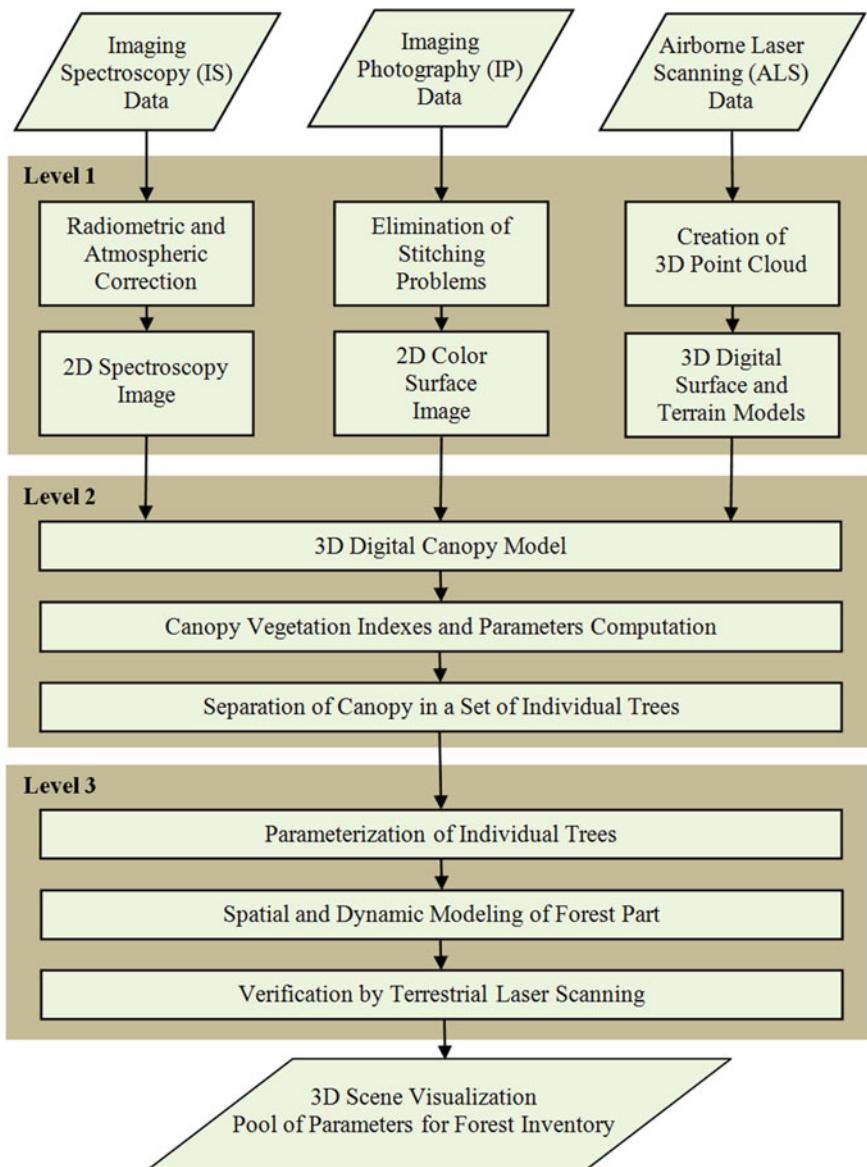


Fig. 4.1 Generalized flowchart for fusion of ALS, IS, and IP data

Thus, a parameterization of the individual tree with evaluation of local parameters is implemented at Level 3. Accuracy parameterization of trees is impossible without their classification in order to predict the main parameters of tree species. The Support Vector Machines (SVMs) and the Linear Discriminant Analysis (LDA) are often used for these purposes. In recent years, the Random Forest

(RF) technique was approved in several experimental studies with better results [57]. The spatial modelling of forest part (with some meteorological and luminance effects) means the detailed visualization of a forest scene during a short time interval, for example, a day. The dynamic modelling of forest part requires the persistent and continue observations and measurements during a long time interval, for example, a year. The final stage is a verification of the obtained models by the TLS, although the TLS does not devoid of own drawbacks and measurement errors.

The main woodland parameters are described shortly in Tables 4.1 and 4.2.

The following discussions will adhere to the concept of generalized flowchart, depicting in Fig. 4.1.

Table 4.1 The main woodland parameters for canopy estimation

Parameter	Short description
Canopy cover	The function is obtained by the direct measurements as the first signals reflected by a target relatively to the first signals reflected by the ground
Canopy height	One of the main parameters for different types of forest (temperate, boreal, and tropical). It is determined from the measurement of the difference between the highest point of a tree and the Earth's surface. The minimum threshold value is near one meter
Canopy height profile	The vertical canopy distribution for any deciduous forest. It defines the sparseness rate of a plantation
Vertical distribution of canopy material	The distribution function for evaluation of the above-ground biomass, predicting a state of forest and determining an age of forest part
Canopy volume profile	This parameter is obtained by modelling. It provides information about a vertical foliage profile and the qualitative and quantitative differences between tree ages of the given tree species
Above-ground biomass	The parameter is calculated from the tree height measurements. It is used as an estimation measure for mixed coniferous/deciduous, dense boreal and tropical forests both in flat ground and mountains
Basal area	Cross sectional area of a trunk at the DBH i.e. 1.37 m
Mean of crown volume	The crown volume is computed knowing the canopy volume, tree density, and tree species
Mean of trunk diameter	A trunk diameter depends strongly from a tree height and species and is calculated by empirical equations
Mean of trunk volume	The trunk volume is inferred according to the mean trunk diameter and the mean three height
Other parameters	There are a density of large trees, also thermal, optical, and radar data for the LAI evaluation and classification of tree species

Table 4.2 The main woodland parameters for estimation of individual tree

Parameter	Short description
Tree height	The parameter is obtained by direct measurements as the first signals reflected by a tree relatively to the first signals reflected by surrounding
Vertical distribution of crown	A distribution function for evaluation the above-ground biomass of a tree
Diameter at breast height	This parameter is calculated by several models
Trunk volume	A trunk volume cannot be directly measured from the ALS data. It can be calculated indirectly using the tree height, crown volume, maximum crown area, and crown height
Shape of crown	A shape of crown is classified by geometrical shapes, such as cone, paraboloid, or hemisphere, and depends on the tree species strongly
Density of crown	This parameter can be estimated by the intensity of laser pulses
Crown volume	This parameter is computed knowing the tree height, vertical distribution, and shape of crown
Type of leaves	This parameter is defined from the texture analysis of optical or photography data and helps to classify the tree species
Tree age	The parameter is predicted from the vertical distribution of crown and tree species
Other parameters	Additional thermal, optical, and radar data can be used for the LAI evaluation and classification of species

4.4 Representation of Airborne LiDAR and Digital Photography Data

According to flowchart depicted in Fig. 4.1, a global point cloud ought to be classified in the Regions Of Interest (ROI)—the individual trees using visual data and utilizing the well-known methods of texture and fractal analysis [58]. Let a point cloud of a single detected ROI be represented by a set $ROI_j = \{(x_1, y_1, z_1, l_1), (x_2, y_2, z_2, l_2), \dots, (x_i, y_i, z_i, l_i), \dots, (x_k, y_k, z_k, l_k)\}$, $j = 1, \dots, N_{ROI}$. The LiDAR points are the reflections of the laser beam at positions (x_i, y_i, z_i) with intensity value of the returned signal l_i . (Not all LiDAR systems provide additional effective parameter for classifying crown and trunk points—the pulse width.) This disadvantage can be removed by application of digital photography data. The LiDAR data are captured by system capable of scanning in three directions—forward, downward, and backward and providing 200–500 thousand pulses/measurements per second (Fig. 4.2). In spite of 30% beams loss through a forest scanning, the highest density of laser scanning pulses achieves 1 pulse per 5–7 cm on the Earth’s surface. Thus, the measurement accuracy of vegetation morphostructural elements, such as the trunks and crowns of trees and other ground objects, is about $\pm 5\text{--}10$ cm.

The tree crown is a penetrable object for a laser scanning that provides two or three back signal responses as it is depicted in Fig. 4.3 schematically. Notice that 3D coordinates of the LiDAR points are the ground truth coordinates.

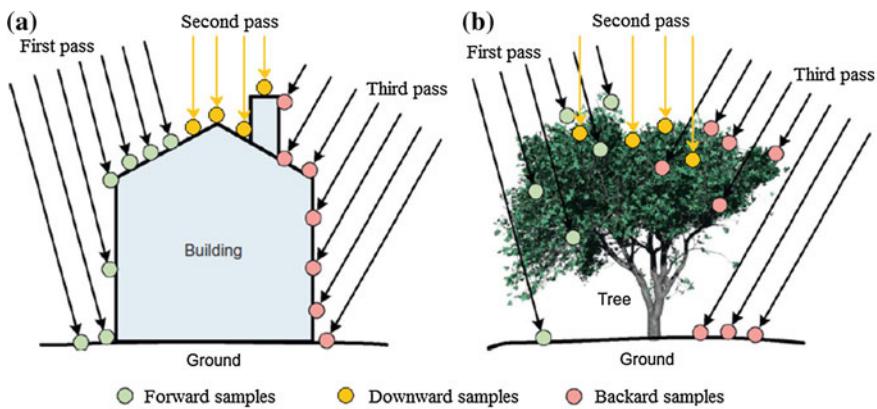


Fig. 4.2 Forward, downward, and backward directions of LiDAR scanning: **a** impenetrable object, **b** penetrable object

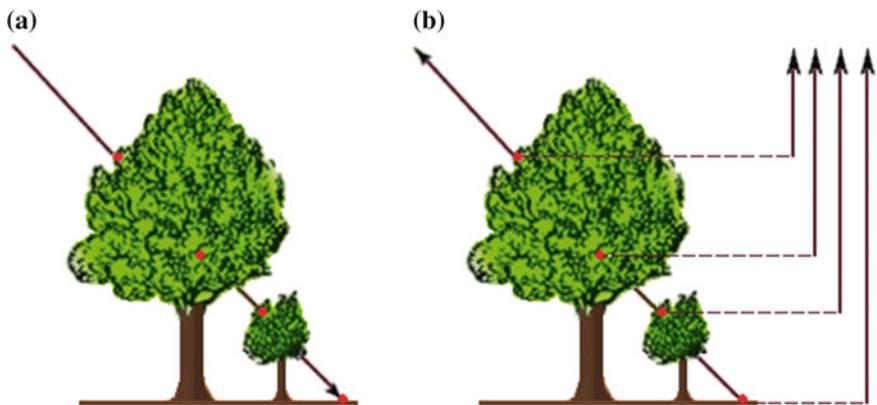


Fig. 4.3 Laser pulses (red dots are the reflected dots): **a** direct pulse, **b** back responses

Due to the parallax effect during an imaging shooting, some artifacts can appear after an image stitching. An approach based on computing of the feature descriptors, line descriptors, and region descriptors in local areas of stitching is reasonable [59, 60]. Full data restoration is impossible, thus this task can be solved as a task with missing data, using the texture parameters and stochastic property of forest's vegetation. This process is illustrated in Fig. 4.4.

Using a density of the LiDAR points, one can judge about a type of known object. Therefore, the LiDAR data help to identify, not to recognize natural or man-caused object. A recognition task is solved using the airborne digital photography data that can be the multispectral monochrome/color, digital color stereo photographs stitching or large-sizes aerial image [61].

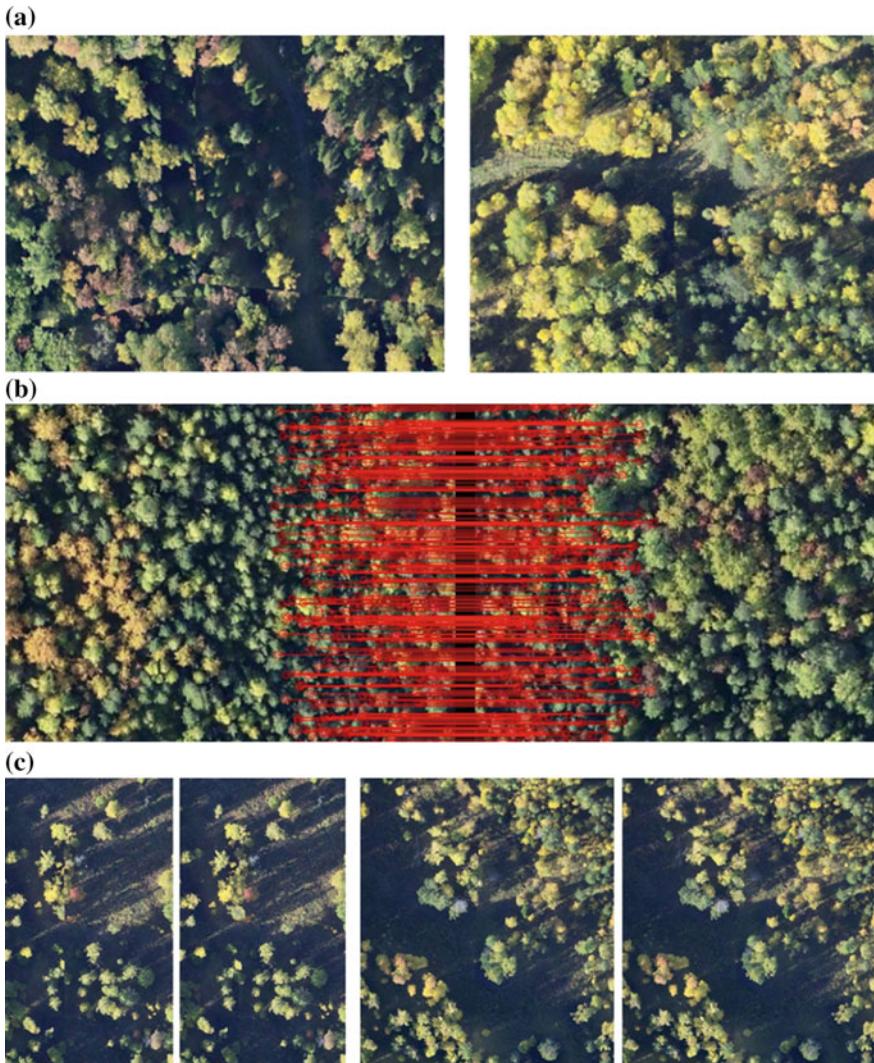


Fig. 4.4 Stitching of photography data: **a** artifacts of stitching, **b** stitching based on detection of feature points and the RANdom SAmple Consensus (RANSAC) algorithm, **c** results of stitching (*left* stitching with artifacts, *right* stitching without artifacts)

However, due to incorrectness of complicated stitching of aerial photographs the essential bias exists between 3D laser coordinates and 2D image coordinates. For this purpose, a procedure for accurate matching of the LiDAR and image coordinates in the local ROI based on a moving cluster was developed. During several iterations as it is shown in Fig. 4.5, the feature points, representing visual coordinates in the proposed center of a tree crown, are moved to the region with higher

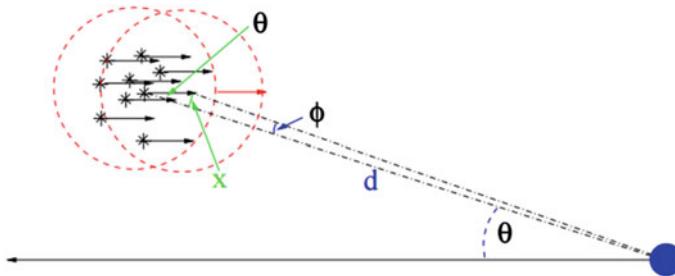


Fig. 4.5 Illustration for accurate matching of laser and image coordinates

values of the Z laser coordinates in the same local ROI. Notice that usually a forest appraiser utilizes a single highest point for a crown top detection. The proposed procedure uses three highest points in order to detect the coordinates of a crown top more accurately.

Then the laser points ought to be separated into the ground, trunk, and crown sets in a vertical direction. The ground points have a small height, not more 1 m in the DTM. The remaining laser points are divided into the trunk and crown points. The trunk detection is predictable due to the typical shapes of trunks, while a crown includes branches and foliage. In the case of individual tree, the task is solved by dispersion calculation of laser points in m layers, in which a virtual tree height is split by 0.5 m. Layers with large values of dispersion are considered to a trunk. Also a heuristic rule exists, according to which 0.15% of tree points correspond to a trunk. Such procedure can be enforced by hierarchical clustering, the RANSAC algorithm, and 3D reconstruction of a trunk. Difficult situation occurs, when several trees are located close each other, and the task cannot be solved without a tree model corresponding to the given type of a tree. Experiments show that the proposed procedure is well for the grown trees separation, when the underwood is rejected. Figure 4.6 depicts a layer cut and examples of a crown projection for the individual tree and trees overlapping.

Method for crown and trunk measurements of individual trees will be discussed in detail in Sect. 4.5.

4.5 Method for Crown and Trunk Measurements

As mentioned in [15], the apparent volumes of the tree crowns V can be estimated from a crown diameter dc and a crown height hc by applying three simple geometric shapes provided by Eq. 4.14, where k is a coefficient defining a geometrical shape, $k = 2$ for cone, $k = 3$ for paraboloid, and $k = 2 \cdot dc/hc$ for hemisphere.

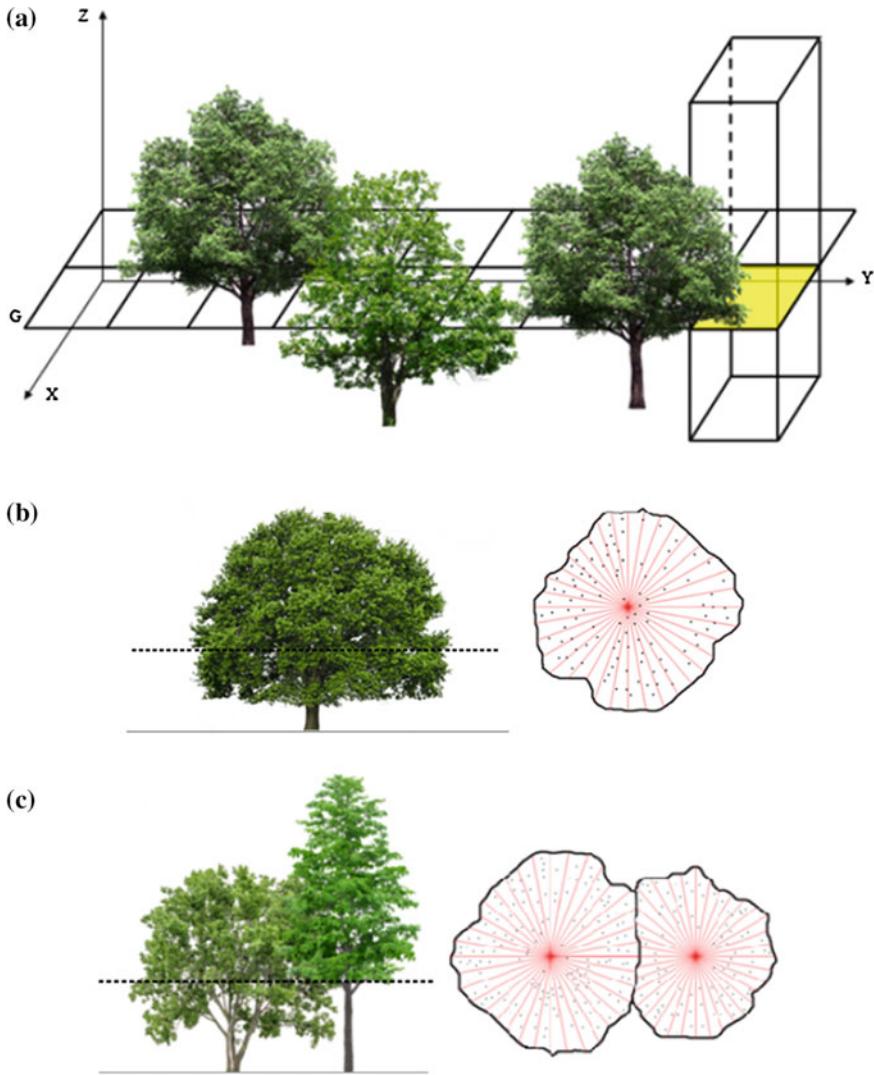


Fig. 4.6 Obtaining of crown projections: **a** separation of canopy by layers, **b** example of crown projection for individual tree, **c** example of crown projection for trees overlapping

$$V = k \cdot \frac{\pi \cdot dc^2 \cdot hc}{24} \quad (4.14)$$

Estimations, using Eq. 4.14, are very approximate. Realistic trees possess the crowns that cannot be approximate by simple geometric shapes. Thus, the accurate crown segmentation is crucial in the forest inventory. In this chapter, a method for

crown segmentation in a plane based on a fusion of the LiDAR and visual data with the following accurate contour segmentation is developed.

The boundaries of tree crowns can be approximate by geometric shapes; however, the accuracy of such approximation is not high. For segmentation of the arbitrary contours, a framework of active contour models is used reasonably. Such models are based on a specific energy function moving and deforming an initial contour that ought to be minimal, when a contour is delineating the object of interest. Different energy functions have been proposed, however, two main groups are distinguished in the active contour framework. First group represents by a parameterized curve called as snakes, when a contour generally converges towards edges or ridges in an image. Second group based on the region properties, such as an intensity, (geometric active contours) apply the energy functions, where a contour is represented using the geodesic level sets. In this research, first approach was implemented.

For low resolution images, one can recommend a contour and texture improvement based on a shearlet transform (Sect. 4.5.1). The boundaries of tree crowns are extracted by a higher order active contour model (Sect. 4.5.2). The crown measurements in 2D plane and 3D space are discussed in Sect. 4.5.3.

4.5.1 Shearlet Transform

The necessity of a noise-robust edge detector leads to the complicated methods based on isotropic and anisotropic Gaussian kernels, the Gabor wavelets, and also the contourlet, tetrolet, and shearlet transforms [62], among others. Each wavelet transform fulfills its own task. In the given case, it is required to preserve the edges and textures in low resolution images, when the noise suppression has become more relevant.

The shearlet transform $SH_{\psi}f(\cdot)$ of function $f(\cdot)$ is determined by Eq. 4.15, where a , s , and t are the scale, shear, and translation parameters in the Euclid coordinates, respectively, $t \in \mathbf{R}^2$.

$$SH_{\psi}f(a, s, t) = \langle f(a, s, t), \psi_{a,s,t} \rangle \quad (4.15)$$

In the case of continuous 2D space, the shearlets are defined by Eq. 4.16, where $\mathbf{M}_{a,s}$ is a transform matrix, ψ is a generating function, x is a coordinate

$$\psi_{a,s,t} = |\det \mathbf{M}_{a,s}|^{-\frac{1}{2}} \psi(\mathbf{M}_{a,s}^{-1}x - t), \quad (4.16)$$

where matrix \mathbf{A}_s is an anisotropic dilation, matrix \mathbf{B}_s is a shearing matrix

$$\mathbf{M}_{a,s} = \begin{pmatrix} a & \sqrt{as} \\ 0 & \sqrt{a} \end{pmatrix} = \mathbf{B}_s \mathbf{A}_s \quad \mathbf{A}_s = \begin{pmatrix} a & 0 \\ 0 & \sqrt{a} \end{pmatrix} \quad \mathbf{B}_s = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}.$$

The discrete form of the shearlets is obtained by the following replacements: $a = 2^{-j}$, $s = -l$, where $j, l \in \mathbb{Z}$, and $t \in \mathbb{R}^2$ is replaced by points $k \in \mathbb{Z}^2$. Then Eq. 4.16 is transformed to a discrete form (Eq. 4.17).

$$\psi_{j,l,k} = |\det \mathbf{A}_0|^{\frac{j}{2}} \psi(\mathbf{B}_0^T \mathbf{A}_0^j x - k) \quad \mathbf{A}_0 = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix} \quad \mathbf{B}_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad (4.17)$$

As a result, the images processed by a shearlet transform are more promising for a crown contour extraction of the individual trees.

4.5.2 Generic, Higher Order, and Probabilistic Active Contour Models

The parametric active contour models (snakes) are based on a compromise balance between the smoothness parameters of contour (internal energy) and object parameters in an image (external energy). Many types of the internal energy functions have been proposed, depending on the application. Kass et al. [63] were the first, who introduced the contour, deformable curve, and its energies based on a variational approach. The conventional energy function $E_{\text{snake}}(\mathbf{r})$ includes an internal $E_{\text{int}}[\mathbf{r}(\cdot)]$ and an external $E_{\text{ext}}[\mathbf{r}(\cdot)]$ energy of contour. This function moves in a spatial domain until reaches the minimum of functions defined by Eq. 4.18, where $\mathbf{r}(s):[0, 1]$ is a parameterized planar curve, $\mathbf{r}(s) = (r_x(s), r_y(s))$, s is a curve parameter.

$$E_{\text{snake}}[\mathbf{r}(\cdot)] = E_{\text{int}}[\mathbf{r}(\cdot)] + E_{\text{ext}}[\mathbf{r}(\cdot)] \quad (4.18)$$

Equation 4.18 can be rewritten by Eq. 4.19, where I is an image, α , β , and λ are real positive constants (α and β impose an elasticity and rigidity of the curve, respectively, and λ means an attraction force of contour towards the object).

$$E_{\text{snake}}[\mathbf{r}(\cdot)] = \alpha \int_0^1 \left| \frac{d\mathbf{r}(s)}{ds} \right|^2 ds + \beta \int_0^1 \left| \frac{d^2\mathbf{r}(s)}{ds^2} \right|^2 ds - \gamma \int_0^1 |\nabla I(r(s))| ds \quad (4.19)$$

The first two terms in Eq. 4.19 define a smoothness of contour (an internal energy) and the third term attracts a contour towards an object in an image (an external energy). For a given set of constants α , β , and λ , a curve C ought to minimize a function $E(\cdot)$. The image energy tends to the negative magnitude of an image gradient. Thus, a snake is attracted to the regions with a low image energy, i.e., edges. The optimization is ended prematurely after one-two hundred iterations.

The generic internal energy functions regularize an active contour by updating a contour point's location based on first and second spatial derivatives in neighboring points. In a discrete case, it means that two previous and two consecutive contour

points are used to update a location of an active contour point. The higher order energy functions generalize this concept, when all contour points within a predefined distance are applied to update a contour point. Rochery et al. [64] proposed a higher order energy function under the assumption that the contour points in each other's vicinity should have a similar normal to the curve. This energy function has a view of Eq. 4.20, where $t(\cdot)$ is a normal to active contour, $h(\cdot)$ are weights, depending on the Euclidean distance between the contour points.

$$E_{int}[\mathbf{r}(\cdot)] = - \int_0^1 \int_0^1 \langle \mathbf{t}(s), \mathbf{t}(u) \rangle h(\|\mathbf{r}(s) - \mathbf{r}(u)\|) ds du \quad (4.20)$$

Both generic and higher order models regularize an active contour using local features, i.e., only a set of the contour points. Another approach is based on a global feature extraction, i.e., all contour points. Among these methods, one can mention the probabilistic active contour models that can segment the object of interest in cluttered and noise background. A probabilistic method does not search the optimal weighting parameters but find a contour based on prior knowledge about the probability of edges. The idea of this method is to find the most likely to be the true border of an object features, especially in noisy or blurred images, containing multiple objects [65]. Let $H(\mathbf{r}(\cdot))$ be a hypothesis that a contour $\mathbf{r}(\cdot)$ delineates an object of interest. Equation 4.21 provides maximum of probability $p(\cdot)$, where $f(\cdot, \cdot)$ is a set of image features $f(x, y)$, e.g., an edge strength in a set of pixels, $\mathbf{r}^*(\cdot)$ is an optimal contour.

$$\mathbf{r}^*(\cdot) = \arg \max_{\mathbf{r}(\cdot)} p(H(\mathbf{r}(\cdot)) | f(\cdot, \cdot)) \quad (4.21)$$

Then Eq. 4.21 can be rewritten according the Bayes' rule in a logarithmic view:

$$\begin{aligned} \log \mathbf{r}^*(\cdot) &= \arg \max_{\mathbf{r}(\cdot)} \log \frac{p(f(\cdot, \cdot) | H(\mathbf{r}(\cdot))) p(H(\mathbf{r}(\cdot)))}{p(f(\cdot, \cdot))} \\ &= \arg \max_{\mathbf{r}(\cdot)} \left(\log \frac{p(f(\cdot, \cdot) | H(\mathbf{r}(\cdot)))}{p(f(\cdot, \cdot))} + \log \frac{p(H(\mathbf{r}(\cdot)))}{p(f(\cdot, \cdot))} \right), \end{aligned} \quad (4.22)$$

where internal and external probabilities are defined as follows:

$$p_{int}(\mathbf{r}(\cdot)) = \log \frac{p(H(\mathbf{r}(\cdot)))}{p(f(\cdot, \cdot))} \quad p_{ext}(\mathbf{r}(\cdot)) = \frac{p(f(\cdot, \cdot) | H(\mathbf{r}(\cdot)))}{p(f(\cdot, \cdot))}.$$

In spite of a probability nature of the landscape images, a method of probabilistic active contours requires a priori information about data and their distributions that is not possible always in the large amount. Therefore, a higher order active contour model is a reasonable approach.

4.5.3 Crown Measurements

The direct measurement of components, such as the crown or trunk, is not a straightforward procedure because of objective circumstances, e.g., the individual tree crown segmentation in a forest image is not perfect in a reason of overlapping leaves of the neighboring trees. The lower part of a tree might be covered by undergrowth vegetation. Data of airborne shooting are difficult matched with data of terrestrial laser scanner directly, without tree modelling. Also a terrestrial laser shooting provides the own measurement errors. The most reasonable approach is based on the Density of High Points (DHP) methods and estimations of a crown distribution by layers [66].

One can estimate an area of each layer by two ways: as a sum of the flat segments with vertices in the center of gravity as it is show in Fig. 4.7b and as a sum of “crown” pixels in a binary image of individual tree [67]. More complex measurement occurs in 3D space because of missing data in a vertical direction.

In the case with a sum of flat segments, the following simple computation can be recommended. Assume that the external contour of a tree projection is a closed polygonal line without the self-intersections. Let x_k, y_k be coordinates of vertices of a polygonal line in a tab order. An area of a polygon is calculated by Eq. 4.23, where if $k = n$, then $k + 1 = 1$.

$$S = \frac{1}{2} \left| \sum_{k=1}^n (x_k + x_{k+1}) (y_k - y_{k+1}) \right| \quad (4.23)$$

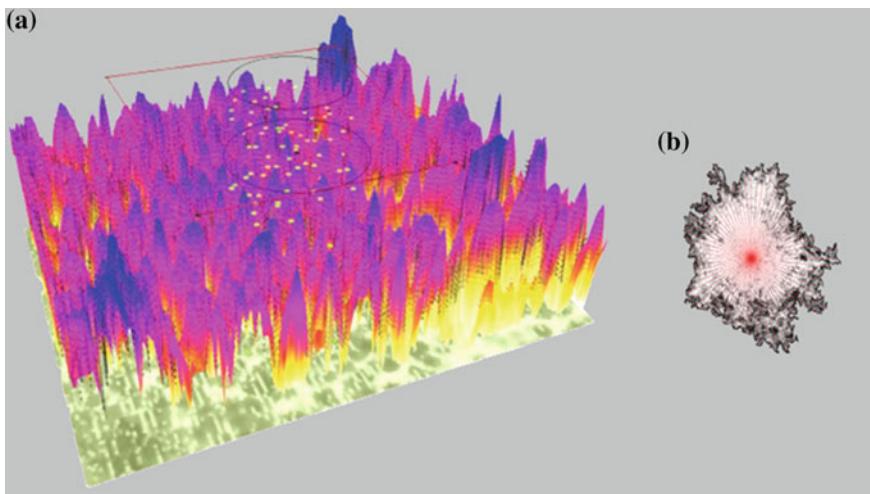


Fig. 4.7 3D visualization of laser data: **a** tree stand with imposed boundaries of measurement area and *inventory circles* with the constant radius, **b** tree crown with the *imposed lines*

The promising decision is to design 3D blobs with the centers in laser points that will fill up the inter-space between the conditional layers of a tree crown. The structure of such 3D blobs is determined by a type of tree and is based on the preliminary observations.

A procedure for calculation of a crown projection area can be applied for a crown volume evaluation. Divide 3D point cloud in a set of layers that are parallel to XOY plane with step h_l . A volume between the nearest layers can be interpreted as a polygonal prism. A volume of such polygonal prism is calculated as multiplying an area by a height h_l . Summation of the local volumes leads to the total crown volume estimation.

4.6 Experimental Results

The airborne LiDAR survey was executed from the airplane AN-2 equipped by laser scanner RIEGL Q560 with digital airborne device IGI DigiCAM that includes digital camera Hesselblad H39/mp and GPS receiver Novatel OEM 4/5. The planning and tracing of route was implemented by satellite images of near infrared range with resolution 50 cm per pixel.

3D visualization of laser data with an example of segmented tree crown is depicted in Fig. 4.7. An image of tree crown is divided into 72 segments with 5° intervals. Sometimes in a single line three edge points appear; in this case only the farthest point is considered. Then a total area is summarized by the areas of separate segments.

One can see a part of a forest aerial image, improving by a fusion of visual and laser data as well as segmented tree crowns, in Fig. 4.8. The shapes of tree crowns are differed from the geometric shapes, such as circles or ellipses. However, such rough approximation is used de facto in most measurement procedures. The proposed approach helps to receive the reasonable measurements of tree crowns in a plane.

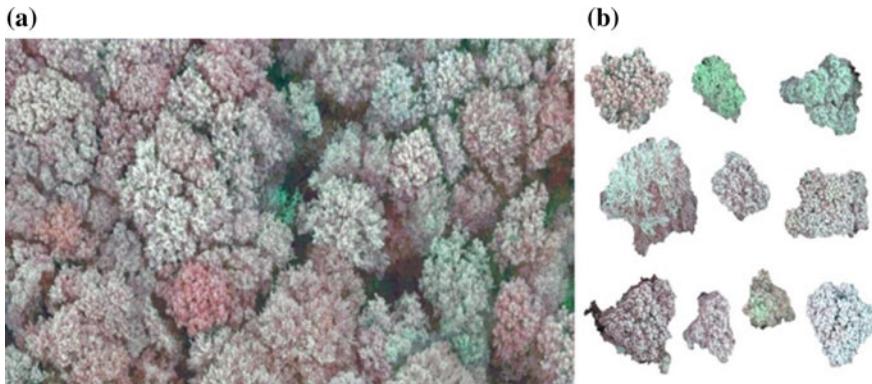


Fig. 4.8 Visual forest image: **a** part of forest aerial image, **b** set of segmented tree crowns

Table 4.3 Accuracy of area measurements of tree crowns

Crown approximation	Tree 1	Tree 2	Tree 3	Tree 4	Tree 5	Tree 6	Tree 7	Tree 8	Tree 9	Tree 10
By circle or ellipse, %	82.67	75.38	81.95	78.09	77.89	78.86	73.08	80.98	76.31	83.67
By active contour, %	91.93	92.94	95.61	93.52	96.05	94.39	93.42	95.83	96.20	94.02

Table 4.3 includes the estimations that were obtained by approximation of geometric shape (circle or ellipse) and higher order active contour model relative an expert segmentation of the individual tree crowns depicted in Fig. 4.8b (the order of visual projections of trees is from top to bottom and from left to right).

As one can see, the area measurements of tree crowns are promising in the last case and coincide with the expert estimation, providing the accuracy 92–96%. The received results are good for forest airborne modelling because of the non-Lambert nature of forest.

4.7 Conclusions

At present, the task of forest modelling based on the fusion of the LiDAR and hyperspectral/multispectral/digital photography shooting data is a non-solved task because of natural variability and complicated conditions of the airborne scanning. In this chapter, the stage devoting to accurate segmentation and crown measurement is developed. First, a procedure for fusion of the LiDAR and digital photography data was proposed. Second, a method for accurate segmentation of tree crowns based on the shearlet transform and high-order active contours model was represented. Finally, the projection areas of tree crowns were computed that have coincided well with the expert estimations, providing the accuracy 92–96%.

References

1. Harding DL, Lefsky MA, Parker GG, Blair JB (2001) Laser altimeter canopy height profiles methods and validation for closed-canopy, broadleaf forests. *Remote Sens Environ* 76:283–297
2. Vauhkonen J, Næsset E, Gobakken T (2014) Deriving airborne laser scanning based computational canopy volume for forest biomass and allometry studies. *ISPRS J Photogramm Remote Sens* 96:57–66
3. Chen Q (2015) Modeling aboveground tree woody biomass using national-scale allometric methods and airborne LiDAR. *ISPRS J Photogram Remote Sens* 106:95–106
4. Bortolota ZJ, Wynne RH (2005) Estimating forest biomass using small footprint LiDAR data: an individual tree-based approach that incorporates training data. *ISPRS J Photogramm Remote Sens* 59(6):342–360

5. Carter J, Schmid K, Waters K, Betzhold L, Hadley B, Mataosky R, Halleran J (2012) National oceanic and atmospheric administration (NOAA) coastal services center. Lidar 101: an introduction to Lidar technology, data, and applications (Revised). NOAA Coastal Services Center, Charleston, SC
6. Holmgren J (2004) Prediction of tree height, basal area and stem volume using airborne laser scanning. *Scand J For Res* 19:543–553
7. Popescu SC, Zhao KG (2008) A voxel-based LiDAR method for estimating crown base height for deciduous and pine trees. *Remote Sens Environ* 112(3):767–781
8. Berni J, Zarco-Tejada P, Suarez L, Fererez E (2009) Thermal and narrow-band multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *IEEE Trans Geosci Remote Sens* 47:722–738
9. Huang Y, Thomson SJ, Hoffmann WC, Lan Y, Fritz BK (2013) Development and prospect of unmanned aerial vehicle technologies for agricultural production management. *Int J Agric Biol Eng* 6(3):1–10
10. Yahyanejad S, Rinner B (2014) A fast and mobile system for registration of low-altitude visual and thermal aerial images using multiple small-scale UAVs. *ISPRS J Photogramm Remote Sens* 104:189–202
11. Duro DC, Franklin SE, Dubé MG (2012) Multi-scale object-based image analysis and feature selection of multi-sensor earth observation imagery using random forests. *Int J Remote Sens* 33(14):4502–4526
12. Bar Massada A, Kent R, Blank L, Perevolotsky A, Hadar L, Carmel Y (2012) Automated segmentation of vegetation structure units in a Mediterranean landscape. *Int J Remote Sens* 33 (2):346–364
13. Junntila V, Finley AO, Bradford JB, Kauranne T (2013) Strategies for minimizing sample size for use in airborne LiDAR-based forest inventory. *For Ecol Manage* 292:75–85
14. Yu X, Hyppä J, Vastaranta M, Holopainen M (2011) Predicting individual tree attributes from airborne laser point clouds based on random forest technique. *ISPRS J Photogramm Remote Sensing* 66(1):28–37
15. Fernández-Sarría A, Velázquez-Martí B, Sajdak M, Martinez L, Estornell J (2013) Residual biomass calculation from individual tree architecture using terrestrial laser scanner and ground-level measurements. *Comput Electron Agric* 93:90–97
16. Zhao K, Popescu S, Nelson R (2009) LiDAR remote sensing of forest biomass: a scale-invariant estimation approach using airborne lasers. *Remote Sens Environ* 113(1):182–196
17. Hollaus M, Wagner W, Eberhöfer C, Karel W (2006) Accuracy of large-scale canopy heights derived from LiDAR data under operational constraints in a complex alpine environment. *ISPRS J Photogramm Remote Sens* 60(5):323–338
18. Keylwerth R (1954) Ein Beitrag zur qualitativen zuwachsanalyse. *Holz als Rohund Werkstoff* 12:41–44
19. Côté JF, Fournier RA, Egli R (2011) An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR. *Environ Model Softw* 26(6):761–777
20. Morsdorf F, Nichol C, Malthus T, Woodhouse IH (2009) Assessing forest structural and physiological information content of multi-spectral LiDAR waveforms by radiative transfer modelling. *Remote Sens Environ* 113:2152–2163
21. Chave J, Coomes D, Jansen S, Lewis SL, Swenson NG, Zanne AE (2009) Towards a worldwide wood economics spectrum. *Ecol Lett* 12:351–366
22. Maltamo M, Peuhkurinen J, Malinen J, Vauhkonen J, Packalén P, Tokola T (2009) Predicting tree attributes and quality characteristics of Scots pine using airborne laser scanning data. *Silva Fennica* 43(3):507–521
23. Vastaranta M, Kankare V, Holopainen M, Yu X, Hyppä J, Hyppä H (2012) Combination of individual tree detection and area-based approach in imputation of forest variables using airborne laser data. *ISPRS J Photogramm Remote Sens* 67:73–79

24. Hyppä J, Kelle O, Lehtinen M, Inkkinen M (2001) A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners. *IEEE Trans Geosci Remote Sens* 39(5):969–975
25. Yao W, Krzystek P, Heurich M (2012) Tree species classification and estimation of stem volume and DBH based on single tree extraction by exploiting airborne full-waveform LiDAR data. *Remote Sens Environ* 123:368–380
26. Koch B, Heyder U, Weinacker H (2006) Detection of individual tree crowns in airborne LiDAR data. *Photogramm Eng Remote Sens* 72(4):357–363
27. Ene L, Næsset E, Gobakken T (2012) Single tree detection in heterogeneous boreal forests using airborne laser scanning and area-based stem number estimates. *Int J Remote Sens* 33 (16):5171–5193
28. Lee H, Slatton KC, Roth BE, Cropper WP Jr (2010) Adaptive clustering of airborne lidar data to segment individual tree crowns in managed pine forests. *Int J Remote Sens* 31(1):117–139
29. Smits I, Prieditis G, Dagis S, Dubrovskis D (2012) Individual tree identification using different LiDAR and optical imagery data processing methods. *Biosyst Inform Technol* 1 (1):19–24
30. Strîmbu VF, Strîmbu BM (2015) A graph-based segmentation algorithm for tree crown extraction using airborne LiDAR data. *ISPRS J Photogramm Remote Sens* 104(2015):30–43
31. Ferraz A, Bretar F, Jacquemoud S, Gonçalves G, Pereira L, Tomé M, Soares P (2012) 3-D mapping of a multi-layered mediterranean forest using ALS data. *Remote Sens Environ* 121:210–223
32. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. *IEEE Trans Pattern Anal Mach Intell* 24:603–619
33. Li W, Guo Q, Jakubowski MK, Kelly M (2012) A new method for segmenting individual trees from the LiDAR point cloud. *Photogramm Eng Remote Sens* 78(1):75–84
34. Lu X, Guo Q, Li W, Jacob Flanagan J (2014) A bottom-up approach to segment individual deciduous trees using leaf-off LiDAR point cloud data. *ISPRS J Photogramm Remote Sens* 94:1–12
35. Kim M, Warner TA, Madden M, Atkinson DS (2011) Multi-scale GEOBIA with very high spatial resolution digital aerial imagery: scale, texture and image objects. *Int J Remote Sens* 32:2825–2850
36. Sasaki T, Imanishi J, Ioki K, Morimoto Y, Kitada K (2012) Object-based classification of land cover and tree species by integrating airborne LiDAR and high spatial resolution imagery data. *Landscape Ecol Eng* 8:157–171
37. Wood EM, Pidgeon AM, Radeloff VC, Keuler NS (2012) Image texture as a remotely sensed measure of vegetation structure. *Remote Sens Environ* 121:516–526
38. Beguet B, Boukir S, Guyon D, Chehata N (2013) Modelling-based feature selection for classification of forest structure using very high resolution multispectral imagery. IEEE international conference on systems, man, and cybernetics SMC 2013, pp 4294–4299
39. Beguet B, Guyon D, Boukir S, Chehata N (2014) Automated retrieval of forest structure variables based on multi-scale texture analysis of VHR satellite imagery. *ISPRS J Photogramm Remote Sens* 96:164–178
40. Zhang J, Sohn G, Brédif M (2014) A hybrid framework for single tree detection from airborne laser scanning data: a case study in temperate mature coniferous forests in Ontario, Canada. *ISPRS J Photogramm Remote Sens* 98:44–57
41. Kalliovirta J, Tokola T (2005) Functions for estimating stem diameter and tree age using tree height, crown width and existing stand database information. *Silva Fennica* 39(2):227–248
42. Kankare V, Vauhkonen J, Tanhuapää T, Holopainen M, Vastaranta M, Joensuu M, Krooks A, Hyppä J, Hyppä H, Alho P, Viitala R (2014) Accuracy in estimation of timber assortments and stem distribution—a comparison of airborne and terrestrial laser scanning techniques. *ISPRS J Photogramm Remote Sens* 97(2014):89–97
43. Takahashi T, Yamamoto K, Senda Y, Tsuzuku M (2005) Estimating individual tree heights of Sugi (*Cryptomeria japonica* D. Don) plantations in mountainous areas using small-footprint airborne LiDAR. *J Forest Res* 10(2):135–142

44. Véga C, Durrieu S (2011) Multi-level filtering segmentation to measure individual tree parameters based on LiDAR data: application to a mountainous forest with heterogeneous stands. *Int J Appl Earth Obs Geoinf* 13(4):646–656
45. Vega C, Hamrouni A, El Mokhtari S, Morel J, Bock J, Renaud JP, Bouvier M, Durrieu S (2014) PTrees: a point-based approach to forest tree extraction from LiDAR data. *Int J Appl Earth Obs Geoinf* 33:98–108
46. Khosravipour A, Skidmore AK, Wang T, Isenburg M, Khoshelham K (2015) Effect of slope on treetop detection using a LiDAR canopy height model. *ISPRS J Photogramm Remote Sens* 104:44–52
47. Brandtberg T (2007) Classifying Individual tree species under leaf-off and leaf-on conditions using airborne LiDAR. *ISPRS J Photogramm Remote Sens* 61(5):325–340
48. Polewski P, Yao W, Heurich M, Krzystek P, Stilla U (2015) Detection of fallen trees in ALS point clouds using a normalized cut approach trained by simulation. *ISPRS J Photogramm Remote Sens* 105:252–271
49. Jonckheere I, Fleck S, Nackaerts K, Muys B, Coppin P, Weiss M, Baret F (2004) Review of methods for in situ leaf area index determination—part I. Theories, sensors and hemispherical photography. *Agri Meteorol* 121:19–35
50. Van der Zande D, Hoet W, Jonckheere I, van Aardt J, Coppin P (2006) Influence of measurement set-up of ground-based LiDAR for derivation of tree structure. *Agric For Meteorol* 141(2–4):147–160
51. Richardson JJ, Moskal LM, Kim SH (2009) Modeling approaches to estimate effective leaf area index from aerial discrete-return LiDAR. *Agric For Meteorol* 149(6–7):1152–1160
52. Zheng G, Moskal LM (2009) Retrieving leaf area index (LAI) using remote sensing: theories, methods and sensors. *Sensors* 9:2719–2745
53. Breidenbach J, Næsset E, Lien V, Gobakken T, Solberg S (2010) Prediction of species specific forest inventory attributes using a nonparametric semi-individual tree crown approach based on fused airborne laser scanning and multispectral data. *Remote Sens Environ* 114 (4):911–924
54. Vastaranta M, Wulder MA, White JC, Pekkarinen A, Tuominen S, Ginzler C, Kankare V, Holopainen M, Hyypä J, Hyypä H (2013) Airborne laser scanning and digital stereo imagery measures of forest structure: comparative results and implications to forest mapping and inventory update. *Can J Remote Sens* 39(5):382–395
55. Torabzadeh H, Morsdorf F, Schaeppman ME (2014) Fusion of imaging spectroscopy and airborne laser scanning data for characterization of forest ecosystems—a review. *ISPRS J Photogramm Remote Sens* 97(2014):25–35
56. Asner GP, Boardman J, Field CB, Knapp DE, Kennedy-Bowdoin T, Jones MO, Martin RE (2007) Carnegie airborne observatory: in-flight fusion of hyperspectral imaging and waveform light detection and ranging (Wilder) for three-dimensional studies of ecosystems. *J Appl Remote Sens* 1(1):013536. doi:[10.1117/1.2794018](https://doi.org/10.1117/1.2794018)
57. Yua X, Hyppää J, Vastaranta M, Holopainen M, Viitala R (2011) Predicting individual tree attributes from airborne laser point clouds based on the random forests technique. *ISPRS J Photogramm Remote Sens* 66(1):28–37
58. Favorskaya M, Zotin A, Chunina A (2011) Procedural modeling of broad-leaved trees under weather conditions in 3D virtual reality. In: Tsirhrintzis GA, Virvou M, Jain LC, Howlett RJ (eds) Intelligent interactive multimedia systems and services. Springer, Berlin
59. Song ZL, Zhang J (2010) Remote sensing image registration based on retrofitted SURF algorithm and trajectories generated from Lissajous figures. *IEEE Geosci Remote Sens Lett* 7 (3):491–495
60. Deshmukh MP, Bhosle U (2011) A survey of image registration. *Int J Image Process* 5 (3):245–269
61. Favorskaya M, Petukhov N (2011) Recognition of natural objects on air photographs using neural networks. *Optoelectron Instrum Data Process* 47(3):233–238
62. Lim WQ (2010) The discrete shearlet transform: a new directional transform and compactly supported shearlet frames. *IEEE Trans Image Proc* 19(5):1166–1180

63. Kass M, Witkin A, Terzopoulos D (1988) Snakes: active contour models. *Int J Comput Vision* 1:321–331
64. Rochery M, Jermyn IH, Zerubia J (2006) Higher order active contours. *Int J Comput Vision* 69(1):27–42
65. De Vylder J, Ochoa D, Philips W, Chaerle L, Van Der Straeten D (2011) Leaf segmentation and tracking using probabilistic parametric active contours. In: Gagolowicz A, Philips W (eds) *Mirage*. Springer, Berlin
66. Jakubowski MK, Guo Q, Kelly M (2013) Tradeoffs between LiDAR pulse density and forest measurement accuracy. *Remote Sens Environ* 130:245–253
67. Favorskaya M, Tkacheva A, Danilin IM, Medvedev EM (2015) Fusion of airborne LiDAR and digital photography data for tree crowns segmentation and measurement. In: Damiani E, Howlett RJ, Jain LC, Gallo L, De Pietro G (eds) *Intelligent interactive multimedia systems and services*. Springer, Switzerland

Part II

Individual Tree Modelling

Chapter 5

Tree Modelling in Virtual Reality Environment

Abstract In this chapter, various approaches for individual tree modelling are considered, among which the L-systems are prevailed. The fundamentals of the L-systems are situated in one of sub-sections. Usually a modelling, using the L-systems, means a viewpoint of a scene from the Earth's surface. The modelling results based only on the L-systems cannot provide good realistic virtual interpretation. However, a compact storage of individual tree in mathematical form has also a great meaning, especially during a modelling of the forest virtual scene. Some methods of realistic modelling are discussed in this chapter. Two principally different types of trees, the broad-leaved and the coniferous, have the own specialties of modelling. Notice that a conifer branching with leaves has more complex structure in comparison to a broad-leaved branching. At the ending of this chapter, some results received by use of the designed experimental software tool TreesEditor, v. 1.04 are presented.

Keywords Tree modelling · L-system · Fractal · Cellular automata · Parametric L-system · Virtual environment

5.1 Introduction

The realistic modelling of nature objects, such as trees, shrubs, flowers, and other vegetation, is necessary in many applications, including the detail visualization of forests and landscapes. In botany and forest inventory, a modelling of plant growth and visualizing of the spatio-temporal processes are required. In movie industry, the special modelling effects or special plants that cannot be found in nature are demanded. In computer games, such models are used for synthetic the virtual environments and backgrounds. In spite of various applications need the different models, the general requirement of the multi-scale descriptions and real-time representation of the landscape scenes increases faster than computers are able to handle.

A tree is a basic and complex object of nature due to its branching structure, foliage, and different species of trees. A bush can be considered as a particular case of a tree. Flowers, grass, and other plants are the separate case of modelling due to the detailed level of visualization in comparison to the trees. The subject of investigation in this chapter is an individual tree modelling. The common way to construct a complex landscape is to clone a single object many times to achieve the geometric complexity of a landscape. In the case of tree modelling, the cloning objects with the similar branches structure is a conventional approach that cannot provide a realistic visualization.

As any computer modelling, the tree modelling can be executed in off-line and real-time rendering [1]. The main goal of this chapter is to discuss the main approaches for realistic off-line rendering. As a result, the virtual natural objects will be created that are suitable for the real-time rendering (i.e., growth behavior, lighting simulation, etc.). The real-time rendering will be discussed in Chaps. 9–11.

The natural likeness of a branch structure for the tree species inspires biologists, mathematicians, and computer scientists to create models with the self-similarity [2]. First attempts tracing back to 1960–1980 are based on the cellular automata theory [3], the fractal theory [4], and the L-systems [5, 6]. The discrete cellular automata theory was modified in more realistic model in a continuous space by Cohen [7]. However, the commonly used framework for a plant simulation is based on so called the L-systems (Lindenmayer systems) proposed by Lindenmayer in 1968 [5]. The L-systems involve a formalism of the Chomsky grammars [8].

The structure of this chapter is the following. Section 5.2 overviews the main existing approaches for a tree modelling. The fundamentals of the L-systems are situated in Sect. 5.3. The procedural modelling of broad-leaved and coniferous trees is discussed in Sects. 5.4 and 5.5, respectively. Section 5.6 includes the modelling results of various tree species. Finally, conclusions are drawn in Sect. 5.7.

5.2 Related Work

In literature, some criteria of classification for tree modelling are represented. According to Boudon et al. [9], the structural criteria of plant representations, the rendering primitive criteria, and the additional classification criteria are recommended. Their short description one can find in Table 5.1. The level of details of a plant representation is provided by the functional-structural models of plants. The rendering primitive has direct impact on the realistic representation and the computational time. The additional criteria are used to estimate a quality of the modelling results.

In current section, the rendering primitive criteria are discussed primarily.

The polygon-based techniques can be classified due to the detailed elaboration of the modelling goals [2]. Figure 5.1 depicts the classification of three types, such as the shape, structure, and growth simulation. The L-systems prevail; they are situated

Table 5.1 Classification criteria with short descriptions

Type of criterion	Criterion	Short description
Structural criteria of plant representations	Detailed representation	The plants are decomposed into the simple modules (leaves, internodes, trunk, etc.) for the realistic representation. A geometrical model is built from these modules. A continuous geometry of the branching system and a realistic geometry of leaves provide a large amount of details for the high resolution views
Global representation		The plants are considered by a single or few primitives in order to drastically simplify a vegetal scene rendering. Such representation is usually adapted for low resolution views
Multi-scale representation		The intermediate approach defines several scales of representations, so called the Levels Of Details (LOD). Multi-scale techniques are divided into two categories. First category uses the hierarchical structures and clones of the similar objects in the structure. Second category is based on the spatial proximity of the elements that makes this approach being relatively independent of the modelling process
Rendering primitive criteria	Polygon-based rendering	The computer graphics representations are commonly based on the polygon technique. This approach, including the huge number of the resulting primitives, is very costly for the rendering of forest scenes. The compact descriptions in a view of grammar rules compensate this weakness. A wide spectrum of the polygon-based methods is available in literature
	Point-based rendering	The idea of drawing many points called as particles can be conveniently applied to a vegetation rendering. The number of particles can be adapted to the distance, they are averaged and simulate the opaque surfaces
	Line-based rendering	The line-based rendering is closed to the point-based rendering with a distinction that the line primitives are used for a branch simulation. This model is not explicitly converted to geometry but interpreted at runtime
	Image-based rendering	The image-based rendering requires a large set of low cost images of the individual objects or a scene. Despite of good representation of very complex objects, many limitations around diversity, lighting, and animation can be mentioned
	Volumetric approach and shader-based method	Since the details of objects merge with a distance, the indistinguishable data can be replaced by the fuzzy primitives that reproduce the same photometric behavior (reflectance model or shader). Shaders are implemented in real time but they cannot be parameterized

(continued)

Table 5.1 (continued)

Type of criterion	Criterion	Short description
Sketch-based modelling		The sketch-based modelling applies the user-defined 2D drawings of plants in order to create 3D model, using the rectilinear models based on the solving constraints, optimization-based algorithms, the reconstruction of a 3D curve using energy minimization, and symmetric relations
Example-based modelling		The example-based modelling supposes a repetition of the end-user' actions by a computer automation
Off-line /real-time rendering		The off-line rendering does not consider the dynamic processing of plants. The real-time rendering means a simulation of growth behavior, lighting simulation, etc.
Distance validity of the technique		A distance validity shows the distance range, at which the method provides the accurate results
Animation		An animation of branches and leaves is mainly addressed to the real-time methods
Lighting characteristics		The lighting characteristics present the self-shadowing (branches and leaves of the tree shadow other parts of the same tree), the external shadowing by the neighboring trees, dynamic lighting conditions, etc.
Need of pre-computation		Use of pre-computation stage or representation on the fly determines many aspects of a following modelling like the diversity allowed, animating, and the capacity of dynamic lighting
Common/specific modelling process		This criterion shows common or particular application of a modelling process
Vegetation specific /general representation		This criterion shows a suitability of method applied to a vegetation modelling. Some methods have general representation in other scopes

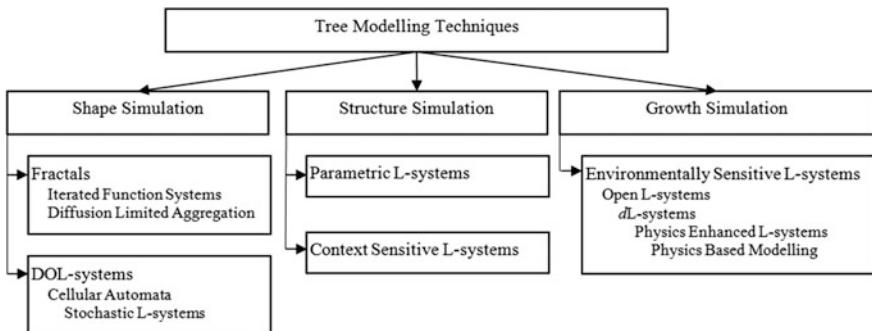


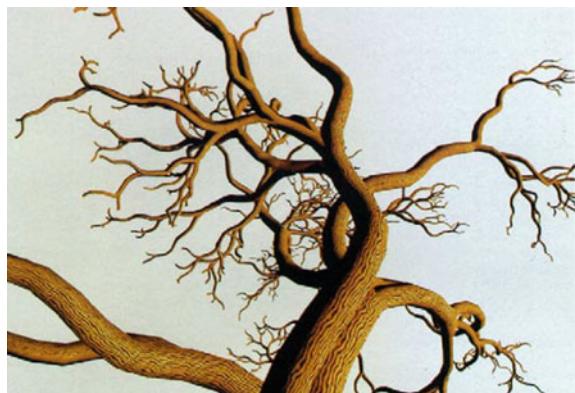
Fig. 5.1 Classification of polygon-based techniques

in three types of modelling. Worth mention that modelling by cellular automata is not popular in the individual tree modelling. However, this approach is very useful for a dynamic modelling of forests and landscapes. Notice that the polygon-based techniques, including the mathematical methods directly in the modelling process, provide 3D multi-resolution representations of scene in a natural way.

Oppenheimer [3] was the first researcher, who proposed to use the fractal theory for a plant modelling. Some production rules based on the fractal property of a self-similarity were applied for the creation of a recursive structure of plant. Oppenheimer proposed the random variables, including the branching-angle and the branch-to-parent-size-ratio that permitted to generate enough realistic fractal trees (an example from [3] is depicted in Fig. 5.2).

Weber and Penn [10] visualized the structure of a tree as a primary trunk, consisting of a variably curved structure similar to a cone. Many attributes are defined relative to the corresponding attribute of their parents. Thus, a child branch's length is specified as a fraction of its parent's length. Then the child branches can have the sub-branches and so on due to the fractal property of a self-similarity. A number of recursion levels is limited to three or four. Different

Fig. 5.2 Example of fractal tree



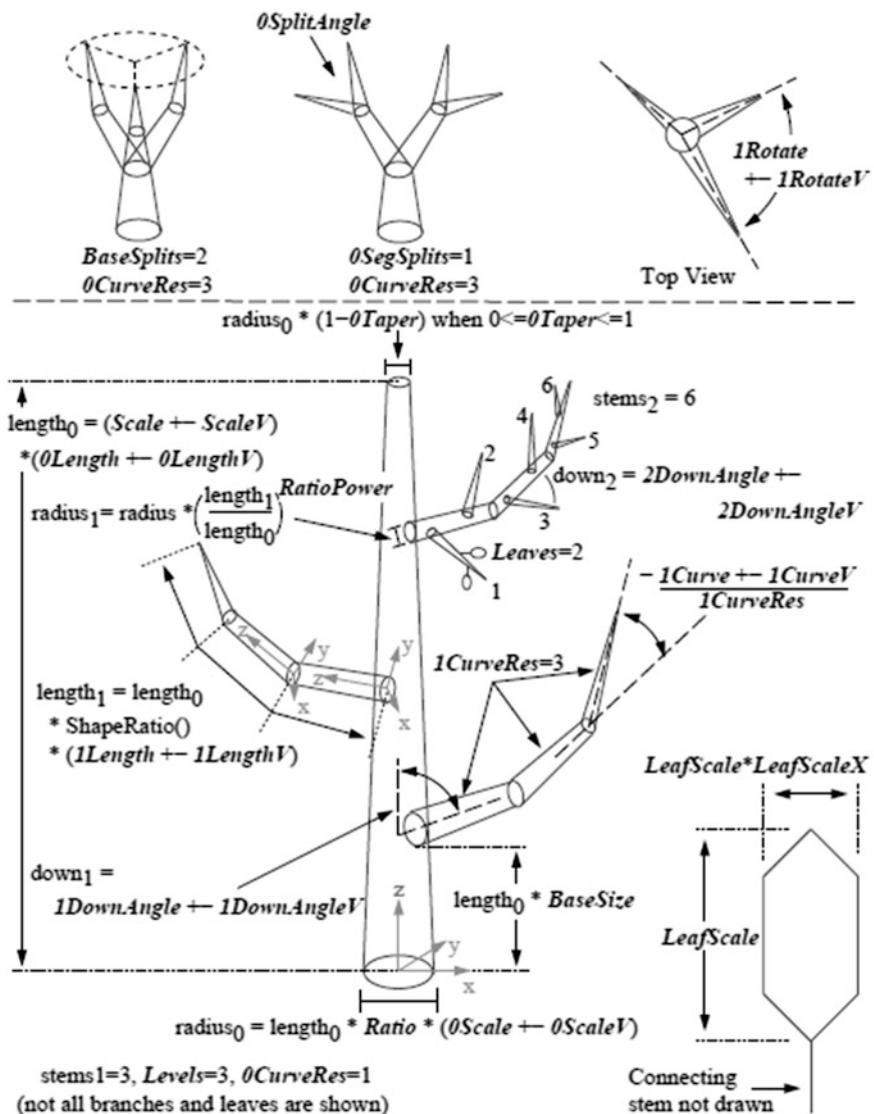


Fig. 5.3 Tree diagram proposed by Weber and Penn

specifications in recursion levels provided to reduce the self-similarity effect of the fractal theory. Tree diagram proposed by Weber and Penn is depicted in Fig. 5.3.

The cellular automata were first defined and studied for various applications since 1940s. They as a modelling formalism were applied widely for a spatial modelling of forests in 1960–1990s [5, 11]. Each automaton can be defined as a triplet G_{aut}

$$G_{aut} = \langle I, S, W \rangle,$$

where I is a set of inputs, S is a set of states, W is a next-state function that is defined on the input-state pairs [12].

The set of inputs is the ordered or non-ordered n -tuples of the states of finite “neighboring” cells. A finite set of “neighboring” cells depends from space dimension. The elementary cellular-automaton theory is based on the simplicity of the rules that produce a complicated behavior. The universal computation, self-reproduction, and conservation laws are the main properties of the cellular-automata. The basic main idea, an iterative creation of branching patterns, was extended to a set of rules in 3D space [13], net-like structures on the triangular grid [14], and 3D voxel space automata for simulation of a growth process in environment [15]. Honda [16] was the first, who created a model of tree under low-cost in computation and simplified assumptions mentioned in bestseller book of Prusinkiewicz and Lindenmayer [17]. However, all models that were proposed in 1960–1990s are far from the realistic models, ignoring the collisions between branches. Hereinafter, the cellular-automaton theory was developed for a modelling of dynamics of vegetation populations [18–20] and even for a forest-fire modelling [21].

As mentioned Prusinkiewicz and Lindenmayer [17], the application of the L-systems for botanical tree generation was first considered by Aono and Kunii [22] but found them unsuitable to model the complex branching patterns of the higher plants. Nevertheless, the following investigations of other researchers, including the founder of this scientific direction—Lindenmayer, show a suitability of the parametric L-systems for a tree modelling.

The realistic extension of the L-systems was proposed by Lintermann and Deussen [23]. A mixture of procedural and rule-based methods is in the basis of this approach. The procedures provide the component structure of plant, using a simple rule-based mechanism. A modelling process is controlled by the graphical user interfaces on the basis of the p -graph and spline functions [24]. Examples of a tree generation, modelling of the rhododendron and sunflower are depicted in Figs. 5.4, 5.5 and 5.6, respectively [23]. These investigations were implemented in the plant modelling tool Xfrog, including a plant library of several hundreds of different plants and landscapes [25]. Also many models for the L-systems have been created.

The concept of a point-based and a line-based rendering lies in the following. In increasingly complex scenes, the triangles (particular case of polygons) become smaller than a single pixel. The computational time in a sub-pixel level is increased dramatically. At the same time, the points merge easily. The adding or the removing of the single points influences on a degree of details adaptively.

A tree modelling, using a particle flow, is concerned to the point-based technique, when the input data is a small set of photographs from different points of view. Even two input photographs are enough for the reconstruction. Neubert et al. proposed a voxel-model of tree volume [26]. Tree images ought to be extracted from a background initially. Each voxel contains a density estimate of a tree biomass. Proportional to the density, the particles are emitted and traced to the tree

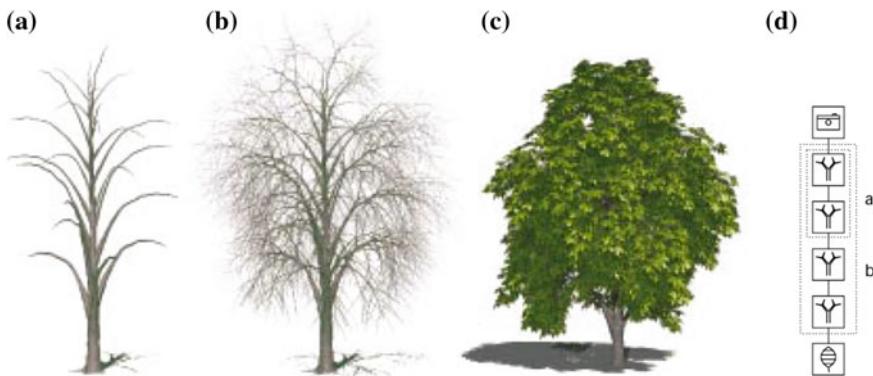


Fig. 5.4 Modelling of a chestnut tree by Lintermann and Deussen approach: **a** trunk visualization by two components, **b** additional branching levels, **c** representation of a tree with the leaves **d** *p*-graph of the chestnut tree (the *dashed regions* indicate the sub-figures corresponding to the sub-graphs)



Fig. 5.5 Modelling of a rhododendron by Lintermann and Deussen approach: **a** leaf with textures, **b** a tiny twig with blossom, **c** a main twig that branches in leaves and in two tiny twigs, **d** the whole bush, **e** *p*-graphs of the rhododendron (the *dashed regions* indicate the sub-figures corresponding to the sub-graphs)

basis, using 3D flow simulation. Simple rules with the internal and external constraints direct the particles to form the subsequent branches. In such manner, a tree skeleton with the branch structures is formed. The adding of leaves is a finalized procedure to complete 3D tree imagery. This approach does not need to adjust many parameters. Randomness is achieved by the painting densities and changing directions for a particle simulation. For medium-sized trees, the authors used 500–1000 particles, for large models 1000–2000 particles were required. The obtained images are realistic. A single disadvantage is a necessity to store a high volume of the particle coordinates. A tree skeleton is created by the manually adapted heuristics without specific botanic measurements. Such tree skeleton determines the initial particle positions. The following improvement of tree skeleton is realized by introducing of the attractor graphs for the particle tracing and additional rules. Then

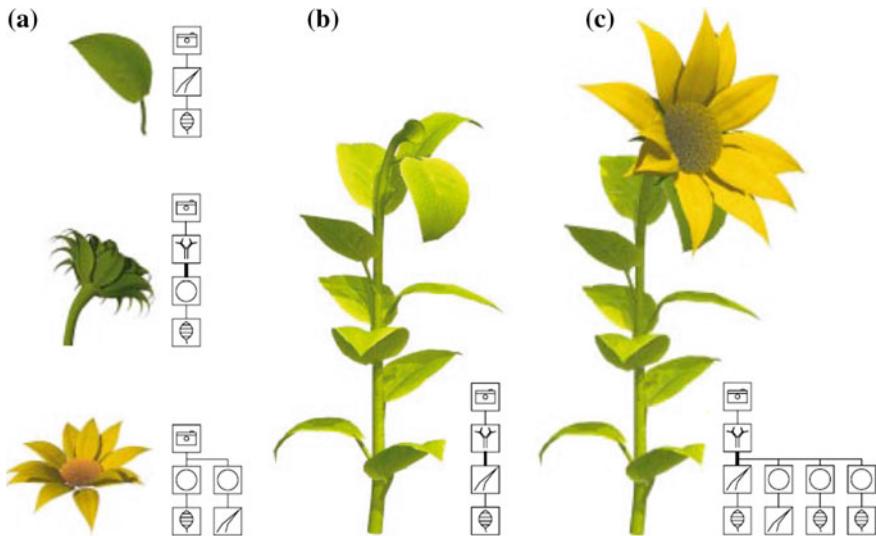


Fig. 5.6 Modelling of a sunflower by Lintermann and Deussen approach: **a** scanned real leaf textures and their p -graphs, **b** arrangement of plant structure with its p -graph by multiplying components, **c** representation of a whole plant and its p -graph

the leaf primitives are imposed on a tree skeleton. This approach is well for a low-scaled visualization of trees.

In the image-based rendering, an image can be considered as a single primitive of very complex object like a realistic tree. Many algorithms are directed to solve the traditional weaknesses of this approach, such as memory cost, static illumination, lack of relief, multi-scale representation, etc. Method of the plant hierarchy allows to bypass the fixed LODs of images. Three main approaches based on the plant hierarchy are mentioned below [9]:

- The hierarchical billboards or the simplified textured trees extension build a simplified tree with a few dozen to a few hundred polygons, approximating the foliage distribution.
- The hierarchical pre-computed multi-layer z -buffers help to create and store the z -buffer images from different fixed viewing positions. This algorithm was proposed by Max and Ohsaki in 1995 [27]. The image-based rendering using the depths may build the Earth's relief in a scene.
- The hierarchical bidirectional texture function makes the billboards fade in the nearest pre-computed views by the alpha blending and the transparency variation (the duplicated features). Multiple depth layers require to partition any object into several slabs. The number of nonequivalent sub-objects in the hierarchy ought to be limited in order to avoid a huge number of textures.

Shlyakhter et al. [28] represented an image-based modelling from a set of 4–15 images covering at least 135° around a tree. The conceptual approach includes four

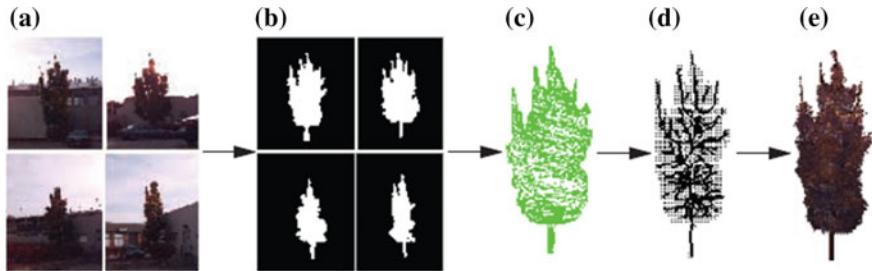


Fig. 5.7 Image-based modelling: **a** initial set of images, **b** binary set of images, **c** visual hull contraction, **d** skeleton reconstruction, **e** L-system application for simulation of small branches and foliage

stages, such as a tree segmentation from every image, a visual hull construction from the segmented binary images (this is possible due to known relative position and orientation of video camera), a constructing of a plausible tree skeleton, using the Voronoi diagrams, and an application of the open L-system for a simulation of the small branches and foliage imposition. This process is illustrated by Fig. 5.7.

Tan et al. [29] derestricted a necessity of external camera calibration. The authors distinguished the terms of trees (large terrestrial flora with small leaves relative to the tree size) and shrubs (terrestrial flora with large discernible leaves relative to the bush size). For experiments, 10–20 initial images were taken for each tree/bush with coverage 120°–200° around the simulated object. The potted flower tree was the exception, when 32 images covering 360° were taken. A scheme of this approach is depicted in Fig. 5.8. The originality consists in a graph construction of visible tree branches and reconstruction of occluded branches (with the help of end-user). The authors provided very good examples of the trees and the potted flower trees modelling.

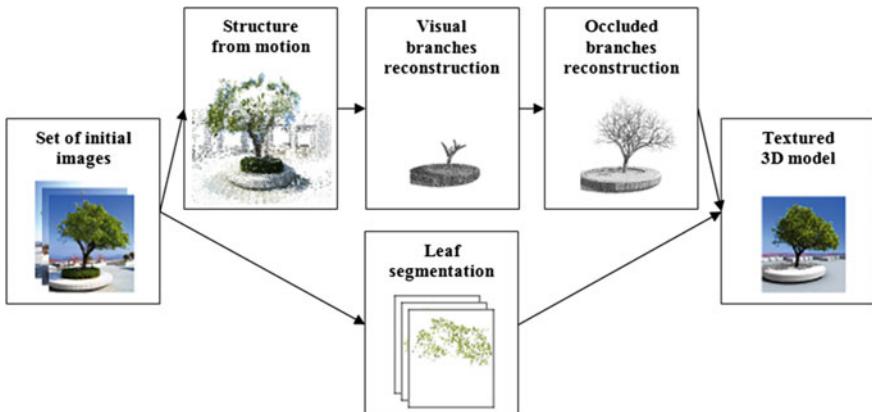


Fig. 5.8 Scheme of image-based modelling using graph structures

A volumetric approach to capture and render trees with a relatively sparse foliage was developed by Reche et al. [30]. The authors estimated the opacity in the grid based on the alpha values extracted from the photographs, while in typical volume rendering applications a sampled opacity field is determined at vertices of a grid coming from a scanning device (X-rays, etc.). The opacity value α_i is assigned to each cell i of the grid. The opacity and the color are applied to standard volume rendering equation (under the assumption that a color c_i was assigned to each cell). The intensity I in pixel p (it is considered as a mix of leaves/branches and background) is defined by Eq. 5.1, where n is a number of grid cells.

$$I = \sum_{i=0}^n \alpha_i c_i \prod_{m=i+1}^n (1 - \alpha_m), \quad (5.1)$$

that is a particular case of Eq. 5.2:

$$I = \alpha_n c_n + (1 - \alpha_n) (\alpha_{n-1} c_{n-1} + (1 - \alpha_{n-1}) (\dots (1 - \alpha_1) \alpha_0 c_0)). \quad (5.2)$$

Reche et al. [30] mentioned that their algorithm uses approximately 20–30 photographs with the following manual calibration of the cameras from these photos. The images are transformed in the Liveness and a and b for the color-opponent dimensions based on a nonlinearly compressed (Lab) color space and clustered by a color distribution. Then the alpha-masks are created for each photograph, in which zones of background and foreground are specified. The alpha-mask calculates a visibility coverage using colors: the alpha-masks of 0 correspond to the background regions and the alpha-masks of 1 correspond to the completely opaque regions. A color of pixel through its alpha value is considered as a best fit interpolation between the distributions of the clusters for the foreground and background colors. Thus, the weighted color of a pixel C_p is a linear interpolation in the LAB color space estimated by Eq. 5.3, where α_p is an alpha value in a pixel p , C_b is the weighted sums of background colors, C_f is the weighted sums of foreground colors.

$$C_p = \alpha_p C_f + (1 - \alpha_p) C_b \quad (5.3)$$

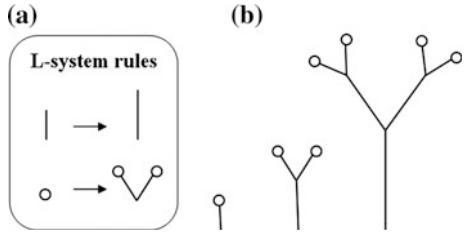
The issues of reconstruction, texture generation and view-dependent rendering of this algorithm are discussed in details in research [30].

Other approaches of a tree modelling from Table 5.1 will be discussed in the following chapters.

5.3 Fundamentals of L-Systems

In 1968, Lindenmayer proposed the L-systems as a formal description of simple multi-cellular organism in development that forms the linear or branching filaments. Afterwards, this self-similarity model was extended to the higher plants [17, 31, 32].

Fig. 5.9 Basic L-system:
a L-system rules (the first rule directs the elongation of existing branches, the second rule allocates the creation of new branches from a bud),
b development of a tree construction in time



After a while, the L-systems model evolved into the open L-systems because a plant appearance depends strongly from the environmental effects [33]. The L-system is a grammar-based parallel string rewriting system. A simulation begins with an initial string (the axiom) that consists of modules or symbols with the associated numerical parameters. In each step of a simulation, the rewriting rules (productions) replace all modules in the predecessor string by the successor modules. The resulting string is visualized by modules' interpreter. All L-systems are able to produce the plant-like structures. Figure 5.9 depicts an example of a simple L-system with two productions. (Notice that more complex p -graphs of Lintemann and Deussen are depicted in Figs. 5.5 and 5.6.)

The original concept of the L-systems was extended by the parametric L-systems, in which the grammar symbols are associated with the numerical parameters, the context-free and context sensitive parametric L-systems with the deterministic or stochastic production rules.

Lindenmayer [5, 6] considered a whole organism as an assembly of the discrete units called modules. The nature of these modules does not predefined by a formalism and may be different, for example, simple and higher botanic organisms or lower biological organisms. Each module is defined by a letter of the L-system alphabet (symbol) that specifies the module's type. Also a module can be characterized by one or more numerical parameters in a view of the structures or lists. The symbols and numerical parameters describe the module's state. The models of the L-systems are inherently dynamic. The development of structures is expressed in terms of production (rewriting rules) that change the module's state usually in a parallel mode or in replacement of current module by zero, one, or several new modules.

Let us briefly discuss the main types of the L-systems and the parametric L-systems in Sects. 5.3.1 and 5.3.2, respectively, with the following representation of the skeleton tree-like structures based on the L-systems in Sect. 5.3.3.

5.3.1 Overview of L-Systems

The context-free, interactionless, or zero-sided L-systems, called as the 0L-systems, do not interchange by information between the coexisting cells [34, 35]. Let

V denote the alphabet, V^* be the set of all words over V , and V^+ be the set of all non-empty words over V . The 0L-system is an ordered triplet

$$G = \langle V, \omega, P \rangle,$$

where $\omega \in V^+$ is a non-empty word called the axiom and $P \subset V \times V^*$ is a finite set of productions. A production $(\alpha, \chi) \in P$, written as $\alpha \rightarrow \chi$, includes the letter α and the word χ , which are called the predecessor and the successor, respectively. It is assumed that for any letter $\alpha \in V$ there is at least one word $\chi \in V^*$, such that $\alpha \rightarrow \chi$. If no production is explicitly specified for a given predecessor $\alpha \in V$, then the identity production $\alpha \rightarrow \alpha$ is assumed to belong to the set of productions P .

The 0L-system is deterministic, it is called as the DOL-system, if and only if for each $\alpha \in V$ there is exactly one $\chi \in V^*$, such that $\alpha \rightarrow \chi$. Lindenmayer rewrote the model of Mitchison and Wilcox [36] as a set of the rewriting rules or productions (taking polarities into account indicated by the arrows)

$$\overrightarrow{S} \rightarrow \overrightarrow{L} \quad \overleftarrow{S} \rightarrow \overleftarrow{L} \quad \overrightarrow{L} \rightarrow \overleftarrow{L} \overrightarrow{S} \quad \overleftarrow{L} \rightarrow \overleftarrow{S} \overrightarrow{L},$$

where L is a long cell and S is a small cell. Notice that Mitchison and Wilcox studied the observed patterns of long and short cells in a vegetative segment of Anabaena as a direct result of a developmental process. Thus, the DOL-system productions have the form:

$$\text{predecessor} \rightarrow \text{successor}.$$

In the L-system formalism, an endogenous control of organism development is implemented by various types of the context-sensitive L-systems (context-sensitive grammars), simulating interactions between the neighboring cells [34, 37]. The productions in the 1L-systems have one-sided context only, and they are written in the forms $\alpha_l < \alpha \rightarrow \chi$ or $\alpha > \alpha_r \rightarrow \chi$, where the letter α , called as the strict processor can produce the word χ if and only if α is preceded by a letter α_l and followed by α_r . The letters α_l and α_r are called the left and right context of α in this production. The 2L-systems use the productions of a form $\alpha_l < \alpha > \alpha_r \rightarrow \chi$. The 0L-systems, the 1L-systems, and the 2L-systems belong to class of the IL-systems also called as the $(m, n)L$ -systems. In the $(m, n)L$ -system, the left context is a word of length m and the right context is a word of length n . Every combination of letters of length m or n must be specified as a context for each letter in the alphabet.

All trees and plants generated by the same deterministic L-system have identical shapes. This predictability is one of the properties that makes the L-systems the useful but non-realistic modelling tool. The use of the stochastic techniques can improve this situation. During a training stage, data can be collected from a number of patterns and analyzed statistically to determine the appropriate probabilities for branching and growth processes of different tree species. The stochastic approach is a good way against the artificial regularity in a scene. In order to prevent this effect, the adding of specimen-to-specimen variation is necessary that will preserve the

general aspects of a plant with its detail modification. Such variation can be achieved by randomizing of the interpretation, the L-system, or both simultaneously. A randomization of the interpretation alone has a limited effect. In the deterministic application, the geometric aspects of a plant (the trunk lengths and branching angles) vary in a random way but the underlying topology remains unchanged. In the stochastic application of productions, both the topology and the geometry of a plant may affect.

Thus, a stochastic IL-system G_π is an ordered quadruplet [38]:

$$G_\pi = \langle V, \omega, P, \pi \rangle,$$

where V is an alphabet, ω is an axiom, P is a set of production, $\pi: P \rightarrow \mathfrak{R}$ is a production that maps a set of productions P into a set of non-negative real numbers called as probability factors \mathfrak{R} .

Let $\hat{P}(\mu, s) \subset P$ denote the subset of productions from the set P , which matches a word μ at position s . If no production in P matches μ at a given position, then $\hat{P}(\mu, s)$ is assumed to contain an identity production for the letter at position s with a probability factor 1. The derivation $\mu \Rightarrow v$ is a stochastic derivation in G_π if for each position s in the word μ , a probability PR of applying production $p_i \in \hat{P}(\mu, s)$ is equal to

$$PR(p_i) = \frac{\pi(p_i)}{\sum_{p_k \in \hat{P}(\mu, s)} \pi(p_k)}. \quad (5.4)$$

An important consequence of below above definition is that different productions with the same strict predecessor can be applied to various occurrences of the same letter in one derivation step. In this case, a static specification of the production probabilities would not be sufficient because a letter at a given position in the word does not dependent only on the strict predecessor but also on the context. Thus, a number of productions, associating with a given letter, may vary from position to position in the string.

5.3.2 Overview of Parametric L-Systems

The first models based on the L-systems were represented by a list of the consecutive words generated during a derivation with the following visualization relied on a human interpretation of the letters as different modules. In order to model and visualize the complex branching plants, Hogeweg and Hesper [39] and Frijters and Lindenmayer [40] proposed the bracketed L-systems that better determine the branching topology of the modelled plants. The geometric aspects, e.g., the branching angles, had been added in a post-processing phase. The results of Hogeweg and Hesper were subsequently extended for the realistic image synthesis

by Smith [41]. In the resulting graphical formalism, this geometry was added to the topological model as a separate interpretation step.

Szilard and Quinton [42] proposed different approach that was assigned an interpretation as a sequence of commands and based on the geometry internal to the L-system. They claimed that simple DOL-systems could generate the convoluted curves—fractals. Prusinkiewicz incorporated the geometric commands directly within the L-systems, and extended this turtle interpretation for the bracketed IL-systems [43] and 3D models [38]. In turtle interpretation of the L-systems, the turtle builds a structure of the line segments as it moves around in 3D world. Its actions are controlled by the sequence of commands generating a string from left to right. Finally, an image had been created by rendering a particular view of the turtle's world, using standard computer graphics techniques.

The L-systems with the turtle interpretation can generate a variety of objects—from abstract fractals to realistic images of trees and plants [44, 45]. However, the discrete nature of formalism imposes a major limitation for modelling. The constant length of the turtle step is also a problem, when the expansion of a structure is simulated over time. Since the line segments are represented by the sequences of F symbols, this expansion is modelled by the increase in their number from one derivation step to the next. Rozenberg and Salomaa [35] show that the growth function $f_G(n)$ of the DOL-system, computing in a derivation step n with a number symbols in a string s , is a combination of the polynomial and exponential functions provided by Eq. 5.5, where $P_i(n)$ denotes a polynomial with integer coefficients, ρ_i is a non-negative integer, n_0 is a total number of letters in the alphabet of G .

$$f_G(n) = \sum_{i=1}^s P_i(n) \rho_i^n \quad \text{for } n \geq n_0 \quad (5.5)$$

However, the natural growth processes cannot be described by Eq. 5.5. The use of the interactive L-systems extends the range of growth functions but still restricts only a logarithmic growth. Also the IL-systems may require the message passing schemes that do not exist in nature. Many processes, including the chemical reactions, diffusion of hormones, and resulting distribution of concentrations, are continuous in nature. It is difficult to capture such phenomena, using the L-systems by discretization of continuous values and introduction of hundreds of symbols and productions. That is why, Lindenmayer proposed to associate the numerical parameters with the L-system symbols and illustrated this idea by referring to the continuous development of the branching [40] and non-branching structures (the last ones for *Anabaena catenula*).

Thus, the definitions for parametric L-systems with the turtle interpretation are based on the ideas of Lindenmayer and are derived from the original formulation by Prusinkiewicz and Hanan [46]. This provides a practical basis for the modelling and visualization of a wide variety of organisms.

Generally, the parametric L-systems are represented by the parametric OL-systems, the parametric IL-systems, and the stochastic parametric L-systems.

The parametric L-systems involve the parametric words as the strings of modules. The letters belong to an alphabet V have the associated parameters belong to the set of real numbers \mathfrak{R} . A module with letter $A \in V$ and parameters $a_1, a_2, \dots, a_n \in \mathfrak{R}$ is denoted by $A(a_1, a_2, \dots, a_n)$. Each module belongs to the set $V \times \mathfrak{R}^*$, where \mathfrak{R}^* is the set of all finite sequences of parameters.

The real-valued actual parameters in the words correspond to the formal parameters represented in the specification of the L-system productions. A letter with an associated sequence of formal parameters is called a formal module and a sequence of formal modules is called a formal parametric word. If \sum is a set of formal parameters, then $C(\sum)$ denotes a logical expression with parameters from \sum , and $E(\sum)$ is an arithmetic expression with parameters from the same set. Both types of expressions consist of the formal parameters and numeric constants, summarized in Table 5.2 and parentheses for grouping. The operator precedences and associativities are presented in Table 5.2. The relational and logical expressions evaluate to zero for false and one for true. A logical statement specified as the empty string is assumed to have value one. The expressions can include the calls to predefined trigonometric and exponential functions. Let the sets of correctly constructed logical and arithmetic expressions with parameters from \sum be denoted $C_l(\sum)$ and $E_a(\sum)$, respectively.

The parametric 0L-system is defined as an ordered quadruplet:

$$G = \langle V, \Sigma, \omega, P \rangle,$$

where V is a non-empty set of letters called the alphabet of a system, \sum is a set of formal parameters, $\omega \in (V \times \mathfrak{R}^*)^+$ is a non-empty parametric word called the axiom, $P \subset (V \times \sum^*) \times C_l(\sum) \times (V \times E_a(\sum))^*$ is a finite set of productions.

A production $(\underline{a}, C, \underline{\chi})$ is noted as $\underline{a} : C \rightarrow \underline{\chi}$, where the formal module $\underline{a} \in V \times \sum^*$ is called the predecessor. The formal parametric word $\underline{\chi} \in (V \times E_a(\sum))^*$ is called the successor. The logical expression $C \in C_l(\sum)$ is called the

Table 5.2 Operator precedence and associativity

Operator	Description	Associativity	Operator	Description	Associativity
$f(\cdot)$	Function call	Left to right	$<$	Less than	Left to right
$-$	Unary minus	Right to left	\leq	Less than or equal to	Left to right
!	Logical negation	Right to left	$>$	Greater than	Left to right
Λ	Exponentiation	Right to left	\geq	Greater than or equal to	Left to right
*	Multiplication	Left to right	$==$	Equality	Left to right
/	Division	Left to right	$=!$	Inequality	Left to right
%	Remainder	Left to right	$\&\&$	Logical <i>and</i>	Left to right
+	Addition	Left to right	$ $	Logical <i>or</i>	Left to right
-	Subtraction	Left to right			

condition. If the condition is empty, then the production can be noted $\underline{a} \rightarrow \underline{\chi}$. For a given production, it is assumed that a formal parameter can appear no more than once in the predecessor and all formal parameters in the condition and successor must appear in the predecessor.

The parametric DOL-system is defined as an ordered quadruplet:

$$G = \langle V, \Sigma, \omega, P \rangle,$$

where V is an alphabet, Σ is a set of formal parameters, ω is a axiom, $P : \{1, 2, \dots, N\} \subset (V \times \Sigma^*)^* \times C_l(\Sigma) \times (V \times E_a(\Sigma))^*$ is a finite ordered set of productions. The production $p_i \in P$ matches a string in a module.

The definitions of predecessor, successor, and condition are the same as in the case of the OL-systems.

The parametric DOL-systems are context and provide a mechanism to model an endogenous control of development, using parameters incorporated into the non-bracketed interactive L-systems. In the non-parametric case, two types of interactive L-systems were distinguished. The IL-systems include the left and right contexts of constant lengths, and the I'IL-systems have not such restriction. In the parametric case, this distinction is not actually. In general case, the more convenient IL-system, in which contexts of different lengths coexist in the same parametric L-system, is used.

The parametric L-system is defined as an ordered quadruplet:

$$G = \langle V, \Sigma, \omega, P \rangle,$$

where V is an alphabet, Σ is a set of formal parameters, $\omega \in (V \times \mathfrak{R}^*)^+$ is a axiom, $P \subset (V \times \sum_l^*)^* \times (V \times \sum_r^*)^* \times (V \times \sum_r^*)^* \times C_l(\Sigma) \times (V \times E_a(\Sigma))^*$ is a finite set of productions.

A production $(\eta_l, \underline{a}, \eta_r, C, \underline{\chi})$ is noted as $\underline{\eta}_l < \underline{a} > \underline{\eta}_r : C \rightarrow \underline{\chi}$, where the formal module $\underline{a} \in V \times \sum^*$ is called the strict predecessor. The formal parametric words $\underline{\eta}_l \in (V \times \sum_l^*)^*$ and $\underline{\eta}_r \in (V \times \sum_r^*)^*$ are called the left context and the right context, respectively. The triplet $(\underline{\eta}_l, \underline{a}, \underline{\eta}_r)$ is called the predecessor. The logical expression $C \in C_l(\Sigma)$ is called the condition. The formal parametric word $\underline{\chi} \in (V \times E_a(\Sigma))^*$ is called the successor. As in the context free case, a formal parameter can appear no more than once in the predecessor of any production, and all formal parameters, appearing in the condition, and the successor must appear in the predecessor.

A production with the left context empty is written as $\underline{a} > \underline{\eta}_r : C \rightarrow \underline{\chi}$, while a production with the right context empty has a similar view $\underline{\eta}_l < \underline{a} : C \rightarrow \underline{\chi}$. A production with the empty condition can be written as $\underline{\eta}_l < \underline{a} > \underline{\eta}_r \rightarrow \underline{\chi}$. Any combinations of an empty condition and empty contexts are possible.

The stochastic capabilities of the parametric IL-systems allow the probability distributions to change over time. The stochastic parametric L-system is defined as an ordered quadruplet:

$$G = \langle V, \Sigma, \omega, P \rangle,$$

where V is an alphabet, \sum is a set of formal parameters, $\omega \in (V \times \mathfrak{R}^*)^+$ is a axiom, $P \subset (V \times \sum_l^*)^* \times (V \times \sum^*)^* \times (V \times \sum_r^*)^* \times C_l(\sum) \times (V \times E_a(\sum))^*$ $\times E_a(\sum)$ is a finite set of productions. If a sextuple $(\underline{\eta}_l, \underline{a}, \underline{\eta}_l, C, \underline{\chi}, \psi)$ is a production, it is noted $\underline{\eta}_l < \underline{a} > \underline{\eta}_r \rightarrow \underline{\chi} : \psi$, where the predecessor $(\underline{\eta}_l, \underline{a}, \underline{\eta}_l)$, the condition C , and the successor $\underline{\chi}$ are defined as in the non-stochastic case and $\psi \in E_a(\sum)$ is called the probability factor expression.

Within a given production, a formal parameter can appear no more than once in the predecessor, and all formal parameters, appearing in the condition, the successor, and the probability factor expression must appear in the predecessor.

Let $\hat{P}(\mu, s) \subset P$ be the subset of productions from the set P , which matches a word μ at position s . If no production in P matches μ at a given position, then the module at position s is replaced by itself. Otherwise, let the probability factor $\pi(p_i, \mu, s) \geq 0$ be the value of the probability factor expression ψ in a production $p_i \in \hat{P}(\mu, s)$ under the assumption that the actual parameter values from the matched string have been substituted for the formal parameters in the predecessor. The derivation $\mu \Rightarrow v$ is a stochastic derivation in G_π if for each position s in the word μ the probability PR_s of applying production $p_i \in \hat{P}(\mu, s)$ is equal to

$$PR_s(p_i) = \frac{\pi(p_i, \mu, s)}{\sum_{p_k \in \hat{P}(\mu, s)} \pi(p_k, \mu, s)},$$

assuming that at least one probability factor $\pi(p_i, \mu, s) \geq 0$.

A formalism of the parametric L-systems was extended to the branching structures, using the branch delimiters “[“ and ”]” as in the non-parametric bracketed L-systems in Lindenmayer’s notion of bracketed strings.

5.3.3 Skeleton Tree-like Structures Based on L-Systems

The parametric L-systems are very suitable to construct the tree skeletons as the branching patterns. Notice that Honda introduced a model, using parameters to define the skeleton of a tree [16]. According to Honda, the shapes of trees are constructed under the assumptions mentioned below:

- The tree segments are straight and their girth is not considered.
- A mother segment produces two daughter segments through one branching process.
- The lengths of the two daughter segments are shortened by constant ratios, r_1 and r_2 , with respect to the mother segment.

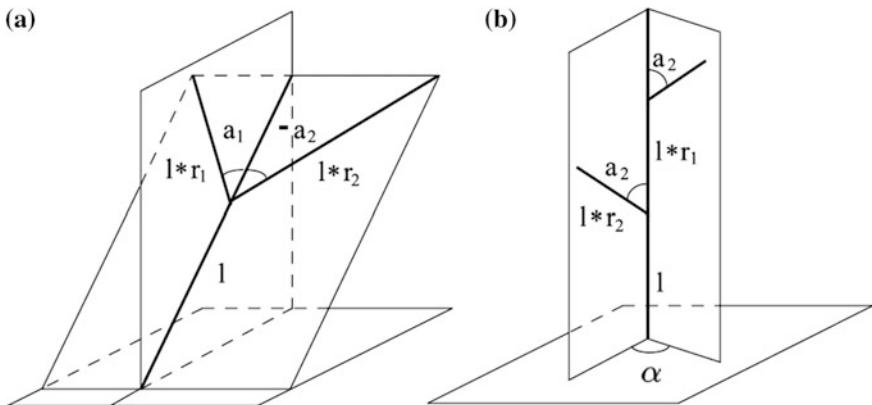


Fig. 5.10 Tree geometry according to Honda: **a** branching of the daughter segments, **b** branching of mother segments

- The mother segment and its two daughter segments are contained in the same branch plane. The daughter segments form constant branching angles, α_1 and α_2 , with respect to the mother branch.
- The branch plane is fixed with respect to the direction of gravity so as to be closest to a horizontal plane. An exception is made for branches attached to the main trunk, when a constant divergence angle α between consecutively issued lateral segments is maintained.

The proposed by Honda [16] tree geometry is depicted in Fig. 5.10.

Figures 5.11, 5.12 and 5.13 illustrates the difference between the monopodial (parallel to the line of gravity) and dichotomous (tending in different directions from the original) branching structures that were constructed using the L-systems paradigms. Examples of compound leaves are depicted in Fig. 5.14.

The wide variety of tree-like shapes can be obtained by changing the numerical parameters and application of different rules for the branching angles that was done by Honda and the following scientists. Thus, the results of Honda were developed by Aono and Kunii [47], who suggested the biasing of branch positions in a particular direction under the effects of wind, phototropism, and gravity. Bloomenthal [48] and Oppenheimer [49] achieved a substantial improvement in the realism of tree models by introduction the curved branches (against the straight branches in Honda model and Aono and Kunii model). Also Bloomenthal and Oppenheimer applied a careful texturing of bark and leaves as this was allowed by the computer technique at that time. In contrast to the deterministic algorithms of branching structures used by Honda, the stochastic approaches were investigated in several publications, for example, in [50, 51]. Reeves and Blau [50] aimed to produce the tree-like shapes without biological details of the modelled structures,

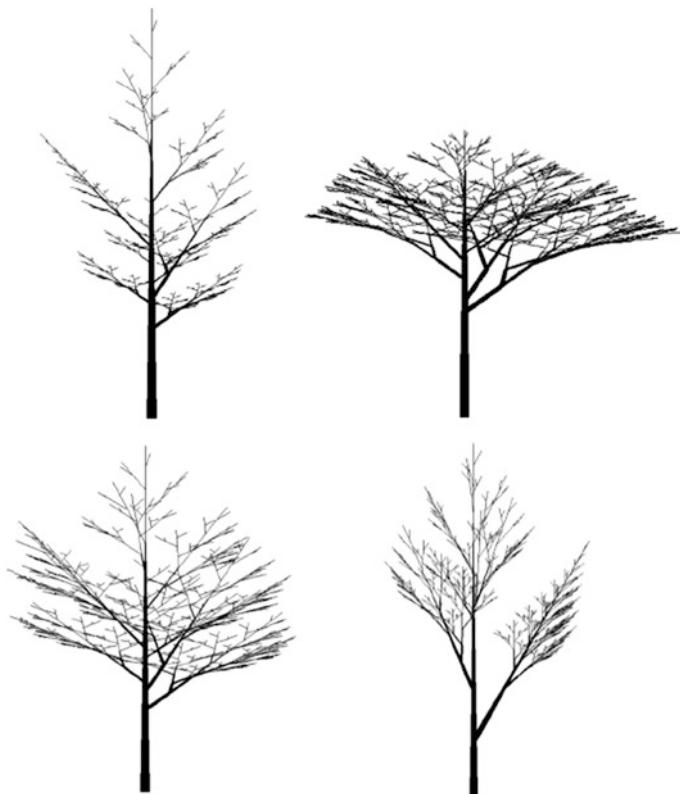


Fig. 5.11 Examples of monopodial tree-like structures

while de Reffye et al. [52] applied a stochastic approach to simulate the real plants in temporal domain by modelling the development of buds at the discrete time intervals.

5.4 Procedural Modelling of Broad-Leaved Trees and Shrubs

The procedural modelling of the individual trees is often used 3D urban scene reconstruction or landscape scenes with rarely trees and shrubs. Three main approaches can be mentioned for a modelling of tree species:

- The non-realistic procedural modelling based on the L-systems or examples. This approach prevailed in 1900–2000.
- The realistic procedural modelling, using a set of images, data of laser scanning, or visual/laser/infrared data. This direction is developed actively at present time.



Fig. 5.12 Examples of symподial tree-like structures

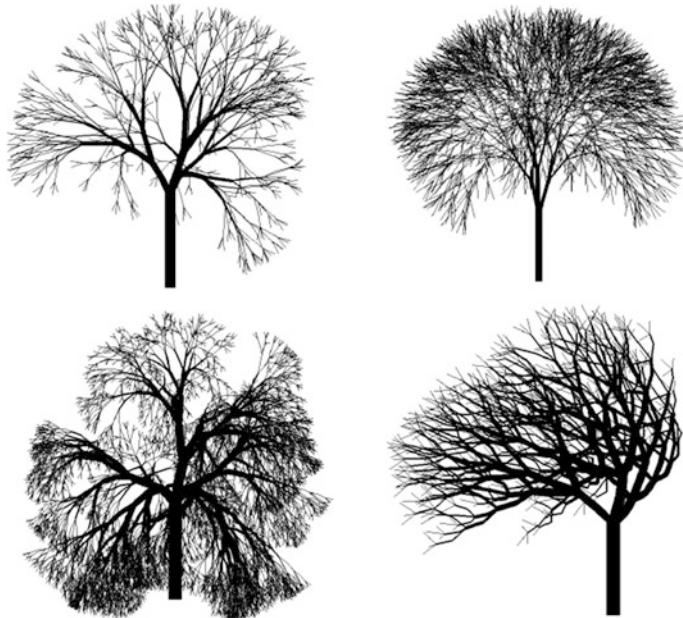


Fig. 5.13 Examples of tree-like structures with ternary branching



Fig. 5.14 Examples of compound leaves

- The hybrid procedural modelling. This is the reasonable approach that integrates the specialties of tree species provided by the L-systems and data received from a laser scanner.

Consider the non-realistic and realistic procedural modelling of the individual broad-leaved trees in Sects. 5.4.1 and 5.4.2, respectively.

5.4.1 Procedural Broad-Leaved Trees Modelling Based on L-Systems

The stochastic grammar of the L-system was extended by use some procedures that include the natural specialties of growth and aging of trees [52]. The initial procedure that permits to choose the parameters of branch 0 was introduced to provide some variety in the species' modelling. This procedure PR_W has a view of Eq. 5.6, where \mathbf{Pb}_0 is a central point of branch 0 with the local coordinates (x, y, z) , \mathbf{Ab}_0 is a set of angle values in 3D-space in the central point \mathbf{Pb}_0 , Tb_0 is a set of thicknesses of branch 0, Lb_0 is a set of proposed lengths of branch 0.

$$PR_W = \{\mathbf{Pb}_0, \mathbf{Ab}_0, Tb_0, Lb_0\} \quad (5.6)$$

A fractal “branching” may be interpreted as a skeleton of a tree. However, the realistic trees have not strong fractal “branching”. Thus, a growing procedure PR_G , which is periodically restarted and recalculates the parameters of all branches, simulating an annual tree volume, was introduced as Eq. 5.7, where GF_i is a growing factor of i branch (in a simple case it may be a constant), GL_i is a variable calling a foliage increasing, \mathbf{Ab}_i is a set of angle values in 3D space in the central point \mathbf{Pb}_i , Tb_{Gi} and Lb_{Gi} are a thickness and a length of i branch increased by a growing factor GF_i , respectively, N is a number of visible branches.

$$PR_W = \{GF_i, GL_i, \mathbf{Pb}_i, \mathbf{Ab}_i, Tb_{Gi}, Lb_{Gi}\} \quad i \in (0, N) \quad (5.7)$$

Equation 5.7 makes a growing tree more realistic that is especially important for the trees' modelling without foliage.

The growing process is not a single process during a tree life-cycle. Under some biological artifacts or weather conditions, a tree changes its branch structure that leads to degenerate a part of branches until a whole tree will not become a dry tree. The degenerating procedure PR_D , which recalling frequency deals with the age of a tree, has view of Eq. 5.8, where DF_i is a degenerating factor of i branch $DF_i = [0, 1]$ (if $DF_i = 0$, then i branch and its foliage are implicitly destroyed), DL_i is a variable calling a foliage decreasing, Tb_{Di} and Lb_{Di} are a decreasing of thickness and a length of i branch by a degenerating factor DF_i correspondingly in the central point \mathbf{Pb}_i ; N is a number of visible branches.

$$PR_D = \{DF_i, DL_i, \mathbf{Pb}_i, Tb_{Di}, Lb_{Di}\} \quad i \in (0, N) \quad (5.8)$$

This procedure is recalled non-periodically during a tree growing but during a tree drying a procedure PR_D will stop the development of branches and foliage.

Equations 5.6–5.8 were applied for a simulation of the broad-leaved trees, shrubs, and forest as well as Eqs. 5.6 and 5.7 were used for a simulation of season foliage effects. The reasonable approach for foliage simulation is to create a set of leaf patterns for the given types of tree species, using the morphological transformations in spring and autumn. The dynamic simulation of a leaf falling in autumn is concerned to a scene modelling, when a texture of 3D Earth's surface is changed.

During visualization (with high resolution) of the L-system, a representation of the branches and the segments of branches by the truncated cones, cylinders, or mesh per each branch may provoke some discontinuities. A visibility problem of a continuous junction between branches is depicted in Fig. 5.15a. To obtain the curved branches, Lluch et al. proposed the generation of a single polygonal mesh [53].

First, the hierarchical structure has been obtained from the interpretation of the chain of symbols produced by the L-system. Second, a triangular mesh is generated that respects the contour shapes and the topology established by the structure. This

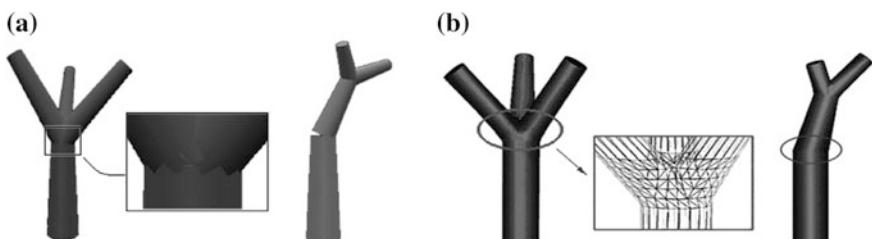


Fig. 5.15 Continuous junction between branches: **a** with artifacts, **b** result of reconstruction algorithm, using triangles

mesh is generated by joining the vertices of the two contours associated to each branch as one can see in Fig. 5.15b. A set of intermediate contours is obtained by a sectioning the trunk cylinders at the specified height.

The task dealing with a more accurate 3D trunk skeleton representation was solved by Teng et al. [54]. The algorithm was based on a self-calibration of camera, using the external parameters. An interactive segmentation to extract a trunk from the image was utilized, using k -means or the Expectation-Maximization (EM) algorithms with the following application of the graph cut theory. The skeleton of 2D trunk structure provides a good way to explore 3D trunk skeleton. In spite 3D reconstruction method using linear triangulation is well-known, the authors applied 3D reconstruction in conjunction with the minimum curvature constraint that makes 3D trunk more realistic with the imposed texture.

Notice that the modelling of shrubs is very similar to a tree modelling with the own parameters of growth maintained manually. Therefore, the bush modelling does not require a detailed description.

5.4.2 Realistic Image-Based Procedural Modelling of Broad-Leaved Trees

The modelling of 3D trees by an iterative reconstruction, using the evolutionary computation, image processing, and computer vision, was proposed by Zamuda and Brest [55]. An evolutionary algorithm iteratively extracted the morphological model parameters from two and more images. The vectorized auxiliary local branch parameters were used to condense and represent the matrix parameters of the branches as an alternative for the use of the L-systems. The end-user enters a minimized set of parameters and then the procedural model designs a tree automatically, determining the positions, rotations, sizes, and textures for several thousand branch segments and leaves. The global parameters of this model with their values and intervals are listed in Table 5.3.

The geometric structure of a branch was approximated by several cones, while the final branches (branches having one strand only) were rendering with the pyramids. The differential evolution optimization algorithm involves three stages, such as the genotype encoding, genotype-phenotype mapping, and its fitness evaluation. The expected reconstruction error was measured by sum of differences in the reference original image and generated rendered images of the evolved parameterized procedural models. The images are compared by a pixel-wise mode to calculate differences. In the evolved image for each pixel rendered as foreground, the Manhattan distance to the nearest non-background pixel of the reference image was computed, and vice versa. The EcoMod software tool includes four main interconnected components [55]:

Table 5.3 Local and global parameters in model of Zamuda and Brest

Local parameters	Values and intervals of global parameters
Types and height of base trunk	The number of strands 0–5000 The height of base trunk 0–10 m
Lower and upper bounds of sub-branch relative length compared to basic branch	The gravity centralisms angle 180°–180° The phyllotaxis angle 0°–360°
Branch length scaling factor	The coefficient of branch thickness 0–0.05
Leaf distribution type	The leaf distribution: Spiral, Stacked, Staggered, Bunched, and Coniferous
Leaf density	The density of leaves 0–30
Leave size	The size of leaves 0–0.3
Bark/leaf texture identification	Defined automatically
Level of detail simplification for a tree structure	Defined automatically
Undirected wind intensity	The speed of undirected wind 0–10 m/s
Branch elasticity for a wind sway	Defined automatically
Directed wind intensity	The speed of directed wind 0–10 m/s
Main direction of a directed wind blowing	The viewing angle 0°–360°

- A tree reconstruction based on differential evolution algorithms as the single-objective and multi-objective versions.
- A modeller of 3D geometrical natural trees (much interactive component of the application).
- A visualization of ecosystems rendering a real-time landscape.
- An artificial life simulation of trees/plants within the ecosystem, the using real-data living condition, plant competition, and plant spread model.

The disadvantages of Zamuda and Brest approach deal with the desirable tree images without leaves, “good” background (for example, clean sky), and the persistent involvement of end-user in a modelling process.

Anastacio et al. [56] proposed an approach dealing with the sketch-based parameterization of the L-systems. The main argument is the increased attention to global features of trees and plants, such as postures, insertion angles, dimensions of the components, and silhouettes. The authors mentioned that the global-to-local specification is introduced into the L-system-style models in a form of user-controlled B-spline functions. The B-spline functions are associated with the parametric production rules and may be used to represent the positional information and morphogenetic gradients in a plant model. In the L-systems, a global structure of plant is provided only by the outcome of the interaction of local rules. Therefore, Anastacio et al. developed the sketch-based approach, when the position of a plant is represented as 3D curve, not by its projection into a plane.

Usually the depth information is provided. It means that an unambiguously mapping a 2D curve in 3D space is very difficult problem and cannot be solved automatically. The algorithm is inspired by the traditional line-based techniques aiming at providing a depth perception in an illustration but uses a tablet that

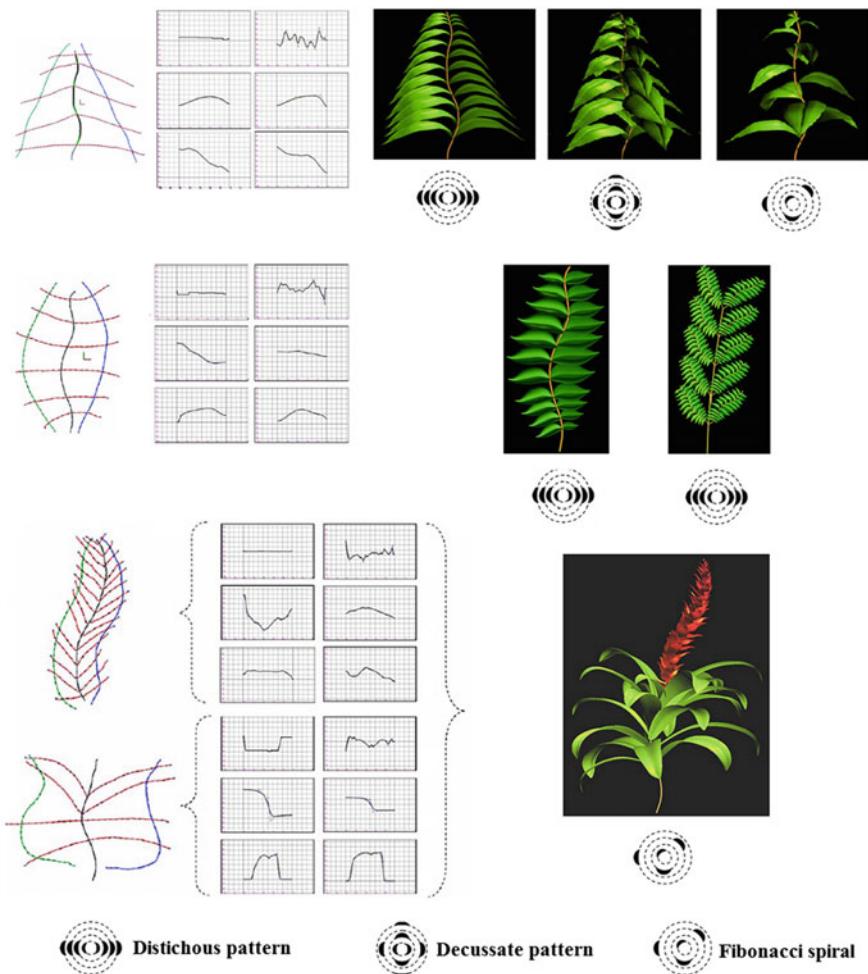


Fig. 5.16 Patterns of curves and phyllotaxis and the sketched-based generated models of branches and plants

outcomes are the pressure data of a pen. The algorithm works under the assumption that the end-user applies more pressure, while drawing the parts of the stroke that are closer, and less pressure for parts that are farther. The patterns of curves and phyllotaxis, storing in a dataset, and the sketched-based generated models of the branches and plants are depicted in Fig. 5.16. This algorithm provides good possibilities for 3D modelling of individual plants with the persistent participation of end-user.

Zen and Wang [57] claimed that in spite of the plant shapes strongly depend from the environmental factors, such as sunlight, water, and neighboring plants, some mathematical rules can be applied in their growth patterns. One of such

mathematical rules is the golden section that can be observed in the branching systems, phyllotaxis, flowers, and seeds. The authors had investigated different tree species and measured the angles between branches of trees. For example, 79 branches of poplars from 100 ones satisfy the golden section of 90° ($(90 - 33.4)/90 = 0.618$). The similar dependences were detected for other space angles between the trunk and branches, the branches and leaves.

The Functional-Structural Plant (FSP) modelling was developed by Evers et al. [58]. An FSP model is a simulation model, describing the development of 3D structure of plants, depending on the environmental factors. An FSP model involves the components taking into account the following issues:

- Acquisition and synthesis of compounds (assimilates, nutrients, and hormones).
- Signalling by compounds (hormones, assimilates, and nutrients).
- Transport of compounds through the plant structure.
- Plant growth and phenological development.
- Light interception, scattering, and signaling.
- Manipulation of plant structure.
- Interactions between individual plants.

The branching as the collective term for all processes, leading to the formation of the side shoots (branches, tillers, and stolons) from the axillary buds on the shoots of a plant, is a key determinant of a plant shape. The branching is highly plastic due to the sensitivity to the environmental factors. A plant can produce a single branchless trunk in a dense stand, whereas the same plant can produce numerous branches, when grown solitarily.

The current investigations show that the best tree modelling is obtained during a portable scanning (i.e., terrestrial laser scanning). Hosoi et al. [59] developed a method to produce 3D voxel-based solid model of a tree based on a portable scanning LiDAR data for accurate estimation of a volume of the woody material. Estimations are enough accurate due to a voxel size $0.5 \text{ cm} \times 0.5 \text{ cm} \times 0.5 \text{ cm}$. First, the algorithm analyzes large components, such as trunk and large branches. Second, the small components are modelled. The results are impressive: the percentage error of the trunk volume and part of a large branch was 0.5% and the estimation error of a certain part of the small branches was 34.0%. However, such approach is suitable only for a finite number of the individual trees.

5.5 Procedural Modelling of Coniferous Trees

A branch structure of the coniferous trees is formed in the same manner as for the broad-leaves trees using the special tuning rules of the L-system. The main difference deals with the foliage modelling. The foliage properties differ from species and depend on tree age as well as on the external environmental factors (e.g., water content, soil nutrients, and light availability). A generic conifer shoot model

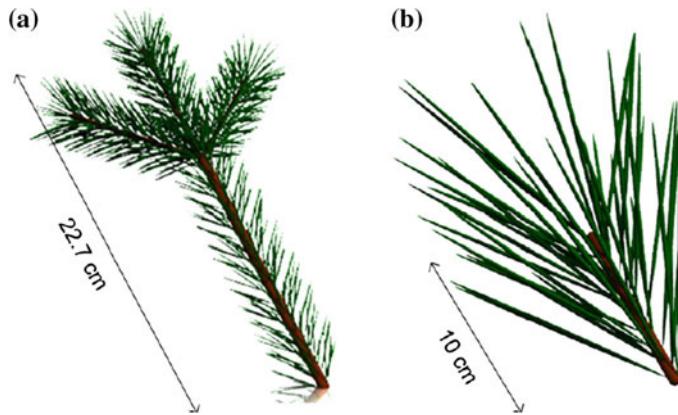


Fig. 5.17 Foliage representations for the 4 conifer species: **a** generic shoot model used for the Douglas fir, western red cedar, and western hemlocks; **b** Aleppo pine shoot model

provides the good results for some species, for example, Côté et al. [60] modelled the Douglas fir, western red cedar, and western hemlock species. Another shoot model was used for the Aleppo pine since its shoot structure differs greatly from the other species (Fig. 5.17).

A reasonable methodology to reconstruct 3D tree architectures based on the use of point clouds from the Terrestrial Laser Scanning (TLS) was proposed by Côté et al. [61]. This methodology is robust and relatively insensitive to wind and occlusion-induced artifacts in 3D point clouds. A series of 3D point clouds from the TLS were aligned and converted into a series of segments geometrically and topologically connected for the representation of a tree using the range (distance) and intensity information of the TLS in three following steps:

- Extraction of vital clues on the main branching structure of a tree including a shape of the trunk and the main branches.
- Adding other branches at locations, where the presence of foliage is very likely using the created skeleton from the previous step.
- Improvement the foliage in the center of the crown, where the TLiDAR information is sparse or absent due to the occlusion effects, as an iterative procedure based on the availability of light.

Côté et al. claimed that their algorithm will not yield an exact copy of the original plant structure. However, the proposed algorithm minimized the impact of the adverse factors like wind, occlusions, and the presence of sub-resolution structure elements that are possible during the TLS. The 3D point clouds taken by the TLS from different viewpoints were aligned into one geometric coordinate system with the software Pointstream 3DImageSuite [62]. This alignment procedure for multiple points of views is done iteratively and provides alignment accuracy less

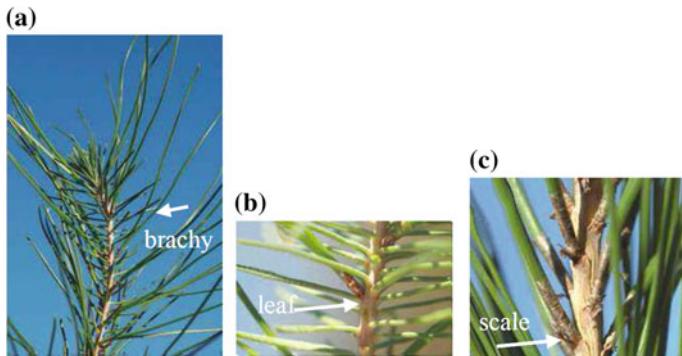


Fig. 5.18 Young trunk of conifer tree: **a** young trunk with branched blasts; **b** the scaled image showing the transition between leaves; **c** the scaled image with the axils, of which a branched blast is developing

than 4 mm for all the co-registered point clouds. The 3D point cloud N is divided into two subsets N_w and N_f for the wood and foliage components, respectively, where $N \supseteq N_w \cup N_f$. The selection of points was performed using two different threshold values t_f , t_w , which are chosen manually according to the bright/dark criterion.

Before adding the foliage, the branching structure was translated into a format compatible with the open L-system growth grammar. In this case, the L-system growth grammar consists of two sets of production rules that are applied to all branch segments. The first set of production rules assumes that every tip of a branching structure supports foliage. The second set of production rules adds new shoot structures according to the light availability at the center point of a crown. Foliage properties of coniferous trees differ from species to species and depend strongly also on the age of the plant—young versus mature—(in comparison to the broad-leaved trees) as well as on external environmental factors (e.g., water content and nutrients). The young needles are depicted in Fig. 5.18.

The detailed description of shoot model for the Aleppo and Brutia pines is considered in the works [63, 64].

5.6 Modelling Results

During experiments, a set of operators for the L-system modelling was specified. For visualization, vectors \mathbf{U} , \mathbf{L} , and \mathbf{H} along the axes in XOYZ coordinate system define a direction of a branch growing. The change of direction is computed by the rotation matrices $\mathbf{R}_U(\alpha)$, $\mathbf{R}_L(\beta)$, and $\mathbf{R}_H(\gamma)$, where α , β , and γ are the angles of rotation in 3D space:

Table 5.4 Main operators and their description

Operator	Description
$F(a)$, $A(a)$ $B(a)$, $C(a)$ $D(a)$, $G(a)$	Move in the predefined direction and draw a line ($a > 0$)
&	Move up using a rotation matrix $\mathbf{R}_U(\alpha)$
^	Move down using a rotation matrix $\mathbf{R}_U(-\alpha)$
\	Turn to the left using a rotation matrix $\mathbf{R}_U(\alpha)$
/	Turn to the right using a rotation matrix $\mathbf{R}_U(-\alpha)$
*	Move up using a rotation matrix $\mathbf{R}_L(\beta)$
%	Move down using a rotation matrix $\mathbf{R}_L(-\beta)$
~	Turn to the left using a rotation matrix $\mathbf{R}_L(\alpha)$
?	Turn to the right using a rotation matrix $\mathbf{R}_L(-\alpha)$
\$	Move up using a rotation matrix $\mathbf{R}_H(\gamma)$
@	Move down using a rotation matrix $\mathbf{R}_H(-\gamma)$
!	Turn to the left using a rotation matrix $\mathbf{R}_H(\gamma)$
#	Turn to the right using a rotation matrix $\mathbf{R}_H(-\gamma)$
[Remember the current position
]	Return to the current position

$$\mathbf{R}_U(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

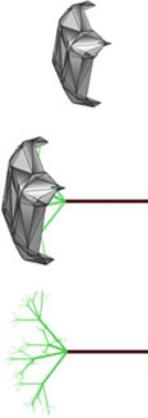
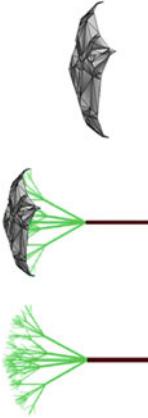
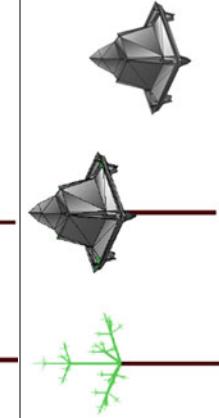
$$\mathbf{R}_L(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix},$$

$$\mathbf{R}_H(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{bmatrix}.$$

The main operators are situated in Table 5.4, based on which the enhanced growing models of tree species were developed. For each branching type (monopodial, sympodial, and ternary), various branching angles as well as proportional coefficients between lengths of the main and lateral branches were selected. Examples are depicted in Table 5.5.

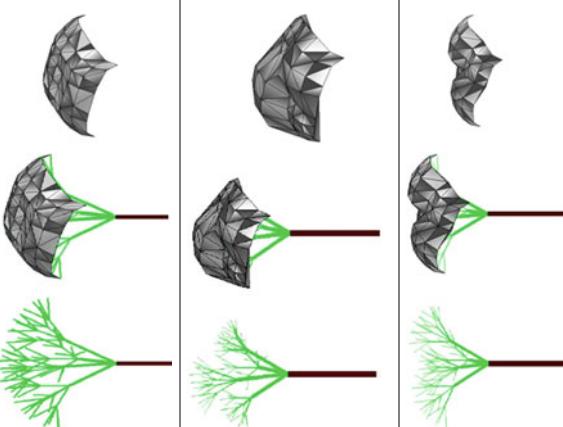
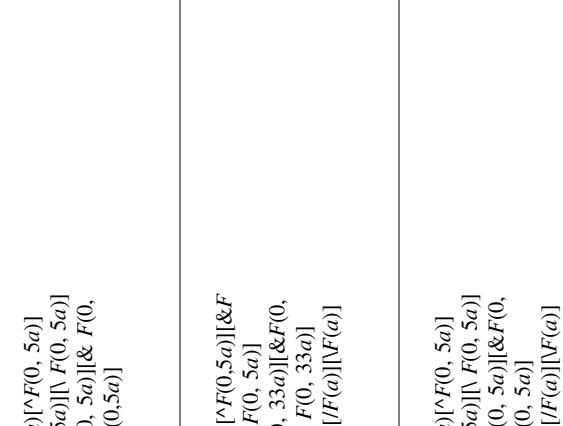
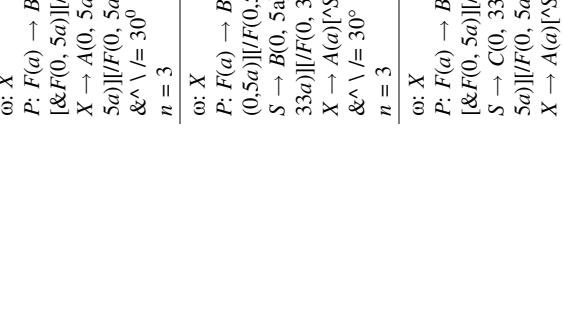
Our experimental software tool “TreeEditor”, v. 1.04 includes three main modules: the module of individual tree generation (six kinds of broad-leaved trees), the simulation module of weather conditions (simulation of wind, luminance, fog, and rain), and the module of a forest generation. A tree generation is based on the L-system with the proposed procedures, dealing with a life-cycle of a tree. A tree model is saved as an object in 3D scene, to which the standard operations, such as the rotation, scaling, or translation, can be applied. If it is necessary, a point light source can be called in a scene and software tool automatically forms a tree

Table 5.5 The developed branching rules

Branching rules	Visualization of tree model
$\omega: F(a)$ $P: F(a) \rightarrow F(a)[F(0, 5a)] [^F(0, 5a)] [\& F(0, 5a)][/F(0, 5a)][\!/F(0, 5a)]$ $\&\wedge \backslash / = 50^\circ$ $n = 3$	
$\omega: F(a)$ $P: F(a) \rightarrow F(a)[F(0, 5a)] [^F(0, 5a)] [\& F(0, 5a)][/F(0, 5a)][\!/F(0, 5a)]$ $\&\wedge \backslash / = 30^\circ$ $n = 3$	
$\omega: F(a)$ $P: F(a) \rightarrow F(a)[F(0, 5a)] [^F(0, 5a)] [\& F(0, 33a)][/F(0, 33a)][\!/F(0, 33a)]$ $\&\wedge \backslash / = 70^\circ$ $n = 3$	

(continued)

Table 5.5 (continued)

Branching rules	Visualization of tree model
$\text{o: } X$ $P: F(a) \rightarrow B(0, 5a)[^F(0, 5a)]$ $[&F(0, 5a)][F(0, 5a)][\backslash F(0, 5a)]$ $X \rightarrow A(0, 5a)[^F(0, 5a)][\& F(0,$ $5a)][\backslash F(0, 5a)][\backslash F(0, 5a)]$ $\&\wedge \vee = 30^\circ$ $n = 3$	
$\text{o: } X$ $P: F(a) \rightarrow B(0, 5a)[^F(0, 5a)][\&F$ $(0, 5a)][\backslash F(0, 5a)][\backslash F(0, 5a)]$ $S \rightarrow B(0, 5a)[^F(0, 33a)][\& F(0,$ $33a)][\backslash F(0, 33a)][\backslash F(0, 33a)]$ $X \rightarrow A(a)[^S][\& S][\backslash F(a)][\backslash F(a)]$ $\&\wedge \vee = 30^\circ$ $n = 3$	
$\text{o: } X$ $P: F(a) \rightarrow B(0, 5a)[^F(0, 5a)]$ $[&F(0, 5a)][F(0, 5a)][\backslash F(0, 5a)]$ $S \rightarrow C(0, 33a)[^F(0, 5a)][\& F(0,$ $5a)][\backslash F(0, 5a)][\backslash F(0, 5a)]$ $X \rightarrow A(a)[^S][\& S][\backslash F(a)][\backslash F(a)]$ $\&\wedge \vee = 30^\circ$ $n = 3$	

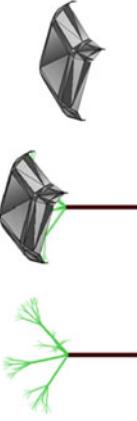
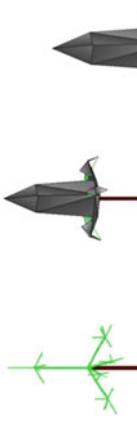
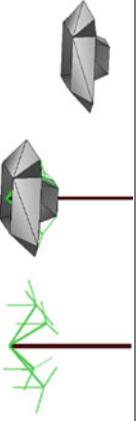
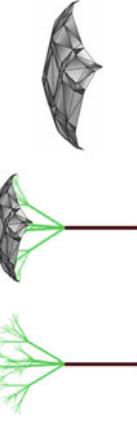
(continued)

Table 5.5 (continued)

Branching rules	Visualization of tree model
$\text{o: } [F(a)][B(a)]$ $P: F(a) \rightarrow F(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $B(a) \rightarrow B(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $\&\wedge \vee / = 30^\circ$ $n = 3$	
$\text{o: } [F(a)][B(a)][C(a)][D(a)]$ $P: F(a) \rightarrow F(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $B(a) \rightarrow B(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $C(a) \rightarrow C(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $D(a) \rightarrow D(a)[^nF(0, 5a)][\&F(0, 5a)][^lF(0, 5a)][^rF(0, 5a)]$ $\&\wedge \vee / = 30^\circ$ $n = 3$	
$\text{o: } F(a)$ $P: F(a) \rightarrow F(a)[^nG(0, 5a)][\& G(0, 5a)][^lG(0, 5a)][^rG(0, 5a)]$ $G(0, 5a) \rightarrow G(0, 5a)[^*B(0, 25a)][%B(0, 25a)][\sim B(0, 25a)][?B(0, 25a)]$ $B(0, 25a) \rightarrow B(0, 25a)[%S(0, 125a)][@S(0, 125a)][! S(0, 125a)][#S(0, 125a)]$ $\&\wedge \vee /* \sim ? = 60^\circ$ $\$ @ ! \# = 20^\circ$ $n = 3$	

(continued)

Table 5.5 (continued)

Branching rules	Visualization of tree model
$\text{o: } B(2a)[@F(a)][\$F(a)][F(a)]$ $[\#F(a)]$ $P: F(a) \rightarrow F(a)[^nF(0, 5a)][\&F$ $(0, 5a)][F(0, 5a)][\ F(0, 5a)]$ $\&\wedge \backslash / @ \$ = 30^\circ$ $! \# = 50^\circ$ $n = 2$	
$\text{o: } F(a)$ $P: F(a) \rightarrow F(a)[^nF(0, 33a)][\&F$ $(0, 33a)][F(0, 33a)][\ F(0, 33a)]$ $\&\wedge \backslash / @ V = 120^\circ$ $n = 2$	
$\text{o: } F(a)$ $P: F(a) \rightarrow F(a)[^nF(0, 5a)][\&F$ $(0, 5a)][F(0, 5a)][\ F(0, 5a)]$ $\&\wedge \backslash / @ V = 120^\circ$ $n = 2$	
$\text{o: } F(a)$ $P: F(a) \rightarrow F(a)[^nF(0, 5a)][\&F$ $(0, 5a)][F(0, 5a)][\ F(0, 5a)]$ $\&\wedge \backslash / @ V = 30^\circ$ $n = 3$	

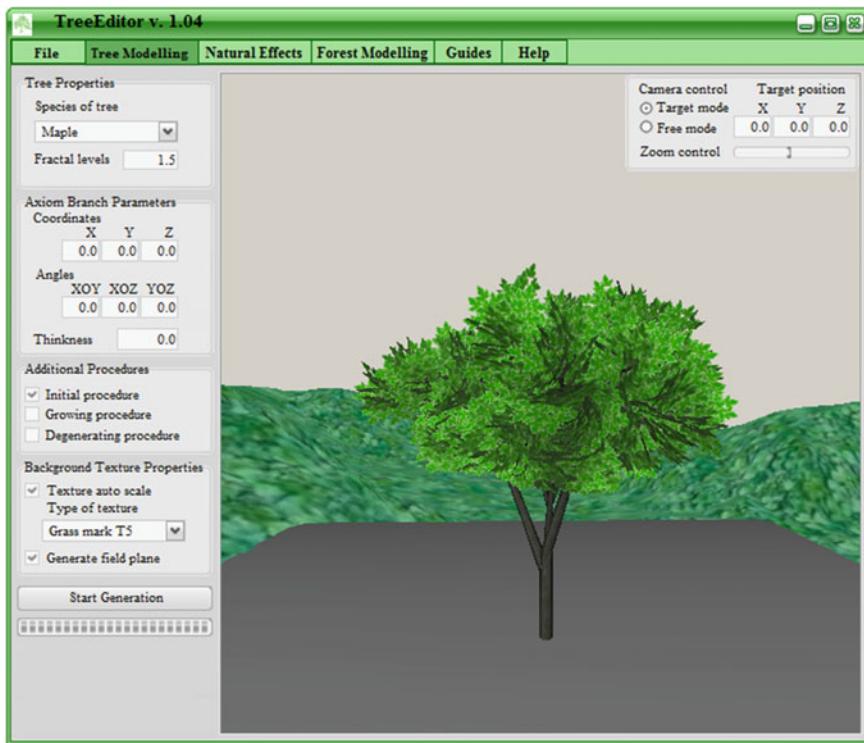


Fig. 5.19 Main screen of experimental software tool TreeEditor v. 1.04

shadows. Using the module of weather conditions, one may locate the “semi-transparent” barriers, back from which only winds penetrate. The sub-module of a wind simulation provided good visible results in real time. The module of forest generation represents a dynamic of growing for closely located trees, however, without considering the ecological factors. This module simulates the Sun movement through a scene with shadow effects and balanced day illumination. At present many functions of this module are executed manually. The main screen of experimental program is presented in Fig. 5.19, while the modelled trees are depicted in Fig. 5.20.

For program realization, the graphical jet GLScene, applying the library OpenGL as application programming interface, was used in the development environment RAD Studio 2010. Some standard objects from the jet GLScene, such as the TGLCamera (a camera object), the TGLSceneViewer (3D viewer object), the TGLMaterialLibrary (a library of materials), the TGLFreeForm (a static 3D model), the TGLLightSource (a light source), among others, were applied. Also the original procedures and components for forest modelling were designed. Software tool TreeEditor, v. 1.04 has user-friendly interface but some complex functions require a preliminary reading of instructions.



Fig. 5.20 Examples of the modelled trees

The experiments includes the modelling of individual trees in 3D scene (six tree species—a white birch, a broad-leaved maple, a green ash, a white poplar, a silky willow, a red oak) with various initial parameters and growth algorithms. Some tree models were tested by the simulation of a wind with various force values and the Sun lighting. The modelling of fog and rain is the additional functions of this software tool. Also relatively simple dynamic examples of forest modelling were received. The generated virtual scenes have a good usability according to the experts' estimations.

5.7 Conclusions

An individual tree is a basic and complex object of nature due to its branching structure, foliage, and different species of wood. It may be considered as an elementary pattern that ought to differ stochastically from the objects of the same tree species. The parametric L-system is the reasonable approach to provide the models closed to realistic with the following improvements based on the LiDAR data. This issue will be discussed in Chap. 6. The results of a tree modelling by some researchers were discussed in this chapter, including the results obtained by the authors.

References

1. Stephan M, Tobler RF, Fuhrmann AL (2003) The state of the art in realtime rendering of vegetation. VRV is Center for Virtual Reality and Visualization
2. Sen SI, Day AM (2005) Modelling trees and their interaction with the environment: a survey. Comp Graph 29(5):805–817
3. Ulam SM (1962) On some mathematical problems connected with patterns of growth of figures. Proc Symposia Appl Math 14:215–224
4. Oppenheimer PE (1986) Real time design and animation of fractal plants and trees. 13th annual conference on computer graphics and interactive techniques, pp 55–64

5. Lindenmayer A (1968) Mathematical models for cellular interactions in development, parts I and II. *J Theor Biol* 18:280–315
6. Lindenmayer A (1971) Developmental systems without cellular interaction, their languages and grammars, parts I and II. *J Theor Biol* 30:455–484
7. Cohen D (1967) Computer simulation of biological pattern generation process. *Nature* 216:246–248
8. Chomsky N (1956) Three models for the description of language. *IRE Trans Inform Theory* 1:113–124
9. Boudon F, Meyer A, Godin C (2006) Survey on computer representations of trees for realistic and efficient rendering. Technical report RR-LIRIS-2006-003, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon
10. Weber J, Penn J (1995) Creation and rendering of realistic trees. 22nd annual conference on computer graphics and interactive techniques, pp 119–128
11. Ulam S (1966) Patterns of growth of figures: Mathematical aspects. In: G. Kepes (ed) *Module, Proportion, Symmetry, Rhythm*, Braziller, New York
12. Hogeweg P (1988) Cellular automata as a paradigm for ecological modeling. *Appl Math Comput* 27:81–100
13. Cohen D (1967) Computer simulation of biological pattern generation processes. *Nature* 216:246–248
14. Meinhardt H (1982) Models of biological pattern formation. Academic Press, New York
15. Greene N (1989) Voxel space automata: modeling with stochastic growth processes in voxel space. *Int Conf Comp Graph SIGGRAPH* 1989:175–184
16. Honda H (1971) Description of the form of trees by the parameters of the tree-like body: effects of the branching angle and the branch length on the shape of the tree-like body. *J Theor Biol* 31:331–338
17. Prusinkiewicz P, Lindenmayer A (1990) *The algorithmic beauty of plants*. Springer, New York
18. Bandini S, Pavese G (2002) Simulation of vegetable populations dynamics based on cellular automata. In: Bandini S, Chopard B, Tomassini M (eds) *cellular automata*. Springer, Berlin Heidelberg
19. Campbell A, Pham B, Tian YC (2004) Mining ecological data with cellular automata. In: Sloot PMA, Chopard B, Hoekstra AG (eds) *cellular automata*. Springer, Berlin
20. Xu J, Gu B, Guo Y, Chang J, Ge Y, Min Y, Jin X (2010) A cellular automata model for population dynamics simulation of two plant species with different life strategies. *Int Conf Int Syst Knowl Eng ISKE* 2010:517–523
21. Iudin DI, Sergeyev YD, Hayakawa M (2015) Infinity computations in cellular automaton forest-fire model. *Commun Nonlinear Sci Numer Simul* 20(3):861–870
22. Aono M, Kunii TL (1984) Botanical tree image generation. *IEEE Comput Graph Appl* 4 (5):10–34
23. Lintermann B, Deussen O (1999) Interactive modelling of plants. *IEEE Comput Graph Appl* 19(1):56–65
24. Deussen O, Lintermann B (2005) *Digital design of nature: computer generated plants and organics*. Springer, Berlin
25. Xfrog inc. <http://www.greenworks.de>. Accessed 15 Aug 2015
26. Neubert B, Franken T, Deussen O (2007) Approximate image-based tree-modeling using particle flows. *ACM Trans Graph* 26(3):article 88
27. Max N, Ohsaki K (1995) Rendering trees from precomputed zbuffer views. 6th Eurographics workshop on rendering, pp 45–54
28. Shlyakhter I, Rozenoer M, Dorsey J, Teller S (2001) Reconstructing 3D tree models from instrumented photographs. *IEEE Comput Graph Appl* 21(3):53–61
29. Tan P, Zeng G, Wang J, Kang SB, Quan L (2007) Image-based tree modeling. *ACM Trans Graph* 26(3):article 87

30. Reche A, Martin I, Drettakis G (2004) Volumetric reconstruction and interactive rendering of trees from photographs. *J ACM Trans Graph* 23(3):720–727
31. Prusinkiewicz P, Lindenmayer A, Hanan J (1988) Developmental models of herbaceous plants for computer imagery purposes. *Comput Graph* 22(4):114–150
32. Prusinkiewicz P (1995) Visual models of morphogenesis. In: Langton CG (ed) *Artificial life: an overview*. MIT Press, Cambridge, Mass
33. Mech R, Prusinkiewicz P (1996) Visual models of plants interacting with their environment. *SIGGRAPH 1996*:397–410
34. Herman GT, Rozenberg G (1975) Developmental systems and languages. North-Holland, Amsterdam
35. Rezenberg G, Salomaa A (1980) The mathematical theory of L systems. Academic Press, New York
36. Mitchison GJ, Wilcox M (1972) Rules governing cell division in anabaena. *Nature* 239: 110–111
37. Salomaa A (1973) Formal languages. Academic Press, New York
38. Prusinkiewicz P (1987) Applications of L-systems to computer imagery. In: Ehrig H, Nagl M, Rosenfeld A, Rozenberg G (eds) *Graph grammars and their application to computer science*. Springer, Berlin
39. Hogeweg P, Hesper B (1974) A model study on biomorphological description. *Pattern Recogn* 6:165–179
40. Frijters D, Lindenmayer A (1974) A model for the growth and flowering of Aster Novae-angliae on the basis of table (1, O)L-systems. In: Rozenberg G, Salomaa A (eds) *L Systems*. Springer, Berlin
41. Smith AR (1978) About the cover: reconfigurable machines. *Computer* 11(7):3–4
42. Szilard AL, Quinton RE (1979) An interpretation for DOL-systems by computer graphics. *Sci Terrapin* 4:8–13
43. Prusinkiewicz P (1986) Graphical applications of L-systems. *Int Conf Graph Interface 1986—vision interface 1986 CIPS 1986*, pp 247–253
44. Prusinkiewicz P, Hanan J (1989) *Lindenmayer systems, fractals, and plants*. Springer, Berlin
45. Hanan JS (1988) *PLANTWORKS: A software system for realistic plant modelling*. Master's thesis, University of Regina
46. Prusinkiewicz P, Hanan J (1990) Visualization of botanical structures and processes using parametric L-systems. In: Thalmann D (ed) *Scientific visualization and graphics simulation*, Wiley, New York
47. Aono M, Kunii TL (1984) Botanical tree image generation. *IEEE Comput Grap Appl* 4 (5):10–34
48. Bloomenthal J (1985) Modeling the mighty maple. *Newslett ACM SIGGRAPH Comp Graph* 19(3):305–311
49. Oppenheimer P (1986) Real time design and animation of fractal plants and trees. *Comput Graph* 20(4):55–64
50. Reeves WT, Blau R (1985) Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Newslett ACM SIGGRAPH Comput Graph* 19(3): 313–322
51. de Reffye P, Edilin C, Francon J, Jaeger M, Puech C (1988) Plant models faithful to botanical structure and development. *Newslett ACM SIGGRAPH Comput Graph* 22(4):151–158
52. Favorskaya M, Zotin A, Chunina A (2011) Procedural modeling of broad-leaved trees under weather conditions in 3D virtual reality. In: Tsihrintzis GA, Virvou M, Jain LC, Howlett RJ (eds) *Intelligent interactive multimedia systems and services*. Springer, Berlin
53. Lluch J, Vivo R, Monserrat C (2004) Modelling tree structures using a single polygonal mesh. *Graph Models* 66(2):89–101
54. Teng CH, Chen YS, Hsu WH (2007) Constructing a 3D trunk model from two images. *Graph Models* 69(1):33–56
55. Zamuda A, Brest J (2014) Vectorized procedural models for animated trees reconstruction using differential evolution. *Inf Sci* 278:1–21

56. Anastacio F, Prusinkiewicz P, Sousa MC (2009) Sketch-based parameterization of L-systems using illustration-inspired construction lines and depth modulation. *Comput Graph* 33 (4):440–451
57. Zeng L, Wang G (2009) Modeling golden section in plants. *Prog Nat Sci* 19(2):255–260
58. Evers JB, van der Krol AR, Vos J, Struik PC (2011) Understanding shoot branching by modelling form and function. *Trends Plant Sci* 16(9):464–467
59. Hosoi F, Nakai Omasa K (2013) 3-D voxel-based solid modeling of a broad-leaved tree for accurate volume estimation using portable scanning LiDAR. *ISPRS J Photogramm Remote Sens* 82:41–48
60. Côté JF, Fournier RA, Egli R (2011) An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR. *Environ Model Softw* 26(6):761–777
61. Côté JF, Widlowski JL, Fournier RA, Verstraete MM (2009) The structural and radiative consistency of three-dimensional tree reconstruction from terrestrial LiDAR. *Remote Sens Environ* 113:1067–1081
62. Identik 300R portable 3D color scanner. <http://www.ariustechnology.com/>. Accessed 2 Sept 2015
63. Fady B, Semerci H, Vendramin GG (2003) EUFORGEN technical guidelines for genetic conservation and use for *Aleppo Pine (Pinus halepensis)* and *Brutia Pine (Pinus brutia)*. International Plant Genetic Resources Institute, Rome, Italy
64. Caraglio Y, Rigolot E, Pimont F (2007) Pinus halepensis mill. Architectural analysis for fuel modelling. International workshop MEDPINE 3: conservation, regeneration and restoration of Mediterranean pines and their ecosystems, pp 43–59

Chapter 6

Realistic Tree Modelling

Abstract The main idea of the realistic tree modelling is to create the algorithm, utilizing the ground-truth data of a laser scanning during the virtual modelling. The reasonable approach deals with voxelization that evaluates a density of points in order to represent 3D grid cells as the high-density, middle-density, and zero-density structures. This permits to make the hypotheses about the individual tree shapes and a tree distribution in the forest. The promising techniques, such as space colonization algorithm, graph-based modelling, self-organizing tree modelling, and inverse procedural modelling, are discussed in details. The shape, geometric, and structural measures help to estimate the obtained modelling results. The experiments show that a shape distance between a modelling tree and a sample tree influences on the similarity significantly. The best results were received using the space colonization algorithm and inverse procedural modelling.

Keywords Tree modelling · Airborne laser data · Terrestrial laser data · Space colonization algorithm · Graph-based modelling · Self-organizing tree modelling · Inverse procedural modelling

6.1 Introduction

In Chap. 5, various models for a tree generation that are mostly based on the procedural approach were discussed. On the one hand, the procedural approach is a low cost procedure and provides a compact description of the branching structures. On the other hand, this method cannot generate the realistic vegetation imagery. The source of a realistic modelling can be different, for example, based on a set of images (in particular case, two images) or a cloud of laser points. Therefore, a procedural modelling ought to be changed and fitted using realistic 2D or 3D data. A production of realistically looking plants based on a set of images is very popular in 2D [1] and 3D variants [2, 3]. The sketching modelling uses the strokes drawn by the end-user [4] with attempts of the accurate reconstruction of branch position in 3D space. Okabe et al. [5] proposed to generate 3D branching structure by

extraction of visible branches with the greedily adjust branches' orientation so that a distance among them will be as large as possible. Reche et al. [6] proposed a volumetric approach to capture and render the trees with the relatively sparse foliage using photographs. The key assumption was the following. At a reasonable distance, each pixel concerning to a tree can be interpreted as a blended projection of a number of leaves/branches and the background. Such mixture model of tree/leaf colors and background modulated by the opacity of the “tree volume” provided a transparent effect, which is easily simulated by the well-known alpha-masking approach. A recursive grid with a cumulative opacity α_i and color c_i for each cell, enclosing a tree, was created. A ray emanating from a pixel p , which traverses n grid cells, forms the intensity I according to Eq. 6.1.

$$I = \alpha_n c_n + (1 - \alpha_n)(\alpha_{n-1} c_{n-1} + (1 - \alpha_{n-1})(..(1 - \alpha_1)(\alpha_0 c_0)..)) \quad (6.1)$$

Such interactive rendering of the captured trees based on the volumetric opacity estimation with a view-dependent texturing is a good solution to simulate the shadows, lighting, and relighting in a scene. The close approach dealing with a texturing of the individual trees was developed by Samal et al. [7]. At the same time, the detailed topology of the basic branches is also possible, for example, in research [1], when the algorithm recursively adds a similar branching structure to a tree model.

Mentioned above approaches are more suitable for visualization purposes. For forest inventory, one can recommend a fusion of modelling results and airborne/terrestrial LiDAR data.

In the following Sect. 6.2, the laser scanning applications for the realistic tree modelling are analyzed. Section 6.3 provides a voxel approach for a vegetation modelling. Section 6.4 explains the algorithms, making the virtual tree models based on laser scanning data. As a result, the realistic tree models are obtained. In Sect. 6.5, some experimental results are presented. Finally, conclusions are drawn in Sect. 6.6.

6.2 Related Work

The Airborne Laser Scanning (ALS) of landscape scenes provides 3D data for estimating the biophysical and structural properties of forests, such as the tree height, trunk volume, and basal area. In practice, the canopy height distribution approach and the individual tree delineation are commonly adopted. The generalized strategy was proposed in outstanding review of Hyypa et al. [8], in which the laser scanning applications for forest inventory were classified in four categories mentioned below:

- Extraction of terrain and canopy height model.
- Feature extraction approaches (canopy height distribution and individual tree-based techniques, techniques based on the synergetic use of the aerial images and LiDAR data, and other new approaches).
- Tree species classification and forest growth, using a laser scanner.
- The use of intensity and waveform data in forest monitoring and analysis.

For optical data received from the passive sensors, both spectral and textural features are usually employed to explore the relationships between the forest attributes and remote sensing data provided by the ALS.

As mentioned in [9], the LiDAR technologies besides the ALS involve the Terrestrial Laser Scanning (TLS) and the Mobile Laser Scanning (MLS). The ALS delivers an adequate point density of large areas like forests. However, the scanning angles do not allow the adequate measurement of points for small or vertical elements [10]. The TLS systems are able to output a higher point density, including the small or vertical objects in comparison to the ALS systems. However, the TLS measurements are usually affected by the occlusions of the ground objects [11]. The MLS systems, deploying in a vehicle (van or car), produce a denser 3D point cloud than the ALS systems using more suitable scanning angles for the measurement of vertical objects. This type of laser scanning is applied successfully in the urban areas because these systems can avoid some occlusions that affect the TLS systems due to the movement of the scanning device [12]. Nonetheless for forest inventory, the ALS as the main shooting of large forest areas and the TLS as a verification shooting of restricted forest parts are recommended.

Xu et al. [13] used knowledge about the structure of trees to produce full polygonal meshes from the point clouds. The main tree skeleton is created step by step in a view of the weighted graph, in which the local neighborhood of each point is defined by a 3D distance (0.2 m in a vertical direction). The weight of each edge in the graph was determined by a length of the edge. The limited scanning resolution and branches occlusion caused a necessity to synthesize the additional branches in order to produce a plausible shape of a tree crown. The appropriate dimensions for each branch segment were estimated using the allometric theory. The authors mentioned that their algorithm can model two categories of trees: those that grow predominantly upwards and those that grow in horizontal plane significantly, for example, the Elm, Ash and Cottonwood trees. The original feature of this algorithm is a possibility to improve the smoothness, when the skeleton nodes are interpolated by a Hermite curve. Leaves are modelled considering the locations of the nearest feasible skeleton nodes applying the alpha textures.

The reconstruction of branching system, using a point cloud, is a crucial problem in a tree modelling. The branches may be hidden by the leaves. Therefore, the algorithms are developed under the assumption that trees have not foliage, for example, during a winter. Yan et al. [14] proposed to reconstruct a pipeline from the laser scanned data points for simulation of a branching system. This algorithm includes three main steps, such as a segmentation based on a variational k -means clustering algorithm, a reconstruction using an adjacency graph, and a modelling by

application of the cylindrical components, which are fitted according to B-spline interpolation. An approach for extracting the topological structure of individual trees from the unfiltered 3D TLS point clouds was proposed by Schilling et al. [15]. First, a tree trunk was extracted by the circular Hough transform to a set of 3D points distributed along the z axis. A trunk can be reconstructed as a set of cross-sections having the circle- or arc-like structures, depending on the scan coverage. The circular Hough transform is represented in details in [16]. Second, a voxel space representation of raw point cloud is created due to the huge number of points, for example, up to 10 million points per tree (obtained during the TLS shooting). A voxel space representation reduces the total number of elements but, hereinafter, some voxels are removed by a thresholding operation in order to make a tree isolated. The authors employed a graph theory algorithm called as the depth first search [17] to retrieve a tree topology.

The fast skeletonization of the botanic trees from a point cloud is represented in research [18]. The skeletonisation algorithm includes three steps: the extraction of a graph from an octree organization, the reduction of this graph to the skeleton, and the embedding of the skeleton into a point cloud. The single point clouds were obtained by the TLS, and the authors deal with the unorganized point clouds, which do not contain the neighboring point information. As a result, they built the octree graph, which is interpreted as a bidirectional graph, representing a tree topology. Such approach demonstrates a complex processing of unorganized point clouds that is suitable to avoid a necessity of the neighboring point information in the modelling algorithms.

6.3 Voxel Modelling of Vegetation

The result of the ALS measurements is a non-uniform 3D point cloud. The methodology deals with a separation of 3D point cloud into the pole-like objects (in our case, trees, shrubs, buildings sometimes) through 3D voxelization (Sect. 6.3.1), 2D analysis of horizontal cuts in a point cloud (Sect. 6.3.2), and 3D reconstruction of the individual trees (Sect. 6.3.3). Consider this methodology in details.

6.3.1 Voxelization

The goal of current discussion is to separate a forest part into the individual trees under the assumption that a density of laser points is higher on the external surface of foliage. Voxelization is an often applied technique in the mentioned aspect. Our contribution is to expand a voxelization technique by space colonization algorithm, graph-based approach, among others, (Sect. 6.4) in order to receive a good realistic model of a tree branching. Consider the main terms and statements of voxelization.

The point cloud from the ALS measurements is a set of 3D points distributed irregularly. Our task is to associate the clusters of points with the individual trees. The studying space is transformed in 3D regular grid. In particular case, a cell of such grid is a cuboid with the equal sizes that corresponds to an elementary structure—volume element (voxel). Then a density of points into each cell is determined with the following clustering of close cells into the high-density, middle-density, and zero-density structures. The analysis of their locations permits to make the hypotheses about individual trees' distribution in a forest.

First, only high-density structures corresponding to the foliage surfaces, trunk, or man-caused buildings are remained and then distributed as the sparse object structures. Data in such sparse object structures are more missing than full. Nobody can guarantee the symmetric representation of the trees in a space of the laser points due to various types of the ALS. It will be good if a trunk position is detected as a reliable feature of a tree. Second, the middle-density structures as the partial penetrable foliage inside a crown are located in the sparse object structures. This verification step is very useful for the assessment of the crown shapes. Third, the zero-density structures mean an empty space separating trees one from others. The high-density, the middle-density, and the zero-density structures are stored in very simple data format, including 3D coordinates of a cuboid vertex, which is accepted as the first vertex in any cell of grid,—XXXXYYYYZZZZ with last digit number—a number of the laser points in the current cell N . The received structures are associated with a sparse object structure with first digit number denoted by S . Thus, full format data is represented as the $SXXXXYYYYZZZZN$ description. A scheme of the high-density and the middle-density voxel representations is depicted in Fig. 6.1.

6.3.2 2D Analysis in Horizontal Cuts

After voxelization, 3D grid is divided in horizontal cuts in order to identify a crown shape. Here, an intersection of the tree crowns is possible. The success is achieved, when the larger part of a crown contour is segmented reliably and the intersected part is interpolated similarly. The 2D analysis includes three following steps:

- Forming a set of crown cuts in a horizontal plane.
- Segmentation of individual trees, using several points of tree apices with maximum values of Z coordinates.
- Interpolation of crown contour in the case of crowns intersection.

The result of this procedure is a set of 2D segments associated with a z coordinate that describe a shape of individual tree. Additional useful information can be extracted from the digital photographs. Using techniques discussed in Chap. 4 as well as the expert's assessment, the assumption about the tree species with shapes, such as cone, paraboloid, cylinder, sphere, or hemisphere, can be formulated. This assumption helps to predict the geometrical projections of crown cuts in the horizontal planes.

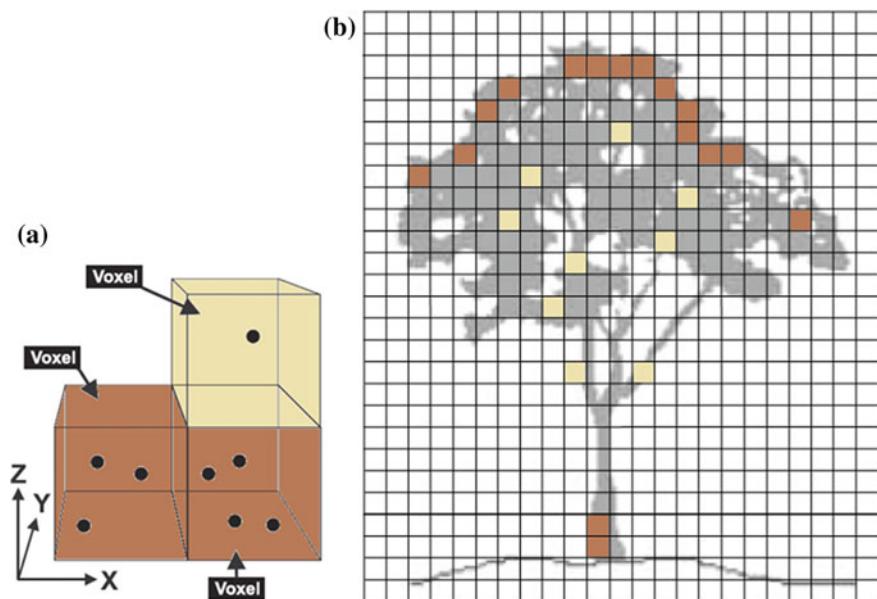


Fig. 6.1 Voxel representation: **a** high-density (brown) and middle-density (yellow) voxels; **b** distribution of the high-density and the middle-density voxels in a plane XOZ

6.3.3 3D Reconstruction

The 3D reconstruction supposes a final separation of a point cloud into the sets, corresponding to the individual objects. Some of laser points are rejected due to the random properties of a laser scanning. The 2D or 3D images of a forest part in representation of the laser points are created. Usually such images are labeled by the colors as it is shown in Fig. 6.2. The obtained modelling scene can be scaled or separated principally in the individual trees, other vegetation, or the man-caused objects.

6.4 Improvement of Tree Models by Realistic Data

Due to that the forest ecosystems demand a combination of the ALS and photography data, the procedural tree modelling ought to be improved by the algorithms that are considered the ground-truth ALS data. A variety of such algorithms is not wide. For these purpose, the space colonization algorithm (Sect. 6.4.1), the graph-based modelling (Sect. 6.4.2), the self-organizing tree modelling (Sect. 6.4.3), or the inverse procedural modelling (Sect. 6.4.4) can be recommended.

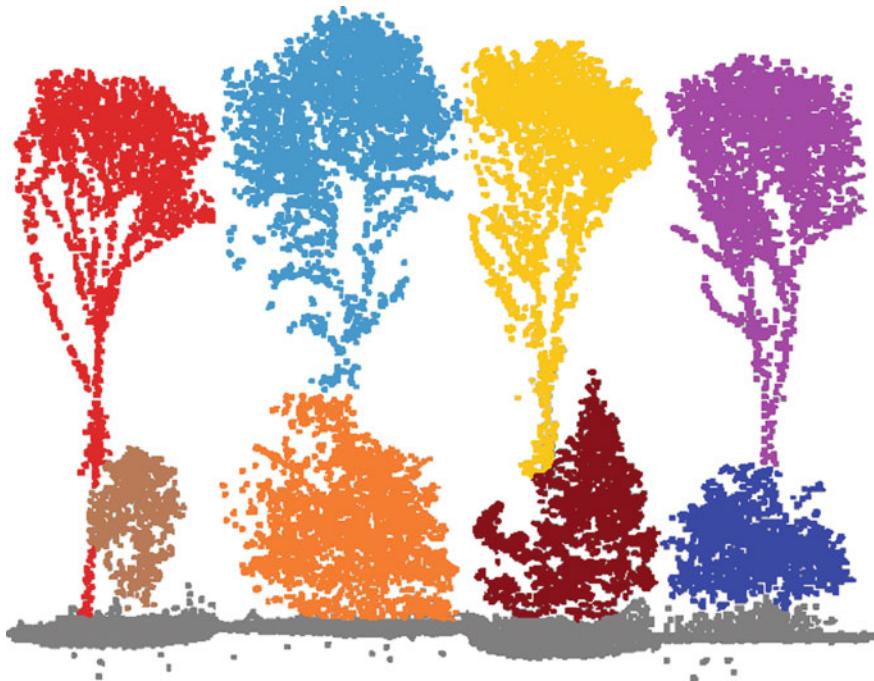


Fig. 6.2 The labeled modelling scene of a forest part

6.4.1 Space Colonization Algorithm

It is reasonable to consider some factors and features of a growing of the young and mature trees and shrubs in a modelling process. Thus, the architecture of the young trees is a result of regular distributed buds, while one cannot say the same regarding to the mature trees, when some buds produce the branches or flowers and other buds remain dormant or abort. This is one of the reasons of an architecture irregularity [19]. Also the initial directions of a branch growth are significantly modified by the branch reorientation, tropisms (the tendency of branches to turn in a particular direction), or a mechanical bending. Due to the environmental conditions, some twigs become major limbs, while others remain the small branches. These phenomena ought to be considered during a tree modelling.

The original space colonization algorithm was proposed to model the leaf venation patterns [20]. An open venation is represented by a tree graph $G = \langle V, E \rangle$. The nodes $v \in V$ of this graph represent the small segments of veins called as vein nodes, which are also embedded in the leaf blade. The adjacent nodes are connected by the edges $e \in E \subset V \times V$. The algorithm proposed by Runions et al. [20] simulates three processes iteratively, such as the leaf blade growth, the placement of auxin sources (according to the biological hypothesis), and the growth of additional

new veins. Three models of a leaf growth are supported. The marginal growth is modelled by scaling the leaf edge with respect to the attachment point of the leaf to the petiole. The uniform growth is regarded as a scaling of the entire leaf, including the veins and auxin sources in a time instant t . The non-uniform anisotropic growth reflects a possible deforming of the initial leaf shape over time. The placement of the auxin sources is assigned farther than a threshold birth distance b_s from all other sources, and farther than a threshold birth distance b_v from all vein nodes. It is assumed that each source influences on the closest vein node (or random vein node in the case of multiple vein nodes). There may be several sources that influence on a single vein node $v \in V$ denoted as $S(v)$. If a set $S(v)$ is not empty, a new vein node v' is created and attached to v by edge (v, v') . The node v' is positioned at a distance D from v , in the direction defined as the average of the normalized vectors toward to all the sources $s \in S(v)$ in a view of Eq. 6.2.

$$v' = v + D \frac{\bar{n}}{\|\bar{n}\|} \quad \bar{n} = \sum_{s \in S(v)} \frac{s - v}{\|s - v\|} \quad (6.2)$$

Notice that the space colonization algorithm can be applied in applications, differing from a simulation of the natural objects. Thus, a virtual crowd simulation inspired by a modelling of biological patterning, was developed by de Lima Bicho et al. [21]. These authors simulated a crowd behavior as a competition for space among the moving agents, considering the collision avoidance, relationship of crowd density, and speed of agents.

Runions et al. [22] developed the basis of the space colonization algorithm for a tree modelling (as 3D extension of space colonization algorithm against 2D leaf venation modelling), using a concept of competition for the space and resources. The input data is 3D envelope of a tree crown specified by any method that can select the points, lying inside or outside of the enclosed volume. Also inside of a crown volume, a set of the attraction points distributed by the end-user control is seeded. These points indicate how a branching structure ought to grow. They are removed after reaching by a branch. It is considered that the attraction points were uniformly distributed in a crown volume. This sub-algorithm is controlled by three parameters: the number of attraction points N , the kill distance d_k (kill distance serves as a threshold parameter), and the radius of influence d_i . A decreasing of N value and an increasing of d_k value create the sparse architecture of crowns. The reduced number N of the attraction points leads to the irregular branches. The increased value of a kill distance results in expansion of the attraction points set, affecting on the individual branch tips. The decreased radius of influence d_i tends to a wiggly or gnarly appearance of the branch tips that violates a tree architecture. One can find the multiple figures, illustrating the influence of these tree parameters on building of a tree skeleton, in researches [22, 23].

The tree skeleton is formed iteratively, when the short branches extend the skeleton in the direction of nearby attraction points. Two stopping criteria exist like as the limited iterative steps or when all attraction points have been removed. The

structure of a tree skeleton may be further improved, for example, by the removal of the closed and duplicate attraction points, the enhancement of curves between the branches, and the replacement of a skeleton model by a pipe model finally [24]. First, all branch tips have the same initial radius r_0 . Second, a calculation proceeds from the branch tips toward the tree base. If branches of radii r_1 and r_2 meet in a branching point, then the radius r of the supporting branch is computed by Eq. 6.3, where n is a tuning parameter (usually between 2 and 3 according to recommendation from [25]).

$$r^n = r_1^n + r_2^n \quad (6.3)$$

Third, an orientation of the generating curves is calculated in a manner that minimizes the twist between the consecutive cross-sections. Forth, leaves, flowers, and/or small branches are added to the tree structure.

The space colonization algorithm is based on a competition for a space as the key factor, determining the branching structure of a tree. The suggestions of improvement proposed by Runions et al. [22] are concerned to the repetitively testing of a set of the attraction points for proximity to the tree nodes, using 3D Voronoi diagram with employing of 3D Delaunay triangulation routines. Nowadays, this algorithm is one of the popular decisions in a tree modelling. A scheme of the space colonization algorithm is depicted in Fig. 6.3.

Livny et al. [26] employed the series of global optimizations to consolidate a laser point cloud received from the TLS into the skeletal structures of trees, including a single tree and the nearest trees. Such automatic algorithm reconstructs the major skeletal branches, forming the Branch-Structure Graph (BSG) of a captured tree from a noisy and incomplete point ground-truth data.

The BSG is defined as a spatially embedded and connected directed acyclic graph with a root node corresponding to the base of a tree. The BSG nodes are connected by the edges as the straight lines. All nodes lie spatially in the center of

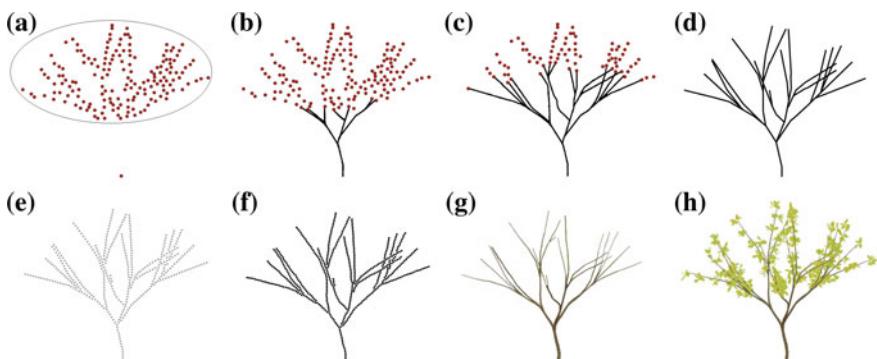


Fig. 6.3 Scheme of space colonization algorithm: **a** envelope with the attraction laser points, **b–d** generation of a tree skeleton, **e** node decimation and relocation, **f** subdivision, **g** construction of branch cylinders, **h** addition of foliage

the tree branches. A branch is a simple branch if it contains no branching points and is represented by the Branch-Chain (BC). The BC is considered as a sub-graph of the BSG. The main optimization criterion is to get the reconstructed BSGs that will be geometrically close to the input point cloud. Additional criteria are based on a set of the weak assumptions on the length, thickness, smoothness, and density of tree branches, which were formulated by Honda as the biological growth properties of the real-world trees [27]:

- The BCs are smooth because they are reflected by the small bending angles between the adjacent edges.
- The BCs are longer and thicker near the root of a tree and shorter and thinner near the crown.
- The density of the BCs is inversely proportional to their corresponding thickness.

Usually laser points that are close together are likely to belong to the same branch. The goal is to find a tree structure with minimum overall edge length as a graph with edges (u, v) weighted by the Euclidean distance $\|u - v\|^2$. The Dijkstra's algorithm was used to extract a minimum-weight spanning tree from this graph [28]. To ensure that all root nodes are contained in this spanning tree all nodes are connected with the temporary zero-weight edges. Hereinafter, the zero-weight edges are removed from a spanning tree, creating a forest $\{T_1, \dots, T_n\}$ of the initial BSGs.

The crucial issue of such graph-based approaches is to construct an orientation field, for example, like as in [3], on the vertices or oriented graph. Let $v \in T_i$ be a vertex with parent v_p . First, the difference $\Delta O(T_i)$ between orientations o_v of both vertices v and v_p is minimized by Eq. 6.4.

$$\Delta O(T_i) = \sum_{v \in T_i} \left(\frac{c_{v_p} + c_v}{2} \|o_{v_p} - o_v\| \right)^2 \rightarrow \min \quad (6.4)$$

Second, the difference $\Delta E(T_i)$ between the orientation at v and the direction of the edge $e(v, v_p)$ is minimized by Eq. 6.5.

$$\Delta E(T_i) = \sum_{v \in T_i} \left(c_v \left\| o_v - \frac{e(v_p, v)}{\|e(v_p, v)\|} \right\| \right)^2 \rightarrow \min \quad (6.5)$$

The smooth orientation field is construct so that minimizes a sum $\Delta O(T_i) + \Delta E(T_i)$. The orientation contribution of the branches is implemented by the weights (c_v, c_{v_p}) , guaranteeing that the noisy branches do not distort the orientation field near a trunk. A minimization of the error functions was introduced in the same manner. Livny et al. [26] claimed that their method had been applied to reconstruct various species of trees; moreover, the multiple overlapping trees can be handled without a pre-segmentation.

6.4.2 Graph-Based Modelling

The irregular individual branches and the multiple levels of details are the typical features of a tree modelling. The procedural method can be applied successfully to model a global tree shape in the scaled image in order to construct a tree skeleton, including the trunk and main branches, and to form a tree crown from small branches as the hierarchical structure. Some investigations are devoted to the modelling using the graphs [29, 30].

The original idea of a procedural modelling is to create a graph with the random edge weights for the detailed tree representation. Xu and Mould [31] developed this approach, starting a simulation as the sequences of graphs with the following linking of the endpoints in all graphs into a single root node. Hereinafter, they proposed to build the initial graph, using the Yao graph in order to reduce the number of edges without the quality degradation. Such graph includes the least-cost paths from a single root node to the destination nodes. As a result, Xu and Mould [32] developed a method that creates a wide range of the tree models by varying the graph shape and edge weights and incorporates the environmental factors into a tree growth process. However, the resolution of this model is limited by the spacing of a graph, when the small-scale features (e.g., tiny twigs) are represented by a high-resolution graph, therefore, the high memory cost graph. Notice that a point cloud of tree data obtained by 3D scanning can be the source of a tree skeleton in order to form a graph [18, 33]. Consider the algorithm based on a graph building in details. The basic algorithm includes the following steps [32]:

- Build a graph and set the edge weights randomly.
- Choose a node to be the root point and some nodes as endpoints of the structure.
- Find the least-cost paths from the endpoints to the root.
- Create the geometry around the path segments and render the resulting model.

Such graphs may be regular or irregular, as it is shown in Figs. 6.4a, b, respectively. However, an irregular graph avoids the lattice artifacts and makes a visibility of a tree model very high. Afterward, Germer and Strothotte [34] developed the orthant neighborhood graphs, yielding a simple and decentralized spatial data structure based on the weak spanners. The proposed algorithm involved the dynamic insertions, deletions and movements of the points as well as the range searching and other proximity queries. For a comparison, orthant neighborhood graph is depicted in Fig. 6.4 c.

A tree model has 3D structure. That is why, it is difficult to use the graphs from Fig. 6.4 in 3D variant. The Yao graph, the edges of which are determined by dividing of the space surrounding in a given node into the sections with rays (in 2D) or planes (in 3D), can be recommended for these purposes [35]. The 2D Yao graphs with six and eight sections are depicted in Fig. 6.5. An integer parameter $k \geq 6$, which is the number of rays and sectors, characterizes the Yao graph.

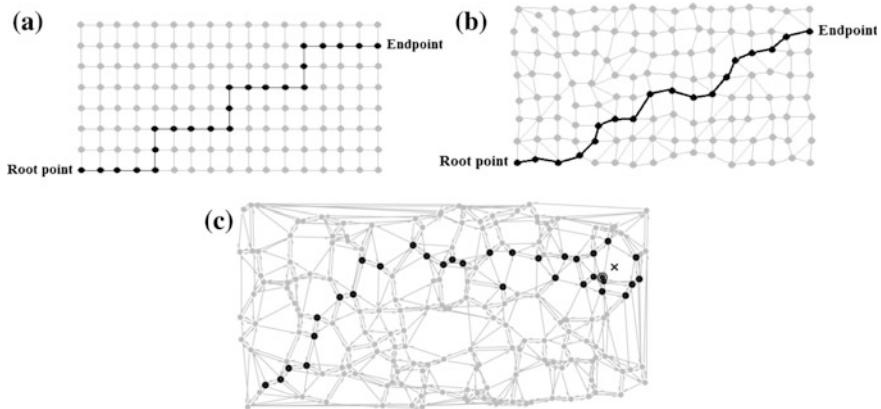


Fig. 6.4 Examples for graph representation: **a** regular graph, **b** irregular graph, **c** orthant neighborhood graph from [34]

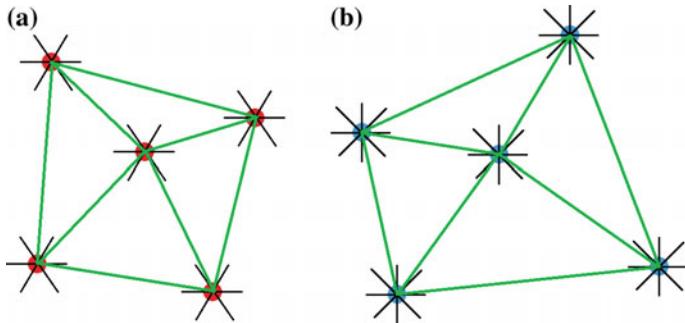


Fig. 6.5 2D Yao graph with: **a** six sections, **b** eight sections

The tree modelling, using the graphs, has some limitations. It forms the broad-leaves tree models only with a lower graph resolution due to a memory restriction. A view of branch (path in a graph) is under more consideration in comparison to the tip position. Also some environmental factors are not explored.

6.4.3 Self-organizing Tree Modelling

Many authors consider that a tree model is largely determined by its branching pattern. Another approach deals with the genetically-controlled distribution of buds, from which branches arise [19]. Some trees and shrubs commonly produce many buds, most of which do not develop into the lasting branches. Sachs and Novoplansky [19] were the first, who proposed the self-organizing model of a tree

development. Based on the concept of the self-organizing, Pałubicki et al. [23] integrated a local control of the branching geometry, a competition of the buds and branches for space or light, and a regulation of this competition through an internal signalling mechanism into a model of a tree development. The self-organizing modelling has the positive features mentioned below:

- The self-organization simplifies the modelling process due to the well-balanced branch distributions, using the generative algorithm.
- A visual realism of the tree models is improved essentially. As a result, a wide range of highly realistic trees can be generated.
- The apical control of a tree development ensures a wide range of the tree forms, in particular with a conspicuous tree trunk and without a pronounced trunk.
- These models can be further extended by the interactive manipulations with the procedural brushes in order to generate the objects most similar to the objects in the natural environment.

In each iteration, two parameters, such as an accumulative amount of light (a number Q) and the optimal direction of a shoot growth (a vector \mathbf{V}) for each bud, are estimated using the modified space colonization algorithm and the shadow propagation calculus. The key modification of the space colonization algorithm consists in consideration only those buds, from which new branches can be produced. Each bud is surrounded by a sphere with radius ρ and a conical perception volume of a growing with a perception angle θ and a distance r . Typical values of these parameters are the following: $\theta \approx 90^\circ$, $\rho = 2IL$, and $r = (4/6)IL$, where IL is a length between two nodes in a trunk or a branch. Therefore, a space available for a tree growth is a set S of marker points M . In the simplest case, these points are generated with a uniform distribution. Then in each iteration, some markers are removed from the set S , and the buds compete for the remaining points. The optimal growth direction \mathbf{V} is calculated as the normalized sum of the normalized vectors towards all markers in $S(A)$, where A means the remaining buds in a perception volume.

The shadow propagation model evaluates a coarse estimate of the exposure of each bud to the light. The space is divided into a grid of voxels; each voxel (i, j, k) has a “shadow value” sh , which is initially set to zero. A bud A located in the highest voxel (I, J, K) creates a pyramidal penumbra from the voxels underneath. The affected voxels have indices $(i, j, k) = (I \pm p, J - q, K \pm p)$, where $q = 0, 1, \dots, q_{max}$ and $p = 1, 1, \dots, q$. The shadow value sh in each affected voxel is increased by $\Delta sh = ab^{-q}$, where $a > 0$ and $b > 1$ (both ones are the end-user defined parameters). The light exposure Q of a sample bud B in a voxel (i, j, k) is calculated as $Q = \max(C - sh + a, 0)$, where C is a constant representing full exposure. Here, it is assumed that a bud does not cast a shadow on itself. The optimal growth direction \mathbf{V} is computed as the negative gradient of the shadow value or by selecting the voxel with the lowest shadow value within the perception volume of the current bud.

Pałubicki et al. [23] proposed the empirical expressions Eq. 6.6 to estimate the amount of resource, reaching a branching point between the continuing main axis (v_m) and the lateral branch v_l , where v is an amount resource, Q_m and Q_l are an amount of light accumulated in internodes m and l , respectively, $\lambda \in [0, 1]$ is a parameter, which controls a process of tree generation. Resource allocation is biased towards the main axis ($\lambda > 0.5$), not biased ($\lambda = 0.5$), or biased toward the lateral branch ($\lambda < 0.5$).

$$v_m = v \frac{\lambda Q_m}{\lambda Q_m + (1 - \lambda) Q_l} \quad v_l = v \frac{(1 - \lambda) Q_l}{\lambda Q_m + (1 - \lambda) Q_l} \quad (6.6)$$

The modelling method produces a plausible visibility for various environmental models, the branching patterns, and the internal regulatory mechanisms. The complex forest scenes can be generated using this approach. Also the resulting shapes can be controlled interactively, e.g., by sketching [36].

6.4.4 Inverse Procedural Modelling

The previous discussed models react to the varying environments differently. Usually the reconstructed and interactive methods provide the static tree models, while the procedural methods are most sensitive to the environmental factors. The main drawback of a procedural modelling, regarding the burdensome manual process of parameters definition, can be overcame by the inverse modelling, i.e., by an automatically computing the parameters of a procedural model for the given tree.

Recently, various inverse modelling approaches have been developed. Ijiri et al. [37] proposed a technique for 2D synthesizing element arrangement patterns from the user-input example. The main idea was to combine the texture synthesis methods based on a local neighborhood comparison and the procedural modelling systems based on a local growth. The inverse procedural modelling by generating the parametric context-free L-systems was developed by Stava et al. [38]. This algorithm obtains an input vector image with objects formed as groups of line segments or Bézier curves and produces the L-system based on the given input. Then the transformation spaces to detect sequential and branching structures of repeating elements with the changing angle, size, and rotation are built in order to create the clusters. Based on the cluster analysis, the most relevant sequence of elements was selected with the following generation of the L-system rules. An extension of this approach to 3D was presented by Bokelohet al. [39]. These authors built a shape grammar that collects and edits the operations and analyzes their dependencies. As a result, the context-free hierarchical rules and grid-based rules were derived directly from the model without user interaction. However, all mentioned above approaches of inverse procedural modelling could be obtained only for highly regular and symmetric input data but not for stochastic data.

The stochastic procedural model of the biological trees is more attractive approach. Notice that the stochastic grammar can be successfully applied in the fields of the urban reconstruction and city modelling. For example, Martinovic and Van Gool [40] presented an approach to create automatically 2D attributed stochastic context-free grammars from a set of the labeled building facades based on the Bayesian model merging.

Stava et al. [41] introduced a framework for the stochastic inverse procedural modelling of the trees involving a compact parametric procedural model, an efficient similarity computation for the comparison of two trees, and a method for the automatic determination of input parameters of the procedural model for the given input 3D polygonal tree model or models, using the Monte Carlo Markov Chain (MCMC) optimization. The LiDAR scans, the Xfrog library models, the SpeedTree models, or the trees generated by the Open L-systems may be among multiple geometric tree models. The model of Stava et al. includes a set $\bar{\phi}$ from 24 parameters, describing an influence of internal and external factors on a tree shape. The geometric parameters are determined a tree shape, viz. the internode lengths of the branches and phyllotaxis. The actual branching structure and its density are controlled by the bud fate parameters. Light and gravity are the main environmental factors, influencing on a tree model. They are concerned to the environmental parameters. A growth time is the 25th parameter, controlling the age and size of the generated trees.

Some geometric parameters control the number and relative orientation of the lateral buds along a branch, while the additional geometric parameters describe the variance of the individual angular variables, including the orientation of apical buds. The initial internode lengths of branch segments decrease with the increasing tree age that is considered by the internode length age factor in the model of Stava et al. [41]. The internal tree structure strongly depends on the fate of apical and lateral buds (this is controlled by the bud fate parameters). A single shoot may contain many buds but many of them cannot grow under the biological or environmental factors. Thus, the parameters that specify a probability for the apical and lateral buds growth during each growth cycle are set. The living buds can either form a new shoot or remain dormant under the apical and lateral light factors and the plant hormones [42]. In normal conditions, the shoots grow in the direction defined by their source bud but under the light (phototropism) and gravity (gravitropism) factors they can bend toward or away from an initial direction. Notice that such bending can be simulated using the approach of Palubicki et al. [23]. Also, a model of Stava et al. contains an increasing pruning factor as a result of the shadowed branches and possible human or animal intervention and the tree growth parameters. During a growth, the branches bend down under the influence of the increased mass of its child branches. This is especially important for many coniferous and other softwood species.

In order to optimize the modelling parameters, Stava et al. [41] introduced the tree similarity measures as mentioned below:

- The shape distance $d_S(\tau_1, \tau_2)$ is defined between two models τ_1 and τ_2 of the botanic trees. This measure evaluates the difference between the overall shapes of the trees. First, the geometric mean of the crown μ_c is computed in order to divide the crown into the upper μ_{uh} and lower μ_{lh} halves, respectively. For each halve, a vector of features as a set of descriptors $\{\lambda_{S,c}\}$ is constructed including a height h , an average radius in the horizontal principal directions of the crown r_{\min} and r_{\max} , a leaf-branch density p_{lb} . Second, the differences δ_{λ_S} between all descriptors λ_S of two trees τ_1 and τ_2 is estimated by Eq. 6.7, where $\sigma_{\lambda_S,i}$ is a normalization factor in Eq. 6.8, which is obtained empirically.

$$\delta_{\lambda_{S,i}} = 1 - \exp\left(-\frac{(\lambda_{S,i}(\tau_1) - \lambda_{S,i}(\tau_2))^2}{2\sigma_{\lambda_{S,i}}^2}\right) \quad (6.7)$$

$$\lambda_{\lambda_{S,i}} = \frac{1}{3} \min(\lambda_{S,i}(\tau_1), \lambda_{S,i}(\tau_2)) \quad (6.8)$$

Third, the shape distance is computed as the Minkowski distance of order p of the vector $\bar{\delta}_S$. The order p is set from the experiments. However, if $p = 1$, the distance is essentially just an averaged sum of the differences, and with increasing p , the weight of larger differences becomes more.

- The geometrical distance $d_G(\tau_1, \tau_2)$ compares the global geometric branching properties. The geometric descriptors λ_G of a tree (branch geometry) are defined by a set of statistics: a total length and a maximal thickness of the branch, an accumulated angular deflection, a ratio between the endpoint distance and a real length of a branch, and the angles between a direction of the branch endpoints and the horizontal plane, between the branch and its nearest sibling, between the branch and its parent branch. Then the geometric distance d_G is computed using the Minkowski length (Eq. 6.7).
- The structural distance $d_T(\tau_1, \tau_2)$ detects the local differences between the individual branches based on their position within the trees. This descriptor evaluates the edit distance between two tree graphs as a minimal cost of transforming all nodes of one tree graph into the nodes of the other tree graph. The assign, insert, and delete operators regarding to the nodes as well as the split and merge operators regarding to a tree topology were introduced. Then the cost functions of operators and edit distance $d_N(t_1, t_2)$, where t_1 is a root of T_1 and t_2 is a root of T_2 , are evaluated. The edit distance is calculated by Eq. 6.9, where ε is a unique symbol, which does not include in a graph [43].

$$d_T(\tau_1, \tau_2) = \frac{d_N(t_1, t_2)}{2 \max(d_N(t_1, \varepsilon), d_N(\varepsilon, t_2))} \quad (6.9)$$

The total distance measure $D_T(\tau_1, \tau_2)$ is defined as the weighted sum of the shape $d_S(\tau_1, \tau_2)$, the geometrical $d_G(\tau_1, \tau_2)$, and the structured $d_T(\tau_1, \tau_2)$ distances with the weights w_S , w_G , and w_T , respectively, $w_S, w_G, w_T \in [0...1]$ provided by Eq. 6.10.

$$w_S + w_G + w_T = 1 \quad (6.10)$$

A value $D_T(\tau_1, \tau_2) = 0$ means the exact similarity and $D_T(\tau_1, \tau_2) = 1$ reflects the full dissimilarity of two botanic trees.

Thus, the optimal parameter values $\bar{\varphi}$ and the optimal growth time t can be generated by the parametric model M . During optimization, the similarity measure SM is maximized by minimizing a sum of the distance measures $D_T(\tau_1, \tau_2)$ between all trees and the reference input tree τ^r . This task cannot be solved analytically. The approximated solution, using the Monte Carlo numerical integration technique, is computed by a randomly sampling in the state space Ω^M , where each sample represents a single generated tree τ^M . The approximated optimization is written in a view of Eq. 6.11, where $\{w_j\}$ are the random samples according to the density $p(w)$, which is specific for every model $M(\bar{\varphi}, t)$. A number of random samples $\{w_j\}$ determines the accuracy of an approximated solution.

$$SM = \arg \min_{\bar{\varphi}_{M,t}} \left(\sum_{w_j} D_T(\tau^r, \tau^M(w_j)) \right) \quad (6.11)$$

The inverse procedural modelling of the trees is very reasonable and perspective approach that combines the procedural L-systems, the scanned and reconstructed models, and even the hand-modelled trees, using the plant libraries.

Also the important issue is the development of algorithms for foliage and leaf generating regarding to the scale of a representation. This issue will be considered in details in Chap. 7. Here, the research of Runions et al. [20] can be mentioned as the productive approach for a high resolution of leaf visualization.

6.5 Experimental Results

During experiments, some models of the trees were generated using four basic algorithms, such as the space colonization algorithm, the graph-based approach with an irregular structure, the self-organizing modelling, and the inverse procedural modelling. The evaluation of a trees' similarity between the modelling trees and the ground-truth trees, representing by a set of images with small amount of foliage, was accomplished according to the similarity measures proposed by Stava et al. [41] (Sect. 6.4.4). The normalized similarity results of a tree modelling by various algorithms are located in Tables 6.1, 6.2, 6.3 and 6.4. For each tree species, there were used 6–10 samples. Values of the weighs in Eq. 6.10 were assigned as $w_S = 0.4$, $w_G = 0.3$, and $w_T = 0.3$, respectively.

Table 6.1 The normalized similarity results, using the space colonization algorithm

Tree species	Shape distance measure, %	Geometrical distance measure, %	Structural distance measure, %	Total distance measure, %
Elm	78.9	69.7	70.8	73.7
English oak	90.6	74.5	88.7	85.2
White birch	85.1	40.9	57.3	63.5
Norway maple	83.4	58.4	62.5	69.6
European ash	76.8	61.4	64.3	68.4
Black poplar	80.3	56.2	61.6	67.5
Salix alba	90.8	38.4	53.6	63.9
Bird cherry	88.4	57.2	42.5	65.3

Table 6.2 The normalized similarity results, using the irregular graph structures

Tree species	Shape distance measure, %	Geometrical distance measure, %	Structural distance measure, %	Total distance measure, %
Elm	77.6	65.5	62.2	69.4
English oak	83.8	59.8	69.9	72.4
White birch	70.8	32.6	34.8	48.5
Norway maple	77.5	62.8	53.4	65.9
European ash	68.5	46.8	60.4	59.6
Black poplar	76.4	55.6	56.7	64.3
Salix alba	60.3	27.6	29.5	41.3
Bird cherry	72.4	34.8	38.1	50.8

The similarity results depend strongly from the tree species. The trees, which crowns are close to a geometric shape, e.g., circle, such as English oak, White birch, Salix alba, and Bird cherry provide better values of shape distance respect to the modelling evaluation of the branches sizes and positions within the trees. Two algorithms, such as the space colonization algorithm and the inverse procedural modelling, demonstrate better results in comparison to the graph-based methods, using the irregular graph structures, and the self-organizing modelling. Value ranges of similarities received by use of four algorithms (Tables 6.1, 6.2, 6.3 and 6.4) are presented in Table 6.5. The best results are pointed by Bold.

Table 6.3 The normalized similarity results, using the self-organizing modelling

Tree species	Shape distance measure, %	Geometrical distance measure, %	Structural distance measure, %	Total distance measure, %
Elm	77.4	66.3	68.4	71.4
English oak	86.1	63.5	73.4	75.5
White birch	78.4	39.5	42.4	55.9
Norway maple	79.2	65.3	56.1	68.1
European ash	70.5	48.3	63.1	61.6
Black poplar	79.1	58.3	59.4	67.0
Salix alba	68.5	29.1	30.2	45.2
Bird cherry	75.3	36.3	39.5	54.2

Table 6.4 The normalized similarity results, using the inverse procedural modelling

Tree species	Shape distance measure, %	Geometrical distance measure, %	Structural distance measure, %	Total distance measure, %
Elm	81.9	68.4	72.6	75.1
English oak	91.2	72.1	89.6	85.0
White birch	87.3	42.2	54.6	64.0
Norway maple	80.3	53.5	61.3	66.6
European ash	75.7	60.2	65.7	68.1
Black poplar	78.5	50.4	62.1	65.2
Salix alba	89.2	39.8	53.1	63.6
Bird cherry	86.9	58.2	58.4	69.7

The large ranges can be explained by various tree species used in the computation. However, all virtual tree models look like the ground-truth samples. This means that a visibility criterion is satisfied. The generalized results of tree species modelling are placed in Table 6.6.

As one can see from Table 6.6, the space colonization algorithm and the inverse procedural modelling provide the best results for the mentioned tree species.

Table 6.5 Value ranges of similarities

Tree species	Shape distance measure, %	Geometrical distance measure, %	Structural distance measure, %	Total distance measure, %
Space colonization algorithm	77–90	38–69	43–70	64–85
Irregular graph structures	60–83	28–65	30–70	41–72
Self-organizing modelling	75–86	29–66	30–68	45–75
Inverse procedural modelling	76–91	40–72	53–89	64–85

Table 6.6 The total distance measures of tree species

Tree species	Space colonization algorithm, %	Irregular graph structures, %	Self-organizing modelling, %	Inverse procedural modelling, %
Elm	73.7	69.4	71.4	75.1
English oak	85.2	72.4	75.5	85.0
White birch	63.5	48.5	55.9	64.0
Norway maple	69.6	65.9	68.1	66.6
European ash	68.4	59.6	61.6	68.1
Black poplar	67.5	64.3	67.0	65.2
Salix alba	63.9	41.3	45.2	63.6
Bird cherry	65.3	50.8	54.2	69.7

6.6 Conclusions

Some reasonable models of a point distribution in 3D space known as voxelization are discussed in this chapter. The task was to create a tree architecture that involves the laser scanning points, depicting a trunk and the main branches. Four main techniques, such as the space colonization algorithm, the irregular graph structures, the self-organizing modelling, and the inverse procedural modelling, were compared for various tree species. The ranges of values are wide that is explained by a high diversity of natural objects. However, all virtual tree models look like the ground-truth samples. The space colonization algorithm and the inverse procedural modelling provided the best results, achieving 75–85% of total similarity with samples.

References

1. Chen X, Neubert B, Xu YQ, Deussen O, Kang SB (2008) Sketch-based tree modeling using Markov random field. *J ACM Trans Graphics* 27(5):109.1–109.9
2. Yu S, Liu F, Li H (2013) Tree modeling based on two orthogonal images. In: Zhong Z (ed) *Proceedings of the international conference on information engineering and applications IEA 2012*. Springer, London, pp 461–468
3. Neubert B, Franken T, Deussen O (2007) Approximate image-based tree-modeling using particle flows. *J ACM Trans Graph* 26(3):88.1–88.8
4. Tan P, Fang T, Xiao J, Zhao P, Quan L (2008) Single image tree modeling. *J ACM Trans Graph* 27(5):108.1–108.6
5. Okabe M, Owada S, Igarashi T (2005) Interactive design of botanical trees using freehand sketches and example-based editing. *Comput Graph Forum* 24(3):487–496
6. Reche A, Martin I, Drettakis G (2004) Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans Graph* 23(3):720–727
7. Samal A, Brandle J, Zhang D (2006) Texture as the basis for individual tree identification. *Inf Sci* 176(5):565–576
8. Hyppä J, Hyppä H, Leckie D, Gougeon F, Yu X, Maltamo M (2008) Review of methods of small-footprint airborne laser scanning for extracting forest inventory data in boreal forests. *Int J Remote Sens* 29(5):1339–1366
9. Cabo C, Ordoñez C, García-Cortés S, Martínez J (2014) An algorithm for automatic detection of pole-like street furniture objects from mobile laser scanner point clouds. *ISPRS J Photogramm Remote Sens* 87:47–56
10. Vastaranta M, Ville Kankare V, Holopainen M, Yu X, Hyppä J, Hyppä H (2012) Combination of individual tree detection and area-based approach in imputation of forest variables using airborne laser data. *ISPRS J Photogramm Remote Sens* 67:73–79
11. Kankare V, Holopainen M, Vastaranta M, Puttonen E, Yu X, Hyppä J, Vaaja M, Hyppä H, Alho P (2013) Individual tree biomass estimation using terrestrial laser scanning. *ISPRS J Photogramm Remote Sens* 75:64–75
12. Puente I, González-Jorge H, Arias P, Armesto J (2012) Land-based mobile laser scanning systems: a review. *Int Arch Photogramm, Remote Sens Spatial Inf Sci* 38(Part 5-W12):163–168
13. Xu H, Gossett N, Chen B (2007) Knowledge and heuristic based modeling of laser-scanned trees. *J ACM Trans Graphics* 26(4):19.1–19.19
14. Yan DM, Wintz J, Mourrain B, Wang W, Boudon F, Godin C (2009) Efficient and robust tree model reconstruction from laser scanned data points. 11th IEEE International conference on computer-aided design and computer graphics, CAD/graphics 2009, pp 572–575
15. Schilling A, Schmidt A, Maas HG, Wagner S (2011) Topology extraction using depth first search on voxel representations of tree point clouds. *Int Arch Photogramm, Remote Sens Spatial Inf Sci, ISPRS Calgary XXXVIII-5/W12:85–90*
16. Sonka M, Hlavac V, Boyle R (2014) *Image processing, analysis, and machine vision*. 4th ed, Cengage Learning, Australia, Brazil, Japan, Mexico, Singapore, United Kingdom, United States
17. Russell S, Norvig P (2010) *Artificial intelligence: a modern approach*, 3rd edn. Pearson Education Limited, Edinburgh
18. Bucksch A, Lindenbergh RC, Menenti M (2009) SkelTre—fast skeletonisation for imperfect point cloud data of botanic trees. In: Pratikakis I, Spagnuolo M, Theoharis T, Veltkamp R (eds) 2th Eurographics conference on 3D object retrieval 3DOR 2009, pp 13–20
19. Sachs T, Novoplansky A (1995) Tree form: architectural models do not suffice. *Israel J Plant Sci* 43:203–212
20. Runions A, Fuhrer M, Lane B, Federl P, Rolland-Lagan A-G, Prusinkiewicz P (2005) Modeling and visualization of leaf venation patterns. *J ACM Trans Graph* 24(3):702–711
21. de Lima Bicho A, Rodrigues RA, Musse SR, Jung CR, Paravisi M, Magalhaes LP (2012) Simulating crowds based on a space colonization algorithm. *Comp Graph* 36(2):70–79

22. Runions A, Lane B, Prusinkiewicz P (2007) Modeling trees with a space colonization algorithm. *Eurographics Workshop Nat Phenomena* 7:63–70
23. Palubicki W, Horel K, Longay S, Runions A, Lane B, Mech R, Prusinkiewicz P (2009) Self-organizing tree models for image synthesis. *ACM Trans Graph* 28(3):58.1–58.10
24. Shinozaki K, Yoda K, Hozumi K, Kira T (1964) A quantitative analysis of plant form—the pipe model theory. I. Basic analyses. *Japan J Ecol* 14(3):97–104
25. MacDonald N (1983) Trees and networks in biological models. Wiley, New York
26. Livny Y, Yan F, Olson M, Chen B, Zhang H, El-Sana J (2010) Automatic reconstruction of tree skeletal structures from point clouds. *J ACM Trans Graph* (6):151.1–151.8
27. Honda H (1971) Description of the form of trees by the parameters of the tree-like body: effects of the branching angle and the branch length on the shape of the tree-like body. *Theor Biol* 31:331–338
28. Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numer Math* 1 (1):269–271
29. Lopez LD, Ding Y, Yu J (2010) Modeling complex unfoliaged trees from a sparse set of images. *Comput Graph Forum* 29(7):2075–2082
30. Estrada R, Tomasi C, Schmidler SC, Farsiu S (2014) Tree topology estimation. *IEEE Trans Pattern Anal Mach Intell* 37(8):1688–1701
31. Xu L, Mould D (2012) Synthetic tree models from iterated discrete graphs. *Graph Int GI* 2012:149–156
32. Xu L, Mould D (2012) A procedural method for irregular tree models. *Comput Graph* 36 (8):1036–1047
33. Xu H, Gossett N, Chen B (2007) Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans Graph* 26(4):19.1–19.19
34. Germer T, Strothotte T (2008) Orthant neighborhood graphs: a decentralized approach for proximity queries in dynamic point sets. In: Braz J, Ranchordas A, Araújo HJ, Pereira JM (eds) Computer vision and computer graphics. Theory and applications, internaional conference VISIGRAPP 2007. Springer, Berlin
35. Yao A (1977) On constructing minimum spanning trees in k-dimensional spaces and related problems. Technical report, Stanford University
36. Longay S, Runions A, Boudon F, Prusinkiewicz P (2012) Treesketch: interactive procedural modeling of trees on a tablet. International symposium on sketch-based interfaces and modeling, pp 107–20
37. Ijiri T, Mech R, Igarashi T, Miller G (2008) An example-based procedural system for element arrangement. *Comput Graph Forum* 27(4):429–436
38. Stava O, Benes B, Mech R, Aliaga DG, Kristof P (2010) Inverse procedural modeling by automatic generation of L-systems. *Comput Graph Forum* 29(2):665–674
39. Bokehlo M, Wand M, Seidel HP (2010) A connection between partial symmetry and inverse procedural modeling. *ACM Trans Graph* 29(4):104.1–104.10
40. Martinovic A, Van Gool L (2013) Bayesian grammar learning for inverse procedural modeling. *IEEE Conf Comput Vision Pattern Recog CVPR* 2013:201–208
41. Stava O, Pirk S, Kratt J, Chen B, Mech R, Deussen O, Benes B (2014) Inverse procedural modeling of trees. *Comput Graph Forum* 33(6):118–131
42. Srivastava L (2002) Plant growth and development: Hormones and environment. Academic Press, San Diego
43. Ferraro P, Godin C (2000) A distance measure between plant architectures. *Ann Forest Sci* 57 (5):445–461

Part III
Landscape Scene Modelling

Chapter 7

Digital Modelling of Terrain Surface

Abstract The digital terrain modelling is a crucial issue in visualization of the forestry and urban areas. Among the digital evaluation, surface, and terrain models, the last ones play the significant role in the GIS applications. The challenges of a terrain modelling lead to the development of complex artificial and statistical methods that include a densification of the LiDAR point cloud, a filtering for extraction of the ground and non-ground points, and an interpolation for generation of the bare Earth's surface. The wide spectrum of methods from each category permit to chose the acceptable solution in practice. However, such conventional way is more available for “heavy weighted” software tools, representing in Chap. 3. The future investigations deal with the design of “light weighted” software tools for unmanned aerial and ground vehicles with a reasonable relation between the accuracy estimators of the models and their computational cost.

Keywords Digital terrain modelling · Lidar data · Densification methods · Interpolation-based filtering · Slope-based filtering · Morphology-based filtering · Segmentation-based filtering · Polynomial interpolation · Delaunay triangulation · Kriging · Thin plate spline

7.1 Introduction

In this part of book, the reconstruction of the forest and landscape scenes will be discussed. The conventional way is a sequence of the following actions: the generation of digital terrain model and its texturing, the vegetation modelling using individual trees, shrubs, underbrush, etc., and the addition of lighting and/or meteorological effects in a modelling scene. The dynamics of vegetation development as a forest ecosystem is considered in Chap. 12.

The term “Digital Terrain Model” (DTM) was originally introduced by Miller and LaFlamme [1] in 1958. For the DTM implementation, three main approaches were suggested as mentioned below [2]:

- The point-wise methods involve the independent determination of different functional parameters and height values in the irregular grids with the following interpolation. These methods interpolate the height values of the randomly measured points in the grid nodes independently from the adjacent points of the terrain model. A continuous surface may be generated through all the derived grid nodes without the discontinuities or boundary problems.
- The patch-wise methods create the local 3D patches, in which the elevations of series of the neighboring grid points, lying within each patch, are interpolated. These methods occupy the intermediate position between the point-wise methods and the global methods. Usually a series of equal sized patches with the identical shapes is mapped in a regular square or a rectangular grid. Then the quite separate mathematical functions are generated to form the surface for each individual patch. Two distinct methods of the patch-wise interpolation exist. The exact fit patch abuts exactly onto its neighbor grid cells. The sharp discontinuities may appear along the junctions of the grid cells. Also if the patches are larger than the grid cells, then several grid nodes will fall within a single patch, creating an uncertainty. In the case of the overlapping patches, the common points, lying within the overlap, are used in the computation of the parameters for each patch.
- The global methods presuppose the existence of a single complex 3D surface with the subsequent interpolation of the terrain heights of all nodes in the regular grid. This means that a high-order polynomial through all of the measured randomly located terrain height points ought to exist. Then the values of the heights in each grid node are interpolated using the polynomial coefficients. The weaknesses deal with many terms in the high order polynomials that ought to be computed and the high computation time for huge data set of the initial measured points. Often these methods produce the poorly interpolated 3D surface, especially for the hill and mountain areas.

The point-wise and global methods are opposite approaches. The patch-wise methods have an intermediate position. The advantages of the patch-wise methods over the global methods are that the quite low order parameters in a polynomial function describe each patch satisfactorily. The disadvantage of the patch-wise methods in a comparison to the point-wise and global methods is the necessity of the well-structured and uniform distributed data. Notice that the point-wise methods have the wide propagation in many practical applications, including the modelling of the Earth's surface based on the LiDAR data.

The most common LiDAR product is a grid or raster elevation surface with a high accuracy of data values over the large areas. Three main elevation products can be obtained from the LiDAR point clouds [3], such as

- The Digital Elevation Model (DEM) includes a general description of an elevation surface often in conjunction with an additional description of the bare Earth DEM.
- The Digital Terrain Model (DTM) is commonly a bare Earth model and provides the best representation of the terrain, incorporating much ancillary information.

- The Digital Surface Model (DSM) may produce a bare Earth surface as well as a surface along the tops of the trees.

The bare Earth DEMs or the DTMs require the removal of the LiDAR points, falling on the non-terrain objects like trees, houses, vehicles, etc. This process may be handled using the specific software tools or implemented in automatic mode to classify the points and create the bare Earth DEM, e.g., using ArcGIS [4], Surfer 13 [5], or other software tools. Besides a forest monitoring and inventory using the DTMs, some other applications, such as the identification of the inherent geomorphological and drainage features [6], the detection of gullies in a roughly textured terrain [7], or the generation of the urban DTM [8] can be mentioned.

Hereinafter, Sect. 7.2 presents related work. The densification methods, improving the initial LiDAR point cloud, are discussed in Sect. 7.3. Section 7.4 presents the filtering methods to classify the LiDAR points on the ground and non-ground clusters. The approaches for the DTM generation are detailed in Sect. 7.5. Section 7.6 contains the experimental results. Finally, conclusions are summarized in Sect. 7.7.

7.2 Related Work

The most widely used approach for the DTM construction from the LiDAR data is based on a detection of local minima in the grid cells. However, this method works well in a flat terrain with few trees and buildings. Usually a high resolution (i.e., a slicing window with the small size) is required for the Earth's point detection, while a low resolution (i.e., a slicing window with the large size) is needed for the non-ground point detection, dealing with the large buildings' detection. This problem has attracted a significant attention in order to generate more accurate DTMs, using the hierarchical or iterative procedures [9, 10].

Another reason in a forest modelling (because of which it is difficult to detect the objects in the LiDAR point cloud) may be a presence of the vegetation on slopes and the low-height vegetation. The reasonable assumption that the vegetation points are significantly higher than their neighborhoods and they can be easily filtered falls away in the steep and rough terrain but sometimes the terrain points may lie at the same height as the vegetation points. Additionally, the detection of low vegetation is very difficult problem due to the possible small but sharp variations in the terrain. This causes a necessity to apply the special algorithms for clustering of the LiDAR point cloud.

Sithole and Vosselman [11] proposed four distinct concepts to separate the points of the objects and the bare Earth:

- The slope-based filter calculated the slope or height difference between two points and compares with a certain threshold.

- The block-minimum filter is based on the discriminant function that defines a horizontal plane with a corresponding 3D buffer zone, the points under which are expected as the bare Earth points.
- The surface-based filter is an extension of block-minimum filter, when the discriminant function is a parametric surface with a corresponding buffer zone.
- The clustering/segmentation filter creates the sets of points belonging to the objects (not facets of objects). These sets are separated by the higher and lower locations.

In the following publications, the mentioned above classification of filters was clarified and reorganized in a new set of filters [12–14]. Nowadays, the following classification of filtering is accepted:

- The surface-based filters build a surface model through the entire point set that iteratively approaches the ground surface. Usually the ground points have negative residuals and the objects' points have positive ones. Thus, a weight function selects the high weights to the points with negative residuals and the low weights to the points with positive residuals. The progressive-densification-based filters are similarly to these filters. These filters work progressively with the difference that there is no need to interpolate the results in the progressive-densification case.
- The slope-based filters imply the assumptions that the gradient of the ground is obviously smoother than that of the non-ground objects [15] and the threshold, separating the ground from the non-ground points, is determined by a monotonically increasing kernel function [16].
- The morphology-based filters help to approximate the terrain surface using the morphological operations, such as the opening or geodesic reconstruction. Most of these methods were proposed under the assumption that a terrain slope is constant. Hereafter, Chen et al. [17] improved a morphological filter, removing the constant slope restriction, and Arefi and Hahn [18] presented a geodesic morphological reconstruction-based algorithm to produce the bare Earth model.
- The segmentation/cluster-based filters are focused on separating of a point cloud into clusters, using elevation differences in the neighborhood and building the normal vectors of clusters [13, 19]. Then each cluster is labeled. Notice that the feature set of a point cloud is poor and reflects only the geometrical properties.

Zhang et al. [20] were the first, who introduced the progressive filtering with a difference threshold. A window with small sizes preserves the ground points over the gradually changing terrain, while a window with larger sizes removes the buildings and vegetation by selecting the minimum elevation points. The window size w_k is progressively increased using Eq. 7.1, where b is a base of the increasing exponential function, k is an iteration step starting at 0.

$$w_k = 2b^k + 1 \quad (7.1)$$

The elevation threshold dhT is calculated based on the condition statements mentioned in Eq. 7.2, where dh_0 is an initial elevation difference threshold, s is a slope terrain, c is a grid cell size, dh_{\max} is a maximum elevation difference threshold.

$$dhT_k = \begin{cases} dh_0 & \text{if } w_k \leq 3 \\ s(w_k - w_{k-1})c + dh_0 & \text{if } w_k > 3 \\ dh_{\max} & \text{if } dhT_k > dh_{\max} \end{cases} \quad (7.2)$$

In order to improve a filter performance and a robustness with respect to the complex scenes, Hu et al. [14] proposed the Adaptive Surface Filter (ASF) for the ALS data processing. These authors mentioned three innovative ideas, including the introduction of a step factor that controls the granularity between the adjacent levels in the data pyramid scheme and improves the robustness of the algorithm, the regularization to overcome a noise during interpolating of surfaces (using progressively densified ground points) by the Thin Plate Spline (TPS) approach, and the bending energy function that explicitly depicts the surface smoothness. Then the raster surface and the bending energy are integrated to label the remaining unclassified points as either ground or non-ground.

Chen et al. [21] designed the mathematical morphology-based multi-level filter to identify and eliminate the non-ground objects, while the terrain objects are preserved. The main idea is to employ the morphological opening operations only in the local approximate object regions. This permits to avoid a smoothing of the terrain as much as possible. The method works under the assumption of a constant slope of area.

Vega et al. [22] developed the algorithm of virtual deforestation that iteratively detect and filter objects above-the ground surface, using the LiDAR points and a raster model. This sequential iterative dual-filter method computes the DTM of the rough and forested terrain and includes four steps:

- Step 1. The extraction of the Lowest Points (LP) from the original LiDAR Point Cloud (LPC) with the minimum elevation in the given pixel or the patch.
- Step 2. The DEM is generated by rasterizing the LPs selected in the given pixel, supplementing the missing values with the neighbor interpolated surface. As a result, a procedure of iterative removal of the points that are distinctly above the ground surface is executed until no more point is removed.
- Step 3. An iterative removal of a point close from the ground surface is implemented. For each pixel, the elevation difference to the eight connected pixel is computed and the mean value of these differences is assigned to the pixel. The LPs located within the resulting mask are removed, and the algorithm iterates Step 3 until no more point is removed.
- Step 4. In order to retrieve the additional ground point, the point densification process is applied using the initial LPC.

This algorithm was tested on badlands with the complex surfaces and provided the modelling results with the False Acceptance Ratio (FAR) up to 7% and the False Rejection Ratio (FRR) up to 20%.

The Multi-resolution Hierarchical Classification (MHC) algorithm with three levels of the hierarchy classifies was proposed by Chen et al. [23]. First, some points are extracted from the raw LiDAR point cloud in advance as the ground

seeds. Second, based on the elevation differences between the sampling points and the interpolated raster surface, the ground and non-ground points are classified. At each level of the hierarchy, the raster surface is iteratively interpolated towards the ground using the TPS until no ground points are classified. The classified ground points serve to update the surface in the next iteration. The cell resolution and the residual threshold are simultaneously increased from the low to the high level of the hierarchy. However, this approach is failed on the complex landscapes with the steep slopes.

The original segmentation of the LiDAR data was proposed by Mongus et al. [24]. The authors suggested to generate a top-hat scale-space using the Differential Morphological Profiles (DMPs) on the points' residuals from the approximated surface. The geometric features, such as the widths, heights, and height differences from the surroundings, were estimated by the mapping characteristic values from the DMPs. The algorithm for the Local Fitting Surfaces (LoFS) was developed for extracting the planar points of building and other above-ground objects in such manner that the achieved completeness was approximately 5% and lower.

The details of processing of the point clouds received by the TLS with the minimal end-user interaction that produce the DTM, the DSM, the CHM, and a vegetation density map are discussed in [25]. As the authors mentioned, their method is of a high importance for many applications, ranging from forestry to hydrology and geomorphology.

Therefore, a processing of the LIDAR data involves some common procedures, such as the improvement (densification) of the initial LiDAR point cloud, the clustering of the LiDAR points on ground and non-ground clusters using filtering methods, and the DTM/DEM/DSM generation applying the spatial, statistical, or curve-fitting interpolations. These issues will be discussed in the following subsections.

7.3 Densification of LiDAR Point Cloud

The preliminary stage deals with the improvement of the initial LiDAR point cloud concerning to the outliers removal and densification of the LiDAR data. The classification of these methods is situated in Sect. 7.3.1, while the measurement errors are pointed in Sect. 7.3.2.

7.3.1 *Densification Approaches*

In nearly all the LiDAR applications, the ground filtering is a necessary step to determine, which the LiDAR returned pulses are from the ground surface and which are from the non-ground surfaces [26]. Generally, the main inaccuracy during the DTM generation is caused by the points that have the lowest elevation

values than their surrounding neighbors. This occurs, when a low-lying ground with a small area is measured in a large scale cell. Some approaches can improve this situation, among which are the following ones:

- The robust linear prediction algorithm proposed by Kraus and Pfeifer [9] involves two steps. First, a rough approximation of a surface is computed using the lowest points. Second, the residuals, (distance vectors from the surface to the measured points) are calculated, and each point receives a weight according to its residual. The points with high weights are attracted to the surface whilst the points with low weights have a little influence on a surface structure.
- The filtering method based on the mathematical morphology preserves the terrain information by analyzing the elevation differences among the neighboring points [16, 17, 27].
- The method based on a local operator that alters the parameters as a function of the slope of a terrain [15].
- The upward-fusion method developed by Chen et al. [8] processed several preliminary DTMs that are generated using the GIS tools or the extended local minimum method in various scales. This approach is especially useful for the urban DTMs building.

Notice that the application of aforementioned approaches depends strongly from the solving task.

7.3.2 Measurement Errors

Due to the integration of different sensors (the Global Positioning System (GPS), the Integrated Navigation System (INS), the laser scanner), the height accuracy of the DTMs does not sufficient. As mentioned Crombaghs et al. [28], the error of a single laser point is a combination of several error components, among which are the following:

- The error per a point. Due to the measurement uncertainty of a laser scanner, each laser point has a random error called as a point noise. A point noise can be estimated on the flat areas without vegetation and buildings, for example, $50 \text{ m} \times 50 \text{ m}$. The height of each laser point (index j in Eq. 7.3) is predicted by the interpolation of the heights in the neighboring points and compared with the originally measured heights (index i in Eq. 7.3). The standard deviation of all differences σ_{pn} in the area is computed by Eq. 7.3, where dH_i is a height difference in point i , dH_j is a height difference in neighboring point j relative to point i , n is a number of laser points in a flat area.

$$\sigma_{pn} = \sqrt{\sum_{i=1}^n \left(dH_i - \left(\frac{1}{n} \sum_{j=1}^n dH_j \right) \right)^2} \quad (7.3)$$

- The error per the GPS observation. This random error is a constant for all laser points measured during a time interval. This error depends on a flying speed and the GPS observation interval. Usually, these points are lying in a strip-wide area of about 100 m in a length. The measured height of a laser point in the current strip is subtracted from the interpolated value of this strip. Equation 7.4 describes a function $C(s)$ of the distance s between these points in time or in space, where $C(0)$ is a variance, n_s is a number of used point pairs for distance s , x_i is a signal in point i , ($x_i = \Delta h_i$), $x_{i'}$ is a signal belonging by point i at distance s .

$$C(s) = C(0) - \frac{1}{2n_s} \sum_{i=1}^{n_s} (x_i - x_{i'})^2 \quad (7.4)$$

- The error per a strip. This systematic error is caused by the GPS and the INS sensors. They introduce the systematic errors in strips, like the vertical offsets, the tilts in along- and across-track direction, and the periodic effects with a period of several kilometers.
- The error per a block. The terrestrial reference measurements (the ground control “point”) are used to transform the blocks of the laser measurements into the national height system. This influence depends on the block configuration (the position and number of strips, the cross strips, and the control “points”) and the correction procedure (the strip adjustment). From the standard deviations of the strip offsets, a single standard deviation σ_{ds} for the precision of entire block can be calculated by Eq. 7.5, where i and j are the numbers of the adjacent strips, σ_{a_i} is a standard deviation of offset a of a strip i , $\sigma_{a_ia_j}$ is a covariance between the offsets of the strip i and strip j , n is a total number of the strips in the laser dataset.

$$\sigma_{ds}^2 = \frac{\sum_{i=1}^n \left(\sigma_{a_i}^2 + \sum_{j=1}^n \sigma_{a_ia_j} \right)}{n^2} \quad (7.5)$$

- The error per a computation. These errors are caused by the applied methodology and the algorithms (the algorithm comparison is discussed in [11]).
- The error per the data characteristics. A quality of the DTM derived from a laser scanning is influenced by the data characteristics (e.g., the point density, the first/last pulse, the flight height, the scan angle, among others).
- The error per a complexity. These errors are caused by the complexity of the targets, e.g., the sloping terrain, the density of the canopy above.

7.4 Filtering of LiDAR Points

The filtering methods classify the LiDAR data on the ground and non-ground sets. The accurate classification provides the generation of the high-quality DTM/DEM/DSM. The main filtering methods fall into the surface-based (Sect. 7.4.1), the slope-based (Sect. 7.4.2), the morphology-based (Sect. 7.4.3), the segmentation-based (Sect. 7.4.4), and the hybrid (Sect. 7.4.5) categories. Each category uses the own assumptions of classification that cause the strengths and weakness of each category. A comparison of these categories is represented in Sect. 7.4.6.

7.4.1 Surface-Based Filtering

The main assumption of the surface-based filter methods is that the terrain has a spatially continuous surface [29]. They are based on the robust interpolation of the ground points, using various evaluations. Thus, a computing of the residual between the observations and the spline interpolated surface helps to classify the points, using a threshold value [30, 31]. The least-squares surface fitting is commonly used in these methods with the main disadvantage—the removal points at the breaklines [32]. Also the ground surface can be estimated by an active shape model. This approach suffers from many parameters and is sensitive to the negative outliers.

The result of the TPS interpolation is a smooth surface $S(x, y)$, which interpolates all the given control-points at minimized bending energy. Under the assumption that the locations (x_i, y_i) are all different and are not collinear [33], the TPS interpolant $S(x, y)$ minimizes the bending energy J_S (Eq. 7.6)

$$J_S = \iint_{\mathbb{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2) dx dy \quad (7.6)$$

and has a view of Eq. 7.7

$$S(x, y) = \mathbf{a}_1 + a_x x + a_y y + \sum_{i=1}^p w_i U(r), \quad (7.7)$$

where \mathbf{a} is a vector defining of affine part of the mapping, \mathbf{w} is a vector denoting the nonlinear part of the deformation, $U(r)$ is a kernel function, $U(r) = r^2 \log r$, $r = \|(x_i, y_i) - (x, y)\|$, (x_i, y_i) are the initial landmarks or the control points.

The constraints

$$\sum_{i=1}^p w_i = 0 \quad \sum_{i=1}^p w_i x_i = \sum_{i=1}^p w_i y_i = 0$$

together with the interpolation conditions, $S(x_i, y_i) = v_i$, yield a linear system for the TPS coefficients:

$$\begin{bmatrix} \mathbf{K} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{o} \end{bmatrix},$$

where $K_{ij} = U(\|(x_i, y_i) - (x_j, y_j)\|)$, the i th row of \mathbf{P} is $(1, x_i, y_i)$, \mathbf{O} is a 3×3 matrix of zeros, \mathbf{o} is a 3×1 column vector of zeros, \mathbf{w} and \mathbf{v} are the column vectors formed from w_i and v_i , respectively, \mathbf{a} is a column vector with the elements a_1, a_x, a_y .

When there is a noise in the specified values v_i , the interpolation can be relaxed by means of the regularization minimizing Eq. 7.8.

$$H[S] = \sum_{i=1}^n (v_i - S(x_i, y_i))^2 + \lambda J_S \quad (7.8)$$

The regularization parameter λ (a positive scalar) controls the amount of a smoothing. The limiting case, $\lambda = 0$, reduces to exact the interpolation.

Zandifar et al. [34] proposed the Multi-Level Fast Multi-pole Method (MLFMM) with an improvement from $O(N^2)$ to $O(N \log N)$ operations. The MLFMM used a hierarchy of grids for subdivision of space in multiple scales and a corresponding hierarchical organization of the charge groups and multi-pole expansions. Ming and Chen [35] adopted this approach for the DTM generation.

Among the surface-based filtering methods, the Progressive TIN Densification (PTD) is widely employed in both the scientific community and engineering applications. The PTD method that was proposed by Axelsson [36] divides the whole point set into the tiles, selects the lowest points in each tile as the initial ground points, and constructs the TIN as the reference surface. For each triangle, two criteria are set: the point's distance to the TIN facet must not exceed the given threshold and the angle between the TIN facet and the line, connecting a point with the facet's closest vertex, must not exceed the given threshold. The PTD is the most promising decision for the Earth's surface modelling that helps to extract the accurate terrain or semantic information from the LiDAR point cloud [37]. This method had been implemented in Terrasolid software tool [38]. The conventional PTD is composed from four steps as mentioned below:

- Step 1. Removing the outliers. When the filters work under the assumption that the lowest point in a grid cell ought to belong to the terrain, they are very sensitive to the outliers in the data sets [11]. Zhang and Lin [13] proposed three-sub-steps to eliminate the outliers. First, the elevation histogram is built, and the lowest and highest values are removed from the distribution. Second, the

remaining outliers are searched using the minimum height difference of each point with respect to all its neighbors, using 2D k dimensional (kd)-tree. Again, the points that are too high or too low with respect to their neighbors are removed from the set. Third, the errors yielded by the automatic outlier classification are corrected manually.

- Step 2. Specifying parameters. There are six key parameters including: the maximum building size m is a length threshold to define the grid cell size, the maximum terrain angle t is a slope threshold that decides, a mirroring point is performed or not (if the slope of a triangle in the TIN is larger than t , then any unclassified/potential point located inside of this triangle should be judged by a corresponding mirror point), the maximum angle θ is the maximum angle between a triangle plane and a line connecting a potential point with the closest triangle vertex, the maximum distance d is the maximum distance from a point to the triangle plane during one iteration, the minimum edge length l is the minimum threshold for the maximum (horizontally projected) edge length of any triangle in the TIN, and the maximum edge length l' is the maximum threshold for the minimum (horizontally projected) edge length of any triangle in the TIN.
- Step 3. Selecting the seed points and constructing the TIN. First, the bounding box of the given point cloud is determined. Second, this region is divided into several tiles in the rows and columns. Each corner's height is equal to the one of its closest seed point on a horizontal plane, and the lowest point in each tile is selected as a seed point. Third, the initial terrain model is constructed based on the seed points. The remaining points, except the seed points, are labeled as the default object measurements.
- Step 4. Iterative densification of the TIN. In each iteration, the judging is performed by a point-wise method under the constraints of the specifying parameters from Step 2. At the end of each iteration, the newly detected ground points are added into the TIN according to some conditions.

The conventional PTD filter fails to detect the ground points around the breaklines and steep natural terrain. Some enhanced methods were developed, for example, the Segmentation Using Smoothness Constraint (SUSC) proposed by Zhang and Lin [13]. Notice that the PTD filter does not need in the interpolation.

Mongus and Zalik [39] had proposed a ground filtering algorithm, in which the TPS interpolation is used iteratively to approximate the ground surface. The approach used residuals of the points from the surface in each iteration with a gradually decreasing a window size in order to filter the ground points. The algorithm coped well with steep terrain, while it became worthless with the low LiDAR point density.

Most of methods employ the initial classification of the ground versus the non-ground LiDAR points, and only few solutions have specifically addressed identification of the non-ground LiDAR points in the forested environments. The landscape objects, such as trees, can be assessed across scales; they can split into the multiple objects, merge with other objects, persist, disappear, or change its

amplitude (a deviation from the surrounding points) [40]. The curvature approaches, such as the virtual deforestation algorithm [41] or the Multi-scale Curvature Classification (MCC) [42], identify the positive local deviations from the surrounding points and then iteratively classify them as the non-ground. Usually these algorithms are based on the identifying of the positive local curvatures in the interpolated Triangulated Irregular Network (TIN) constructed from the LiDAR point data that becomes a reasonable solution in the complex terrain evaluation.

The implementation of the MCC algorithm is the following [42]. Let $P(x, y, z)$ be a set of 3D coordinates x , y , and z (elevation) of the LiDAR points. This set is employed to interpolate a raster surface, using the TPS with different cell resolution defined by a scale parameter λ . The λ parameter is calculated in three scale domains as 0.5λ , 1.0λ , and 1.5λ . The TPS is fitted using a slicing window with the 12 nearest neighbors and an invariant tension parameter f defined as 1.5 across all scale domains. A mean kernel 3×3 is passed over the interpolated raster defining a new vector $P'(x, y, m)$, which is coincident with $P(x, y, z)$, consisting of x and y coordinates and a mean surface value. The curvature tolerance t is added to $P'(x, y, m)$, and the points are classified as the non-ground by applying the conditional statement. In each iteration, a set $P(x, y, z)$ is redefined as only points not yet classified. Then these remaining points are used at the start of the next iteration. The scale domain l is defined in interval 1–3. A curvature tolerance parameter t is determined by the end-user. Usually its value increases on 0.1 in each scale domain.

The classification steps of the MCC are mentioned below:

- A surface is interpolated, using $P(x, y, z)$ and the TPS. The scale parameter λ and the curvature tolerance t are applied (initial values of λ and t parameters are defined by the end-user).
- A mean kernel 3×3 is passed over the interpolated surface, and a new set $P'(x, y, m)$ is declared that is coincident with $P(x, y, z)$.
- The curvature C_l in the scale domain l is calculated by Eq. 7.9, where $P'(x, y, m)$ is a new mean elevation set, t is a curvature tolerance parameter in the scale domain l .

$$C_l = \|P'(x, y, m)\| + t \quad (7.9)$$

- The LiDAR points are classified as the non-ground and removed under the following condition:

IF $\|P(x, y, z)\| > C_l$ THEN a point is classified as a non – ground

- When the convergence threshold j is assessed, the model either iterates or starts with the next scale domain.

The MCC algorithm has two loops: the nested loop, defining a model convergence j within a scale domain l , and the main loop in the multi-scale facet. The MCC effectively identifies the non-ground returns in the vertical distribution of

elevations in the LiDAR point cloud due to the dynamic scale and the curvature tolerance parameters. This algorithm was successfully implemented in the software tool Workstation AcrInfo [43].

7.4.2 Slope-Based Filtering

The slope-based filtering relies on the premise that the gradient of a natural slope of the terrain is distinctly different from the slopes of the non-terrain objects (trees, buildings, etc.). In other words, any feature in the laser data that has slopes with the gradients larger than a certain predefined threshold cannot belong to the natural terrain surface. First researches, devoting to analysis of a sloped terrain, are concerned to 1990s, and investigations had developed in two ways:

- Kilian et al. [44] suggested the morphological approach with the series of opening operations with different structure element sizes. A weight in the current point was assigned proportionally to the distance between this point and the opened surface. Such weights were used for modelling of a smoothness ground surface.
- Pfeifer et al. [45] applied a filter method based on an iterative linear least squares interpolation. A robust estimation of the ground surface was obtained using a weight function, which assigned the low weights to the points with relatively high values.

In the following investigations, the first approach prevailed, and was supported in pioneer works of Vosselman and Sithole. Vosselman [16] utilized the mathematical morphological method and created a kernel function based on terrain slope to filter the non-ground points. However, instead of weights Vosselman suggested to introduce a non-decreasing filter function between two points $\Delta h_{\max}(d)$, defining the acceptable height difference between these points. Three ways to derive a filter function was proposed by Vosselman [16], at that, the first one used knowledge about the terrain shape and the precision of the height measurements, while the other two were based on a training data set:

- The synthetic functions, for example, under the assumption that the slopes in the terrain are not steeper than 30%:

$$\Delta h_{\max}(d) = 0.3 d$$

and in the case of the noise measurements with adding a confidence interval:

$$\Delta h_{\max}(d) = 0.3 d + 1.65 \sqrt{2} \sigma.$$

- The functions, preserving the important terrain features, can be built knowing the cumulative probability distribution for the height differences between the points at distance d in the training data.
- The functions, minimizing the classification errors, estimate the height differences between the point pairs in a training set of the ground points and between the point pairs with one point from a training set of the ground points and the other point from a training set of unfiltered data of the same area.

Based on any type of the filtering functions, Vosselman [16] introduces the DEM as a set of the ground points in a view of Eq. 7.10, where A is a set of all points.

$$DEM = \{p_i \in A \mid \forall p_j \in A : h_{p_i} - h_{p_j} \leq \Delta h_{\max}(d(p_i, p_j))\} \quad (7.10)$$

Hereinafter, Sithole [15] studied the case with the gentle sloped terrains and improved Vosselman's method using a gradient map to automatically adapt the change of a terrain. The filter function was an inverted bowl, whose shape was defined by a probabilistic function, minimizing the classification errors. The filter function is often replaced by a cone for simplicity. The vertex of an inverted cone sweeps under each point from the filtered point set. Wherever a cone cuts the point set, the point at the vertex of a cone is filtered off. Another way to utilize this method is to introduce a surface referred to as a point-slopes surface. A point-slopes surface is negative relative to the absolute value of the gradient of a cone's surface. If a cut-off plane cuts a point p_i in a point-slopes surface, then this point is filtered off. The curvature of a cut-off plane is determined by a probabilistic function derived from a training data set. According to Sithole, the classifier is expressed by Eq. 7.11, where p_j is a point in the data set ($p_i \neq p_j$), h_{p_i} and h_{p_j} are the heights of p_i and p_j , respectively, $h_{p_i} - \Delta h(p_i, p_j)$ is the height of a point directly above or below p_j and the point p_i on the lateral surface of a cone, m is an absolute value of the gradient of a cone's surface (the negative value of m is a height of a cut-off surface), A is a set of the LiDAR points to be filtered in order to extract the DEM.

$$\forall p_j \in A : h_{p_i} - \Delta h(d(p_i, p_j), m) \leq h_{p_j} \quad (7.11)$$

The radius of the base of a cone defines the operating range of a filter. Therefore, the parameter m in Eq. 7.11 can be replaced by m_i for a point p_i . In order to consider the special cases of cuts, Sithole [15] introduced additionally a predefined factor and a minimum threshold. Equation 7.11 can be written in a view of Eq. 7.12, where m_i is a maximum slope of the terrain at the point, where a cone's vertex touches the surface, s_{m_i} is a predefined factor, by which m_i is multiplied, m_{\min} is a minimum threshold value for $s_{m_i} \cdot m_i$.

$$\forall p_j \in A : h_{p_i} - \Delta h(d(p_i, p_j), m_i, s_{m_i}, m_{\min}) \leq h_{p_j} \quad (7.12)$$

Then the DEM is expressed by Eq. 7.13.

$$DEM = \{p_i \in A | \forall p_j \in A : h_{p_i} - \Delta h(d(p_i, p_j), m_i, s_{m_i}, m_{\min}) \leq h_{p_j}\} \quad (7.13)$$

7.4.3 Morphology-Based Filtering

The morphology-based filtering assumes that the ground objects are associated with the relatively lower LiDAR points, while the non-ground objects are described by the relatively higher LiDAR points. The mathematical morphology is the theory for processing of the geometrical structures based on a set theory. It is a commonly used to process the digital images but it can be utilized on the graphs, surface meshes, and many other spatial structures. These filters are based on a series of 3D morphological closing and opening operators. The filter parameters highly depend on a terrain slope [16] as well as on the filtering window size [46, 47]. The computational cost is not high but these filters need the accurate a priori knowledge about a terrain topology.

The morphological reconstruction of the Earth's surface based on a geodesic dilation was suggested by Arefi and Hahn [18]. Such morphological grayscale reconstruction employs two input images, marker and mask images, which have the equal sizes and the mask image I has intensity values larger or equal to the marker image J . In a geodesic dilation, the marker image is dilated by an elementary isotropic structuring element, while the mask image acts as a limit for the dilated marker image. The conventional grayscale dilation of J with a structuring element B is given by Eq. 7.14, where the symbol \oplus is a dilation operation.

$$\delta(J) = J \oplus B \quad (7.14)$$

The geodesic dilation of size 1 of the marker image J with respect to the mask image I is defined by Eq. 7.15, where the symbol \wedge means the point-wise minimum between the dilated marker image and the mask image.

$$\delta_I^{(1)}(J) = (J \oplus B) \wedge I \quad (7.15)$$

The geodesic dilation of size n of the marker image J with respect to a mask image I is obtained by performing n successive geodesic dilations of size 1 of J with respect to I (Eq. 7.16).

$$\delta_I^{(n)}(J) = \underbrace{\delta_I^{(1)}(J) \circ \delta_I^{(1)}(J) \circ \dots \circ \delta_I^{(1)}(J)}_{n \text{ times}} \quad (7.16)$$

The morphological reconstruction is achieved by the repeated dilations of the marker image J until the contour of the marker image J fits under the mask image

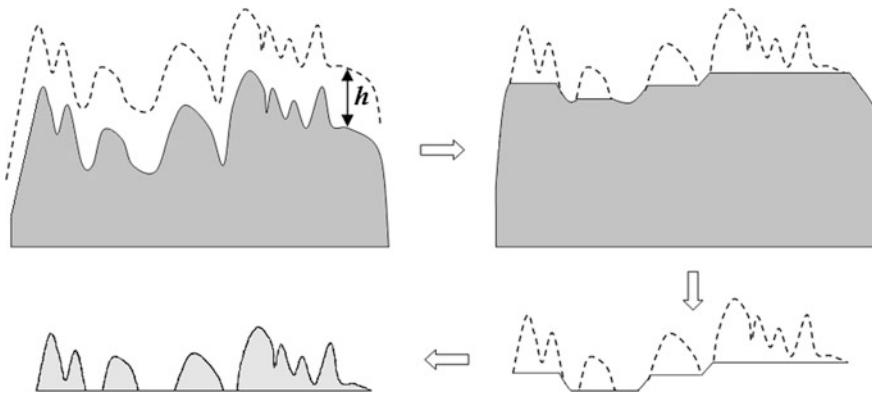


Fig. 7.1 Scheme of morphological reconstruction using the geodesic dilations

1. As a result, the peaks in the marker image J are dilated. Each dilation operation is forced the marker image to lie underneath the mask. The processing is finished, when further dilation operations do not change the marker image any more. Figure 7.1 depicts the morphological reconstruction, using the geodesic dilations of 1D signal I from a marker signal $J = I - h$.

Applied to the surfaces instead of images, the term “normalized digital surface model” is often used. A significant advantage of the geodesic filtering is that a dilation uses an elementary isotropic kernel. During the sequential processing, a classification is carried out to separate the off-terrain and terrain regions. The off-terrain regions are further analyzed by investigating the local height difference along the boundary of the objects.

The idea of some morphological methods is based on the approximation of a terrain surface, using the opening and closing operations that are built on the dilation and erosion operators. The morphological opening operation with a small window size can effectively remove the above-ground objects and generate the Earth’s surface. However, in the case of the sparse LiDAR points, the window size for morphological opening operation ought to be large to remove the objects that caused the artifacts in a surface reconstruction. Chen et al. [17] investigated this challenge, using the algorithm with the following steps:

- *The rasterization and filling missing data.* Due to the morphological operations are typically performed over a grid, the elevation of the last return of each pulse ought to be written into a grid. The grid size was set to $1 \text{ m} \times 1 \text{ m}$ according to the test data set. If a cell is without pulses, then this cell is iteratively filled with the elevations of the nearest cells. However, this strategy does not work in large areas of missing data. Notice that sparse laser data can be caused by highly absorptive materials of the above-ground objects. In order to locate the areas of having missing data, the morphological closing operation (a dilation followed by an erosion) is implemented in a binary image with a structural element, having radius r , in a view of Eq. 7.17, where d is a pulse density (number of

pulses per square meter), c is a cell size, m, $(1/d)^{0.5}$ is an average distance between pulses.

$$r = \left(\frac{1}{d}\right)^{0.5} \cdot \frac{1}{c} \quad (7.17)$$

Equation 7.17 provides the average cell number between two cells with values. After iterative application of the closing operation, the small data gaps caused by a sparse pulse density disappear and the large areas with missing data are located.

- *Objects removal.* The objects on the ground are removed by the morphological opening operations with the progressively increased windows sizes. The natural objects, such as trees, shrubs, canopy, or undergrowth vegetation, are removed using the morphological opening operations with the minimum window size. Large objects, such as buildings, are remained.
- *Low outliers detecting and filling.* The detection of the lower outliers is more difficult against the higher outliers. The higher outliers can be removed by the morphological opening operation, while the lower outliers cannot be removed. Therefore, an empirical equation with the erosion operators was proposed [17]. Then the areas with the lower outliers are filled by the approximate neighboring values.
- *Filtering over areas with buildings.* The buildings will be removed using the progressively increased window sizes. This permits to avoid a cutting of the terrain features.
- *Extraction of the terrain point.* Finally, the terrain points are extracted from the approximated surface and the DEM is generated.

The discussed algorithm works well in the urban and forested areas. The major factor is a discontinuity caused by missing data. This situation can be improved with either the advances in the sensor design or by overlapping swaths of the data.

7.4.4 Segmentation-Based Filtering

The segmentation and clustering techniques can be successfully used to separate the LiDAR points. Such methods define multiple types of the points to categorize the initial data set in order to provide finally two classes, the ground points and non-ground points. Filin [19] proposed the algorithm that extracted the surface clusters from the airborne LiDAR data. The algorithm aimed to distribute a point cloud into four different surface categories: the forested/wooded area, the low vegetation areas and rough surfaces, the smoothly varying topography, and the planar surfaces. The clustering was performed based on an attribute vector $v_i = \{x_i, y_i, z_i, \vec{n}_{\{1,2\}i}, \rho_i, d_i\}$, where (x_i, y_i, z_i) is a point position, $(\vec{n}_{\{1,2\}i}, \rho_i)$ are the parameters of the tangent plane to that point (normal measured by two parameters

and a constant), d_i is a relative height difference between the point and its neighbors. The data clustering was conducted in this space via the unsupervised classification. To accelerate the data clustering, this algorithm partitioned the feature vector into 4D attribute space, consisting of the tangent plane parameters and the height differences, and 3D point position in the object space. The clusters in this space can be considered as the “surface classes” that contain all of the points that share similar features. Then points are grouped in a cluster according to the belonging criteria. The process is repeated, until no meaningful surfaces can be proposed. Finally, each cluster is extended until no more points can be added, and then clusters merge with similar attributes. The clustering algorithm developed by Filin [19] is mentioned below:

- Step 1. Initialize n_{\min} the minimal number of the points per a cluster and s_{\max} an accuracy threshold
- Step 2. Compute attributes d_i and $(\vec{n}_{\{1,2\}i}, p_i)$ belonging to the laser points
- Step 3. Generate a feature space
- Step 4. Propose a surface class and identify the points associated with the class
- Step 5. Group the points according to the neighborhood system
- Step 6. **for** each group
 - if** group size $\leq n_{\min}$ **then** dismiss a group
 - else** compute the surface attributes for a group in particular the estimated standard deviation s
 - if** $s > s_{\max}$ **then** test for the existence of outliers
 - if** $s > s_{\max}$ **then** test for the existence of more than one class and split if it is needed
 - ednfor**
- Step 7. **repeat** steps 3–6 **until** no meaningful surfaces are proposed
- Step 8. Extend each cluster based on its attributes **until** no further points can be added or another cluster was reached
- Step 9. Merge the clusters that share similar attributes and test for a surface model
- Step 10. Analyze and group the unclassified points based on a height variation

A mode seeking algorithm better suited for identifying the planar surface elements was implemented as an unsupervised classification. The experiments show that the algorithm is robust to the existence of errors even with the relatively sparse datasets.

Jacobsen and Lohmann [48] also used the segmentation method to classify data points into several classes to separate the terrain points. Tóvári and Pfeifer [49] proposed a segmentation-based ground filtering method, which has two major steps. First, a segmentation process was initialized by a region growing method from a randomly picked seed point. Second, the method gradually added the neighboring points based on three criteria that are: a similarity of normal vectors, a distance of candidate point to the adjusting plane, and a distance between the current point and candidate point. Data points were separated into segments according to the surface objects. Then these segments were used as initial elements. During the second step,

a least squares linear interpolation based on an adaptive weight function that minimized the weights of segments was carried out.

Since many segmentation and cluster-based filtering experiments were tested on relatively flat ground surfaces [19, 27, 48, 49], the further experiments on more complex surfaces with the rough terrain are necessary to evaluate the performance, because the surfaces with less homogenous height texture could be challenging for the segmentation-based methods [26].

Charaniya et al. [50] developed the supervised parametric classification of the LiDAR and imagery data in four classes: trees, grass, roads, and buildings. Feature vector x involves five parameters, such as the normalized height, height variation, multiple returns, luminance, and intensity. The posterior probability $P(i|x)$ of a data sample x belonging to a particular class i is computed using the Bayes rule, Eq. 7.18, where $p(x|i)$ is a class condition density, $P(i)$ is a prior probability of class i , C is a number of classes.

$$P(i|x) = \frac{p(x|i)P(i)}{\sum_{i=1}^C p(x|i)P(i)} \quad (7.18)$$

Assuming that there are no prior information about $P(i)$, the prior probabilities for all classes are equal $1/C$. The unknown class condition densities are often estimated by the Gaussian mixture model provided by Eq. 7.19, where (μ_i, Σ_i) are the model parameters, $P_i(j)$ are the mixing parameters, M is a number of mixtures. Notice that the model and the mixing parameters are estimated iteratively using the Expectation Maximization (EM) algorithm that is trained on the training samples.

$$p(x|i) = \sum_{j=1}^M P_i(j)G(x|\mu_j, \Sigma_j) \quad (7.19)$$

Finally, the data sample x is assigned to the class i , for which $P(i|x)$ is maximum using the marginal histograms, the observations, a spatial coherence, etc. This method was not directed on the ground and non-ground points classification but it can be useful for classification of the above-ground objects.

The approach to segment a point cloud into the smooth segments that may still contain the height discontinuities was developed by Sithole and Vosselman [51]. Instead of classifying points in a local neighborhood, a point cloud is segmented into the patches, in which all points can be connected through a smooth path of the adjacent points. Then the detected segments are classified based on their geometric relationships with the surrounding segments. The assumptions, laying in the basis of this approach, are mentioned below [51]:

- The surface of each object is encompassed in one or more continuous surface segments.
- All points in a surface segment that can be described by the locally continuous shape functions belong to the same surface.

- Object segments are separated from the bare Earth segments by discontinuities.
- The perimeter of each object is mostly raised above its neighborhood.

In order to get a disconnected graph, a cuboid of the LiDAR point cloud is segmented by the scan lines with multiple orientations. Then a weighted minimum spanning tree is generated for a profile, and all edges with a weight greater than a defined threshold are removed.

The weight of edges w in the weighted minimum spanning tree is computed using the proximity function provided by Eq. 7.20, where (x_i, z_i) and (x_j, z_j) are the profile coordinates of the end points i and j of an edge, k ($k > 1$) is the end-user defined parameter that scales the proximity function so that the points along the OX-axis are more proximal than the points along the OZ-axis.

$$w = (x_j - x_i)^2 + k(z_j - z_i)^2 \quad (7.20)$$

Such approach permits to detect the continuous smooth segments for the following classifying. Six topological relationships, such as raised, lowered, terraced, high, low, and no shape, were identified between the adjacent line segments. In a landscape object surface, segments are mainly associated with the raised and high line segments. The lowered and low line segments are mainly associated with the bare Earth surfaces. The terraced line segments are associated with both the bare Earth and object surface segments. Therefore, the surface segments, containing a majority of the raised and high line segments, are classified as an object. The following step is a removal the large above-ground objects from a point cloud. The point cloud is again segmented (with generation of the surface segments) but with a larger value of k increased by a factor of 4 and the increased user-defined threshold during each repetition. This segmentation-classification step is repeated several times until large objects would not be removed. The final step is devoted to form a bare Earth point cloud without small objects like shrubs, cars, park benches, etc. The point is deemed to belong to small object if it is above the plane fitted to its nearest neighboring points and if the standard deviation of the residuals is above a user-defined threshold. The main advantage of this algorithm is a possibility to work on raw and irregular spaced LiDAR points. Due to this possibility, the algorithm may be applied to analyze many types of landscape.

An unsupervised filtering based on k -means clustering of 3D-point cloud was proposed by Chehata and Bretar [52] as the first step of their algorithm. Data filtering into the ground and non-ground points is based on a multi-resolution approach with features (the mean and standard deviation of heights) and a tuning parameter (the window size d_s). The small window size provides the true ground points within the neighborhood. On the contrary at the second step, the large window size tends to smooth the DTM that is built using the Markovian regularization process in the Bayesian framework.

Consider the DTM as a georeferenced regular gridded surface with a resolution r . The local 3D environment V_s in local region s has a view of a cylindrical neighborhood centered on s and of diameter $d_s = 2 * r$. A hierarchical k -means

algorithm that provides an accurate filtering by an iterative split of the ground cluster can avoid the LiDAR point overfiltering by the following procedure. At first, 3D point cloud is initialized as the non-ground. The cylindrical neighborhood V_s for each region s is analyzed on the presence of the negative outliers. If the percentage of negative outliers is over 80%, then the LiDAR points are labeled as the non-determined points. Then the centroids are initialized at equal distance on Z interval. The number of the initial centroids is initialized to 1, and it increases iteratively up to three (the correspond to ground, non-ground, and low non-ground classes), while $\sigma_C > 1$ m. The cluster, whose distance Z has a minimum value, is considered as the ground cluster. In the following iteration, the algorithm splits the ground cluster into two classes: if the standard deviation σ_C of each cluster is higher than a threshold T_n , where n is the number of cluster splits. At each split, a threshold is changed as $T_{n+1} = T_n / 2$. When the algorithm converges, the ground cluster is labeled, and the propagation continues through the 3D point cloud without processing of the labeled ground points. This method provides a robust filtering of the ground points. The result of the first step is the coarse DTM with an approximate surface and a corresponding map, depicting the number of the cluster splits per a region. The less the number of splits, the more reliable the classification can be obtained. During the second step, the DTM regularization, a terrain surface becomes smooth due to k -means clustering, using the large window size.

It is interesting that the modern technologies of computer vision can be employed in the GIS application, for example, Guo et al. [53] proposed a supervised classification method to identify types of the man-made objects, including the power-lines and pylons from the point clouds by use of the JointBoost classifier. The parameters for the learning model were estimated with various features computed based on the geometry and echo information of the LiDAR point cloud.

The analysis of 3D urban scenes represented as the LiDAR point cloud in the terms of semantic labeling is discussed in [54]. Weinmann et al. adapted the conventional approach, including a neighborhood selection (7 neighborhood definitions), a feature extraction (21 geometric features), a feature selection (7 approaches), and a classification (10 classifiers) for the photogrammetric tasks of the urban areas. For experiments, the Oakland 3D Point Cloud Dataset was utilized, and a distribution of the assigned optimal neighborhood for the different classes, such as Wire, Pole/Trunk, Facade, Ground, and Vegetation, was built.

7.4.5 Hybrid Methods

Recently, some researchers became to develop the hybrid filtering algorithms that do not directly fall into any of the above categories. They make use of a combination of the filtering techniques. Kobler et al. [55] proposed a two-stage algorithm. In the first stage, the negative outliers are removed using a morphological filter and then a slope-based filter removes the most non-ground points. This stage produces partially filtered data that enters the second stage of the algorithm. In the second

stage, the elevations at unsampled locations are estimated from the repeated independent estimates taken from the sampled points. By combining of different filtering methods, the algorithm performed better results than the individual filters.

Maguya et al. [56] developed a new adaptive algorithm that generates the DTM for a complex terrain, including hills, steep slopes, and plateau. The main idea is to partition the predicted terrain into the patches with various types of the surfaces. For relatively less complex patches, a combination of the trend surfaces can be used, while for more complex patches, the cubic splines are adopted. The partitioning was done by dividing the XOY plane into the square patches (tiles) of the given width W . The algorithm proceeded in two steps: a filtering of non-ground points and the DTM interpolation. The first step is controlled by two parameters: a quality of fit r^2 , $r \in [0...1]$, and a search window width (w). The parameter r^2 influences on an accuracy of the DTM directly (the higher value is the better) and enables the algorithm to adapt to the changes in a terrain by choosing an appropriate interpolation strategy (a planar trend surface, a parabolic trend surface, or a cubic spline interpolation). Value of r^2 was estimated using Eq. 7.21, where $\{z_i\}$ are the observed minimum elevation values in the current patch, $\{\hat{z}_i\}$ are the predicted elevation value in the current patch, \bar{z} is a mean of $\{z_i\}$, n is a number of observed elevation values. The sum in a numerator of Eq. 7.21 is a residual sum of squares.

$$r^2 = 1 - \frac{\sum_{i=1}^n (\hat{z}_i - z_i)^2}{\sum_{i=1}^n (\bar{z} - z_i)^2} \quad (7.21)$$

The value of w determines the area to search for a point with the lowest elevation value. Maguya et al. [56] used values $r^2 = 0.95$, $w = 5$ m, and $W = 40$ m. The high value of r^2 used was chosen in order to attain the high DTM accuracy. In general, the smaller value of w is the better because it helps to minimize an interpolation error. However, there are no LiDAR points that can be detected with values of w less than 5 m. The optimal value of W was chosen experimentally with a range of values (10–80 m) looking for a value that is robust and minimizes the border effects.

For each trend surface, the residuals and r^2 values are computed. The points with the negative residuals (set A) below the surfaces are assumed to be the ground points. The points with the positive residuals (set B) above each surface consist of a mixture of both ground and non-ground points. At this stage, if either model has a value of r^2 below the threshold value, it is discarded. Then a filtering of the non-ground points is done on points in a set B . The points in a set B are sorted in the descending order of residuals. Starting with the point with the largest positive residual, each point is left out in turn and the corresponding surface is re-fit using the remaining points in both sets A and B . The value of r^2 for a new surface is computed and compared to the previous value. If the value of r^2 increases, the point is considered to be the non-ground point and discarded from a set B ; otherwise the point is considered to be the ground point. In such manner, a set of ground points C is formed. Data from set C is the input data for the second interpolation stage of

the algorithm. Spline interpolation is performed, when both linear and quadratic models fail. Due to the patches technique applied, sometimes the area covered by the input LiDAR is not a rectangular in a shape, and some of the square patches along the diagonal have the non-significant LiDAR data. In this case, a linear or a quadratic trend surface cannot be fitted to the data as a linear model, and a spline interpolation fails. Also the embedded gaps may appear in the initial data that leads to gaps in the resulting interpolated DTM. The impact of this effect can be reduced by decreasing the patch width w but a final feasibility of this solution depends on the data density or the use of the higher-order polynomial interpolations [57].

The surface patches based on the triangular splines allow the enhanced geometric control. Amongst others, the Bezier splines [58] and the Coons patches [59] have been successfully applied. Similar to the interpolation techniques like the Bezier and B-splines, the Coons patches have been developed in the mid 1960s for the digital design of car bodies [60]. Hereinafter, these interpolation techniques have been widely applied in the natural scenes, involving the DTM generation. However, a major drawback is the inability to represent the breaklines.

7.4.6 Comparison of Filtering Methods

The LiDAR points involve the ground points (the points of the bare Earth's surface), the non-ground points (the points of vegetation, buildings, bridges, and other man-made constructions), and the noise points as the error measurements. The ground points have general physical characteristics that are commonly used as the assumptions in many filtering methods. These general physical characteristics can be divided into the following categories [26]:

- *Lowest elevations.* The ground points usually have the lowest elevations among their neighboring points in a certain local area. Many filtering methods employ this assumption to search the initial ground surface points for filtering.
- *Ground surface steepness.* The surface slope between two neighboring ground points is often smaller than between the ground point and the non-ground point. Therefore, many filtering methods use the surface slope threshold as the criteria to distinguish the ground and non-ground points. However, the threshold values may vary significantly even on the same sloped terrain; in other words, these slope threshold values are not constants. Notice that the steeper slopes are the mountain terrain, high-relief forest canopy surfaces, and even urban surfaces.
- *Ground surface elevation difference.* The elevation difference between two neighboring ground points is normally smaller than between the ground and non-ground point. Many filtering methods use this parameter as the criterion to separate the ground and non-ground points.
- *Ground surface homogeneity.* The ground surfaces are relatively continuous and smooth, especially in a flat terrain. Trees and buildings are the major non-ground objects that should be removed from the measurements. The trees

are usually less smooth in a texture than the building surfaces, and the morphological or fractal methods can be utilized to remove them.

The main problems of filtering methods are caused by the following aspects:

- Low vegetation, such as shrubs.
- Complex man-made constructions, such as bridges.
- Buildings with complex shapes and different sizes.
- Hill cut-off edges.
- Mixing of terrain types.
- Lack of reliable accuracy assessment.

The low vegetation measurements are usually mixed with the ground surfaces because the elevation change on the low vegetation measurements is very similar to the terrain variation. This results to identical distribution of these low elevation measurements, and the problem becomes incorrect without additional information, for example, from the digital photographs. The morphological filters are more often employed in the mixed types of a terrain. The complex man-made constructions, especially the complex bridges and power transmission lines, are also difficult to filter. It is difficult to separate the constructions that are connected with ground surfaces smoothly. Many filters are not able to handle the hill cut-off edges, such as cliffs, riverbanks, and buildings, well because they have a sharp change on the slope or elevation. This promotes to continue the development of slope-based methods.

Zhang and Whitman [61] investigated three methods for the non-ground measurements of the LiDAR data, including the Elevation Threshold with Expanding Window (ETEW), the Maximum Local Slope (MLS), and the Progressive Morphological (PM) filters. The test data sets were taken from [62] and involved the low and high-relief data sets with various densities of trees, houses, and sand dunes. Zhang and Whitman [61] reported that all three methods removed effectively the non-ground points in both low-relief urban and high-relief forested areas. The PM filter generated the best result in the coastal barrier island areas, whereas the MLS and the PM tended to remove the tops of the steep sand dunes. The omission or commission errors were estimated using a set of cell sizes: 2 m × 2 m, 3 m × 3 m, 4 m × 4 m, 5 m × 5 m, and 6 m × 6 m. It was found that a topographic slope is the most sensitive parameter for all filtering methods.

Seven interpolation routines, using the small footprint LiDAR data collected over a range of vegetation classes on Vancouver Island, British Columbia, Canada, were tested by Bater and Coops [63]. A set of the DEMs were generated using a linear, a quintile, a natural neighbor, a regularized spline, a spline with tension, a finite difference approach, and an inverse distance weighted interpolation routines, at spatial resolutions of 0.5, 1.0, and 1.5 m and various terrain slope, vegetation structural class, the LiDAR ground return density, and the Normalized Difference Vegetation Indices (NDVI). The spline and the Inverse Distance Weighed (IDW) algorithms appear to be more sensitive to the changes in a slope than other methods. The TIN-based routines demonstrated better results in comparison to the splining and the IDW approaches.

Nevertheless, the hybrid techniques provide objectively the reasonable advantages against the individual filtering methods with the losses in the computational costs.

7.5 Generation of Digital Terrain Model

The processed LiDAR data do not distributed regularly. Thus, the LiDAR point cloud ought to be interpolated in any grid, and a grid size ought to be selected according to the resolution of the LiDAR data set. The DTM implementation is based on two main approaches, when the height data are arranged in a form of the regular grid (e.g., square, rectangle, triangular, or hexagonal) or they are collected randomly in a form of the irregular grid (often triangular) with various sizes and orientations in 3D space. Fisher and Tate [64] pointed out that there seemed to be no preferable interpolation algorithm for the LiDAR data. As a result, researchers can choose any interpolation method, e.g., the IDW method that is widely available in most GIS software.

Further the most interesting approaches (from an algorithmic point of view) for the DTMs generation will be discussed including the spatial relationship methods (Sect. 7.5.1), the statistical methods (Sect. 7.5.2), and the curve-fitting methods (Sect. 7.5.3). The interpolating techniques, using in the ArcGIS Spatial Analyst, are presented in Sect. 7.5.4. Section 7.5.5 provides the recommendations of the accuracy estimators.

7.5.1 *Spatial Relationship Methods*

The spatial relationship methods are called as the deterministic interpolation techniques that generate the surfaces from the measured points based on either the extent of similarity (the IDW method) or the degree of smoothing (radial basis functions). The deterministic interpolation techniques can be divided into two groups: the global techniques calculate the predictions using the entire dataset, while the local techniques calculate the predictions from the measured points within neighborhoods. The IDW, local polynomial, and radial basis functions are the local interpolators, while the global polynomial and triangulation are the global interpolators. Another classification deals with a forcibly passing of the resulting surface through the measured data values. An interpolation technique that predicts a value identical to the measured value at a sampled location is an exact interpolator. An inexact interpolator predicts a value that is different from the measured value (this can be used to avoid sharp peaks or troughs in the output surface). The IDW and radial basis functions are exact interpolators, while the local and global polynomials are inexact. Consider these methods successively.

The IDW method. The IDW method employs the measured values surrounding the prediction location in order to predict a value of any unmeasured location. The measured values closest to the prediction location have more influence on the predict value than those farther away. The influence is reflected in the weighting values of the points, when the weights of the points closer to the prediction location greater than those farther away, hence the method is called the inverse distance weighted. The predicted values are calculated by Eq. 7.22, where $\hat{Z}(\mathbf{s}_o)$ is a predicted value in location \mathbf{s}_o (symbol s contains x (longitude) and y (latitude) coordinates), λ_i is a weight assigned to each measured points (the weights are decreased with a distance), $Z(\mathbf{s}_i)$ is an observed value at the location \mathbf{s}_i , N is a number of measured sample points surrounding a prediction location.

$$\hat{Z}(\mathbf{s}_o) = \sum_{i=1}^N \lambda_i Z(\mathbf{s}_i) \quad (7.22)$$

The weights are determined by Eq. 7.23, where p is a reduced power factor, d_{io} is a distance between the prediction location \mathbf{s}_o and the measured location \mathbf{s}_i .

$$\lambda_i = d_{io}^{-p} \sum_{i=1}^N d_{io}^{-p} \quad \sum_{i=1}^N \lambda_i = 1 \quad (7.23)$$

The optimal power parameter p can be determined by minimizing the Root-Mean-Square Prediction Error (RMSPE) that is calculated by a cross-validation method. The RMSPE is a summary statistic quantifying the error of the prediction surface. Usually several different powers are defined for the same data set. The power that provides the smallest RMSPE is determined as the optimal power.

As it follows from Eq. 7.23, if $p = 0$, then there is no decrease with distance because each weight λ_i will be the same and all the measured values will be involved in the prediction. If the p value increases, then the weights of distance points decrease rapidly. If the p value is too high, then only few surrounding points will influence the prediction. In practice, a number of measured values are limited in such manner that the farther away points with little influence are discounted to zero. The shape of a neighborhood can be uniform, when there is no directional influence on the weighting of data, and non-uniform, when, for example, a prevailing wind can be presented. In the first case, a shape of neighborhood is a circle. In the second case, a shape of neighborhood may be an ellipse with the major axis parallel with the wind direction.

The radial basis functions interpolation. In this case, the predictor is a linear combination of the basis functions provided by Eq. 7.24, where $\phi(r)$ is a radial basis function, $r = \|\mathbf{s}_i - \mathbf{s}_o\|$ is the Euclidean distance between the prediction location \mathbf{s}_o and each data location \mathbf{s}_i , $\{\omega_i\}$, $i = 1, 2, \dots, n + 1$ are the weights to be estimated, ω_{n+1} is a bias parameter, n is a number of points.

$$\hat{Z}(\mathbf{s}_o) = \sum_{i=1}^n \omega_i \phi(\|\mathbf{s}_i - \mathbf{s}_o\|) + \omega_{n+1} \quad (7.24)$$

Let $\mathbf{w} = (\omega_1, \omega_2, \dots, \omega_n)$, which are found by solving a system of equations in a view of Eq. 7.25, where Φ is a matrix with (i, j) th element $\phi(\|\mathbf{s}_i - \mathbf{s}_o\|)$ for the (i, j) th data pair, $\mathbf{1}$ is a column vector of all ones, \mathbf{z} is a column vector containing the data.

$$\begin{pmatrix} \Phi & \mathbf{1} \\ \mathbf{1}' & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \omega_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ \mathbf{0} \end{pmatrix} \quad (7.25)$$

If Φ is a vector containing $\phi(\|\mathbf{s}_i - \mathbf{s}_o\|)$, then the predictor is

$$\hat{Z}(\mathbf{s}_o) = \mathbf{w}' \Phi + \omega_{n+1}.$$

The radial basis functions enable to generate a surface that captures the global trends and picks up the local variations and demonstrate the ability to bend and stretch the predicted surface so that it passes through all of the measured values. The radial basis functions may be used to form various curves (a thin plate spline) or to control the edges of the surface (a spline with tension).

The spline with tension is described by Eq. 7.26, where I_o is a modified Bessel function, γ is the Euler's constant, σ is an optimal smoothing parameter that is calculated by minimizing the root-mean square error using a cross-validation [65].

$$\phi(r) = \ln(\sigma r / 2) + I_o(\sigma r) + \gamma \quad (7.26)$$

The local polynomial interpolation. The local polynomial interpolation is similar to the global polynomial interpolation except that it uses data within the local surroundings (localized slicing windows). The surface value at the center of the window called as $\mu_o(x, y)$ is estimated at each point. The weighted least squares method is used by minimizing Eq. 7.27, where n is a number of points within the window.

$$\sum_{i=1}^n w_i (Z(x_i y_i) - \mu_o(x_i y_i))^2 \rightarrow \min \quad (7.27)$$

The local weight w_i is determined by Eq. 7.28, where d_{io} is a distance between the point and the center of the window, a is a parameter, which controls how fast weights decay with distance.

$$w_i = \exp(-3d_{io}/a) \quad (7.28)$$

The term $\mu_o(x, y)$ is the first-order or the second-order polynomial defined by Eqs. 7.29 and 7.30, respectively.

$$\mu_0(x_i, y_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i \quad (7.29)$$

$$\mu_o(x_i, y_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_i^2 + \beta_4 y_i^2 + \beta_5 x_i y_i \quad (7.30)$$

The global polynomial interpolation. The global polynomial interpolation uses the ordinary multiple regression methods, processing all data. A trend surface is fitted to (x, y) coordinates, which are the covariates. Equations 7.31–7.33 provide the models for the first-order trend, the second-order trend, and the third-order trend, respectively, where $Z(x_i, y_i)$ is the datum at location (x_i, y_i) , $\{\beta_j\}$ are the parameters, $\varepsilon(x_i, y_i)$ is a random error.

$$Z(x_i, y_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i + \varepsilon(x_i, y_i) \quad (7.31)$$

$$Z(x_i, y_i) = \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_i^2 + \beta_4 y_i^2 + \beta_5 x_i y_i + \varepsilon(x_i, y_i) \quad (7.32)$$

$$\begin{aligned} Z(x_i, y_i) = & \beta_0 + \beta_1 x_i + \beta_2 y_i + \beta_3 x_i^2 + \beta_4 y_i^2 + \beta_5 x_i y_i + \beta_6 x_i^3 \\ & + \beta_7 y_i^3 + \beta_8 x_i^2 y_i + \beta_9 x_i y_i^2 + \varepsilon(x_i, y_i) \end{aligned} \quad (7.33)$$

These trends can be continued up to 10 degree. Fitting regression models by the estimating parameters $\{\beta_j\}$ uses the ordinary least squares method.

The triangulation interpolation. The triangulation interpolation is another popular spatial interpolation method that is normally based on the Delaunay triangulation. The triangulation serves two purposes in a terrain modelling: as a basis for the TIN data structure and as the basis for the DTM interpolation. The Delaunay triangulation algorithms can be grouped into three categories: a triangulation growth, a divide-and-conquer algorithm, and a point by point insertion method. The last approach is the most popular and includes the following steps. First, a set of discrete points of convex hull is established. Second, depending on the nature of the Delaunay triangulation, a triangulation of the convex hull is achieved generating the initial triangulation. Third, the points inside the convex hull are inserted into the initial triangular network to generate the final Delaunay triangulation.

The triangulation was invented by Boris Delaunay [66]. In mathematics and computational geometry, a Delaunay triangulation for a set S of points in a plane is a triangulation $D(S)$ such that no point in S is inside the circumcircle of any triangle in $D(S)$. The Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid the skinny triangles. In other words, a triangulation of a set of points is the Delaunay triangulation if, and only if, the circumcircle of any of its triangles does not contain any other point in its interior. The Delaunay triangulation is bounded by the convex hull.

The dual of the Delaunay triangulation is called the Voronoi diagram (or the Thiessen polygons, or the Dirichlet tessellation). Nodes of the Voronoi polygon are coincident with the circumcentres of the Delaunay triangles (Fig. 7.2). The perpendicular bisectors of the Delaunay edges form the edges of the Voronoi polygons. Since they are dual, the Delaunay triangulation may be constructed from its

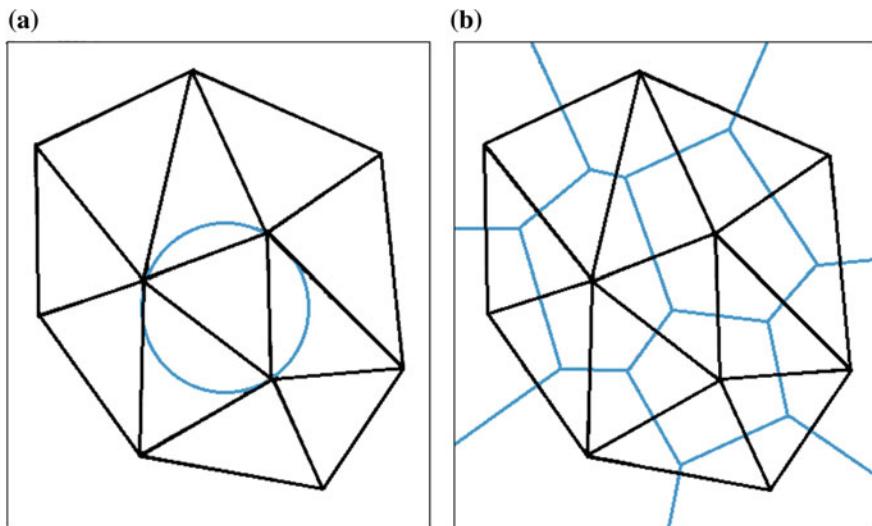


Fig. 7.2 Scheme of the Delaunay triangulation: **a** the Delaunay triangulation (black) with a circumcircle (blue) of one of the triangles, **b** the Delaunay triangulation (black) and its dual Voronoi diagram (blue)

Voronoi diagram and vice versa. The statement of the Voronoi diagram is as the following. Consider a set V of n points. The closest-site Voronoi diagram of V subdivides a plane into the regions. Each region is associated with one site $v_i \in V$ and contains all points, for which v_i is the closest among all sites of V [67].

Sometimes in practice, it is impossible to apply the pure Delaunay triangulation due to the geometrical constraints, which break the Delaunay criterion. Sometimes a triangulation ought to consider the contour data. This difficult issue is under consideration during many years [68]. The generating of realistic terrains with the higher-order Delaunay triangulations is discussed in [69].

Another problem is a computation time of the LiDAR data process. Isenburg et al. [70] proposed the greatly accelerate algorithms of the Delaunay triangulations for well-distributed point sets in 2D and 3D space by exploiting the natural spatial coherence in a stream of points. A special program technique was introduced, viz. a spatial finalization into the point streams, when a space was portioned into the regions and a stream of input points with finalization tags that indicate, when a point is the last in its region was enlarged. This has made possible representation of billion-triangle terrain (11.2 GB of the LiDAR data) to process in 48 min using only 70 MB of memory on a laptop with two hard drives. The acceleration techniques based on the multi-core architectures are discussed in [71].

The fully-automated Rational Polynomial Coefficients (RPCs) bias compensation of the high resolution satellite imagery was proposed by Oh and Lee [72]. The map features are scaled and aligned to the image features through the RPCs-based image projection, and then the local shifts are automatically determined by

template-based edge matching of the heterogeneous data set. The map features are selected based on four suggested rules:

- The map features are excluded if they do not appear in the real world.
- The map features do not selected if their images may be different over seasons.
- The topography layers, such as contours, are also not selected.
- The elevation of the map features should be known.

These rules help to simplify the bias compensation and enable to model the RPCs bias compensation parameters for the accurate georeferencing.

Notice that some authors investigate new type of splines, for example, the locally refined B-splines based on the non-uniform B-splines, which allow all refinements, where B-spline is split [73]. The use of a combination of the interpolation methods in order to improve an accuracy of the DTM generated is discussed in [74].

7.5.2 Statistical Methods

The geostatistical interpolation techniques (kriging) utilize the statistical properties of the measured points. In 1963, the French mathematician G. Matheron called the optimal or the Best Linear Unbiased Prediction (BLUP) method as a kriging named after the South African mining engineer D. G. Krige, as it is still known in spatial statistics today. The geostatistical techniques quantify the spatial autocorrelation among the measured points and account for the spatial configuration of the sample points around the prediction location. The kriging [75] is very popular geostatistical interpolation method that produces the visually engaging maps from the irregularly spaced data. This technique belongs to the family of the linear least squares estimation algorithms that achieve the accurate or smoothing interpolation by specifying the appropriate variogram model.

The kriging is similar to the IDW: it weights the surrounding measured values to derive a prediction for each location [76]. However, the weights are based not only on the distance between the measured points and the prediction locations but also on the overall spatial arrangement among the measured points. The spatial autocorrelation ought to be quantified in order to use the spatial arrangement in the weights. The kriging technique supposes the following steps:

- *Calculate the empirical semivariogram.* The kriging is built on the conventional (for interpolation techniques) assumption: the points that are close to one another are more alike than those farther away. The empirical semivariogram is a mean to explore this relationship (Fig. 7.3). Pairs that are close in distance should have a smaller measurement difference than those farther away from one another.
- *Fit a model.* This is done by defining a line that provides the best fit through the points in the empirical semivariogram cloud graph. Often the weighted

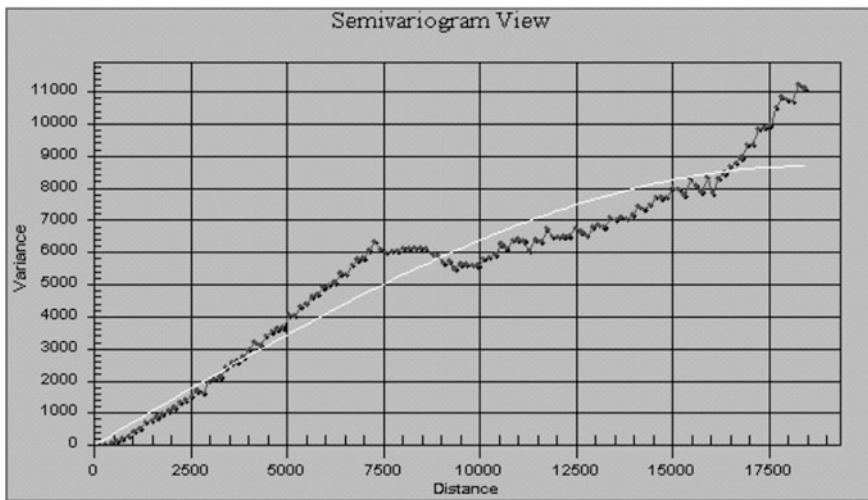


Fig. 7.3 An example of an omni-direction semivariogram. The *white line* represents the model that will be used in the kriging interpolation

least-square fit is used. The line is considered a model that quantifies the spatial autocorrelation in data.

- *Create the matrices.* The equations for the ordinary kriging are contained in matrices and vectors that depend on the spatial autocorrelation among the measurement sample locations and the predicted location. The matrices and vectors determine the kriging weights that are assigned to each measured value. The autocorrelation values are taken from the semivariogram model.
- *Make a prediction.* A prediction for the location with the unknown value can be calculated from the kriging weights of the measured values.

Besides the ordinary kriging that will be considered in details hereinafter, one can mention the simple kriging, the universal kriging, the lognormal linear kriging, the transgaussian kriging, the indicator kriging, the probability kriging, the disjunctive kriging, and the cokriging. The reader can find more information about the family of these statistical methods in literature [77–83].

The geostatistical data are expressed in a simple Eq. 7.34, where $Z_t(\mathbf{s})$ denotes the t th realization at location \mathbf{s}_i (the index i is omitted for simplicity) that is decomposed into a deterministic trend $\mu(\mathbf{s})$ and a random autocorrelated error $\varepsilon_t(\mathbf{s})$.

$$Z_t(\mathbf{s}) = \mu(\mathbf{s}) + \varepsilon_t(\mathbf{s}) \quad (7.34)$$

Let n_i be a number of measurements at location \mathbf{s}_i . Often $n_i = 1$, and if $n_i > 1$, it forms a measurement error model.

Decompose a random error $\varepsilon_t(\mathbf{s})$, using Eq. 7.35, where $Y(\mathbf{s})$ is an error of smooth second-order autocorrelation, $\eta(\mathbf{s})$ is a noise error, $\delta_t(\mathbf{s})$ is a measurement error of the t th realization at location \mathbf{s}_t .

$$\varepsilon_t(\mathbf{s}) = Y(\mathbf{s}) + \eta(\mathbf{s}) + \delta_t(\mathbf{s}) \quad (7.35)$$

The following assumptions are made:

- $\mu(\mathbf{s}) = \mu$ is an unknown, deterministic mean value.
- $Y(\mathbf{s})$ is a smooth second-order stationary process, whose range of autocorrelation is detectable with the empirical semivariogram or covariance.
- $E(Y(\mathbf{s})) = 0$.
- $\text{Cov}(Y(\mathbf{s}), Y(\mathbf{s} + \mathbf{h})) = C_Y(\mathbf{h})$. There is no additional nugget effect in the process $Y(\mathbf{s})$.
- $h(\mathbf{s})$ is a smooth second-order stationary process, whose variogram range is so close to 0 that it is smaller than all practical distances between data and prediction locations.
- $E(h(\mathbf{s})) = 0$.
- $\text{Cov}(h(\mathbf{s}), h(\mathbf{s} + \mathbf{h})) = C_h(\mathbf{h})$ with $C_h(Y) = 0$.
- $d_t(\mathbf{s})$ is a white-noise process composed of measurement errors.
- $E(d_t(\mathbf{s})) = 0$ for all \mathbf{s} and t .
- $\text{Cov}(d_t(\mathbf{s}), d_u(\mathbf{s} + \mathbf{h})) = s^2$, if $\mathbf{h} = 0$ and $t = u$, otherwise it is 0.
- $Y(\cdot)$, $h(\cdot)$, and $d(\cdot)$ are independent of each other.

Assume that the nugget effect, which is called v , is composed of two parts: the micro-scale variations and measurement error. That is, $v = C_\eta(\mathbf{0}) + \sigma^2$. A covariance for different cases is estimated by Eq. 7.36.

$$\text{Cov}(Z_t(\mathbf{s}), Z_u(\mathbf{s} + \mathbf{h})) = \begin{cases} C_Y(\mathbf{h}) + C_\eta(\mathbf{h}) & \text{if } \mathbf{h} \neq \mathbf{0} \\ C_Y(\mathbf{0}) + C_\eta(\mathbf{0}) & \text{if } \mathbf{h} = \mathbf{0} \text{ and } t \neq u \\ C_Y(\mathbf{0}) + C_\eta(\mathbf{0}) + \sigma^2 & \text{if } \mathbf{h} = \mathbf{0} \text{ and } t = u \end{cases} \quad (7.36)$$

If there is measurement error, a prediction of the filtered (noiseless) quantity will be $S(\mathbf{s}_o) \equiv \mu + Y(\mathbf{s}_o) + \eta(\mathbf{s}_o)$ at location \mathbf{s}_o . If there is no measurement error, $S(\mathbf{s}_o) = Z(\mathbf{s}_o)$. The ordinary kriging with measurement error is obtained for the linear predictor

$$\hat{S}(\mathbf{s}_o) = \boldsymbol{\lambda}' \mathbf{z},$$

then minimize

$$E(S(\mathbf{s}_o) - \boldsymbol{\lambda}' \mathbf{z})^2,$$

where \mathbf{z} is a vector of the observed data and $\boldsymbol{\lambda}$ is a vector of the kriging weights.

An unbiased condition

$$E(S(\mathbf{s}_o) - \boldsymbol{\lambda}' \mathbf{z}) = 0$$

implies $\boldsymbol{\lambda}' \mathbf{1} = \mathbf{1}$, which causes the need to use the Lagrange multipliers, when minimizing. Thus, obtain the kriging equations

$$\begin{pmatrix} \boldsymbol{\Sigma}_z & \mathbf{1} \\ \mathbf{1}' & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ m \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ 1 \end{pmatrix},$$

where m is the Lagrange multiplier, $\boldsymbol{\Sigma}_z$ is the covariance matrix among the data, \mathbf{c} is $\text{Cov}(\mathbf{z}, S(\mathbf{s}_o)) = \text{Cov}(\mathbf{z}, Y(\mathbf{s}_o) + \eta(\mathbf{s}_o))$.

Assuming that the range of $\eta(\cdot)$ is very close to 0, suppose $\text{Cov}(\mathbf{z}, \eta(\mathbf{s}_0)) = 0$ for all practical distances except when $\mathbf{s}_o = \mathbf{s}_t$, where \mathbf{s}_t is one of the spatial locations for observed data. Then $\text{Cov}(Z(\mathbf{s}), \eta(\mathbf{s}_o)) = C_\eta(\mathbf{0})$, which needs to be estimated. The total nugget effect can be estimated as $v = \sigma^2 + C_\eta(\mathbf{0})$. If there is a separate estimate of σ^2 , then $C_\eta(\mathbf{0}) = v - \sigma^2$. Identify $\sigma^2 = \pi v$, $0 \leq \pi < 1$, and $C_\eta(\mathbf{0}) = (1 - \pi)v$ can be specified. If there are multiple observations per location, then a measurement error can be estimated as shown earlier.

The parameters σ^2 and $C_\eta(\mathbf{0})$ are specified, proceed to solve the kriging equations. If all of the nugget effect is a micro-scale variation due to $\eta(\cdot)$ (no measurement error), then the solution to the kriging equations yields the exact kriging. Solving for $\boldsymbol{\lambda}$, obtain for the ordinary kriging predictor:

$$\boldsymbol{\lambda} = \boldsymbol{\Sigma}_z^{-1}(\mathbf{c} - \mathbf{1}m), \text{ where } m = (\mathbf{1}' \boldsymbol{\Sigma}_z^{-1} \mathbf{c} - 1) / (\mathbf{1}' \boldsymbol{\Sigma}_z^{-1} \mathbf{1}).$$

Substituting in this $\boldsymbol{\lambda}$, the mean-square predication error will be received:

$$\begin{aligned} E(S(\mathbf{s}_o) - \boldsymbol{\lambda}' \mathbf{z})^2 &= C_Y(\mathbf{0}) + C_\eta(\mathbf{0}) - \boldsymbol{\lambda}' \mathbf{c} - m \\ &= C_Y(\mathbf{0}) + (1 - \pi)v - \boldsymbol{\lambda}' \mathbf{c} - m, \end{aligned}$$

so the prediction standard errors are

$$\hat{\sigma}_S(\mathbf{s}_o) = \sqrt{C_Y(\mathbf{0}) + (1 - \pi)v - \boldsymbol{\lambda}' \mathbf{c} - m}.$$

The predicting of new value for the cross-validation and validation is estimated in the same manner. During a cross-validation, the noiseless version of data $S(\mathbf{s}_o)$ ought to be predicted. At first, predict $Z_u(\mathbf{s}_o)$ with measurement error. Prediction of “new value” is obtained for the linear predictor

$$\hat{Z}_u(\mathbf{s}_o) = \boldsymbol{\lambda}' \mathbf{z},$$

then minimize

$$E(Z_u(\mathbf{s}_o) - \lambda' \mathbf{z})^2 \rightarrow \min.$$

Assume that if $\mathbf{s}_o = \mathbf{s}_i \in D$, then $u > n_i$. Notice that if $\mathbf{s}_o = \mathbf{s}_i \in D$, the prediction $\hat{Z}_u(\mathbf{s}_o)$ is usually not equal to one of the observed values $z_t(\mathbf{s}_i)$, $t \leq n$. Substituting in mentioned above expression λ , get the mean-square prediction error

$$\begin{aligned} E(Z_u(\mathbf{s}_o) - \lambda' \mathbf{z})^2 &= C_Y(\mathbf{0}) + C_{\eta}(\mathbf{0}) + \sigma^2 - \lambda' \mathbf{c} - m \\ &= C_Y(\mathbf{0}) + v - \lambda' \mathbf{c} - m, \end{aligned}$$

so the prediction standard errors are

$$\hat{\sigma}_Z(\mathbf{s}_o) = \sqrt{C_Y(\mathbf{0}) + v - \lambda' \mathbf{c} - m}.$$

Notice that, when $\mathbf{s}_o = \mathbf{s}_i$ for one of the observed data locations, $\mathbf{s}_i \in D$, then neither of the prediction standard errors will be 0.

If the random errors are normally distributed with a stationarity, then the prediction error $\hat{S}(\mathbf{s}_o) - S(\mathbf{s}_o)$ has a normal distribution with zero mean and variance $\hat{\sigma}_s^2(\mathbf{s}_o)$. Normality allows the computing of a probability map or a quantile map.

Chaplot et al. [84] investigated the accuracy of the interpolation techniques of the point height data with density values from 4 to 109 points/km² for six territories with various surface areas from micro-plots, hill-slopes, and catchments. The spectrum of the interpolation techniques included the inverse distance weighting, the ordinary kriging, the universal kriging, the multi-quadratic radial basis function, and the regularized spline with tension. Irrespective of the spatial scales or the variability and spatial structure of altitude, few differences existed between the interpolation methods if the sampling density was high. The multi-quadratic radial basis function performed slightly better. At lower sampling densities, the kriging yielded the best estimations for the landscapes with the strong spatial structure, low coefficients of variation, and low anisotropy. The regularized spline with a tension yielded the best estimations for the landscapes with the low coefficients of variation and weak spatial structure. Under the conditions of the high coefficients of variation, strong spatial structure, and strong anisotropy, the inverse distance weighting performed slightly better than the other method. These results indicate that an accuracy of the interpolation techniques for the DEM generation should be tested not only in relation to the landform types and data density but also in multi-scales domain.

7.5.3 Curve-Fitting Methods

The minimum curvature is a popular interpolation method in the Earth science. This method generates a smoothest possible surface as a thin and linearly elastic plate, passing through each of the data values with a minimum amount of bending [85].

In 3D space, the TPS is the solution to define a continuous and smooth surface. According to Szeliski [86], the smoothness data term can be defined as the squared first or second derivatives of the unknown function $f(x, y)$, from which the data point $d(x_i, y_i)$ were sampled. These derivatives have a view of Eqs. 7.37 and 7.38.

$$\varepsilon_1 = \int [f_x^2(x, y) + f_y^2(x, y)] dx dy = \int \|\nabla f(x, y)\|^2 dx dy \quad (7.37)$$

$$\varepsilon_2 = \int [f_{xx}^2(x, y) + 2f_{xy}^2(x, y) + f_{yy}^2(x, y)] dx dy \quad (7.38)$$

In addition to the smoothness term, a regularization requires a data term ε_d (data penalty) that measures the distance between the function $f(x, y)$ and a set of data points $d_i = d(x_i, y_i)$. Equations 7.39 and 7.40 provide a view of function ε_d in the discrete and continuous forms, respectively.

$$\varepsilon_d = \sum_{i=1}^n [f(x_i, y_i) - d_i]^2 \quad (7.39)$$

$$\varepsilon_d = \int [f(x, y) - d(x, y)]^2 dx dy \quad (7.40)$$

The global energy ε , which ought to be minimized, is determined by Eq. 7.41, where ε_d is a data term (measures the fitness of the surface and the control points), ε_s is a smoothness penalty (ε_1 , ε_2 , or other), λ is the regularization parameter.

$$\varepsilon = \varepsilon_d + \lambda \varepsilon_s \quad (7.41)$$

If the regularization parameter $\lambda = 0$, then no regularization is imposed, and the surface will pass exactly through all the given points. If the parameter λ is over regularized, then the surface will become a completely smooth surface like a least-square fitted plane. The intermediate values of parameter λ will produce a compromise between the fitness of the data and the smoothness of the surface.

The lifting scheme to generate the multi-scale DEMs based on the granulometric analysis of curvature regions in the terrain was employed by Hani et al. [87]. The proposed algorithm is based on the granulometric analysis to compute the average size of the curvature regions and the average roughness of a terrain. The close approach was developed by Mongus and Zalik [39]. The TPS control-points were arranged within a multi-resolution hierarchical structure, which enabled it to iteratively advance the TPS toward the ground. The filtering was performed with the

gradually decreasing window size. At each level of the hierarchy, the points were filtered according to their residuals (height distances) from the interpolated surface.

7.5.4 Surfaces' Interpolation in ArcGIS Spatial Analyst

The ArcGIS Spatial Analyst [88] employs several interpolation tools. Thus, the ArcGIS 9 Spatial Analyst offers PointInterp, Natural Neighbors, Trend method, and Topo to Raster command that had been added to the Inverse Distance Weighed (IDW), Spline, and Kriging interpolation methods provided by the ArcGIS 8.3 Spatial Analyst. The Natural Neighbors and Trend methods were available in ArcGIS 8.3 3D Analyst, although the Trend method was accessible programmatically. A choice of an appropriate method depends on the distribution of the sample points and the phenomenon being studied.

The IDW method should be used, when the set of points is dense enough to capture the extension of the local surface variations. The IDW determines the cell values using a linear-weighted combination set of the sample points. The weight assigned is a function of the distance of an input point from the output cell location. The greater the distance, the less influence the cell has on the output value.

The Spline estimates values using a mathematical function that minimizes overall surface curvature. As a result, the surface becomes smooth. It can predict the ridges and valleys in the data. There are two types of Spline, regularized and tension. A regularized Spline incorporates the first derivative (slope), second derivative (rate of change in a slope), and the third derivative (rate of change in the second derivative) into its minimization calculations. A tension Spline uses only the first and second derivatives but it includes more points in the Spline calculation that permits to create the smoother surfaces, spending longer computation time.

The Kriging is a powerful statistical interpolation method. It assumes that the distance or directions between the sample points reflects their spatial correlation. It fits a function to a specified number of points or all points within a specified radius to determine the output value for each location. The Kriging is most appropriate, when a spatially correlated distance or directional bias in the data is known (soil and geological data). The predicted values are derived from the measure of relationship in samples using the weighted average technique. The Kriging uses a search radius that can be fixed or variable. The generated cell values can exceed value range of samples, and the surface does not pass through samples. The ordinary Kriging is the most common method. It assumes that there is no constant mean for the data over an area mean (no trend). The universal Kriging does assume that an overriding trend exists in the data, which can be modelled.

The PointInterp method is similar to the IDW and allows more control over the sampling neighborhood. The influence of a particular sample on the interpolated grid cell value depends on whether the sample point is in the cell's neighborhood and how far from the cell being interpolated it is located. The points outside the neighborhood have no influence. The weighted value of the points inside the

neighborhood is calculated using the inverse distance weighted interpolation or inverse exponential distance interpolation. This method interpolates a raster using the point features but allows for different types of neighborhoods. The neighborhoods can have shapes, such as the circles, rectangles, irregular polygons, annuluses, or wedges.

The Natural neighborhood interpolation can be used for both interpolation and extrapolation and generally works well with the clustered scatter points. Another weighted-average method used in natural neighbor interpolation, called as the basic equation, is identical to the one applied in the IDW interpolation. It can efficiently handle large input point datasets.

The Trend is a statistical method that finds the surface that fits the sample points using a least-square regression fit. It fits one polygonal equation to the entire surface. This leads to the surface that minimizes surface variance in relation to the input values. The surface is constructed so that for every input point, the total of the differences between the actual values and the estimated values (the variance) will be small as possible. It is an inexact interpolator, and the resulting surface rarely passes through the input points. However, this method is similar to natural phenomena that typically vary smoothly.

The Topo to Raster method, interpolating the elevation values for a raster, imposes the constraints that ensure a hydrologically correct digital elevation model. It contains a connected drainage structure and correctly represents ridges and streams from input contour data. The Topo to Raster method uses an iterative finite difference interpolation technique that optimizes the computational efficiency of local interpolation without losing the surface continuity of a global interpolation. It was specifically designed to work with the contour input data.

The results of some interpolation methods applied in the ArcGIS Spatial Analyst are depicted in Fig. 7.4.

7.5.5 Accuracy Estimators

The accuracy of the interpolation process can be evaluated from different aspects [89]. The most straightforward is to predict some single, global accuracy measures that characterize the interpolation accuracy via different validation techniques, among which the cross-validation prevails [90–92].

Another approach is to employ a secondary dataset that is never used in the interpolation process [91, 93]. For each point, the deviation between the measured and predicted values is calculated, and the accuracy is tested according to these values. The Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) indices calculated from the difference between the surveyed and the predicted values for each point were examined to compare the interpolation techniques [65]. The MAE and the RMSE indices are computed by standard Eqs. 7.42 and 7.43, where $\{\hat{z}_i\}$ are the predicted values, $\{z_i\}$ are the observed values, n is a number of elevation values.

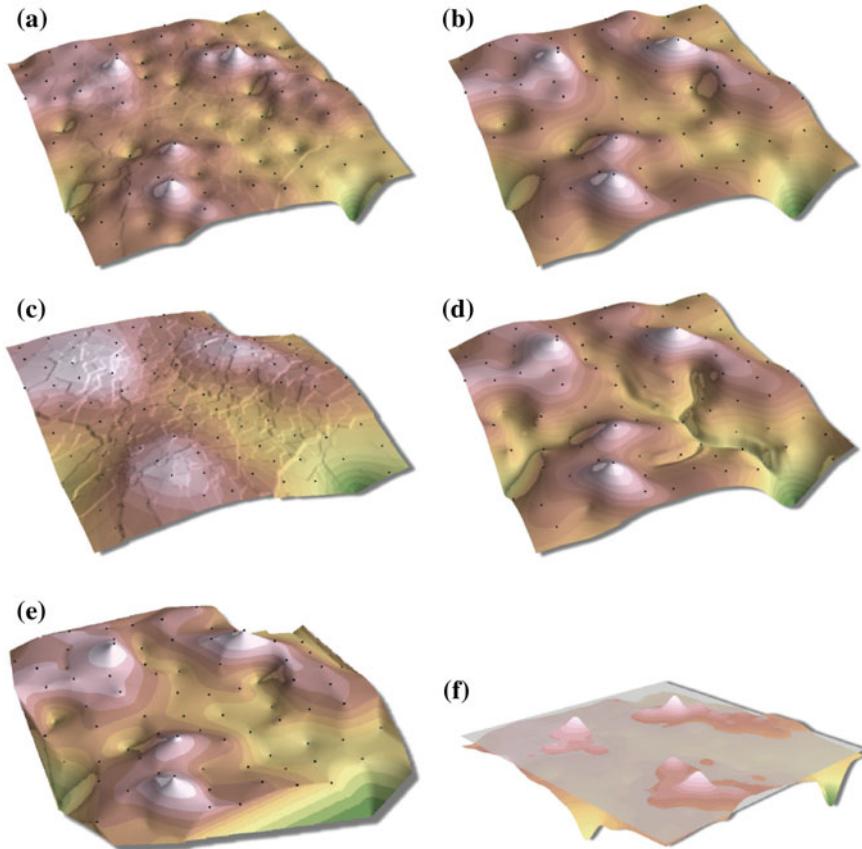


Fig. 7.4 Surfaces created in the ArcGIS spatial analyst, using: **a** the IDW method, **b** spline interpolation, **c** Kriging technique, **d** PointInterp method, **e** natural neighborhood interpolation, **f** trend method

$$MAE = \frac{1}{n} \sum_{i=1}^n |(\hat{z}_i - z_i)| \quad (7.42)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (z_i - \hat{z}_i)^2}{n}} \quad (7.43)$$

Also the approach of the global correlation (Moran's I, Geary's C, or Getis-Ord's General/Global G) and the local spatial correlation (Local Moran's I, Getis-Ord's Gi*) measures examine the extent of error clustering. The global Moran's I statistic is defined by Eq. 7.44, where $\{z_i\}$ and $\{z_j\}$ are values at different particular

locations, \bar{z} is the mean values, $\{w_{ij}\}$ are the weight indexing locations of i relative to j , n is a number of observations (points or polygons).

$$I = \frac{n \sum_{i=1}^n \sum_{j=1}^n w_{ij} (z_i - \bar{z})(z_j - \bar{z})}{\left(\sum_{i=1}^n \sum_{j=1}^n w_{ij} \right) \sum_{i=1}^n (z_i - \bar{z})^2} \quad (7.44)$$

The range of possible values of Moran's I is -1 to 1 ; the positive values indicate a spatial clustering of the similar values, while the negative values indicate a clustering of the dissimilar values. For Moran, the cross-product is based on the deviations from the mean for the two location values.

The Geary's C (Contiguity) Ratio is determined by Eq. 7.45.

$$C = \frac{n \sum_{i=1}^n \sum_{j=1}^n w_{ij} (z_i - z_j)^2}{2 \left(\sum_{i=1}^n \sum_{j=1}^n w_{ij} \right) \sum_{i=1}^n (z_i - \bar{z})^2} \quad (7.45)$$

Geary's C varies on a scale from 0 to 2 . For Geary, the cross-product uses the actual values themselves at each location.

The G statistic distinguishes between the hot spots and the cold spots. It identifies the spatial concentrations: G is relatively large if high values cluster together and G is relatively low if low values cluster together [94]. It is calculated using Eq. 7.46, where d is a neighborhood distance, w_{ij} is a weights matrix (has only 1 or 0 values: 1 if j is within d distance of i and 0 if it beyond that distance). Thus, any point beyond distance d has a value of zero and, therefore, is excluded.

$$G(d) = \frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij}(d) z_i z_j}{\sum_{i=1}^n \sum_{j=1}^n z_i z_j} \quad (7.46)$$

Because of the insufficiency of the global indices to explain the DEM uncertainty well, the spatial distributions of the model residuals using error surfaces and the global and local spatial autocorrelation indices are welcome.

7.6 Experimental Results

The experimental software tool TerrainBuilder, v. 2.01 was designed for generation of the Earth's surface, using the LiDAR point cloud and visualization of the DTM and the above-ground objects [95]. The main functions are the following:

- The reading of file with the LiDAR data.
- A visualization of the LiDAR point cloud in 3D scene.
- A filtering of the LiDAR points on the ground and non-ground clusters using the surface-based methods.

- The DTM generation, using the adaptive Delaunay triangulation.
- A visualization of the DTM in 3D scene.
- A segmentation of the non-ground LiDAR points on the individual trees.
- A visualization of the simplified tree models in 3D scene.
- A saving the intermediate states of a program.

The file structure of the software tool includes the execution file TerrainBuilder.exe, which runs the application and a set of file, dynamic libraries with extensions *.dll. Also a set of the generated scene objects is available for first program execution. The main screen is depicted in Fig. 7.5.

The programming environment Visual Studio 2010 was used for design of experimental software tool. C# is a programming language. Libraries of platform. NET Framework 4.0 and additional libraries, such as OpenTK, OpenGL, Open Toolkit, were applied. The input data is the LiDAR data, representing in the text format *.XYZ. This format has the following structure. Each line describes a single point as a set of four numbers with a floating point. First three numbers are latitude, longitude, and elevation values, respectively. The forth number is a special sign, which does not influence on data processing. All numbers are separated by the spaces.

Also the input and output data can store in a format *.obj, in which data is located in lines, and the first symbol in each line means a type of data in this line:

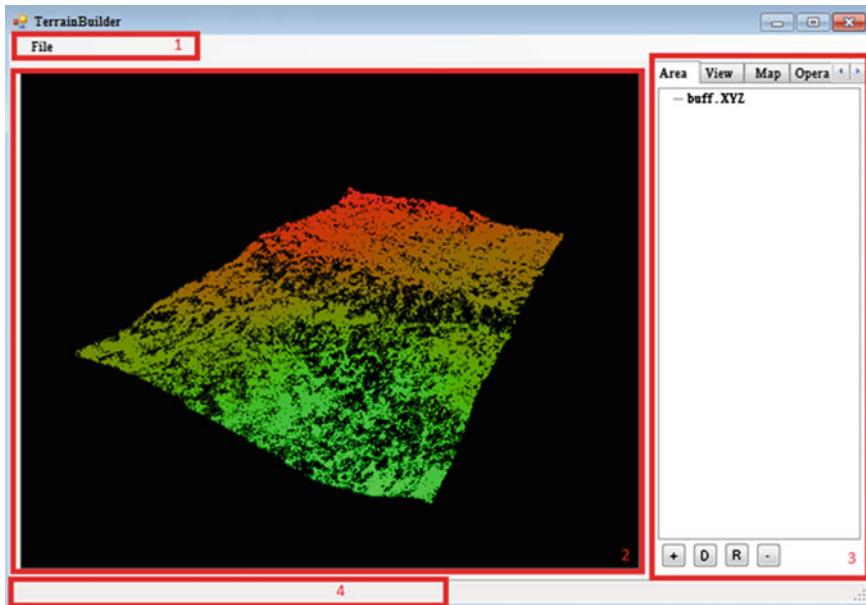


Fig. 7.5 Main screen of software tool TerrainBuilder with segmented working areas, such as 1 is the main menu, 2 is an area of 3D scene visualization, 3 is an area of operations, 4 is an area for visualization of operations' execution

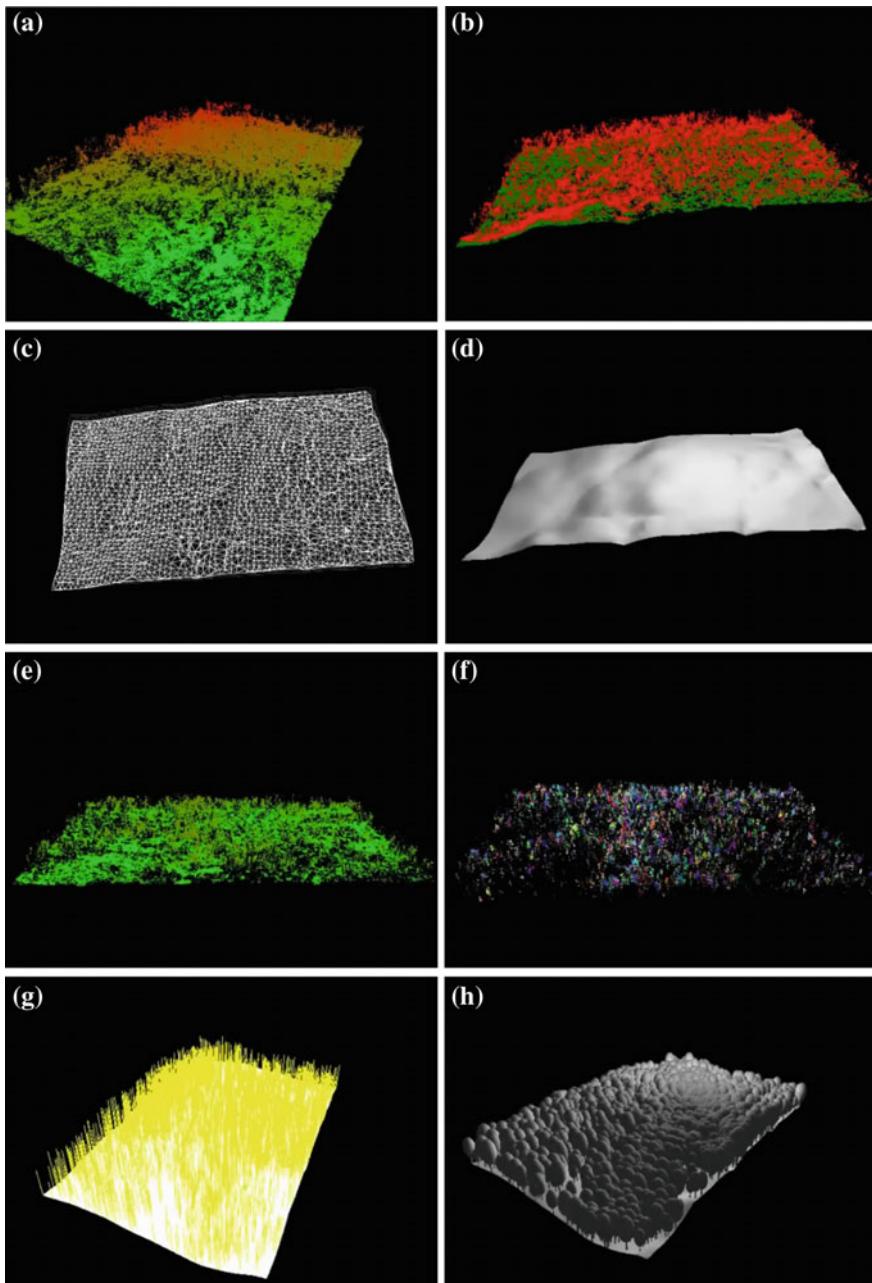


Fig. 7.6 A sequence of processing steps: **a** initial point cloud with elevation marking, **b** elevation excesses, **c** the Delaunay triangulation, **d** interpolation, **e** vegetation in green color, **f** vegetation in marked color, **g** trunks as lines, **h** simplified models of trees

- Symbol v means an apex. The following values are determined a location of current apex in 3D space.
- Symbol vn means a normal. The following values are determined a direction of current normal in 3D space.
- Symbol vt means the texture coordinate. Values are determined the shifts of texture in current point.
- Symbol f means a facet. A facet is presented by three triples mapping three points of a facet. Values in each triple are separated by symbol \|. The first number in a triple is an index of the current apex in a set of the apexes, the second number is an index of the texture coordinate, and the third index is an index of the normal.

A sequence of processing steps is depicted in Fig. 7.6.

7.7 Conclusions

In this chapter, many methods for the DTM construction from LiDAR data were discussed and compared. This task is very difficult due to the statistical nature of the LiDAR point cloud with a restricted set of the initial and computed features. The main challenges of a modelling deal with the slope and steep natural terrain, mountains, and breaklines detection, for example, in the urban areas. Many conventional filtering and interpolation methods fail to detect the grounds points around these discontinuities. The hybrid approach, combining different methods, performs objectively better results than single methods. This is a good subject for investigation in future as well as the implementation of these complex algorithms using the recent achievements in software techniques.

References

1. Miller C, LaFlamme RA (1958) The digital terrain model—theory and applications. Photogram Eng 24(3):433–442
2. Petrie G, Kennie TJM (1987) Terrain modelling in surveying and civil engineering. Comput-Aided Des 19(4):171–187
3. Carter J, Schmid K, Waters K, Betzhold L, Hadley B, Mataosky R, Halleran J (2012) National oceanic and atmospheric administration (NOAA) coastal services center. “Lidar 101: an introduction to lidar technology, data, and applications.” Revised. NOAA Coastal Services Center, Charleston, SC
4. ArcGIS 10.3. Environmental Systems Research Institute, Inc. <http://www.esri.com>. Accessed 22 Oct 2015
5. Surfer 13. Powerful contouring, gridding, and 3D surface mapping software for scientists and engineers. <http://www.goldensoftware.com/products/surfer>. Accessed 11 Nov 2015
6. Zhou Q, Yumin Chen Y (2011) Generalization of DEM for terrain analysis using a compound method. ISPRS J Photogramm Remote Sens 66(1):38–45

7. Baruch A, Filin S (2011) Detection of gullies in roughly textured terrain using airborne laser scanning data. *ISPRS J Photogramm Remote Sens* 66(5):564–578
8. Chen Z, Devereux B, Gao B, Amable G (2012) Upward-fusion urban DTM generating method using airborne LiDAR data. *ISPRS J Photogramm Remote Sens* 72:121–130
9. Kraus K, Pfeifer N (2001) Advanced DTM generation from LiDAR data. *Int Archives Photogramm Remote Sens XXXIV(Part3/W4)*:23–30
10. Wack R, Wimmer A (2002) Digital terrain models from airborne laser scanner data—a grid based approach. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXIV(Part3B)*:293–296
11. Sithole G, Vosselman G (2004) Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds. *ISPRS J Photogramm Remote Sens* 59 (1–2):85–101
12. Chehata N, David N, Bretar F (2008) LiDAR data classification using hierarchical k-means clustering. *Int Arch Photogramm Remote Sens Spat Inf Sci. Vol. XXXVII. Part B3b* 325–330
13. Zhang J, Lin X (2013) Filtering airborne LiDAR data by embedding smoothness-constrained segmentation in progressive TIN densification. *ISPRS J Photogramm Remote Sens* 81:44–59
14. Hu H, Ding Y, Zhu Q, Wu B, Lin H, Du Z, Zhang Y, Zhang Y (2014) An adaptive surface filter for airborne laser scanning point clouds by means of regularization and bending energy. *ISPRS J Photogramm Remote Sens* 92:98–111
15. Sithole G (2001) Filtering of laser altimetry data using a slope adaptive filter. *Int Achieves Photogramm Remote Sens XXXIV(Part 3/W4)*:203–210
16. Vosselman G (2000) Slope based filtering of laser altimetry data. *Int Achieves Photogramm Remote Sens XXXIII(Part B3/2)*:935–942
17. Chen Q, Gong P, Baldocchi D, Xie G (2007) Filtering airborne laser scanning data with morphological methods. *Photogram Eng Remote Sens* 73(2):175–185
18. Arefi H, Hahn M (2005) A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXVI(Part3/W19)* 120–125
19. Filin S (2002) Surface clustering from airborne laser scanning data. *Int Achieves Photogramm Remote Sens XXXIV (Part 3A)*:119–124
20. Zhang K, Chen SC, Whitman D, Shyu ML, Yan J, Zhang C (2003) A progressive morphological filter for removing non-ground measurements from airborne LIDAR data. *IEEE Trans Geosci Remote Sens* 41(4):872–882
21. Chen D, Zhang LQ, Wang Z, Deng H (2013) A mathematical morphology-based multi-level filter of LiDAR data for generating DTMs. *Sci China Inf Sci* 56 102314:1–102314:14
22. Vega C, Durrieu S, Morel J, Allouis T (2012) A sequential iterative dual-filter for LiDAR terrain modeling optimized for complex forested environments. *Comput Geosci* 44:31–41
23. Chen C, Li Y, Li W, Dai H (2013) A multiresolution hierarchical classification algorithm for filtering airborne LiDAR Data. *ISPRS J Photogramm Remote Sens* 82:1–9
24. Mongus D, Lukac N, Borut Zalik B (2014) Ground and building extraction from LiDAR data based on differential morphological profiles and locally fitted surfaces. *J Photogramm Remote Sens* 93:145–156
25. Pirotti F, Alberto Guarneri A, Vettore A (2013) Ground filtering and vegetation mapping using multi-return terrestrial laser scanning. *ISPRS J Photogramm Remote Sens* 76:56–63
26. Meng X, Currit N, Zhao K (2010) Ground filtering algorithms for airborne LiDAR data: a review of critical issues. *Remote Sens* 2(3):833–860
27. Roggero M (2001) Airborne laser scanning: Clustering in raw data. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXIV(Part3/W4)*:227–232
28. Crombaghs M, Elberink SO, Brügelmann R, de Min E (2002) Assessing height precision of laser altimetry DEMs. *Int Arch Photogramm Remote Sens XXXIV(Part3A)*:85–90
29. Kraus K, Pfeifer N (1998) Determination of terrain models in wooded areas with aerial laser scanner data. *ISPRS J Photogramm Remote Sens* 53:193–203
30. Brovelli MA, Cannata M (2004) Digital terrain model reconstruction in urban areas from airborne laser scanning data: the method and the example of the town of Pavia (Northern Italy). *Comput Geosci* 30(4):325–331

31. Vosselman G (2012) Automated planimetric quality control in high accuracy airborne laser scanning surveys. *ISPRS J Photogramm Remote Sens* 74:90–100
32. Bruegelmann R (2000) Automatic breakline detection from airborne laser range data. *Int Arch Photogramm Remote Sens XXXIII(B3)*:109–115
33. Donato G, Belongie S (2002) Approximate thin plate spline mappings. In: 7th European conference on computer vision ECCV 2002, 3:21–31
34. Zandifar A, Lim S, Duraiswami R, Gumerov N, Davis L (2004) Multi-level fast multipole method for thin plate spline evaluation. *Int IEEE Conf Image Process ICIP* 3:1683–1686
35. Ming Y, Chen CJ (2013) A robust filtering algorithm of LiDAR data for DTM extraction. *Adv Mater Res* 765–767:639–642
36. Axelsson PE (2000) DEM generation from laser scanner data using adaptive TIN models. *Int Arch Photogramm Remote Sens Spat Inf Sci XXXII(PartB4/1)*:110–117
37. Zhang JX (2010) Multi-source remote sensing data fusion: status and trends. *Int J Image Data Fusion* 1(1):5–24
38. TerraSolid software. Software for processing LiDAR point clouds and images. <http://www.terrasolid.com/home.php>. Accessed 25 Oct 2015
39. Mongus D, Zalik B (2012) Parameter-free ground filtering of LiDAR data for automatic dtm generation. *ISPRS J Photogramm Remote Sens* 67:1–12
40. Hay GJ, Dube P, Bouchard A, Marceau DJ (2002) A scale-space primer for exploring and quantifying complex landscapes. *Ecol Model* 153(1):27–49
41. Haugerud RA, Harding DJ (2001) Some algorithms for virtual deforestation (VDF) of LiDAR topographic survey data. *Int Arch Photogramm Remote Sens XXXIV(Part3/W4)*:211–217
42. Evans JS, Hudak AT (2007) Multiscale curvature algorithm for classifying discrete return LiDAR in forested environments. *IEEE Trans Geosci Remote Sens* 45(4):1029–1038
43. Workstation ArcInfo. <http://www.esri.com/software/arcgis/arcinfo>. Accessed 06 Nov 2015
44. Kilian J, Haala N, Englisch M (1996) Capture and evaluation of airborne laser scanner data. *Int Arch Photogramm Remote Sens XXXI(PartB3)*:383–388
45. Pfeifer N, Köstli A, Kraus K (1998) Interpolation and filtering of laser scanner data—implementation and first results. *Int Arch Photogramm Remote Sens XXXII(Part3/1)*:153–159
46. Zhang K, Chen SC, Whitman D, Shyu M, Yan J, Zhang C (2003) A progressive morphological filter for removing nonground measurements from airborne LiDAR data. *IEEE Trans Geosci Remote Sens* 41(4):872–882
47. Pingel TJ, Clarke KC, McBride WA (2013) An improved simple morphological filter for the terrain classification of airborne LiDAR data. *ISPRS J Photogramm Remote Sens* 77:21–30
48. Jacobsen K, Lohmann P (2003) Segmented filtering of laser scanner DSMs. *Int Arch Photogramm Remote Sens* 34(3/W13):14.1–14.6
49. Tóvári D, Pfeifer N (2005) Segmentation based robust interpolation—a new approach to laser filtering. *Int Arch Photogramm Remote Sens* 36(3/W19):79–84
50. Charaniya AP, Manduchi R, Lodha SK (2004) Supervised parametric classification of aerial LiDAR data. In: IEEE conference on computer vision and pattern recognition workshop CVPRW 2004, pp 25–32
51. Sithole G, Vosselman G (2005) Filtering of airborne laser scanner data based on segmented point clouds. *Int Arch Photogramm Remote Sens XXXVI(Part3/W19)*:66–71
52. Chehata N, Bretar F (2008) Terrain modeling from LiDAR data: hierarchical k-means filtering and Markovian regularization. In: 15th IEEE international conference on image Processing ICIP 2008, pp 1900–1903
53. Guo B, Huang X, Zhang F, Sohn G (2015) Classification of airborne laser scanning data using JointBoost. *ISPRS J Photogramm Remote Sens* 100:71–83
54. Weinmann M, Jutzi B, Hinz S, Mallet C (2015) Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J Photogramm Remote Sens* 105:286–304
55. Kobler A, Pfeifer N, Ogrinc P, Todorovski L, Ostir K, Dzeroski S (2007) Repetitive interpolation: a robust algorithm for DTM generation from aerial laser scanner data in forested terrain. *Remote Sens Environ* 108(1):9–23

56. Maguya AS, Juntila V, Kauranne T (2013) Adaptive algorithm for large scale DTM interpolation from LiDAR data for forestry applications in steep forested terrain. *ISPRS J Photogramm Remote Sens* 85:74–83
57. Hugentobler M, Schneider B (2005) Breaklines in coons surfaces over triangles for the use in terrain modelling. *Comput Geosci* 31(1):45–54
58. Schneider B (2001) Phenomenon-based specification of the digital representation of terrain surfaces. *Trans GIS* 5(1):39–52
59. Hugentobler M, Purves RS, Schneider B (2005) Evaluating methods for interpolating continuous surfaces from irregular data: a case study. In: Fisher P (ed) *Developments in spatial data handling*. Springer, Berlin
60. Coons S (1967) Surfaces for computer aided design of space forms. Technical Report, MIT, Project MAC-TR 41, Cambridge (MA)
61. Zhang K, Whitman D (2005) Comparison of three algorithms for filtering airborne LiDAR data. *Photogram Eng Remote Sens* 71(3):313–324
62. Puget Sound Lidar Consortium. <http://www.pugetsoundlidar.org>. Accessed 06 Nov 2015
63. Bater CW, Coops NC (2009) Evaluating error associated with LiDAR-derived DEM interpolation. *Comput Geosci* 35(2):289–300
64. Fisher PF, Tate NJ (2006) Causes and consequences of error in digital elevation models. *Prog Phys Geogr* 30(4):467–489
65. Johnston K, Hoef JMV, Krivoruchko K, Lucas N (2001) Using ArcGIS geostatistical analysis. *GIS User Manual*. ESRI, New York
66. Delaunay B (1934) Sur la sphère vide. *Izvestia Akademii Nauk SSSR. Otdelenie Matematicheskikh i Estestvennykh Nauk* 793–800
67. Aurenhammer F (1991) Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput Surv* 23(3):345–405
68. Agarwal PK, Arge L, Har-Peled S, Yi K (2005) I/O-efficient construction of constrained Delaunay triangulations. In: Brodal GS, Leonardi S (eds) *Algorithms—ESA 2005*. Springer-Verlag, Berlin Heidelberg
69. de Kok T, van Kreveld M, Löffler M (2007) Generating realistic terrains with higher-order Delaunay triangulations. *Comput Geom* 36(1):52–65
70. Isenburg M, Liu Y, Shewchuk J, Snoeyink J (2006) Streaming computation of delaunay triangulations. *ACM Trans Graph* 25(3):1049–1056
71. Wu H, Guan X, Gong J (2011) ParaStream: a parallel streaming delaunay triangulation algorithm for LiDAR points on multicore architectures. *Comput Geosci* 37(9):1355–1363
72. Oh J, Lee C (2015) Automated bias-compensation of rational polynomial coefficients of high resolution satellite imagery based on topographic maps. *ISPRS J Photogramm Remote Sens* 100:14–22
73. Skyyt V, Barrowclough O, Dokken T (2015) Locally refined spline surfaces for representation of terrain data. *Comput Graph* 49:58–68
74. Gosciewski D (2014) Reduction of deformations of the digital terrain model by merging interpolation algorithms. *Comput Geosci* 64:61–71
75. Lloyd CD, Atkinson PM (2006) Deriving ground surface digital elevation models from LiDAR data with geostatistics. *Int J Geogr Inf Sci* 20:535–563
76. Stein ML (1999) Interpolation of spatial data. Some theory for Kriging. Springer, New York
77. Sullivan J (1984) Conditional recovery estimation through probability Kriging—theory and practice. In: Verly G, David M, Journel A, Marechal A (eds) *Geostatistics for natural resources characterization, part 1*. Reidel, Dordrecht
78. Cressie N (1985) Fitting variogram models by weighted least squares. *J Int Assoc Math Geol* 17:653–702
79. Cressie N (1986) Kriging nonstationary data. *J Am Stat Assoc* 81:625–634
80. Cressie N (1988) Spatial prediction and ordinary Kriging. *Math Geol* 20:405–421
81. Cressie N (1990) The origins of Kriging. *Math Geol* 22:239–252
82. Cressie N (1993) *Statistics for spatial data*, Revised edn. Wiley, New York

83. Rivoirard J (1994) Introduction to disjunctive Kriging and non-linear geostatistics. Oxford University Press, Oxford
84. Chaplot V, Darboux F, Bourennane H, Leguédois S, Silvera N, Phachomphon K (2006) Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density. *Geomorphology* 77(1–2):126–141
85. Yang CS, Kao SP, Lee FB, Hung PS (2004) Twelve different interpolation methods: a case study of Surfer 8.0. XXth ISPRS Congress 35(B2):772–777
86. Szeliski R (2011) Computer vision: algorithms and applications, 1st edn. Springer-Verlag, London
87. Hani AFM, Sathyamoorthy D, Asirvadam VS (2011) A method for computation of surface roughness of digital elevation model terrains via multiscale analysis. *Comput Geosci* 37 (2):177–192
88. ArcGIS Spatial Analyst (2016) <http://www.esri.com/software/arcgis/extensions/spatialanalyst>. Accessed 25 July 2016
89. Erdogan S (2010) Modelling the spatial distribution of DEM error with geographically weighted regression: an experimental study. *Comput Geosci* 36(1):34–43
90. Smith SL, Holland DA, Longley PA (2005) Quantifying interpolation errors in urban airborne laser scanning models. *Geogr Anal* 37(2):200–224
91. Robinson TP, Metternicht GI (2005) Comparing the performance of techniques to improve the quality of yield maps. *Agric Syst* 85(1):19–41
92. Wise S (2011) Cross-validation as a means of investigating DEM interpolation error. *Comput Geosci* 37(8):978–991
93. Desmet PJJ (1997) Effects of interpolation errors on the analysis of DEMs. *Earth Surf Process Land* 22:563–580
94. Getis A, Ord JK (1992) The analysis of spatial association by use of distance statistics. *Geogr Anal* 24(3):189–206
95. Favorskaya M, Zotin AG, Danilin I, Smolentcheva S (2010) Realistic 3D-modeling of forest growth with natural effect. In: Phillips-Wren G, Jain LC, Nakamatsu K, Howlett RJ (eds) Advances in intelligent decision technologies. Springer-Verlag, Berlin Heidelberg

Chapter 8

Texturing of Landscape Scenes

Abstract The efficient texturing and rendering of vegetation is a crucial problem for enhancing of the landscape scenes. The realistic visualization is a challenging problem due to a huge amount of information even for the individual tree. In recent years, the geometry-based and image-based approximations as the main approaches have developed and demonstrated their possibilities in many applications. The explicit and induced surface parameterizations lay in the basis of a texture mapping. A model of multi-resolution level of details is the well-known method that is an alternative to a polygonal modelling of the natural objects. The basic idea is that the highly detailed models do not always need to be presented in full details, especially in far-distance levels. The reasonable approaches for modelling of the digital Earth' surface and individual trees including other vegetation are analyzed in details in this chapter.

Keywords Landscape scene · Texture mapping · Geometry-based approximation · Image-based approximation · Level of details · Procedural multi-resolution

8.1 Introduction

A modelling of the landscape scenes includes three complex issues, dealing with the landscape modelling, short-term rendering (in the hours), and long-term modelling (in the years). The landscape design is enough close for their purposes. These issues are discussed in the current chapter. The short-term rendering is represented in the following Chap. 9. The goal of the long-term forest modelling is to estimate the biological, geodesic, climatic, and man-made factors. This issue is on the border of forestry, computer graphics, and computer vision.

Any landscape scene involves many natural and, possibly, the man-made objects. The main attention will be paid to the natural objects like trees, plants, grass, among others. Herein, the computer vision methods are overlapping with the computer graphics requirements and acquisitions. The 3D scene rendering requires different representation of the natural objects due to the resolution properties.

The conventional way to design such scenes is a multi-resolution modelling. The accurate models of the tree or plant are worthless for far part of a scene. Also, a real-time modelling requires a compact description in computer memory, usually the Random Access Memory (RAM) for a single object because a number of these objects can achieve several hundreds and even thousands. The cost of compact storage is implemented by a complex uploading of the Levels Of Details (LODs) according to the end-user requirements. The LOD technique is very effective approach to model not only trees but also the Digital Earth Surface (DES) model.

As it was shown in Chaps. 5 and 6, the procedural tree models based on the L-systems with considering the objective LiDAR data is very reasonable approach for the forest inventory. However, in a visualization stage the most graphical applications convert such models to a polygon-based representation for hardware-accelerated rendering [1]. Hundreds of thousands of polygons, representing many trees in a landscape scene, is not the appropriate decision for current hardware equipment. A multi-resolution modelling reduces a number of the required polygons but a polygon-based method fails to capture trees and plants. In this case, the image-based tree models called imposters can replace the polygon-based models at close views [2]. Such technique is called a procedural multi-resolution [3].

Notice that the landscape scenes are designed according to the purposes of the following application. Thus, the software Object Raku Technology's Sextant [4] is focused on the fast generation of urban environments but not on the visualization inside large forest scenes. This software tool offers three levels of forest abstraction, such as the forest silhouette, trees with canopy, and groups of tree models. The software Generating Enhanced Natural Environments and Terrain for Interactive Combat Simulations (GENETICS) [5] applied two tree levels of details: the full geometry and view-aligned images called as billboards. Some software tools present a visibility-based walkthrough framework. This is the most complex type of design and rendering due to the objects are rendered in a front-to-back order with good visibility in addition to the conventional projected object sizes and the LOD technique. The object visibility ought to be computed for each frame at run-time, thereby allowing a dynamic scene modification, for example, the swaying tree branches due to a wind or the burning trees due to the explosions and fires.

It can be stated that two different approaches are known. The geometry-based approximations are represented by the polygonal meshes and can have the constant (non-adaptive approximations) or varying LODs (adaptive approximations). The image-based approximations employ impostors [6], billboards [7], billboard clouds [8], multi-layer Z-buffers [9], sprites with depth [10], the Layered Depth Image (LDI) [11], hierarchical bi-directional textures [12], and volumetric textures [13].

Vegetation as the object of modelling has some particular qualities, dealing with the foliage and trunk approximation. Thus, due to the density of leaves in a tree, the occlusion exists widely among its crown, and it is not reasonable to render the leaves inside of the crown accurately. This proposition decreases greatly the number of the rendered primitives, improving a rendering efficiency. At the same time, the trunk and branch structures are preferable to simulate using the

geometry-based approximation. It is interesting that some authors classify the geometry-based approach into the point-based and the polygon-based ones [14].

An impostor is a simple primitive with the most realistic visual features of the original object that is used for the creation of different object views within a small range. Usually it is represented by a few textured quadrilaterals or by a few simple textured polygonal meshes in the special cases [15]. Herewith, a depth map is associated with each quadrilateral or mesh. If it is available, the content of the depth map is used to generate a warped texture. Billboards also consist of the textured single-layered or multi-layered and world-aligned or screen-aligned polygons. The billboard texture is used for creation of the arbitrary views of the object, while the impostor texture is created for a small range of the object views. However, the ideas of both approaches penetrate each other forming the hybrid-based visualization techniques.

The structure of chapter is the following. Section 8.2 provides a short description of previous works in visualization issues. Fundamentals of texture mapping are discussed in Sect. 8.3. A multi-resolution texturing for the digital Earth's surface model and vegetation models with the following rendering details are considered in Sects. 8.4 and 8.5, respectively. Section 8.6 presents some experimental results. Conclusions are given in Sect. 8.7.

8.2 Related Work

Methods for the landscape scene visualization have not wide diversity due to the main requirement for providing of the minimum computer resources. Consider the background of visualization of the DES model and tree models in Sects. 8.2.1 and 8.2.2, respectively.

8.2.1 *Visualization Techniques for Digital Earth Surface Model*

The texturing of complex scenes has passed several stages of its development. In 1990s, the first methods were based the image depth or image correspondences to a transform one viewpoint to another viewpoint using information about a camera position but without any computational optimization [16]. The main challenge was to use the geometry theory in order to generate the novel views of a scene fast and realistically-based on the real views. The View Dependent Texture-Mapping (VDTM) technique rendered each pixel of a novel view as a blend of its corresponding pixels in the original views. However, this approach did not guarantee the smooth blending of images and good scaling in the viewpoint changes. Debevec and Borshukov [17] developed the VDTM by the preprocessing, polygon view

maps, and projective texture mapping of scene surfaces. Their rendering algorithm involved three stages:

- Draw all polygons that are seen in none of the original views using the vertex colors determined during a hole filling.
- Draw all polygons that are seen in just one view.
- For each polygon that is seen in more than one view, calculate its viewing direction for the desired novel view. Render the polygon using Draw all polygons that are seen in none of the original views using the vertex colors determined by the alpha-blending of the textures applying the projective texture mapping.

The projective texture mapping was introduced Segal et al. [18], and now it is a part of the OpenGL graphics standard. Afterwards, the VDTM modifications were proposed. For example, Xu et al. [19] generated the seamless texture maps for 3D models using the real-world photos under the uncontrolled environment. Their method addressed the texture discontinuity problem and rendering color/lighting differences among images due to the real-world uncontrolled environments. A general texture mapping framework included a geometric part, where the Markov Random Field optimization was implemented to realize a global registration of the texture patches, and a radiometric part, where the local radiometric correction among pieces of texture patches minimized the color difference, seams and blurs. A flowchart of the texture mapping is depicted in Fig. 8.1.

Some image-based rendering approaches use the dynamically generated impostors [20] or layered depth images [11] for the objects or object groups. Before rendering the next frame, a decision is made whether a cached impostor of the object can be reused, a new impostor has to be generated for the object, or the original geometry may use. In this case, the number of polygons of the dynamically generated impostors is reduced dramatically. The main drawback remains the high computational cost for the impostors that are generated based on the original geometry.

The video-based rendering is an extension of the image-based rendering to a sequence of frames. The Free-Viewpoint Video Renderers (FVVRs) proposed by Zitnick et al. [21] allow the end-user to view the captured video footage from any position and direction in the dynamic scenes. The FVVR technique was adapted by Imber et al. [22] to the mobile platforms as a layered FVVR. This approach presents

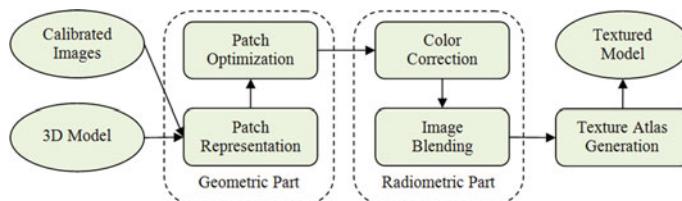


Fig. 8.1 Flowchart of the texture mapping with seamless texture maps

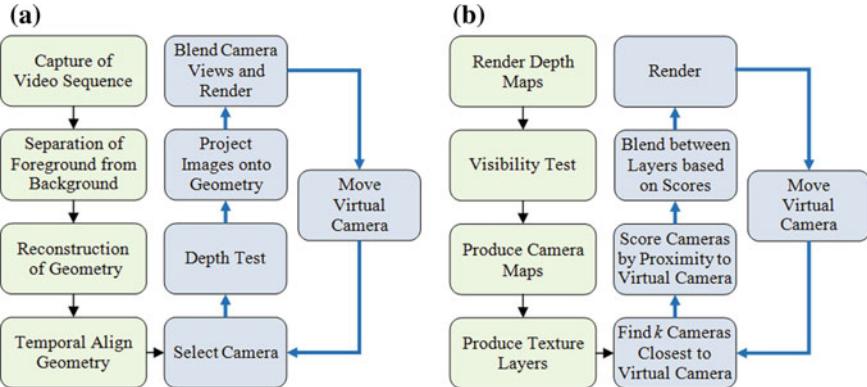


Fig. 8.2 Flowcharts of the FVVR: **a** the original open-source FVVR [23], **b** the FVVR for rendering on mobile devices, assuming availability of geometry [22]

the relighting content and does not require prior knowledge of the scene illumination or accurate surface geometry. A surface is separated into the detailed set of materials with the surface colors and specular behavior and provides the scene shading. Flowcharts of different types of the FVVR are depicted in Fig. 8.2 (blue arrows indicate the render loop).

The LOD-based rendering approaches are applied either for the whole scene or for each object in a scene. One of the crucial issues is the proper choice of the approximation that, on the one hand, should be less detailed and, on the other hand, should look very similar to the original object from a current viewpoint. The first LOD techniques were based on the discrete multi-resolution models. Such model contains a set of discrete object approximations that were generated by a repeated polygonal simplification. At run-time, only one object approximation can be selected for rendering. Some authors proposed a set of pre-generated impostors as the approximations of the objects or object groups [15]. In this case, an appropriate impostor was selected for rendering each object or object group depending on the view-independent criteria. The discrete LOD techniques suffer from visual artifacts, when a currently used object approximation is replaced by another one. It is impossible to speak about the optimal model because only a small number of approximations can be available for each object.

The current LOD-based rendering approaches use the continuous multi-resolution models as the abstract data structures, representing an individual object or the whole scene. Two main categories of continuous multi-resolution models are known:

- The incremental continuous multi-resolution model [24] contains a reduced basic model and a restricted sequence of the refinement operations. The major disadvantage is a non-adaptive approximation, when the refinement operations in the refinement stream can only be executed in that order they were entered in the refinement stream.

- The hierarchical continuous multi-resolution model [25] is based on a dependency graph of the refinement or simplification operations. The extracted adaptive approximations are always optimal for the current viewing configuration. However, these multi-resolution models are very time-consuming.

Notice that the hierarchical image caches are the sophisticated application of image-based and geometry-based rendering techniques. The basic idea is to partition a scene into a hierarchy of boxes and create an impostor for each box. Before a frame is rendered, the impostors that become invalid are updated and propagated towards the root of the hierarchy. These systems use a hierarchy of impostors to reduce a time of the impostor regeneration. However, this approach cannot be used to process the dynamic scenes, and it is difficult to be parallelized [26].

8.2.2 *Visualization Techniques for Tree and Plant Models*

The simplest tree models use two or more orthogonal rectangular impostors, forming a cross or apply a single rectangle called a billboard that always faces the end-user. These techniques are worthless for a trees' rendering from above and at close range. The different representation that is often applied in computer games simulates a tree by the trunk tubes and branch billboards. This approach is the core of commercial software tool SpeedTree [27] with wide possibilities of a landscape rendering with a wind movement but this software tool requires the end-user skill of modelling.

Notice that the LOD modelling technique is a cornerstone of 3D perspective scene, e.g., the cluster-based hierarchical polygon decimation, the LOD foliage with continuous multi-resolution and hardware-assisted rendering, the image-based representations including the layered depth images, the volumetric textures, and the bidirectional textures. The pioneer research of Heckbert and Garland [28] introduced a multi-resolution model, where each object was represented by the LODs.

The finest LOD represents the full-resolution model, while the coarser LODs represent the lower resolution versions of the vertices, edges and polygons suitable for the faster rendering. The advantages of a multi-resolution modelling are mentioned below:

- The size of the model must not increase with the number of the LODs.
- The extraction of the LODs ought to be fast to support an interactive rendering.
- The finest LOD must reproduce the original model at its full resolution.
- The changes between the LODs ought to be smoothed.

Methods for a tree modelling in the LOD representation can be roughly categorized into two approaches, such as the geometry-based and image-based (or texture-based). Oppenheimer [29] was the first, who proposed to use the polygonal cylinders to render the large branches, replacing the small ones with the lines. Weber and Penn [30] simplified this approach, replacing the cylindrical branch

segments with the lines and the leaf polygons with the points or not rendering them at all. At far LODs, the branches and leaves are grouped into “masses” and are rendered as a mass. A multi-resolution simplification scheme by iteratively merging of the leaves was proposed by Remolar et al. [31]. This method provides the overall shape of a tree but the popping artifacts occur. This approach was developed in the recent works [32].

Sometimes the image-based methods help to reduce the geometric complexity of the trees. Thus, Jakulin [33] approximated a tree with the slicings, each consisting of multiple layers of the textured polygons. The primary and secondary slicings were blended together to create a solid-looking rendering of a tree. The most important drawback of this method is the consumption of a texture memory that limits a resolution of the slice textures and a variety of the models. Meyer et al. [12] modelled a tree as a 3-level hierarchy in such manner that each level was composed of the transformed instances of the lower level elements. A set of bidirectional textures was obtained for each hierarchy level and partially for the direct and ambient lighting. The texture maps for a tree reconstruction were proposed by Max [9]. These texture maps contain the normal and depth information in addition to Red, Green, Blue, Alpha (RGBA) color for each pixel. Several texture maps for each hierarchical level of a tree are pre-computed from a number of the viewing directions along a bounding sphere of a tree. A time expensive pre-computation creates the conditions of an individual tree rendering in a few seconds.

8.3 Fundamentals of Texture Mapping

The surface parameterization specifies how points in 2D texture space are projected onto a surface in 3D object space. Two types of parameterization, such as the explicit mapping, using the basic primitives, and the induced mapping, employing n -point correspondence, are known [34]. Examples of the texture mapping on geometrical shapes and terrain surfaces are depicted in Fig. 8.3.

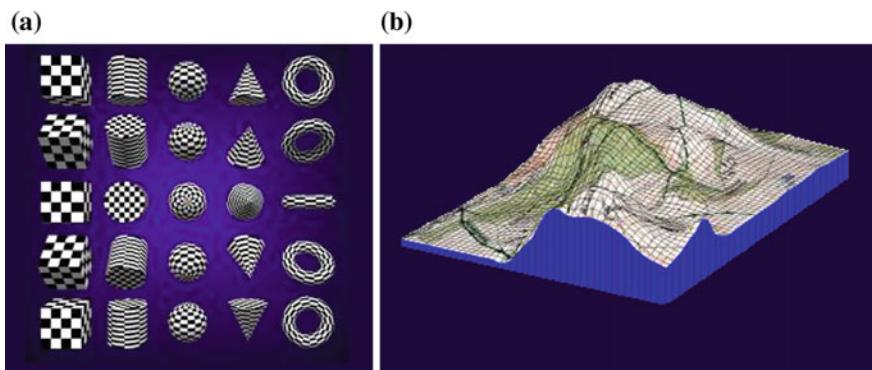


Fig. 8.3 Examples of texture mapping: **a** explicit, **b** induced

The explicit mapping is implemented using a composition of closed-form analytic expressions. This type of mapping is associated with the parametric surfaces, such as patches, cylinders, spheres, cones, and surfaces of revolution. The explicit mapping is typically linear and is expressed in a form of Eq. 8.1, where (u, v) are the coordinates of texture, (θ, φ) are the coordinates of the parametric surface's orthogonal coordinate system, $g(u, v)$ is a surface of revolution.

$$g(u, v) = (f(u, v), f(\theta, \varphi)) \quad (8.1)$$

For example, the sphere's parametric representation is written in a form of Eq. 8.2.

$$\begin{aligned} x &= \sin \theta \sin \varphi & \varphi &= \cos^{-1} y & 0 < \varphi \leq \pi/2 \\ y &= \cos \varphi & \text{or} & \theta &= \sin^{-1}(x/\sin \varphi) & 0 \leq \theta < 2\pi \\ z &= \cos \theta \sin \varphi \end{aligned} \quad (8.2)$$

If it is necessary to map a texture around the sphere in a counter-clockwise direction, then the mapping of a parametric space in a texture space is specified as the linear mapping $u = \theta/2\pi$ and $v = 2\varphi/\pi$. Therefore, the explicit inverse surface parameterization mapping has a view of Eq. 8.3.

$$u = \frac{1}{2\pi} \sin^{-1}(x/\sin(\cos^{-1} y)) \quad v = \frac{2}{\pi} \cos^{-1} y \quad (8.3)$$

The texture mapping is classified as the non-parametric and parametric (according to any model) texture mapping. However, some unexpected warping is also possible as it is depicted in Fig. 8.4.

The use of the polygonal models to create an explicit parameterization is an acceptable but very expensive approach, when every screen pixels is evaluated and it is difficult to represent the resulting surface parameterization in a matrix notation. The induced mapping employs another way of a representation. The induced mapping is computed by specifying the n -point correspondences between the parametric and texture spaces. Then these points are used to solve a series of the simultaneous equations to define a geometric mapping function, which defines a behavior of a texture between the polygon vertices. The resulting 2D–3D continuous mapping defines a direct relationship between the texture space and object space coordinates. In addition, this type of mapping can be evaluated

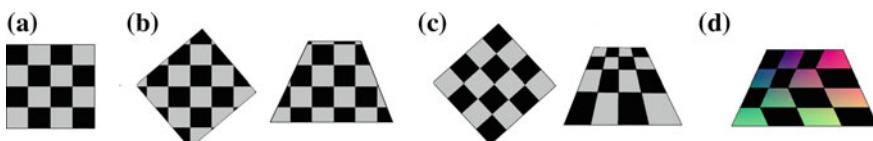


Fig. 8.4 Texturing mapping of a checkerboard text image: **a** original image, **b** non-parametric mapping, **c** parametric mapping, **d** warping

incrementally during the texture rendering that makes it more computationally efficient than the direct explicit mapping.

A surface parameterization mapping is required for each object in a scene because the ray traces rely on the intersecting rays with the objects and these 3D intersection points ought to be mapped into a texture space. Usually for this purpose the affine or bilinear induced mappings are used. However, the scan-line renders avoid the full surface parameterization models traditionally. Instead of this, they usually break up all 3D model data (especially the parametric surfaces) into the basic polygons, assigning the texture coordinates to each vertex and using the explicit or induced parameterizations. Then the texture coordinates across the surface of each polygon are computed during the rendering by the linearly interpolating vertex texture coordinates, rather than using the inverse surface parameterization. However, the resulting surface parameterization mapping may not be the consistent because of a possible mismatch between the texture and screen spaces. In this case, the additional interpolation process is required.

Consider the conventional induced surface parameterizations, such as the affine, bilinear, and projective models. Each model is characterized by a number of the points used in the inference. Thus, the affine mapping requires three vertex points for inference, while the bilinear and projective mappings require four corresponding points [35]. These mappings can be expressed in a matrix notation, thus they have the unique inverses models.

The affine mapping is composed of the linear transformations and translations. It can map the triangles to the triangles and the rectangles to the parallelograms but not be generalized to map the rectangles into the quadrilaterals. A mapping $T(x)$ is linear iff $T(x + y) = T(x) + T(y)$ and $T(\alpha x) = \alpha T(x)$ for any scalar α , and a mapping $T(x)$ is affine iff a constant c and a linear mapping $L(x)$ exist such that $T(x) = L(x) + c$ for all x . The affine mapping is characterized by the preservation of the parallel lines and equal space points along such lines. The 2D texture space to 3D object space mapping is inferred directly from three non-collinear points in both spaces. The mapping of coordinates between a plane in the texture space (u, v) and a plane in the object space (x, y, z) is provided by Eq. 8.4, where $\{p_{ij}\}$ are the parameters of the affine mapping.

$$[x \ y \ z] = [u \ v \ 1] \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \quad (8.4)$$

Equation 8.4 can be represented in a matrix view $\mathbf{x} = \mathbf{u} \ \mathbf{P}_{af}$, where \mathbf{P}_{af} represents the inferred affine mapping. Equation 8.4 can be rewritten for three point correspondences simultaneously.

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \quad (8.5)$$

Equation 8.5 can be represented in a matrix view $\mathbf{X} = \mathbf{U} \mathbf{P}_{af}$, where \mathbf{X} and \mathbf{U} are the three object space and three texture space coordinates, respectively. The matrix \mathbf{P}_{af} may be solved using the Gaussian elimination or directly by multiplying both sides by \mathbf{U}^{-1} .

However, the parameterizations that map four-sided figures to the arbitrary quadrilaterals require the non-affine mappings. The bilinear transformation can transform rectangles into the non-planar quadrilaterals using a linear interpolation along two edges of a quadrilateral. The forward transmission preserves the horizontal or vertical lines in the source space as well as the corresponding points along these lines but the non axis-aligned lines become curved in the destination space. The bilinear interpolation is a common method for interpolating values between four points (the neighboring texel values in our case) and can be used with the planar and non-planar surfaces. The bilinear interpolation can be recommended as a forward direction mapping (from the texture space to the object space) but it is not recommended as a surface parameterization because its inverse is multi-valued and the inverse mapping is not bilinear.

Another class of the unusual non-affine mapping called the conformal maps allows for the derivation of a continuous, bijective map from a polygonal region in the texture space to an arbitrary n -sided simple polygon in another 2D space. The angles are preserved during a mapping but the line segments are typically mapped to the arcs. Such mapping permits to perform the intermediate mapping of the upper half of the complex plane using the Schwarz–Christoffel formulae. These mappings are expensive to compute. The conformal maps are pre-evaluated for a mesh of points in texture space and stored in the look-up tables before a rendering. During a run-time, the approximated solutions to the mapping in a point are evaluated by the interpolating from samples in this table.

The projective mapping causes all lines not parallel in the projection plane to converge to a vanishing point. This convergence induces a foreshortening affect, when more distant lines become closer together and for which equal spaced points are not preserved. The projective mapping has some benefit properties, when the arbitrary oriented line segments remain the lines during a projection, the quadrilaterals can be mapped to the quadrilaterals, and the composition of two projective mapping is still projective. Also, this mapping can be expressed in a matrix notation and the mapping can be easily inverted.

For a projective geometry, the rational linear expressions in the 4-vector homogeneous notation (xw, yw, zw, w) , $w \neq 0$ are represented. The 4-vector (xw, yw, zw, w) represents a line in 4-space and is projected onto 3D space if $w = 1$. Thus, both affine and projective coordinates may share the same notation depending on the value of w ($w = 1$ for the Euclidian coordinates), and the representative point in \Re^3 , (x, y, z) , can be recovered by dividing through the homogeneous point (xw, yw, zw, w) by w . Notice that vectors of the form $(xw, yw, zw, 0)$ including $(0, 0, 0, 0)$ are not defined. The homogeneous coordinates in the texture space are referred to by the notation (uq, vq, q) and those in the object space are (xw, yw, zw, w) .

Four-point correspondences, defining 16 equations in 16 unknowns, are provided by Eq. 8.6.

$$\begin{bmatrix} x_1w_1 & y_1w_1 & z_1w_1 & w_1 \\ x_2w_2 & y_2w_2 & z_2w_2 & w_2 \\ x_3w_3 & y_3w_3 & z_3w_3 & w_3 \\ x_4w_4 & y_4w_4 & z_4w_4 & w_4 \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \\ u_4 & v_4 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \quad (8.6)$$

In a matrix notation, Eq. 8.6 can be expressed as $\mathbf{X} = \mathbf{U} \mathbf{P}_{pr}$. The elements $\{p_{ij}\}$ can be determined from the 3×4 surface parameterization matrix \mathbf{P}_{pr} . The four $\{w_i\}$ values can be computed by multiplying the i th row of the \mathbf{U} matrix with the fourth column of the solved \mathbf{P}_{pr} matrix. The $\{p_{ij}\}$ elements can be solved using the Gaussian elimination, if the $\{w_i\}$ unknowns are eliminated from the equations, by multiplying each of four equations by $w_i = p_{14}u_i + p_{24}v_i + p_{34}$ and equating them to 0. In a pre-processing step, the projective surface parameterization would be derived for each polygon and stored in a matrix form. The run-time intersection process may be simplified because a matrix multiplication with 3D object space coordinates is only required.

8.4 Multi-resolution Texturing for Digital Earth Surface Model

The modelling and rendering of large natural scenes are not the easy tasks. The modelling of terrains ought to be realistic, and the rendering of such scenes requires a low computational cost in order to obtain the real-time application. During texturing, three challenges can appear. The first one occurs in incorrect mapping of texture, when a viewpoint is changed due to a texture represents an arbitrary subset of the model from a single viewpoint. In scenes with a complex geometry, a switching between the geometric shapes and textured shapes (or vice versa) may cause a sudden jump in the image. This is the second problem. This means that a transition between the geometry and texture ought to be smoothed. The third problem is caused by a storing of many texture samples that requires a vast amount of textures in the computer memory. The reduction of this subset can be obtained by the image warping for possible solutions.

In this chapter, the modelling issues are considered, while the rendering is discussed in Chap. 9. In their way, the modelling approaches passed a long way of a development beginning from the tiles and finishing by the modern multi-resolution LOD-techniques. The basic techniques are situated in Sect. 8.4.1, while the extended LOD-versions are described in Sect. 8.4.2.

8.4.1 Basic Techniques

One of the computational models available for texturing a plane is so called the Wang tiles proposed by the mathematician Hao Wang in 1961 [36]. The Wang tiles are the square tiles with the colored edges or colored triangles. The number of tiles is fixed, and these tiles are called prototiles. The edges of any two adjacent tiles ought to be matched, and the tiling supports only translations of the prototiles (rotations and reflections are not allowed). The basic question is whether the given number of prototiles can completely tile a plane. The tiling algorithm assumed that any set of tiles able to tile a plane would be periodic. However, later it was shown that the aperiodic tilings exist and that the Wang tiles could be used to simulate the Turing machine. Initially the Wang tiles were four colored tiles that can be used to tile a plane. The first known set of the aperiodic Wang tiles was discovered by Berger in 1966 [37]. The Berger's original aperiodic set had 20,426 prototiles but later he reduced this number to 104. The recent development demonstrated the existence of prototiles' sets, which admit infinitely many tilings of the plane, none of which are periodic. The samples of the small aperiodic Wang tiles are depicted in Fig. 8.5.

The applications of the Wang tiles include the graphics of a texture generation. The examples of application of several modelling techniques are depicted in Fig. 8.6.

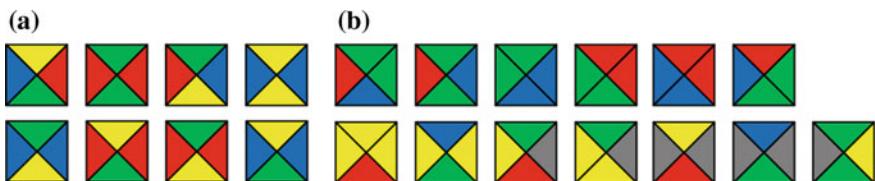


Fig. 8.5 Samples of small aperiodic Wang tiles: **a** the set of 8 prototiles with 4 colors, **b** the set of 13 prototiles with 5 colors

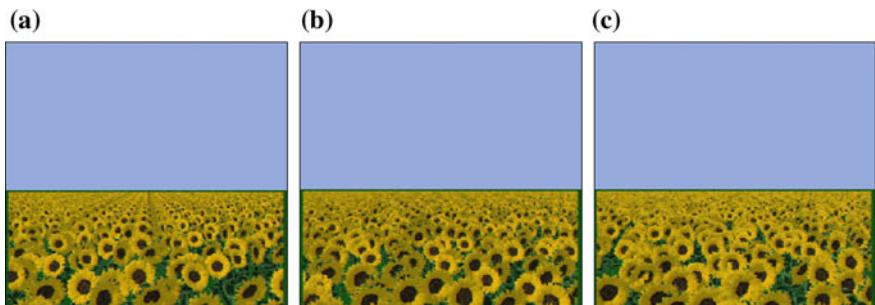


Fig. 8.6 Modelling a scene with tiles [38]: **a** modelling with simple tiles, **b** modelling with the Wang tiles (8 prototiles from Fig. 8.3.a), **c** modelling with the LDI tiles

The special type of the impostors called the Textured Depth Meshes (TDM) is a texture with the detailed scene information that is mapped to a simple polygonal mesh emulated the rough scene structure. The advantages of the TDM technique are mentioned below:

- The parallax movements are correctly represented so that the impostor is valid longer than, for example, billboards.
- The storage cost of the TDMs is low compared to more complex image-based representations.
- The polygonal nature of the TDMs is implemented by graphics hardware well.

The TDMs are very useful to accelerate the rendering of large scenes. However, their creation is a difficult task that restricts their use in practical applications. Some extensions of this approach, such as the incremental TDM [39] or the polygon meshes using the voxel layers [40], were proposed. Consider the original algorithm for the fast TDM implementation based on detection and removal the invisible voxels due to the occlusions in a scene. This algorithm was developed by Jeschke and Wimmer [40]. Often many voxels in a far layer are occluded by the voxels in a close layer. The algorithm for removing of the invisible voxels includes several steps repeated for each pair of the adjacent sampling layers depending on a number of the layers. All steps of 1D projective algorithm for three layers (and texels instead of voxels) are depicted in Fig. 8.7.

Such approach makes the sampling layers optimally placed. The parallax error remains bounded.

The texture is only prospectively correct, when it is viewed from a reference viewpoint. When a viewpoint is changing, the discontinuous of the texture matching appears. As a part of pre-computation, the static LODs are generated for each object. Based on a viewpoint, the portions of the hierarchy outside the view are culled and the suitable LODs for each visible object are rendered. Moreover, some techniques for rendering large environments include the occlusion culling that accelerates the rendering of high depth complexity environments.

8.4.2 *Extended Techniques*

The extended techniques use the LODs as the necessary component in the virtual reality applications. Such frameworks include the interesting composition of the geometry-based and image-based LOD-techniques. Consider the ways to improve and extend the basic LOD approach.

Hilbert and Brunnett [26] performed a scene as a scene graph, and all objects are represented as the textured polygonal meshes of an arbitrary topology. For each LOD object, a continuous hierarchical multi-resolution model is created previously. An adaptive approximation is extracted based on the view-dependent criteria. The vertex tree and the list of active triangles are the main sources to create a series of

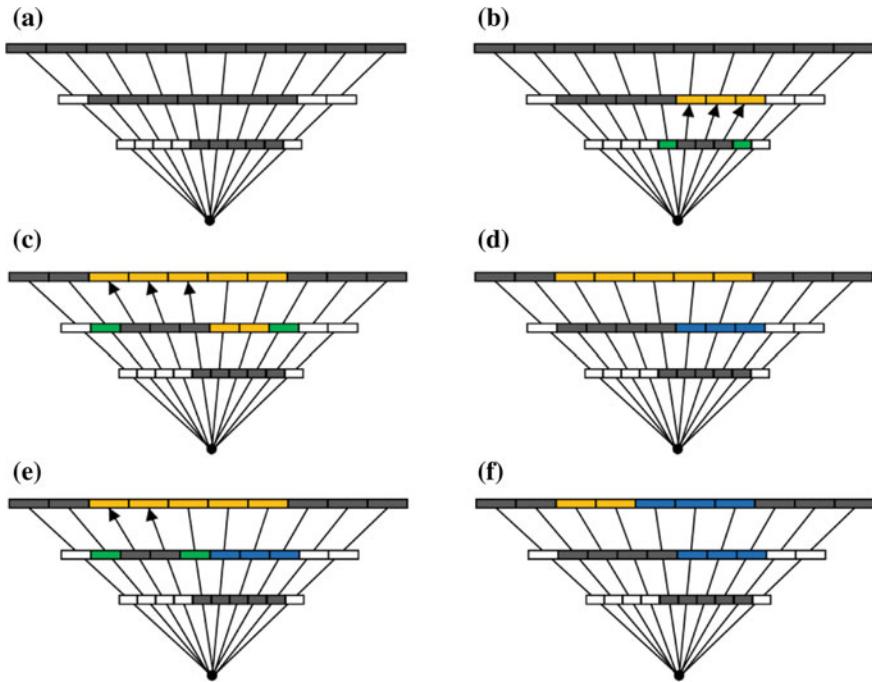
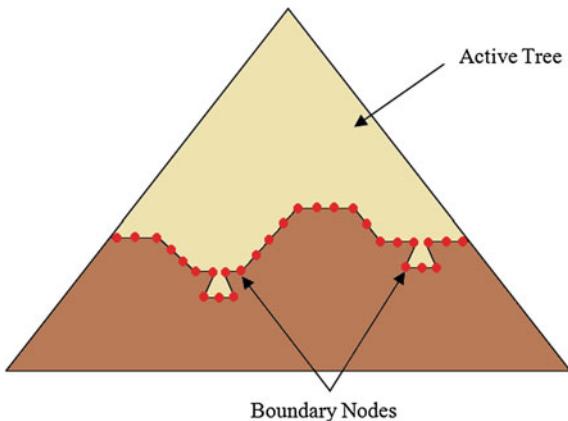


Fig. 8.7 Removing texels in the sampling layers (*gray texel* is an opaque texel, *green texel* is a boundary texel, *yellow texel* is a hidden texel, *blue texel* is excluded from computation invisible texel): **a** initial configuration, **b** detection of the hidden texels in the second layer, **c** detection of the hidden texels in the third layer, **d** the hidden texels from the second layer are marked as the invisible texels and excluded from further consideration, **e** detection of the hidden texels in the third layer, **f** the hidden texels from the third layer are marked as the invisible texels and excluded from further consideration

MIP maps (MIP maps or mipmaps are the sequences of images, each of which is a progressively lower resolution representation of the same image.). The coordinates of the vertices are saved as the coordinates of the object coordinate system, and all computations are done in the object coordinate system. The structure of a vertex tree was proposed by Luebke and Erikson [41]. The nodes of the vertex hierarchy for a virtual mapping store the following data:

- *Id* is a bit vector, which labels the path from the root of the vertex tree to the node. Each 3-bit triple in the vector denotes the correct branch for the vertex octree.
- *Depth* is the depth of a node in the vertex tree. The Depth and Id together uniquely identify the node.
- *Label* shows the node's status as active, boundary, or inactive.
- *Revert* contains the coordinates of the node's representative vertex.
- *Texcoord* includes the texture coordinates of a representative vertex.

Fig. 8.8 Mapping of a vertex tree



- *Center, Radius* are the center and radius of a bounding sphere, containing all vertices in the octree cell.
- *ConeNormal* and *ConeAngle* define a cone that contains the Gaussian image of all normals of the triangles at least partially included in the octree cell.
- *Tris* is a list of the triangles with exactly one vertex in the node. These are the triangles, whose vertices must be adjusted, when the vertex tree node is folded or unfolded.
- *Subtris* is a list of triangles with two or three vertices within the octree cell but no more than one vertex within any child of the octree cell. These triangles will be filtered out, if a node is folded, and re-introduced, if a node is unfolded.
- *Parent, Numchildren, Children* are the parent and children of this node in the vertex tree.

Mapping of vertex tree is depicted in Fig. 8.8.

At run-time, the texture parameters for the vertices of active triangles are computed that permits to avoid the occurrence of discontinuities in the texture of whole scene. Such hierarchical structures are created for each object that allows to create the dynamic scenes. The caching mechanism both for the geometry-based and image-based approximations was employed.

Before rendering a frame, Hilbert and Brunnett checked the availability of valid approximation in cache for each LOD object using a bounding-box-based view-dependent. If a valid approximation of the LOD object is detected, then the cache data can be used for rendering. For all other LOD objects, a new approximation can be created by extraction data from the object's vertex hierarchy. At this step, the geometry-based approximation can be replaced by an impostor using three mentioned below criteria:

- Check, the object is separated from all other objects or not using a bounding box-based collision mechanism.
- Do not approximate an object by an impostor if its distance to the viewer falls below the given threshold.
- Create an impostor for the object if the impostor can be reused for a sufficient number of the frames.

Sometimes a last recently invented mechanism is used, when a memory volume is not available to save the current approximation. More, all approximations of all objects can be saved in the approximation cache for further usage that permits only to select the approximation during a rendering.

8.5 Multi-resolution Texturing for Vegetation Models

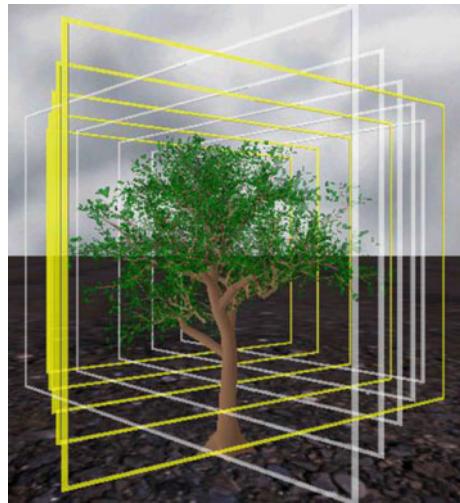
The main criteria of a virtual vegetation modelling are the unique visibility of each tree and shrub and their persistent positions that are given by various viewpoints.

The reasonable hybrid model was introduced by Lluch et al. [1]: for the trunk and branches, a procedural multi-resolution representation was introduced based on the parametric L-systems and, for the leaves, a hierarchical and multi-resolution image-based representation was applied. For the LODs construction, some metrics on the tree branches were proposed that preserves the structure of a tree. An intermediate data structure called a tree Abstract Data Type (t-ADT) involves the tree structure and required metric. A tree structure is represented by an input chain S . The t-ADT contains the nodes for each branch resulting from the interpretation of chain S ; a weight to each node is applied by a metric at each branch. Lluch et al. [1] introduced four metrics: the number of children $B(n)$, the number of descendants $R(n)$, the longest path to a leaf node $P(n)$, and the branching length $L(n)$ provided by Eq. 8.7, where n is a set of children, m is a subset of n , $d(n)$ is a set of descendants, $l(n)$ is a length of the branch.

$$\begin{aligned} B(n) &= |n| \\ R(n) &= |d(n)| \\ P(n) &= l(n) + \max(P(m)) \\ L(n) &= l(n) + \sum_{m \in \{n\}} l(m) \end{aligned} \tag{8.7}$$

The multi-chain contains the modules of the output chain and two types of the modules SAVE(id) and RESTORE(id) that help to generate the next LOD fast. For a foliage rendering, each node was associated with a bounding box that enclosed the geometry and orientation of that node and its descendants. Then six images of the leaves as the impostors (one for each side of the box) are rendering. In such manner, the bounding box hierarchy is created. However, this rendering does not look

Fig. 8.9 Tree approximation by blending two 60° sets of slices (the borders of the quads are rendered yellow and white for the left and the right set of slices, respectively)



deep. The modified version of leaves rendering was proposed in [42], when a volume of the box is divided into the parallel slices and an image for each slice is computed (Fig. 8.9). This technique provides a better sense of the depth and smoother transitions between the LODs with a shortcoming of large amounts of storage for the textures.

The idea of using a hierarchy of the LODs is a cornerstone in many algorithms. Thus, Erikson et al. [43] introduced some innovative techniques in their approach, formulating them in the following manner:

- The high fidelity approximations are achieved by grouping objects during design of the Hierarchical Levels of Detail (HLODs). For a simplification, the polygons from different objects are merged that increases the visibility of a landscape scene.
- The algorithm can automatically compute the HLODs of the scene graph.
- For rendering the LODs and the HLODs, the display lists are supported to make the best performance of current high-end graphics systems.

Notice that the topological challenges, such as vertex clustering, edge collapses, vertex merging, voxelization, hole removal, among others, exist, and each algorithm ought to compensate these artifacts usually merging the disjoint regions of an environment based on a topological performance.

The LDI is a type of an image-based representation that allows to model the objects with a high degree of a depth complexity. Moreover, for a terrain rendering with the homogeneous 3D textures, the tiling LDI was proposed with the multi-view and multi-resolution structure of the layered depth images and a hierarchical representation of the non-periodic tiling of the plane [38]. The polygonal environment can be represented using a scene graph as a directed acyclic graph. A node contains a polygonal representation of an object, while a directed arc

connects the parent and child node. The arc also contains a child-to-parent transformation matrix. A simple scene graph with the LODs and the HLODs is depicted in Fig. 8.10. The HLODs of a scene graph are computed as follows. The HLODs are recursively generated from the LODs in a bottom-up manner for each node in a scene graph. The HLODs of a leaf node are equivalent to its LODs. The HLODs of an intermediate node in a scene graph are computed by combining the LODs of the node with the HLODs of its children.

A hierarchical scheme for computing the tree levels of detail was proposed by Micikevicius and Hughes [44]. In the k th level of detail, the LOD- k , each k -sub-tree is replaced with a textured cross-polygon impostor that can transform in 3D space. The same set of textures is used for all trees of the same species. The silhouette of a sub-tree is the same from any two views at an angle of 180° to each other that makes possible to use the same texture for both sides of a quadrangle. The 12 LODs are used for a tree approximation. To reduce the popping artifacts during a switching between the discrete LODs, a discrete model was extended to a continuous one. The continuous model renders two LODs linearly interpolating the semi-transparency. The continuous LOD model nearly eliminates the popping artifacts but a visibility of an object changes smoothly. The following proposed model became the Inertial LOD model. If a change in detail is needed, the inertial LOD model interpolates the blend factor of the continuous LOD over a number of frames. Two LODs are maintained, referred to as past and target, at that the target LOD may need to be updated. Let a and b , $a < b$, be the past and target LODs, respectively. If the new target LOD c is less than a , then b becomes the past LOD; otherwise a remains the past LOD. The case, where $a > b$, is processed similarly. The blend factor has to be adjusted considering its current value in order to avoid the abrupt changes in the past LOD.

The simplification of a foliage geometry is also in a scope of interests. The Foliage Simplification Algorithm (FSA) was proposed by Remolar et al. [45] for the quadrilateral leaves. The FSA culls a pair of leaves, using the minimal value of the defined cost function to implement a leaf collapse. Then two leaves are replaced by a new larger leaf with similar position and shape as the original ones. This method was improved in the Progressive Leaves Union (PLU) algorithm to preserve a better silhouette and the overall foliage area of a crown. Zhang and Blaise [46] proposed a more reasonable cost function, which includes six items to measure the similarities of the leaf pairs, made the PLU as a view-dependent algorithm. The Hierarchical Union of Organs (HUO) algorithm [3] extends the PLU to simplify both quadrilateral and triangular leaves. The main improvement in the HUO deals with a hierarchically simplification of leaves with respect to the natural hierarchy of trees. The crown of a tree is composed by many leaf clusters arranged according to the important botanic concept of a phyllotaxy. Three levels in the HUO exist: in the first level, each single leaf is confined by the leaf cluster constructed using a phyllotaxy, the second level is the leaf cluster, and the third level is the whole foliage. Compared with the FSA and the PLU, the HUO can preserve the botanical architecture of the original models better, and it is more efficient in a pre-processing. All mentioned above algorithms simplified data on a pre-processing

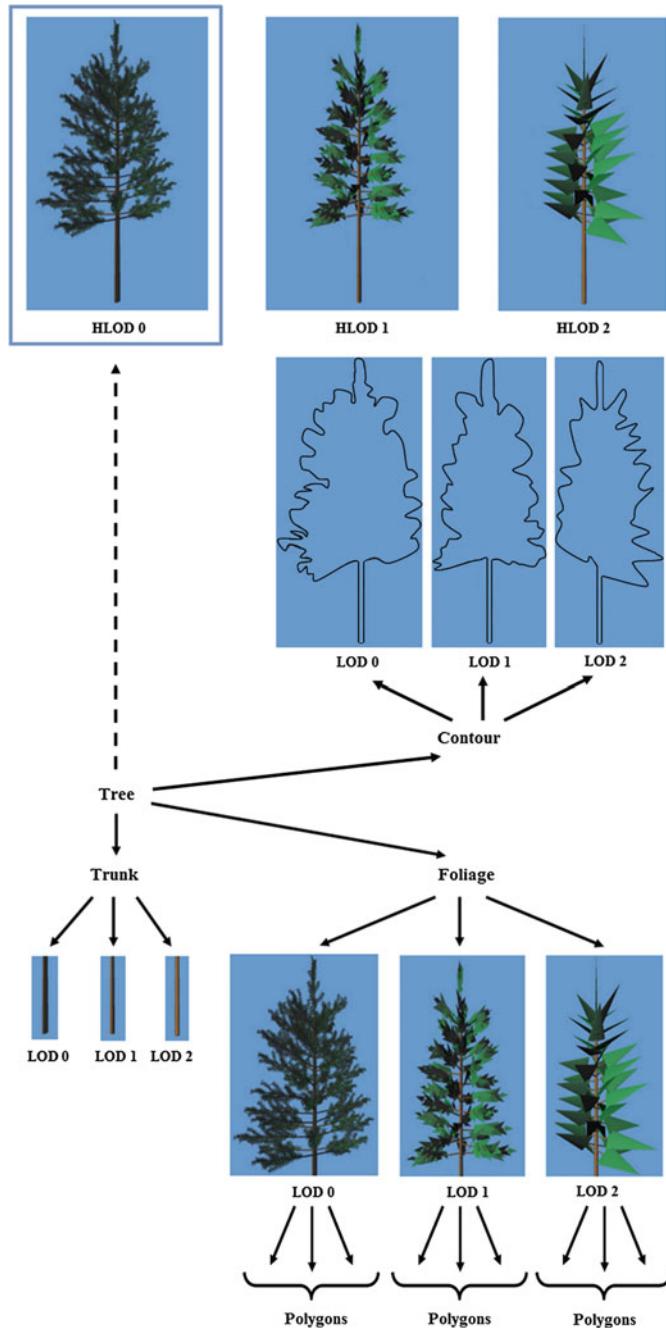


Fig. 8.10 Simple scene graph with the LODs and the HLODs

stage and recorded them on the hard disk. At run-time, the data is loaded into the RAM, and their appropriate models will be chosen for different viewpoints automatically.

Consider a linear cost function to show the priority of the pair formed by the leaves lf_i and lf_j for a leaf collapse how it was proposed in the PLU algorithm [46]. The PLU cost function $S(lf_i, lf_j)$ is defined by Eq. 8.8, where $S_1(lf_i, lf_j)$ is a normal similarity, $S_2(lf_i, lf_j)$ is a positional similarity, $S_3(lf_i, lf_j)$ is an area similarity, $S_4(lf_i, lf_j)$ is a diameter similarity, $S_5(lf_i, lf_j)$ is a union age similarity, $S_6(lf_i, lf_j)$ is a diameter penalty of the leaves lf_i and lf_j , $\{k_l\}$ are the coefficients with values $k_1 = k_3 = k_4 = 0.0375$, $k_2 = 0.3125$, $k_5 = k_6 = 0.2875$ with the total sum equaled 1.

$$S(lf_i, lf_j) = \sum_{l=1}^6 k_l S_l(lf_i, lf_j) \quad (8.8)$$

Using Eq. 8.8, the more preferential the pair should be simplified. However, a human vision is sensitive to the leaves close to the silhouette of a crown and insensitive to the leaves inside of the crown. Deng et al. [14] modified the cost function in Eq. 8.8 by adding the seventh item: a centricity $S_7(lf_i, lf_j)$, its value is a real in the range of $[0, 1]$, which is defined in Eq. 8.9, where q is a center of the crown, p_i and p_j are the center points of the leaves lf_i and lf_j , respectively, $d(\cdot)$ is a distance between two points, D_{crown} is a diameter of a crown.

$$S_7(lf_i, lf_j) = (d(q, p_i) + d(q, p_j)) / D_{crown} \quad (8.9)$$

The smaller value of $S_7(lf_i, lf_j)$ for the leaves lf_i and lf_j means that the two leaves are more likely to be simplified. Then Eq. 8.8 is modified by Eq. 8.10.

$$S(lf_i, lf_j) = \sum_{l=1}^7 k_l S_l(lf_i, lf_j) \quad \sum_{l=1}^7 k_l = 1 \quad (8.10)$$

Deng et al. [14] had offered $k_7 = 0.3$, while the sum of the other six weights was 0.7 due to a requirement of normalization.

Hereinafter, Deng and Zhang extended their approach by the idea of hierarchical simplification in the HUO algorithm [47]. The essential difference consists in a regular 3D grid to partition the crown volume. Then for each cell, the continuous two-level LOD models are generated with hierarchical simplification within each cell. The criterion of a leaf cluster belonging to a grid cell is that its center is inside the cell. To discriminate leaves in different positions of the crown, the spatial error for each cell was computed using a global spatial error e , which is estimated by the distance from the camera to the tree and the camera parameters, and local spatial error e_k in cell k according to the occlusion degree hi provided by Eq. 8.11, where d is a distance from the viewer to the center of the crown, d_i is the distance between

the viewer and the center of the cell i , \bar{v} is a the normalized vector from the camera to the center of the crown, \bar{p}_k is a normalized vector from the center of the crown to the center of the cell i .

$$e_k = (1 + h_k) e \quad h_k = \begin{cases} 0 & \text{if } d_k \leq d \\ \bar{v} \cdot \bar{p} & \text{if } d_k > d \end{cases} \quad (8.11)$$

Such spatial errors of cells are non-uniform through a tree crown.

Besides the broad-leaves tree approximation, other models can be incorporated in the landscape scenes, such as models of the coniferous trees and grass simulation. Deng et al. [48] presented a geometric simplification method for modelling of the leaves of the pine-trees or fir trees. The method is based on a cylinder and line representation combining with the LOD technique. The simplest cylinder models with two tunable parameters, including the centerline and the radius of each needle leaf, are used to represent needles near the end-user, while the translucent lines depict needles at a far distance. Each line is determined as a skeleton of a cylinder plane projection. These lines are merged together for the farthest views. This process is depicted in Fig. 8.11.

The grass as a plant family that occupies the greatest area of the world's land surface is very difficult to model and render in 3D scenes. A natural surface is composed of the grass blades, which density can be estimated very relatively. Huge number of the grass cannot be fully stored in the RAM and rendered directly. The goal is to render the surfaces of a grass with the high accuracy in real-time mode. In comparison to a tree modelling, a grass modelling is represented in literature limitedly [49–52]. It may be explained that a grass in large landscape scenes is often simulated as a texture. The reasonable approach of a grass modelling one can find in the researches of Boulanger, for example, in [52, 53]. The approach combines the

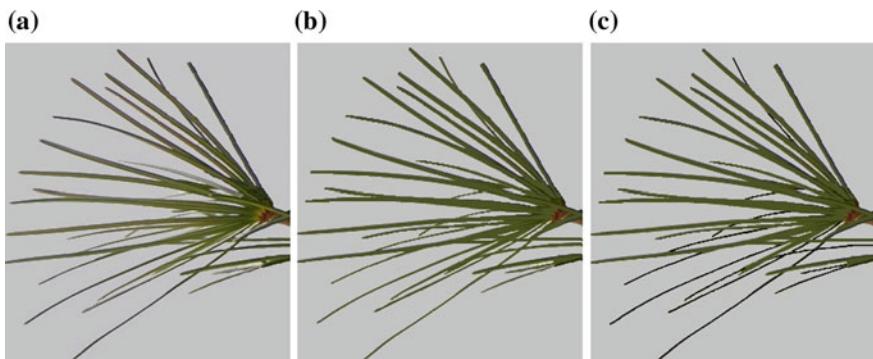


Fig. 8.11 Coniferous foliage approximation: **a** original image, **b** cylinder representation, **c** far needles are represented by *black lines*

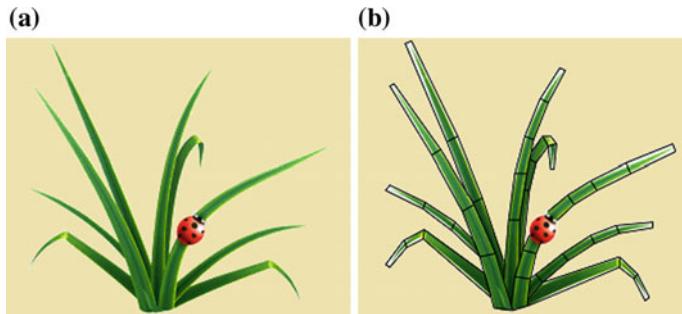


Fig. 8.12 Grass blades: **a** original image, **b** grass blades defined with semi-transparent quadrilateral strips

geometry and volume rendering, using the LODs and a simple non-periodic tiling scheme with only four different versions of the unique grass patch. Boulanger et al. [52] used the geometrically modelled grass blades for rendering close to the camera. Grass blades are represented as a volume slices data. Each grass blade is modelled by a billboard in a view of quadrilateral strip in with two-sided quadrilaterals of a zero thickness (Fig. 8.12). Notice that before this a direction of a strip ought to be determined. Usually it begins from the root of the blade, almost vertically, with influence of gravity. The alpha channel of the texture and covering the quadrilateral strips give the correct shape of a grass blade. In such manner, many cereal plants can be modelled.

The Lambert reflection model is used for simulation of the grass blade surfaces, representing the reflectance of diffuse only surfaces. The color of each blade is slightly modified to simulate different ages and levels of a degradation. To simulate the ambient occlusion, the color of the blades close to the ground is set darker. The coefficient of the ambient occlusion per a vertex is calculated as a linear function of the height of the vertex respect to the Earth's surface. Examples of a grass modelling are depicted in Fig. 8.13.

Additional challenge is to simulate a vegetation in the close distance to the camera for the complex terrains with the differences in the elevation.

8.6 Experimental Results

For texturing of large landscape scenes, the software tool “TerraTex”, v. 1.0, was designed using environment Microsoft Visual Studio 2008 Professional Edition and language C++. The graphical package Direct X SDK, which is distributed free and includes all necessary libraries and the header files for support of 3D graphics.

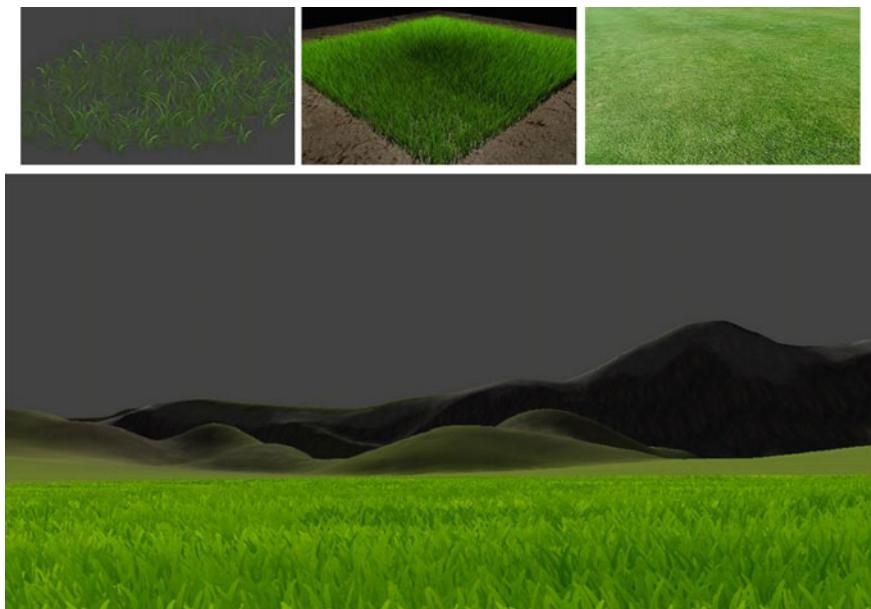


Fig. 8.13 Examples of a grass modelling

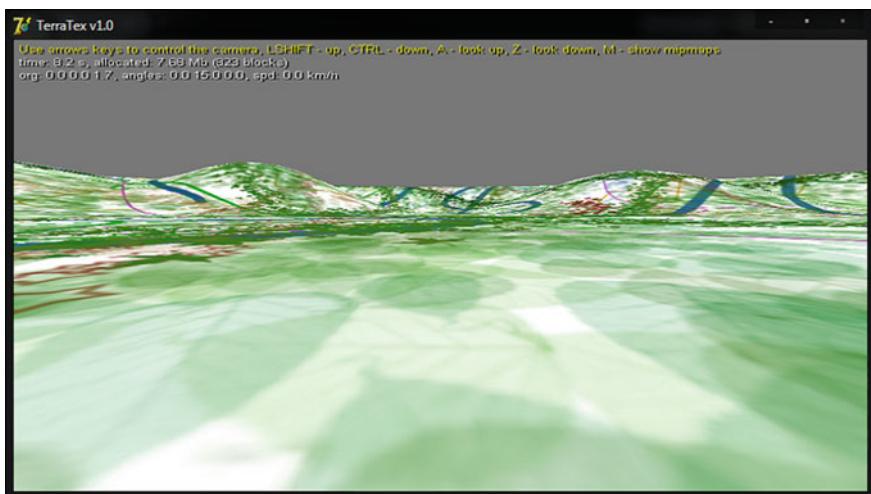


Fig. 8.14 Main screen form of software tool TerraTex, v. 1.0

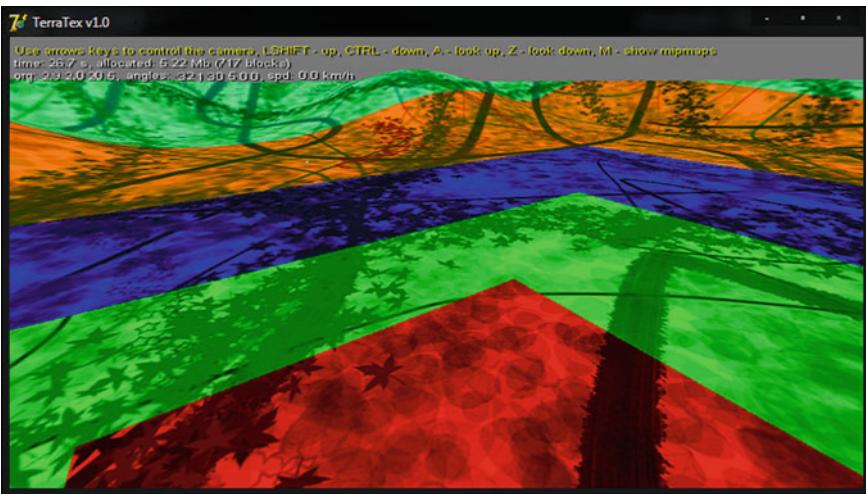


Fig. 8.15 Screen form with five LODs

The transfer of software tool at other software platforms, for example, in open graphical library OpenGL is possible.

The software tool TerraTex, v. 1.0 is executed under the operating system Windows, version XP and higher. For successful work, it is required to install the package MS DirectX 9.0c. The main screen form is depicted in Fig. 8.14.

The designed software tool has a simple interface with the following functionality: the camera translation in the horizontal and vertical planes, the camera rotation, and the camera tilts up and down. Also the tuning parameters that can increase the processing speed are available:

- The choice of a graphical adapter if computer contains several adapters.
- The choice of a rendering utility.
- The switching between the windowed and full screen modes.
- In full screen mode, it is possible to select the colors, the resolution and the updating frequency.

The software tool “TerraTex”, v. 1.0 allows to visualize the LODs with the different colors as it is shown in Fig. 8.15. The generated landscape scenes with the modelled trees and different types of textures of landscape surface are depicted in Fig. 8.16.

The software tool TerraTex, v. 1.0 is the experimental tool that implements the LODs technology in the large landscape scenes with the modelled vegetation.

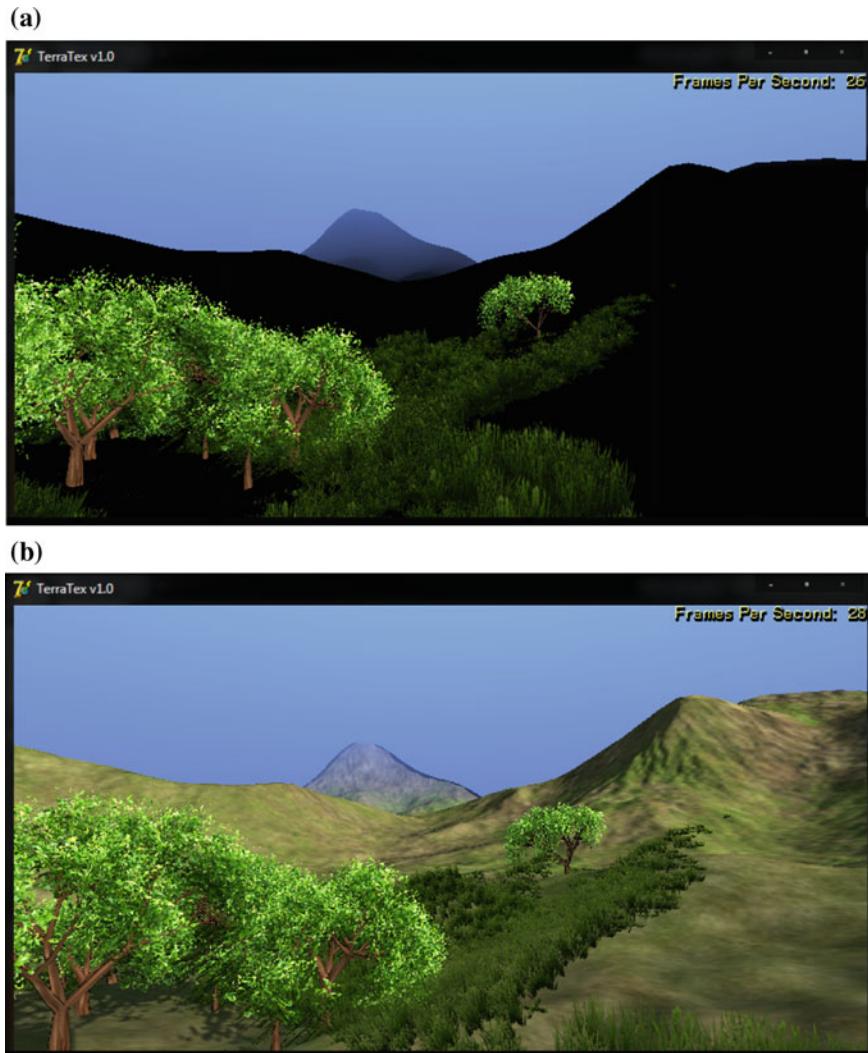


Fig. 8.16 Examples of different textural surfaces: **a** scene without a texture, **b** scene with a grass texture, **c** scene with a sand texture, **d** scene with an artificial texture

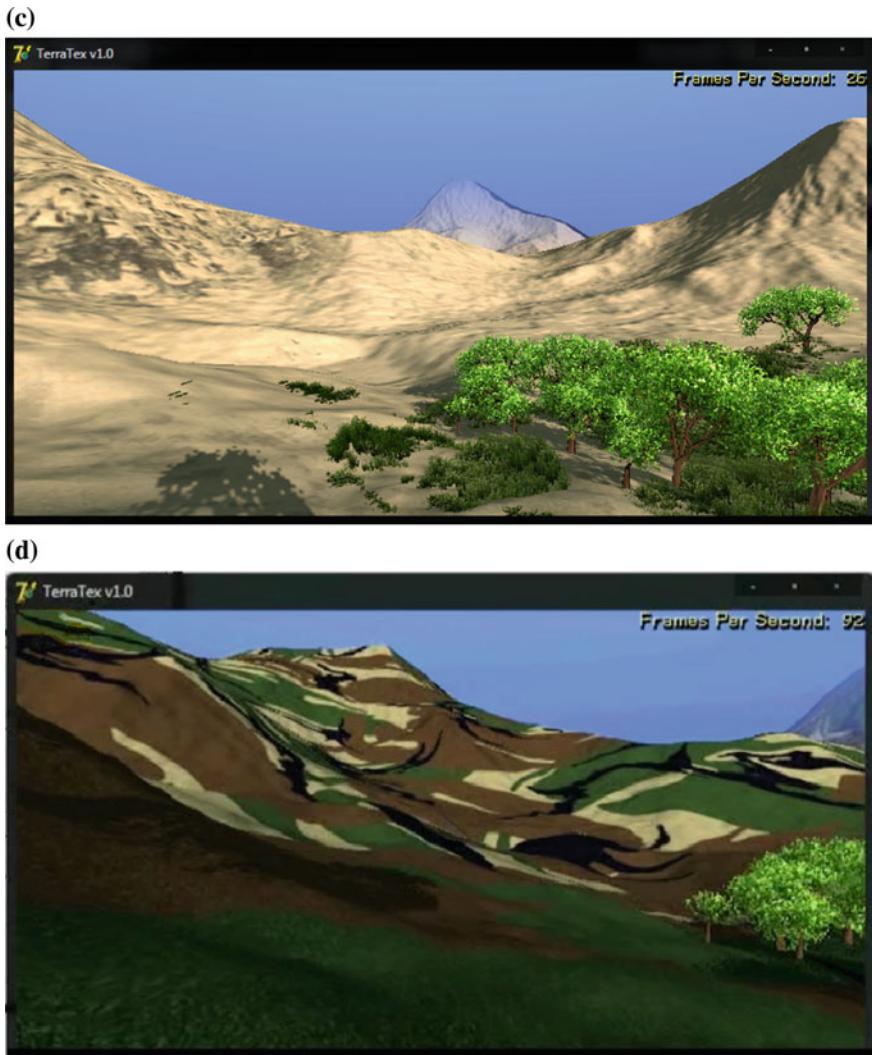


Fig. 8.16 (continued)

8.7 Conclusions

The adaptive approximations (or the varying LODs) are widely used techniques in a representation of the large landscape scenes as well as the individual objects in the perspective projections. The first LOD techniques were based on the discrete multi-resolution models, while the following LOD-based rendering approaches use the continuous multi-resolution models as the abstract data structures. The current investigations are directed on the algorithmic optimization of the storage and

mapping of huge data with a high resolution simultaneously. In dependence of the goal, the textural techniques can vary in order to receive the performance benefits. A vegetation modelling is also very difficult issue with the topological artifacts caused by edge collapses, vertex merging, voxelization, hole removal, among others. The following development of these methods is surely required.

References

1. Lluch J, Camahort E, Hidalgo JL, Roberto Vivo R (2010) A hybrid multiresolution representation for fast tree modeling and rendering. *Proc Comput Sci* 1(1):485–494
2. Quan L, Tan P, Zeng G, Yuan L, Wang J, Kang SB (2006) Image-based plant modeling. *ACM Trans Graph* 25(3):599–604
3. Zhang X, Blaise F, Jaeger M (2006) Multiresolution plant models with complex organs. *ACM SIGGRAPH international conference on virtual reality continuum and its applications (VRCAIA 2006)*, pp 331–334
4. Object Raku Technology, Inc. <http://www.objectraku.com/>. Accessed 1 Jan 2016
5. Delta3D Open Source Engine. <http://sourceforge.net/projects/delta3d/>. Accessed 1 Jan 2016
6. Garcia I, Sbert M, Szirmay-Kalos L (2005) Leaf cluster impostors for tree rendering with parallax. *Eurographics 2005, Short presentation* 1–4
7. Fuhrmann A, Umlauf E, Mantler S (2005) Extreme model simplification for forest rendering. In: 1st Eurographics conference on natural phenomena (NPH 2005), pp 57–67
8. Behrendt S, Colditz C, Franzke O, Kopf J, Deussen O (2005) realistic real-time rendering of landscapes using billboard clouds. *Comput Graph Forum* 24(3):507–516
9. Max N (1996) Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In: Pueyo X, Schröder P (eds) *Rendering techniques 1996*. Springer, Wien
10. García I, Patow G, Szirmay-Kalos L, Sbert M (2007) Multi-layered indirect texturing for tree rendering. In: 3rd Eurographics conference on natural phenomena (NPH 2007), pp 55–62
11. Shade J, Gortler S, Wei HL, Szeliski R (1998) Layered depth images. In: Annual conference on computer graphics (SIGGRAPH 1998), pp 231–242
12. Meyer A, Neyret F, Poulin P (2001) Interactive rendering of trees with shading and shadows. In: Gortler SJ, Myzskowski K (eds) *Rendering techniques 2001*. Springer, Wien
13. Decaudin P, Neyret F (2004) Rendering forest scenes in real-time. In: *Eurographics symposium on rendering rendering techniques*, pp 93–102
14. Deng Q, Zhang X, Jaeger M (2006) Efficient multiresolution of foliage for real-time rendering. In: IEEE 2nd international symposium on plant growth modeling and applications (PMA 2006), pp 307–314
15. Sillion F, Drettakis G, Bodelet B (1997) Efficient impostor manipulation for real-time visualization of urban scenery. *Comput Graph Forum* 16(3):207–218
16. McMillan L, Bishop G (1995) Plenoptic modeling: an image-based rendering system. In: 22nd annual conference on computer graphics and interactive techniques (SIGGRAPH 1995), pp 39–46
17. Debevec P, Yu Y, Borshukov G (1998) Efficient view-dependent image-based rendering with projective texture-mapping. In: Drettakis G, Max N (eds) *Rendering techniques 1998*, Eurographics. Springer, Wien
18. Segal M, Korobkin C, Van Widenfelt R, Foran J, Haeberli P (1992) Fast shadows and lighting effects using texture mapping. *ACM SIGGRAPH Comput Graph* 26(2):249–252
19. Xu L, Li E, Li J, Chen Y, Zhang Y (2010) A general texture mapping framework for image-based 3D modeling. In: IEEE 17th international conference on image processing (ICIP 2007), pp 2713–2716

20. Schaufler G (1995) Dynamically generated impostors. In: Fellner DW (ed) Modeling—virtual worlds—distributed graphics MVD 1995 Workshop, pp 129–136
21. Zitnick C, Kang S, Uyttendaele M (2004) High-quality video view interpolation using a layered representation. ACM Trans Graph 1(212):600–608
22. Imber J, Volino M, Guillemaut J-Y, Fenney S, Hilton A (2013) Free-viewpoint video rendering for mobile devices. In: 6th international conference on computer vision/computer graphics collaboration techniques and applications (MIRAGE 2013), article No. 11
23. Starck J, Kilner J, Hilton A (2009) A free-viewpoint video renderer. J Graphics GPU Game Tools 14(3):57–72
24. Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Werner Stuetzle W (1995) Multiresolution analysis of arbitrary meshes. In: 22nd annual conference on computer graphics and interactive techniques (SIGGRAPH 1995), pp 173–182
25. Xia J, El-Sana J, Varshney A (1997) Adaptive real-time level-of-detail-based rendering for polygonal models. IEEE Trans Visual Comput Graph 3(2):171–183
26. Hilbert K, Brunnett G (2004) A hybrid LOD based rendering approach for dynamic scenes. In: IEEE International conference on computer graphics (CGIV 2004), pp 274–277
27. SpeedTree UE4. <http://www.idvinc.com/>. Accessed 20 Dec 2015
28. Heckbert PS, Garland M. (1994) Multiresolution modeling for fast rendering. Graphics Interface 1994, Canadian Inf Proc Soc, pp 43–50
29. Oppenheimer P (1986) Real time design and animation of fractal plants and trees. Newslett ACM SIGGRAPH Comput Graph 20(4):55–64
30. Weber J, Penn J (1995) Creation and rendering of realistic trees. In: 22nd international ACM conference on computer graphics and interactive techniques (SIGGRAPH 1995), pp 119–128
31. Remolar I, Chover M, Belmonte O, Ribelles J, Rebollo C (2002) Geometric simplification of foliage. Eurographics 2002:397–404
32. Zhang S (2014) Foliage simplification based on multi-viewpoints for efficient rendering. J Software 9(7):1655–1665
33. Jakulin A (2000) Interactive vegetation rendering with slicing and blending. In: de Sousa A, Torres JC (eds) Proceedings of Eurographics 2000, Short Presentations 1–9
34. Heckbert PS (1989) Fundamentals of texture mapping and image warping. Master's thesis. Computer Science Division ((EECS), University of California, Berkeley. Report No. UCB/CSD 89/516
35. Heckbert PS, Moreton HP (1991) Interpolation for polygon texture mapping and shading. In: Rogers DF, Earnshaw RA (eds) State of the art in computer graphics: visualization and modeling. Springer, New York
36. Wang H (1961) Proving theorems by pattern recognition II. Bell Syst Tech J 40:1–41
37. Berger R (1966) The undecidability of the domino problem. Mem Am Math Soc 66:1–72
38. Shade J, Cohen MF, Mitchell DP (2002) Tiling layered depth images. University of Washington, Department of Computer Science and Engineering Technical Report #02-12-07
39. Wilson A, Manocha D (2003) Simplifying complex environments using incremental textured depth meshes. ACM J Trans Graph 22(3):678–688
40. Jeschke S, Wimmer M (2002) Textured depth meshes for real-time rendering of arbitrary scenes. In: 13th Eurographics workshop on rendering (EGRW 2002), pp 181–190
41. Luebke D, Erikson C (1997) View-dependent simplification of arbitrary polygonal environments. In: 24th annual conference on computer graphics and interactive techniques (SIGGRAPH 1997), pp 199–208
42. Meyer A, Neyret F (1998) Interactive volumetric textures. Render Tech 1998:157–168
43. Erikson C, Manocha D, Baxter III WV (2001) HLODs for faster display of large static and dynamic environments. In: Symposium on interactive 3D graphics I3D 2001, pp 111–120
44. Micikevicius P, Hughes CE (2007) Visibility-based forest walk-through using inertial level of detail models. J Defense Model Simul 4(2):80–96
45. Remolar I, Chover M, Ribelles J, Belmonte O (2003) View-dependent multiresolution model for foliage. J WSCG 11(2):370–378

46. Zhang X, Blaise F (2003) Progressive polygon foliage simplification. In: International symposium on plant growth modeling, simulation, visualization and their applications (PMA 2003), pp 217–230
47. Deng Q, Zhang X (2008) Grid-based view-dependent foliage simplification. *J Comput Inf Syst* 4(4):1643–1650
48. Deng Q, Zhang X, Gay S, Lei X (2007) Continuous LOD model of coniferous foliage. *Int J Virtual Reality* 6(4):77–84
49. Bakay B, Lalonde P, Heidrich W (2002) Real-time animated grass. *Eurographics* 2002:234–237
50. Pelzer K, Bytes P (2004) Rendering countless blades of waving grass. In: Kirk D (ed) GPU Gems. Addison-Wesley, New York
51. Shah MA, Konttinen J, Pattanaik S (2005) Real-time rendering of realistic-looking grass. In: 3rd international conference on computer graphics and interactive techniques in Australasia and South East Asia Graphite, pp 77–82
52. Boulanger K, Pattanaik S, Bouatouch K (2006) Rendering grass in real-time with dynamic light sources and shadows. Research Report PI 1809
53. Boulanger K (2005) Real-Time realistic rendering of nature scenes with dynamic lighting. PhD dissertation, University of Rennes I, France

Chapter 9

Large Scene Rendering

Abstract Large scene rendering causes many issues, including the algorithmic support and software/hardware implementation. The Level Of Details (LOD) architecture is the basis of terrain and vegetation rendering. The texturing techniques are strongly connected with a category of the LOD algorithm. Multi-texturing, clipmaps, and virtual texturing are the main methods, applying in the LOD algorithms. The classification of the forest rendering techniques demonstrates a great variety of methods that were developed actively since 1990s. The realistic leaves and grass rendering in the nearest LOD are the special issues to that ought to be given a lot of attention respect to the current situation in computer graphics. The realistic lighting and shadow mapping technique are also in the focus of consideration.

Keywords Mathematical morphology • Shape representation • Continuous skeleton • Morphological spectrum • Image analysis • Change detection

9.1 Introduction

The real-time rendering of the large landscape scenes is employed in many applications, such as video games, serious gaming, landscape and urban scene walk-through, simulators in forest inventory, web applications, among others. Forests contain many objects that are complex for a geometric modelling and its real-time rendering remains the great challenge. On the one hand, the algorithmic implementation of the rich plant details in large landscape scene ought to be compact and optimal in accuracy. On the other hand, a computer memory should be enough for the storage and real-time rendering of big modelling data that are loaded partly according to the end-user viewing.

The mapping of the large terrains is a multi-stage process, when the coarser algorithm detects the active region of a terrain, the more accurate algorithm calculates, what portion of data ought to be loaded, and the precise algorithm displays only the necessary data. The rendering of the large terrains means the mapping of a terrain during the camera movement. A continuity of a mapping requires the

algorithms, fetching the data that are necessary in future time instants. Thus, the rendering of the large detailed terrains implies the development of algorithms, including the selection of terrain parts that are likely required in the near future, rendering of visible terrain parts, loading, managing, disposal of the loaded data, and balancing of algorithms in order to maintain the high performance. The data are distributed in three levels of storage pools, such as the hard drive memory, containing a complete terrain data stored as small blocks in each detail level, the system memory, including the texture blocks from a wide spherical area around the camera, and the Graphics Processing Unit (GPU) memory, mapping the visible texture blocks with surrounding in the case of a rapid motion. These algorithms organize the transfers between the hard drive memory and the system memory independently of the GPU workload. The GPU memory has to wait until the transfers are complete. Hence, the transfers to the GPU memory have to be limited per a frame.

This chapter is organized as follows. In the next Sect. 9.2, some related works are reviewed. In Sect. 9.3, the challenges of a large landscape scene rendering with the possible decisions are discussed. The large terrain and vegetation renderings are described in Sects. 9.4 and 9.5, respectively. The realistic lighting is presented in Sect. 9.6. Information about shaders on the example of Unity 3D 3.x package is located in Sect. 9.7. Finally, the concluding remarks are given in Sect. 9.8.

9.2 Related Work

The rendering is a widely spread procedure but a type of the object impose the special requirements. Consider the overview of the terrain, object and lighting rendering methods in Sects. 9.2.1–9.2.3, respectively.

9.2.1 *Overview of Terrain Rendering Techniques*

The terrain LOD algorithms can be categorized as follows:

- The irregular meshes or triangulated irregular networks provide good approximation but require the tracking of the mesh adjacencies and refinement dependencies. Some of the irregular meshes use the Delaunay triangulation. The Real-time Optimally Adapting Meshes (ROAM) [1] is a recursive method of triangles' merging, depending on the placement in the view-frustum and the difference between the triangle planes. Similar algorithms were designed in the 1990s, when the GPUs were the simple processors and the Central Processing Units (CPUs) were the relatively powerful. The meshes were processed by the CPU for every camera movement.

- The bin-tree hierarchies use the recursive bisection of the right triangles. They require the RAM volume and the immediate mode rendering.
- The bin-tree regions define the refinement operations on regions associated with a bin-tree structure. The pre-computed triangulated regions are uploaded to the buffers cached in a video memory.
- The tiled blocks partition a terrain into the square patches that are tessellated at the different resolutions. The seamless stitching of the block boundaries is the main challenge.
- The regular grids using the geometry clipmaps define a hierarchy centered about the viewer (A clipmapping is a method of clipping a mipmap to a sub-set of data pertinent to the geometry being displayed). This simplifies the inter-level continuity in the spatio-temporal domain. The refinement hierarchy is based on the viewer-centric grids, providing the inter-level continuity with the geomorphs. The refinement criterion still considers a viewer distance but it ignores a local surface geometry.

All view-dependent LOD algorithms coarsen adaptively the mesh based on the screen-space geometric error that estimates the pixel deviation between the mesh and the original terrain. Generally, a screen-space error depends on the viewer distance, surface orientation, and surface geometry.

The texturing techniques are strongly connected with a category of the LOD algorithm. The multi-texturing, clipmaps, and virtual texturing are the main methods, applying in the LOD algorithms. The multi-texturing can be concern to the simple methods used to blend the multiple textures [2]. The GPU creates a large high-detail texture by repeating the high resolution terrain detail textures and use of a low resolution mask texture or a height map. This method is simple and effective but offers a very limited amount of the textures to blend. Also, it creates the repeating patterns of the texture tiles that leads to a poor visibility.

The clipmaps [3] is a common algorithm for the LOD of the textures and meshes that has been used since the 1990s. This algorithm provided an effective method of the displaying textures under the conditions of the limited GPU memory and restricted rendering possibilities of old computer assemblies. The clipmaps stores a cutting of the textures centered in the camera viewpoint at different LOD levels that are restored during a rendering. When a camera moves, a clipmap has to be recalculated. New data is refreshed as rows or columns in a clipmap. If a new pixel is outside of the clipmap borders, then it overflows to the opposite side of the texture. This procedure is called a toroidal addressing [3]. The updating of the rows is done in a single operation as the data is stored sequentially, while the updating of columns require one operation for each pixel in the column. This makes the clipmaps to be less effective on the modern hardware. The geometry clipmaps [4] utilize the same principle of the clipmaps applied to the meshes but additionally interpolate the vertex coordinates at the border of two detail levels. The geometry clipmaps suffer from the same problem of data uploading in the higher detail levels, while typically 60–75% of the data in a clipmap do not used. When a camera moves in one direction, it still needs to fetch the lower levels textures in a 360° range. The

use of the clipmaps may cause visual artifacts, for example, sharp corners. The geometrical MIP-maps are closely related to clipmaps [5] but loads the small blocks instead of single rows or columns of the textures and meshes. This may cause a “popping effect”, when the large blocks are updated during a camera movement.

A virtual texturing is a technique for storage of the active sub-sections of the texture as small tiles in a single texture storage that then can be reassembled and rendered as if the complete texture was stored in the GPU memory. Each LOD of the complete texture is divided into blocks of the constant size, where the total block structure will have resemblance with an Octree structure [2]. This block structure is very similar to the geometrical MIP-maps. Each of these blocks is checked for intersection with the field of view in order to determine, which blocks are visible. The active blocks are then stored in a single texture in the GPU memory, and may be extracted in any random order. Since the neighboring blocks in the texture storage might not be neighboring in the complete texture, the blocks need in the padding to avoid the issues with clipping, when the texture is used in a rendering. Due to the padding, the block sizes are no longer power of two. The use of the texture filtering, such as trilinear filtering, will increase the need for larger margins, which further increases the storage overhead. The Sparse Virtual Textures, the MegaTexture, and the Partial Resident Textures are similar implementations of the virtual textures. The Sparse Virtual Textures load the texture blocks, when it is required for a rendering, and the discard blocks using the Least Recently Used (LRU) queue. The MegaTexture employs a single large texture space for a static terrain. The Partial Resident Texture is a new way of getting some hardware support for the Virtual Textures and the MegaTextures.

9.2.2 *Overview of Object Rendering Techniques*

Since 1990s, the real-time forest rendering has been intensively studied, and the accumulated experience can be presented in the following classification:

- *The point-based rendering.* Reeves and Blau [6] were the first, who used the point models to render the trees and grass. Their stochastic modelling approach called the particle systems was introduced to provide the visual richness and realism of vegetation. First, they had chosen a shape of a tree with five global parameters (height, width, first branch height, mean branch height, and branching angle). A recursive algorithm generated some branches. Then the leaves or needles were added to the branches that had no sub-braches. Finally, the colors of the trunk, branches, and leaves had been assigned. The clump of a grass was simulated using the stochastic parameters of its position, orientation, area, density, and types, also as a particle system. These authors understand that an individual tree might be composed of over millions independent particles. They proposed the approximate probabilistic shading model for both trees and grass. Thus, the trees are internally (from branches and leaves of the same tree)

and externally (by neighboring trees) shadowed. The shading functions provide ambient, diffuse, and specular components. Nowadays, the point-based rendering of trees is considered efficiently only for the distant objects.

- *The line-based rendering.* The lines are efficiently rendered by the small polygons. Both points and lines usually combine with the polygons to construct the hybrid tree models. The polygonal geometry is used for the close distances but the practice of the random points and lines application is not reasonable.
- *The polygon-based rendering.* The polygon represents the geometric model of a plant. The main shortcoming is a visibility of the geometric details in the close distance. The multi-resolution and the LOD algorithms are frequently used these methods with a geometry compression. The polygon-based rendering efficiently simplifies the smooth objects but cannot represent a complex topology of a tree.
- *The image-based rendering.* The Image-Based Rendering (IBR) is the efficient technique that is applied for many years. In the IBR, several types of billboards are employed. A billboard consists of the triangles or quadrilaterals covered by a semi-transparent 2D texture. The simplest billboard represents the entire model constantly facing the camera. The IBR technique can render the complex natural objects, such as grass. This method shows no parallax, when the camera is moving. However, some unrealistic artifacts can appear in the close distance due to a weak geometry. The simplest billboard is often used in the background, while an approximation of the trees requires more complex constructions. For example, the fixed crossed quadrilaterals of two or three billboards, crossing one another, represent 3D impression of a tree. Decoret et al. [7] introduced the billboard clouds as a means of an extreme simplification, when a set of the arbitrary oriented billboards had been generated the visually pleasing results with a smooth geometry model. Nevertheless, the billboard clouds fail for the plant models with a high complexity. The IBR renders more efficiently than the polygon-based rendering because a single primitive replaces large number of 2D geometrical shapes. It is often used in the industrial simulators.
- *The volume-based rendering.* A volume implies 3D reference box, containing one or more instances of the rendering object. The lighting and shadowing of the complex natural scenes using the GPU provides the realism to the volume-encoded forest scenes. However, this method requires the pre-computed 3D textures [8]. Decaudin and Neyret [9] proposed no periodic volumetric texture tiles to render the dense forests. The volumetric textures were generated as a series of the parallel image slides and then the tiles over the DTM to represent the forests. Hereinafter, Decaudin and Neyret [10] extended their previous method by the volumetric billboards. The volume-based rendering offers a full parallax effect from any viewing direction and the objects are correctly rendered during a walkthrough. In spite of the memory cost is very high; this type of rendering overcomes the IBR. The volume-based rendering is often used in the flight simulators.

The mentioned above classification of the rendering techniques in the forest scenes is the approximate. Hereinafter, some approaches were developed. Zhang

et al. [11] proposed the hierarchical layered depth images that were generated from the sampled depth images of the polygonal tree models and performed as a hierarchy structure. In order to decrease a storage cost of this approach during the real-time shadow and dynamic lighting effects, Liu et al. [12] suggested an approach that adopts a dynamic quad stream on the GPU buffers for a view-dependant rendering. A quad stream arranged as the pre-sampled parallel billboards represented a tree model. In the run-time rendering stage, a geometry tessellation increases the emitted primitives to refine the tree models. This approach provides very high visual quality and can render large-scaled forest scenes with the high efficiency and low memory and data transfer cost. At the same time, it is limited in receiving a realistic rendering effect, such as shadowing.

The rendering ought to support the realistic representation not only of forests but also foliage and leaves. Some good results were implemented in this direction. Gumbau et al. [13] developed a foliage pruning method for the real-time vegetation rendering under the assumption that some of the foliage are not visible depending on the viewer's location. In a pre-processing step, the LOD foliage model is divided into a cloud of cells. Each cell's visibility is computed from a set of the external viewpoints. In the run-time step, the LOD models are interactively altered respect to the viewpoint position, size, and color of the rendering visible polygons. This method is efficient for the GPU, however, it cannot achieve the real-time shadowing effects. Bao et al. [14] proposed a leaf geometry modelling approach based on a special representation of a leaf in alpha format, thereby, the leaf models can be replaced without damaging a visibility. Deng et al. [15] proposed the LOD-base simplification method for the coniferous leaves as a cylinder and line representation to model the nearby and faraway coniferous leaves, respectively. The lines can be merged for further simplification.

It is necessary to mention the achievements in the hardware implementation suitable for the realistic forest scene rendering, such as the progressive transmission and hardware instancing. The Progressive Meshes (PMs) in a view of arbitrary triangle meshes proposed by Hoppe [16] permit to execute the progressive transmission and loss less compression of a model data over a communication line. The view-dependent progressive meshes were introduced by Kim et al. [17]. They gave a priority on the visual impact of the transmitting large patches and reducing a time cost. Unfortunately, the complexity of the view-dependent progressive meshes limited their flexibility. Maglo et al. [18] proposed a progressive mesh compression method as the high-quality intermediate LODs that can handle 3D objects associated attributes. This method has the adaptation procedures to optimize the LOD management of 3D scenes including the network bandwidth, device's graphic capability, display resolution, and user preferences.

A visualization of the large-scale forests is a great challenge not only due to the high geometric complexity but also due to the batch problem. This is caused by generating of large number of graphics Application Programming Interface (API) draw calls in every frame. Most previous methods work by the following algorithm. First, all geometry instances are updated in the view of a frustum.

Second, the update are put data into several vertex streams. Third, the streams are rendered in a few draw calls. These methods need the culling instances against the view frustum, determining the appropriate LOD models. Shopf et al. [19] proposed to use the geometry shaders to perform the culling and dynamic LOD selection on the GPU.

9.2.3 Overview of Realistic Lighting

The realistic lighting is still a challenge because the light interaction of vegetation, consisting of thousand millions of elements, is quite complex. Qin et al. [20] proposed a fast photo-realistic rendering of trees using a quasi-3D tree database. The geometrical information, including an original color for each visible polygon, normal vector, relative depth, shadowing of direct sunlight and skylight from every direction, and shading information, estimating the transparency, specular reflection, and inter-reflection of leaves, were the main parameters for a modelling. Qin et al. [20] introduced the innovative for that time ideas and technical solutions for a photorealistic representation of the forests. For example, each type of a tree owns a deep-frame buffer, which can be repeatedly used for the same type of a tree in other places, self-shadow information for sunlight and skylight are kept for each pixel of a tree, and plural sun shadow buffers are employed to assure the accuracy of the external sun shadows. The use of a quasi-3D tree helped to reduce the computational time and memory requirements.

Behrendt et al. [21] proposed a method that renders the complex geometries in a dynamic low-frequency lighting environment using a special representation for the reflection function. A spherical harmonic lighting projects the rendering equation onto an ortho-normal basis (the Legendre polynomials) over a sphere. In a pre-processing step, a set of coefficients is computed from the original object. During a run-time, the lighting environment is projected onto the basis, and the evaluation of the scalar product of the coefficients provides a final luminance result. For diffuse a light interaction, a single vector of real numbers is associated with each vertex of the geometry. However, the complexity of this method is high. A number of bands used to calculate the coefficients ought to be restricted.

The shadow mapping has been used extensively in 3D virtual reality. The standard shadow mapping suffers from its well-known aliasing problems, when the sampling frequency used to render the shadow map from a viewpoint of the light is lower than the frequency, at which the shadow map is sampled during a rendering from a viewpoint of the eye. Many approaches have been developed for the anti-aliasing. They can be classified as the warping and partitioning methods.

Some perspective warping methods are introduced in literature [22]. The Perspective Shadow Maps (PSMs) were introduced by Stamminger and Drettakis [23]. Their method was quickly computed and reduced a shadow map aliasing. Hereinafter, the PSMs were extended as the Light-space PSMs (LiPSM) by

Wimmer et al. [24]. The LiPSM allows a treating of all lights as the directional lights and does not change the direction of the light sources. The LiPSM algorithm includes the following steps:

- Focus the shadow map on the convex body B that encloses all light rays in the view frustum, involving all objects that can cast the shadows into it.
- Enclose the body B with an appropriate perspective frustum P (P has a view vector parallel to the shadow map).
- Control the strength of the warping effect by choosing the distance n of the projection reference point p to the near plane of the frustum. If the distance n is chosen close to the near plane of P , then a perspective distortion will be strong like as in the original perspective shadow maps. If it is chosen far away from the far plane of P , then the perspective effect will be very light approaching the uniform shadow maps.
- Apply P both during shadow map generation rendering just as in standard shadow mapping.

This shadow mapping technique combines the advantages of the perspective and uniform shadow maps, providing the overall high shadow quality. The algorithm is robust, simple to implement, and fast as the standard shadow maps.

Martin and Tan [25] changed the frustum in the LiPSM on a bounding trapezoid of the camera frustum. A logarithmic parameterization closer to optimal was proposed in some researches, e.g., the Logarithmic PSMs (LogPSMs) [26], which provide significantly less error than the competing algorithms. The main disadvantage of these warping methods is their capability to generate only a single shadow map (this is insufficient in the large-scale forest).

In contrast to the warping methods, the partitioning methods approximate the ideal sample distribution through the multiple shadow maps. Zhang et al. [27] introduced the Parallel-Split Shadow Mapping (PSSM), which splits the view frustum into different depth ranges using the split planes parallel to the view plane. However, this concept leads to an extensive amount of geometry for each shadow map in every frame. The using of the multiple shadow maps is also applied in other algorithms, such as plural sunlight buffers [28], z-partitioning [29], or the Cascaded Shadow Maps (CSMs) [30]. The CSMs can be considered a discretization version of the PSMs. This algorithm proceeds as follows:

- For every light's frustum, render the scene depth from the lights point of view.
- Render the scene from the camera's point of view. Depending on the fragment's z -value, pick an appropriate shadow map to the lookup into.

Figure 9.1 shows the parallel splits in the CSM. The splits are planes parallel to the near (red lines) and far (blue lines) planes and each slice is a frustum itself.

The common major advantage of the aforementioned algorithms is that they are scene-independent. However, these approaches are limited with the local aliasing effects due to different surface orientations. The adaptive partitioning algorithms are

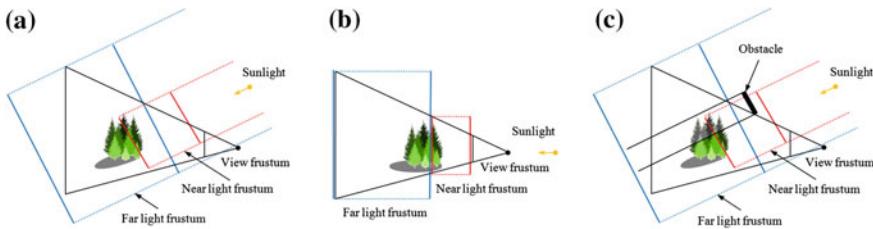


Fig. 9.1 Schemas of the CSMs implementation: **a** different locations of the sunlight and viewpoint, **b** the same locations of the sunlight and viewpoint, **c** different locations of the sunlight and viewpoint with an obstacle

recommended if higher quality is desired. For a static scene with a static light source, the temporal reprojection is a powerful method, which provides the high-quality shadows.

Trees and plants in the nearest LODs have very complex optical behavior. Much effort to simulate leaf optics in botany cannot provide efficient methods due to the complexity of the natural scenes. Franzke and Deussen [31] provided the ray tracing algorithms to render the trees in a botanically correct way. A leaf is usually modelled by four layers with different optical properties [32]. The upper and lower epidermis cover the leaf, in the interior a layer of elongated palisade parenchyma is arranged densely in parallel to the incident radiation as it is shown in Fig. 9.2.

The light passes through these cells into the spongy layer of the leaf. The elliptical cells, which are interspersed with the inter-cellular air spaces, cause the internal scattering of the incident radiation and distribute the light within the leaf. Each of the layers causes the reflection, refraction, and scattering.

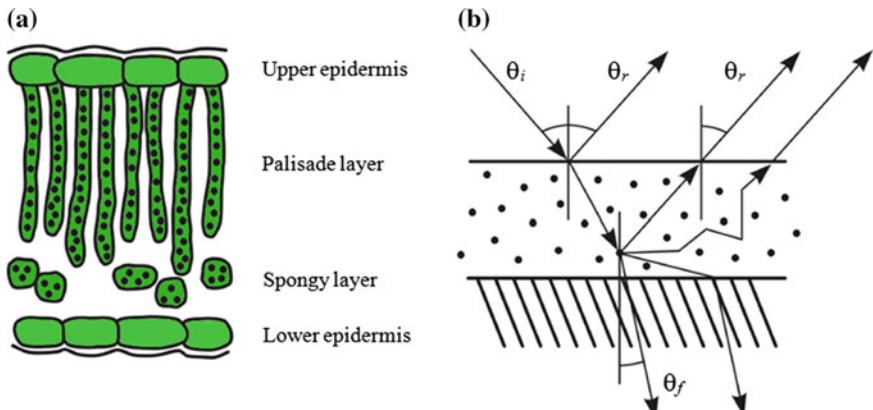


Fig. 9.2 Optical models of leaves: **a** leaf interior, **b** scattering of light in an optical layer (θ_i is the incident angle of the light ray with respect to the normal, θ_r is the reflected angle with respect to the normal, θ_f is the refracted angle with respect to the normal)

Four layers, depicting in Fig. 9.2 a, were simulated by Tucker and Garatt in one of the first stochastic models by a Markov chain [33]. Four light states are defined as solar, reflected, absorbed, and transmitted. For each transition between two layers, a transition probability from each light state to any other was given. In a simpler model the layers are defined by an absorption coefficient and a scattering coefficient. Using the Kubelka-Munk scattering theory, it is possible to simulate the general leaf properties accurately [34].

Such sophisticated simulation models cannot be supported by the real-time applications. A simplification of the light interaction with the cells using the light rays was performed by Haberland in 1914 [35]. Allen et al. were the first, who studied a light interaction with the entire leaves [36]. Their model, which consisted of circular cells embedded in air, was improved by Kumar and Silva [37], who added two more internal layers to the model. Govaerts et al. [38] implemented the ray tracing procedures that extended the models to real 3D-structures. Baranosi and Rokne [39] presented a stochastic forward ray tracing method that followed up the photons in the interior of a leaf. The model had five parameters that describe the optics of a leaf: η_c (a refractive index of the upper epidermis), ob (an oblateness of the epidermis cells), η_m (a refractive index of the mesophyll cell wall), t_m (a thickness of the mesophyll tissue), c (a concentration of the pigments), and η_a (a refractive index of the lower epidermis). The model faithfully returns the reflection and absorption of leaves but due to its stochastic nature many photons are needed to obtain the results with low noise.

9.3 Large Landscape Scene Rendering

The widely spread in computer graphic technique based on the level of details uses the paradigm of decreasing the complexity of 3D object representation at far levels from a viewpoint according to a predefined set of rules under the assumption that the reduced visual quality of the far away objects is usually not noticeable. The main benefit of the LODs application is the increasing the efficiency of a rendering due to the substantial reduction of a computational cost. Most of the researchers assume that the term LOD is referred to decreasing the geometric complexity of an object reducing its number of polygons. Generally, the LOD concept includes additionally the decreasing of shader and texture complexities. Notice that the LOD technique for textures that has been developed in recent years is well-known by a different name, the MIP-mapping.

Two types of the LOD algorithms, such as the Continuous LOD (CLOD) and the Discrete LOD (DLOD), are known. The CLOD algorithm is executed in the real-time using the high complexity objects with their following simplifying. This approach is more difficult than the DLOD algorithm because the creating process of the simplified object versions is often complicated and time consuming. The DLOD algorithm uses the previously created LOD objects and exchanges them according to the distance from the camera in the discrete intervals. Often the CLOD technique

cannot be covered as this is a complex method. In many publications, the LOD term is referred to the DLOD. For simplicity, the DLOD term will be replaced as the LOD term.

The LOD technique is based on a sub-dividing of 3D space into the finite amount of regions, when each region is assigned with a different level. The basic concept of the LOD is to provide various models to represent the same object. The LOD models can be obtained by the complicated polygon reduction techniques or created manually. Many smoothing techniques, such as the alpha blending and morphing, are used to avoid a visual popping that is appeared because of the discrete levels.

The Occlusion Culling (OC) technique, also known as the Hidden Surface Removal (HSR), determines what surfaces or parts of the surfaces are not visible from a certain viewpoint. The reliable definition of only visible objects for rendering reduces the number of draw elements and greatly increases the performance of a heavy scene. However, the trees are often characterized by a transparency for their foliage that makes them excluded from the occlusion computation. Only trunks as the opaque objects can be occluded. In this case, not only foliage area but caves, hills, cliffs, boulders, and other terrain objects can be considered in the OC technique.

There are many different algorithms used for the occlusion culling, such as the Z-buffering, binary space partitioning, or ray tracing. One can mention very fast techniques that are applied in the commercial software tools. The Potentially Visible Sets (PVSs) is an occlusion algorithm that pre-computes candidates for potentially visible polygons. Then they are indexed at run-time in order to obtain fast an estimation of the visible geometry. To determine the PVS, the camera view-space is sub-divided into (usually convex) regions, and the PVS is computed for each region. The benefit of this method is its fast run-time performance. However, it is required to store the large amount of the pre-computed data, and the main drawback is impossibility to use it for the completely dynamic scenes. The portal rendering as a sub-type of the PVS divides the scene into the cells/sectors (rooms) and portals (doors). Each cell is a sub-division of the entire bounding volume of the scene, and together they form a binary tree. The portal rendering is able to create the portals at run-time. This modification is more accurate than the general PVS but it requires more processing time.

9.4 Large Terrain Rendering

The terrain LOD techniques that were considered in Sect. 9.2.1 are based on the local terrain geometry according to a scheme, when the planar regions are assigned with the larger triangles, resulting in the irregular meshes to render. This means that the refinement criteria and/or refinement operations must be pre-computed and consume the additional memory. A changing of tessellation requires the immediate-mode rendering, while the caching static regions hinder a temporal continuity that leads to failing of a constant frame rate. Also, the surface shading

requires the texture images that are stored separately and use the entirely different LOD structure. The cornerstone is to develop the LOD framework that can optimally feed the graphics pipeline.

The MIP-mapping is a technique for obtaining of the pre-calculated and optimized sequences of the textures, when a texture in each level has a progressively lower resolution representation. The letters “MIP” are an acronym of the *Multum In Parvo* (in Latin), meaning “much in a small place”. The term MIP-map was introduced by Williams [40]. A high-resolution MIP-map image is used for the high-density objects close to the camera. The lower-resolution images are used as the object appears farther away. The MIP-mapping with the bilinear and trilinear filtering are commonly used the filtering techniques for reducing the spatial aliasing artifacts. The task of a texture filter is to determine, which texels contribute to each screen space pixel.

The MIP refers to the pre-filtering and storage of multiple texture maps with the decreasing resolution [41]. This is achieved by the progressively averaging groups of four neighboring texels to form each new layer of the image pyramid from the initial (base) level referred to as the level 0 until the final single texel level is reached. The initial texture dimensions must be a power of two magnitudes. The image pyramid and the texture storage in a framework of the MIP-mapping are depicted in Fig. 9.3.

The further explanation of the algorithms for faster calculation of the texture and the level selection for the MIP-map trilinear filtering one can find in [41].

The View Dependent Texture-Mapping (VDTM) is the efficiently implemented technique for generating of novel views in a scene with the approximately known geometry. This method was presented by Debevec et al. [42] as a rendering method of the interactively constructed 3D architectural scenes using the images taken from the multiple locations. In spite of the VDTM cannot be applied directly for a texture

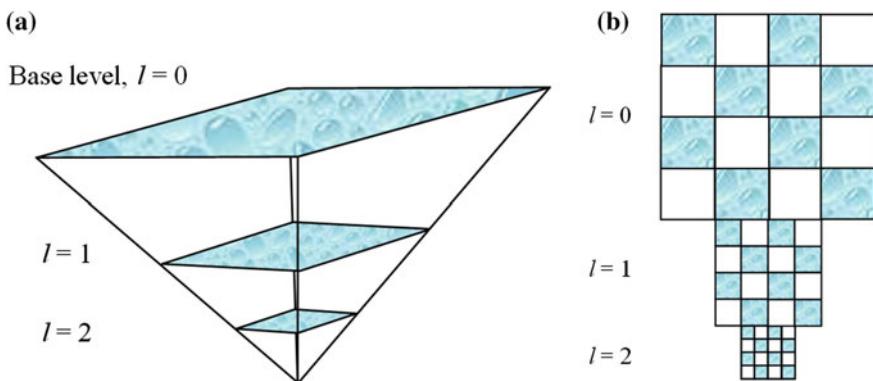


Fig. 9.3 MIP-mapping with three levels: **a** image pyramid, **b** texture storage

mapping in the forest scenes, it includes the interesting algorithms of the visibility analysis and hole filling, when some polygons become undefined during a viewpoint changing.

Debevec et al. [43] proposed to build a pyramid for the occluding polygon with the apex at the camera position and then clip the polygon with the bounding faces of the pyramid. This algorithm executes a shallow, clipping a number of rendering polygons efficiently. If a polygon A occludes a polygon B , then only a polygon A is used to clip a polygon B , and any polygons behind a polygon B are unaffected. Suppose that a polygon P has a list of occluders $O = \{p_1, p_2, \dots, p_m\}$. First, the overlapping area between a polygon P and a set O is analyzed and the polygon p_i with maximum overlapping area is determined. Second, the maximum overlapping of a polygon P is divided into two parts: a part P_O as the occluded part and a part P_S as the non-occluded part with a set of convex polygons S . The algorithm is recursively applied to each element of a set S detecting its occluders and then performing the clipping. Additionally, a lower threshold on the size of polygons is set. If a polygon's size is below the threshold value, then it is assigned a constant color based on the textures of its surrounding polygons instead of texturing. This permits to further reduce the number of the analyzing polygons.

The appearance of undefined regions is a usual challenge in computer graphics. The image-space hole-filling algorithms can cause a flickering in a scene. The object-space hole-filling algorithm can guarantee that the derived appearance of each invisible polygon is consistent between the viewpoints. After detection of the polygon connectivity, the holes are reconstructed by filling with colors close to the colors of the surrounding visible polygons.

The pioneer research of Losasso and Hoppe [4] deals with a geometry clipmap that caches the terrain in a set of the nested regular grids centered about the viewpoint. These grids represent the terrain tile at the power-of-two resolutions, and are stored as the vertex buffers in video memory. During rendering, the clipmap levels are shifted and incrementally refilled with data. The illustration of a geometry clipmap is depicted in Fig. 9.4.

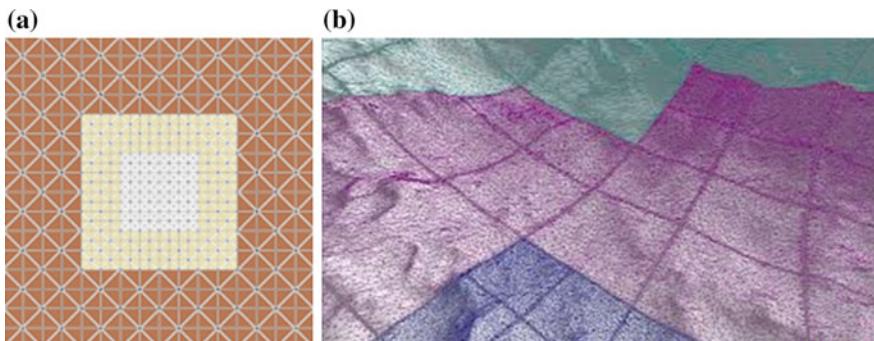


Fig. 9.4 Geometry clipmap: **a** nested regular grid, **b** terrain rendering, using the geometry clipmap technique

The geometry clipmap is referred to the MIP-map pyramid of the power-of-two grids (Fig. 9.3). The MIP-map level is a function of the screen-space parametric derivatives that depend on the view parameters and do not depend on the content of the image. The LODs in the world space are used as a set of the nested rectangular regions about the viewpoint. The transition regions ought to be smoothly blend between the levels and avoid the T-junctions by stitching the level boundaries using the zero area triangles. The advantages of the geometry clipmaps in comparison of other terrain LOD techniques are the following [4]:

- *Simplicity.* There is no irregular traversal of pointer/index-based structures. There is no tracking of refinement dependencies.
- *Optimal rendering throughput.* The clipmap vertices are fitted in video memory. Their grid structure is suitable for the indexed triangle-strip rendering with optimal vertex-cache reuse.
- *Visual continuity.* The inter-level transition regions provide a spatio-temporal continuity for both geometry and texture using a few instructions in the vertex and pixel programs, respectively.
- *Steady rendering.* The rendering rate is nearly constant since the tessellation complexity is independent of a local terrain roughness.
- *Immediate complexity throttling.* Even with a fixed clipmap size, the rendered regions can be shrunk in order to reduce a rendering load.
- *Reasonable degradation.* When a viewpoint is moving quickly, as many levels as possible are updated. The effect is such that the fast moving terrain loses its high-frequency details.
- *Surface shading.* The normal maps are computed on-the-fly from the geometry and use the same LOD structure as the geometry.
- *Compression.* Since only the clipmap needs to be expanded into the vertex data, the remainder of the terrain pyramid can be stored in the compressed form. The high data coherence allows for compression factors around 60–100.
- *Synthesis.* The simple grid structure permits on-the-fly terrain synthesis so that coarsely specified geometry can be amplified by the procedurally generated details.

The shortcomings of the geometry clipmap application deal with several main issues. The rendered mesh is more complex than in other LOD schemes but at the same time the mesh regularity is high that provides the optimal rendering performance for this worst case. Another limitation is that a terrain is assumed to have the bounded spectral density (for example, a terrain has not the tall needle-like features). Also buildings, vegetation, and other objects that populate the environment are rendered separately using other LOD techniques.

Losasso and Hoppe [4] implemented the geometry clipmaps in each clipmap level as a traditional vertex buffer. This had leaded to the significant load on the CPU to update and render of the clipmap because at that time the GPU could not

modified the vertex buffers. A year ago, Asirvatham and Hoppe proposed the implementation of the geometry clipmaps using the vertex textures [44]. Their approach involved a splitting the (x, y, z) geometry of the samples into two parts:

- The (x, y) coordinates were stored as the constant vertex data.
- The z coordinate was stored in the elevation map. For each clipmap level, a separate $n \times n$ elevation map texture was defined. These textures were updated as the clipmap levels shift with a viewpoint change.

Due to the uniform 2D grids with the regular (x, y) coordinates, 2D “footprints” both within and across resolution level, including a small set of read-only vertex and index buffers, were implemented. The z elevations of vertices were obtained by sampling the elevation map as a vertex texture. The vertex shader DirectX 9 Shader Model 3.0 was used. Such implementation permitted to perform nearly all computations on the GPU itself, thereby reducing the CPU load.

Asirvatham and Hoppe [44] simplified a clipmap rendering introducing a set of active levels $0 \dots L' - 1, L' \leq L$, where L is a total number of levels, depending on the height of the viewpoint over the underlying terrain. A level is deactivated, if a grid size is less than $2.5 h$, where h is a viewer height above the terrain. Therefore, a level is either fully updated or declared inactive in order to avoid the aliasing artifacts caused by the unnecessarily dense of the finest clipmap levels and to reduce a computational time. The view frustum culling is done at the block level on the CPU. Each block is extruded by $[z_{\min}, z_{\max}]$ and intersected with a view frustum in 3D space. It is rendered only if this intersection is non-empty. Depending on a view direction, the rendering load is reduced by a factor of 2–3 for a 90° field of view.

Consider some shader implementations proposed by Asirvatham and Hoppe. For the given footprint (x, y) coordinates, the vertex shader computes the world (x, y) coordinates using a simple scaling and translation (the vertices correspond to the texture samples). The z value is extracted from the elevation map stored in the vertex texture. For smooth transitions, the vertex shader blends the outer boundary of level with the next-coarser level. The blending parameter α is computed based on the (x, y) location relative to the continuous viewer position (v_x, v_y) using Eq. 9.1, where $[0 \dots n - 1]$ is a range of the clipmap grid, $w, w = n/10$, is a transition width.

$$\alpha = \max \left(\text{clamp} \left(\left(|x - v_x| - \left(\frac{n-1}{2} - w - 1 \right) \right) / w, 0, 1 \right), \text{clamp} \left(\left(|y - v_y| - \left(\frac{n-1}{2} - w - 1 \right) \right) / w, 0, 1 \right) \right) \quad (9.1)$$

(In computer graphics, a clamping is the process of limiting a position to an area. Unlike wrapping, a clamping merely moves the point to the nearest available value.)

For geometric blending, an interpolation of z coordinate as z' is executed as a linear interpolation between the current (fine-level) elevation z_f and the following (next-coarser level) elevation z_c at the same (x, y) location using Eq. 9.2.

$$z' = (1 - \alpha)z_f + \alpha z_c \quad (9.2)$$

Asirvatham and Hoppe [44] computed z_c as a part of the clipmap update and pack both values z_f and z_c into the same one-channel floating-point texture in the pixel shader. The pixel shader accesses the normal map and shades the surface. The normal map is incrementally computed from the geometry, whenever the clipmap is updated. For the smooth shading transitions, the shader looks up the normals for the current level N_x, N_y and the next-coarser level N_{cx}, N_{cy} and blends them using the α value computed in the vertex shader assuming that $N_z = 1$ and $N_{cz} = 1$. As the viewer moves through the environment coherently, each clipmap window translates within its pyramid level centered about the viewer. Generally, a small region needs only to be incrementally processed each frame. The active clipmap levels are updated in coarse-to-fine order using a pixel shader. The finer-level geometry is predicted from the coarser level using the well-known four-point sub-division curve interpolant with the Kobbelt's mask weights $(-1/16, 9/16, 9/16, -1/16)$ [45]. The shader that updates the normal map for a level takes as the input the elevation map at the same level and computes the current normal as the cross product of two grid-aligned tangent vectors.

Instead of relying on the vertices, the heightmaps can be rendered directly using the fragment shader. Dick et al. [46, 47] proposed the Axis-Aligned Bounding Boxes (AABB) method instead of the surface patches of the HLOD to represent the height map tiles. Each AABB is rendered using a ray matching, i.e., a ray-casting in the incremental steps from the camera through the bounding box. This approach uses almost no geometry, this means that the CPU processing is minimum. However, the analytical solution is absent and it is impossible to calculate dynamically the step size. To counteract this, Dick et al. propose the use of a maximum MIP-map pyramid.

Most popular terrain rendering algorithms deal with the planar terrain, while another type—the spherical terrain is studied for both planets and planar terrain representation. The rendering algorithm for the spherical terrains was developed by Clasen and Hege [48]. They replaced the planar geometry with a geometry that maps on the sphere. This leads to the spherical geometry clipmap, using the concentric rings instead of the rectangles. The plane coordinates (x, y) cannot be transferred directly to the sphere. A point p with coordinates (x, y, z) on the unit sphere can be parametrized by its angle θ to the OZ axis, and the angle ϕ from p projected on the XOY plane to the OX axis. Thus, the spherical coordinates (ϕ, θ) are belong to interval $(\phi, \theta) \in [0, 2\pi] \times [0, \pi]$. The points with coordinates $(0, 0, 1)$ and $(0, 0, -1)$ can be denoted as the north pole and the south pole, respectively. All points with the same angle ϕ belong to a meridian; the zero-meridian intersects the positive OX axis. During a mapping, the world space (x, y, z) provides an absolute orientation of the spherical terrain and the view space

$(\tilde{x}, \tilde{y}, \tilde{z})$, which locates the viewer at the north pole. It is assumed that the view space is described by a static geometry, when all coordinates are calculated and transferred to the GPU only once. The hemisphere around the viewer is parametrized by $(\tilde{\phi}, \tilde{\theta})$ whereas the terrain is parameterized by (ϕ, θ) . The mapping between both spaces depends on the position of the viewer (in the world space), (ϕ_v, θ_{view}) . These authors assumed (without loss of generality) that the viewer can located exactly above the 0-meridian at $(0, \theta_{view})$ since any deviation in ϕ_{view} translates directly to a simple ϕ -offset in the height map. Thus, the following mapping is searched:

$$f(\theta_{view}, \tilde{\phi}, \tilde{\theta}) \rightarrow (\phi, \theta) \quad (9.3)$$

The mapping provided by Eq. 9.3 is a rotation around the OY axis as the zero-meridian intersected with the positive OX axis was chosen. The third space is a texture space. A map of the height texture parameterized by (u, v) is transformed to the spherical surface: $(u, v) = (\tilde{\phi}, \tilde{\theta})$. The hemisphere $(\tilde{\phi}, \tilde{\theta})$ is discretized into quads in such manner that angle $\tilde{\phi}$ is divided into n fixed steps, while the discretization of angle $\tilde{\theta}$ depends on the distance to the viewer. The first level of discretization divides the hemisphere into the concentric rings that shrink exponentially. The level i covers elements with $\tilde{\theta} \in (2^{-i}\pi, 2^{-i-1}\pi)$. This sequence is terminated by a fill level, which covers $\tilde{\theta} \in (2^{-i-1}\pi, 0)$. Then each level is subdivided into m rings by $\tilde{\theta}_{i,j} = \theta_{i,0} \cdot 2^{-j/m}$. Each discrete element of the hemisphere is then partitioned into two triangles. The resulting geometry ensures that the triangles have about the same size in the screen space.

Not all circular rings of the hemisphere are visible from each position. Also a visible surface is restricted by a view frustum determined from a viewpoint location, especially the height. A viewpoint height with the high value provides the low detailed Earth's curvature, while a viewpoint height with low value covers the high detailed local surface. The difference between the LODs of geometry clipmaps and spherical geometry clipmaps is depicted in Fig. 9.5.

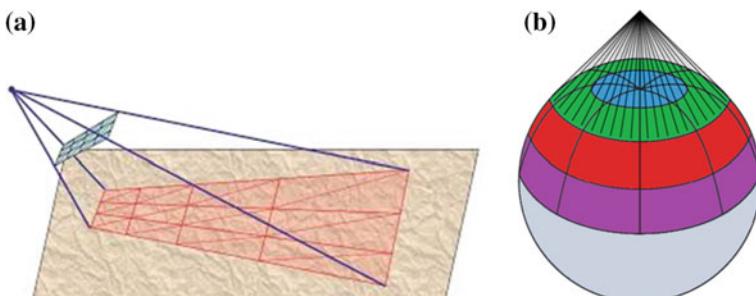


Fig. 9.5 Frustum location: **a** geometry clipmap, **b** spherical geometry clipmap

Notice that the mapping scheme for the planet approximation may be various, for example, the cylindrical projection map measured in longitude and latitude coordinates or (ϕ, θ) coordinates to the respective x and y dimensions of an image, the cubemaps mapping imagery to a cube, the tessellated octahedral adaptive sub-division transform with the aim to pack the spherical data into the square images, among others.

A clipmap storage as a stack texels and pyramid texels is very efficient. Consider how a clipmap is update in cache according to technique that was proposed by Tanner et al. [3]. A necessity to update a cache emerges, when a viewpoint is changing. At initial stage, a set of 2D images for each LODs is built and loaded into the texture memory. Due to the significant frame-to-frame coherence in cache contents, the cached texels in the subsequent frames can be reused. The complete MIP-map can be done using roam (three steps) or toroidal addressing (four steps) as it is shown in Fig. 9.6. The process begins with 2D image centered in the cache. Then new clip center is specified (image translation), and new regions appear (in the current case, there are three regions—top, right, and corner). Using the toroidal addressing, new data at the top of the image is loaded at the bottom, data on the right is loaded at the left, and the upper right corner is loaded at the lower left. The complete multi-level update process is visualized in Fig. 9.6b and d for the roaming stack and toroidal update, respectively.

Feldmann et al. [49] proposed the flexible clipmap that can incrementally generate the large virtual texture of dynamically changing content and the growing extent. This technique uses the tile-based clipmap approach and common spatial indexing data structure with their implementation on the GPU. Also, it is independent of the underlying geometry data.

The clipmap structure proposed by Tanner et al. [3] is based on a two-level cache hierarchy, adapting to the hardware and improving the support of dynamic textures that are continuously updated as a function of time. The first cache level is the texture sub-set stored in the Transputer Random Access Memory (TRAM),

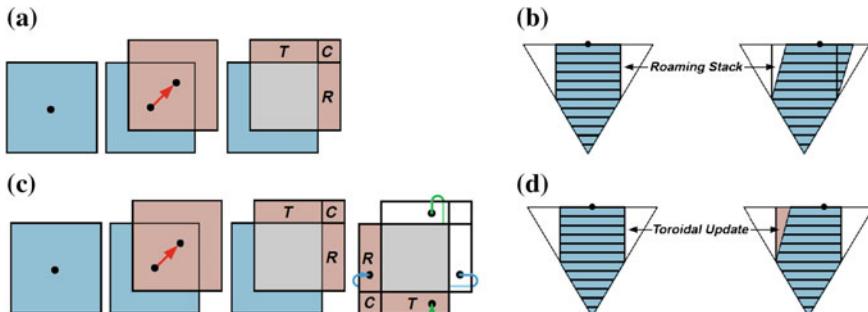


Fig. 9.6 Update clipmap (blue color is an old image, brown color is new image, gray color is the same image, black point is the center of 2D image, T is a top, R is a right, C is a corner): **a** 2D image with roam addressing, **b** 2D image roam of complete stack, **c** 2D image with toroidal addressing, **d** 2D image toroidal of complete stack

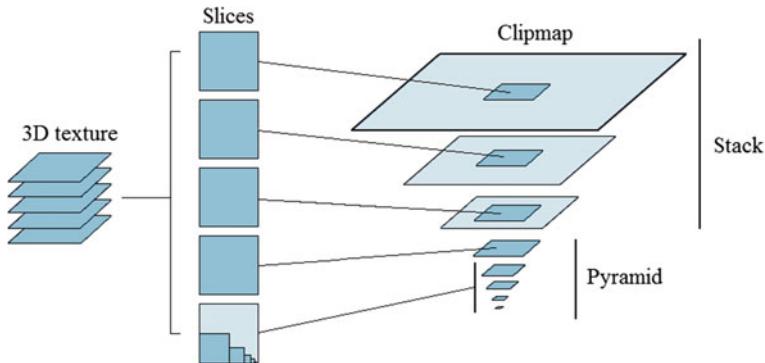


Fig. 9.7 Clipmap structure on 3D texture

while the second level is stored in the RAM. Each cache level has an associated component, whose task is to load or generate the contents stored in this cache. Following the clipmap structure, the pyramid (complete levels) as well as the stack (incomplete levels) is cached on the TRAM. The toroidal updating and addressing of the stack levels has two main advantages. Each stack level on its own texture slice keeps the continuity; thus a bilinear filtering can be automatically done by hardware without noticeable overhead. Also this continuity simplifies a texture addressing. The storage of the clipmap structure in 3D texture is depicted in Fig. 9.7.

A new approach to high resolution and a real-time texturing of the dynamic data based on the clipmap structure was developed by Taibo et al. [50]. However, they had focused on the terrain texture applications, where details are located around a unique area, such as the planetary sized dynamic textures with the sub-metric resolution.

9.5 Vegetation Rendering

The rendering of vegetation covered the landscapes is especially complex task because of the huge amount of details, complex lighting conditions, and complex visibility parameters. The species of trees and their realistic representations under various lighting conditions provide factually the infinite set of details. The natural hierarchical structures of the trees and plants and the LOD techniques allow to simplify a modelling by creation and storage a collection on similar and simpler few base models.

Some modifications of the LOD rendering techniques improve the property of a performance. Chen et al. [51] proposed the LOD-sprite rendering technique, when the objects are modelled as both the LOD models and sprites. Each 2D sprite is coupled with the LOD representation, and the renderer utilizes both the LOD and

the sprite as the inputs to create the output image. The main idea of the LOD-sprite technique is to cache the rendered view of a high-resolution representation of the dataset and to form a sprite. The frame, where the sprite was created, is called a keyframe. The LOD-sprite renders the subsequent frames referred to as novel views with a lower resolution. These novel frames are applied as the texture maps. The LOD-sprite measures the error caused by the divergence of a viewpoint from the keyframe as each novel view is rendered. When this error exceeds a threshold value, the LOD-sprite renders new keyframe. Such technique is repeated in a cycle.

The efficient rendering of the trees including the shading and shadows ought to provide the real-time rendering of the dense forests with good image quality during walkthrough and flythrough of the natural landscapes. Some famous investigations were generalized by Meyer et al. [52] in order to receive the high quality forest scenes. Four main aspects, such as the Bidirectional Texture Functions (BTFs) [53], the hierarchical IBR [54], the billboards, and the lightfields, were taken into consideration. Based on the LOD technique, an object type is encoded either the collection of smaller objects with the standard geometry or the large object depending on the level. The BTFs in [53] were pre-computed using the multiple local pre-renderings with the six degrees of freedom: the position on texture, view direction, and light direction. Meyer et al. sampled the pre-determined directions using only 3 of the 4th degrees of freedom in such manner that the BTFs produced images of an object, when the observer and light locations vary. For lightfield and the Bidirectional Reflection Distribution Functions (BRDF) representations, 4D direction space was applied. The billboards were used for representation of 3D objects with textures that is adapted to the point of a view. The Hierarchical Bidirectional Texture (HBT) structure extended the existing hierarchical scene graph of this object with the level of detail information.

The auxiliary representation had been introduced by an acceleration of the shadow modelling inspired by the horizon maps [55] and visibility pre-computed for the cells [56]. The structure for shadows was built using the hierarchy of visibility of the cube-maps. During rendering, an image for the given view and light directions was reconstructed by interpolating between the closest pre-computed billboards. Either 6, 18, 66, or 258 samples for the view directions, which were evenly distributed on a sphere, were used. The different types of the billboards are depicted in Fig. 9.8.

Instead of the standard volume rendering techniques, Behrendt et al. [21] proposed to use the slices (representing a plant layer) that are offset respect to the terrain surface. Notice that the slice geometry can be generated on the fly from the Earth's geometry. These data are stored in the GPU and can be used, while a viewpoint remains at its position. This approach assumed that the y-components of the sample coordinates have fixed values. Thus, 3D texture can internally be represented as a set of 2D textures. As it is known, 2D textures are more efficient for the GPU processing and more cache efficient than 3D textures. The main difficulty of this approach is that with a small incidence angle very few samples could lie on a viewing ray (in the extreme case the viewing direction is parallel to the slice

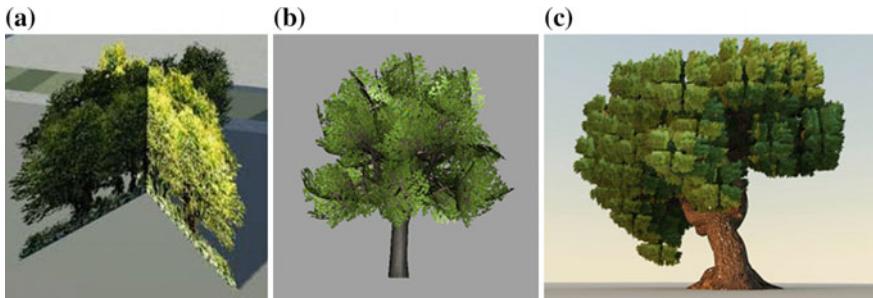


Fig. 9.8 Examples of billboards: **a** billboards of the whole tree, **b** flat billboards, **c** volume billboards (model was generated by Strom using 3D-Coat generator and rendered in LightWave 9.5)

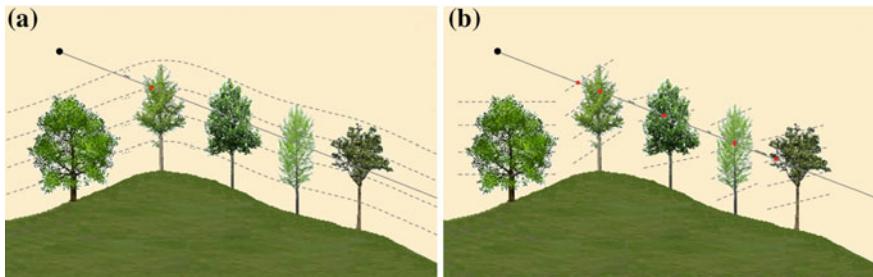


Fig. 9.9 Viewing slices with small incidence angle (*black dots* are the viewpoints, *red dots* are the intersections between slices and a view line): **a** traditional display with low quality, **b** slanting of the slices

geometry). Behrendt et al. [21] investigated a simple way to solve this problem. They slant the triangles of the slices towards the viewer as it is shown in Fig. 9.9.

Practically anyone landscape scene cannot be represented without a grass. A hybrid method as a combination of the geometry-based and volume-based approaches (the volume data are defined using the BTFs) was proposed by Boulanger et al. [57]. The direct rendering of the geometric blades of the grass with lighting is impossible in the real-time application. Additionally, the system ought to integrate the dynamic illumination and good parallax effect in the real-time applications. According to these requirements, a rendering of very large surfaces, containing grass, is based on the LODs paradigm. If the grass is close to the camera, then the best rendering quality is performed using the lit and shadowed geometry. When the grass is farther, a volume rendering, using the semi-transparent axis-aligned slices, is applied. A palette of the grass and the Earth's texture and the generated LODs of the grass (three levels with a final view) are depicted in Fig. 9.10.

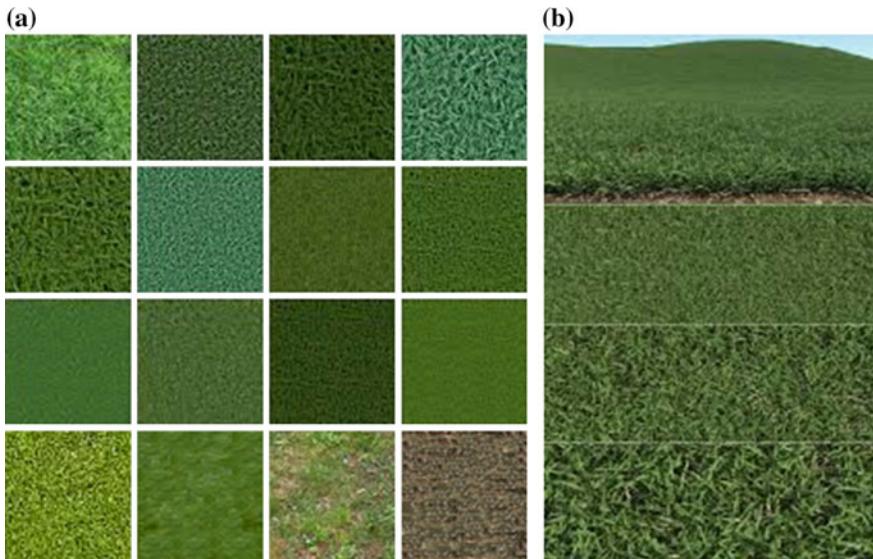


Fig. 9.10 Grass representation: **a** palette of a grass, **b** the LODs of a grass

All parameters of every blade of a grass in the terrain, laying on the cells of an uniform grid, cannot be defined. A technique of the grass patches over the ground surface was implemented by Boulanger et al. [57] as a set of the geometric grass blades distributed inside a rectangle and a set of axis-aligned slices, using the semi-transparent textures. The slices provides a good parallax effect, when the grass patches do not look flat seen from any direction. Notice that the slice textures use the semi-transparent BTFs, not 2D classic billboards.

Generally, the BTF is a 6-dimensional discrete function, defining the reflectance at each point of a surface for any incidence and reflection direction. A simulation of the ground surface makes enough 5-dimensional BTF because the camera and the light source cannot be under the ground. Due to the zero thickness of the grass blades, the slices are invisible from the remaining 3 directions. The remaining 2-dimensional BTF provides the real-time implementation. The macro-structures on a surface represent the small variations of a height with simulation of the self-shadowing and self-occlusions. Due to far distance, a surface of the grass looks flat, thus only the slices parallel to the ground are kept.

One of the main shortcomings of the LOD technique lies in the seamless transition management. For each grass blade processed at rendering time, the local density taken from the pre-defined density map is multiplied by the weight function corresponding to the current rendering method with the following comparison to the blade density threshold. Boulanger et al. elaborated several types of density maps, including the maps for geometry, vertical slices, and horizontal slices, and obtained very realistic results.

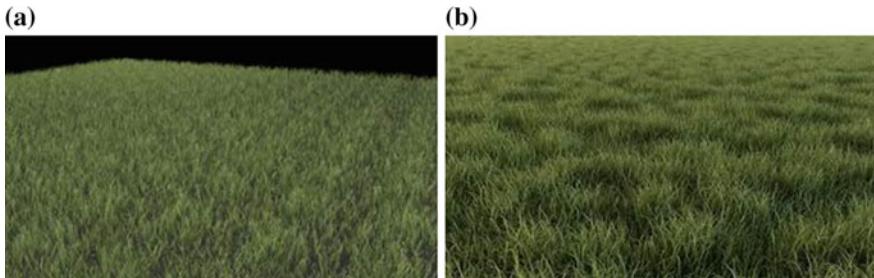


Fig. 9.11 Examples of grass tiling: **a** periodic, **b** aperiodic

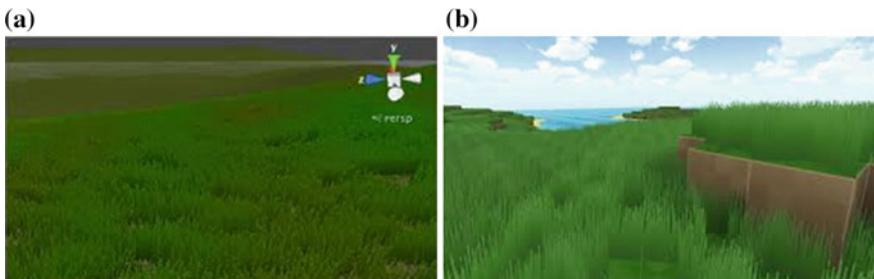


Fig. 9.12 Examples of grass modelling in the unity 3D

Repeating the same patch of the grass many times over the whole terrain generates a distracting visual pattern. Therefore, a simple aperiodic tiling scheme with the random patches provides better results. The illustration of this effect is depicted in Fig. 9.11. The results of grass modelling in package Unity 3D are situated in Fig. 9.12.

Nevertheless, a rendering of grass blades without natural effects like smoke or drops in close distance cannot provide the highest degree of visibility. For comparison, the aesthetic examples of grass and plants are shown in Fig. 9.13.

The BTF is a function, depending on the light and view directions. The sampled BTF data is interpolated according to the incident light direction (Sect. 9.6). For a view direction, the interpolation is not useful since only 2-dimensional BTFs are defined.

9.6 Realistic Lighting

During a daytime, the sunlight passes through the atmosphere and forms a distributed skylight. The sunlight has its solid angle and causes the umbrae and penumbras. In fine weather, a direct sunlight plays an important role in displaying of bright green and red foliage. Inter-reflections among the leaves are comparatively



Fig. 9.13 Aesthetic examples of the realistic plants and grass

intensive because they are semi-transparent and close to each other. In some time period, the skylight serves as the main light source, for example, on a cloudy day or at sunrise and sunset. Generally, a forest structure is complicated due to the randomness and occluding of trees. It is enough difficult to estimate the skylight illumination. All these phenomena make a rendering of the forests a challenging task. However, the shapes and sizes of the trees species growing close together are fairly similar that provides a possibility to create the realistic landscape scenes.

The Lambert's cosine law states that the luminance E , falling on any surface, depends on the cosine of the light's angle of an incidence, θ (see Fig. 9.14). The angle θ is measured from a line normal to the surface. A luminance on the surface E_θ is computed by Eq. 9.4.

$$E_\theta = E \cdot \cos \theta \quad (9.4)$$

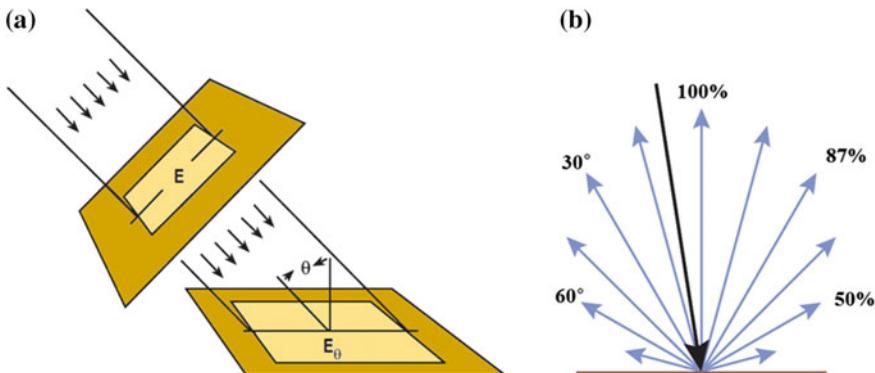


Fig. 9.14 Emission and reflection: **a** the Lambert's cosine law, **b** the Lambertian surface

The Lambertian surface reflects or emits an isotropic luminance in every direction. For example, an evenly illuminated diffuse flat surface is approximately Lambertian because the reflected light is the same in every direction. However, it does not have an isotropic intensity because the intensity varies according to the cosine law. Figure 9.14b shows the Lambertian reflection from a surface—the amount of reflected energy in a particular direction (the intensity) is proportional to the cosine of the reflected angle. The luminance is the intensity per a unit area. Because both intensity and apparent area follow the cosine law, they remain in proportion to each other as the viewing angle changes. Therefore, the luminance remains constant, while the luminous intensity does not.

Consider the rendering of the semi-transparent leaves. A lot of papers were devoted to the general problem—a subsurface scattering. First model was introduced by Hanrahan and Kruger [58], who computed the scattering in a homogenous material and derived an analytic expression for a single scattering. Pharr and Hanrahan [59] investigated several scattering functions to model a scattering. The direct rendering based on these models is very time consuming, especially if the highly scattering materials are used. Jensen et al. [60] propose an efficient method for a sub-surface scattering. The scattering process is separated in a single scattering term $L^{(1)}$ and a diffusion approximation term L_d . Franzke and Deussen [31] used three layers in a leaf model: the upper and lower epidermis layers and scattering layer for the scattering interior. They proposed a ray tracing procedure for rendering the leaf considering the leaf geometry, leaf texture, and alpha channel in two states: the opaque and fully transparent.

Wang et al. [61] presented a framework for the real-time rendering of the plant leaves with the global illumination effects. This model described a leaf appearance in terms of a few parametric Bidirectional Reflectance Distribution Functions (BRDFs) and the Bidirectional Transmittance Distribution Functions (BTDFs). The reflectance and transmittance properties of two-layered model, describing the upper and lower surfaces, were simulated using a pre-computed radiance transfer approach to all-frequency lighting of leaves. The excellent illumination effects were achieved due to the combined low-frequency environment light and high-frequency sunlight. The local incident radiance of a sunlight was decomposed into the direct and indirect components. The designed by Wang et al. two-pass algorithm with a sunlight is applicable to the rendering of other types of scenes, e.g., illuminated by an environment map or by several small area light sources. Also, there is no restriction on the shapes of the area light sources.

In general, the dense forests have a low albedo (albedo is Latin, meaning whiteness). This means that the majority of the ultraviolet and visible spectrum is absorbed through photosynthesis [62]. Thus, the Shell's law and the Lambert's law may be inadequate in the nearest LODs. The large terrain surfaces, which properties are spatially uniform over the large enough scales and the individual surface elements can be treated as planar. In this case, the reflection is described by the Fresnel's laws. The lighting rendering of the forests strongly depends on the climate, season, and the most species of trees. In the realistic models, an atmosphere

effects ought to be considered. However, this factor is usually rejected in the computer graphics' applications.

Consider a lighting model of a grass, representing in [57]. According to the Lambert model for a single omni-directional light source applied to a point of a surface, the rendered pixel intensity I is described by Eq. 9.5, where K_d is a diffuse reflectance factor of the material, I_a is an intensity of the ambient light, I_d is an intensity of the point light source, d is a distance between the surface point and the light source, β is an attenuation factor, \bar{N} is a normal to the surface, \bar{L} is a light direction from the surface point.

$$I = I_{ambient} + I_{diffuse} = K_d I_a + K_d \max(\bar{N} \cdot \bar{L}_i, 0) \frac{I_d}{1 + \beta d^2} \quad (9.5)$$

The 1 in the denominator avoids an infinite intensity, when being very close to the light source. Only 5 light directions are available, such as $X+$, $X-$, $Y+$, $Z+$, and $Z-$. The interpolation of the BTF values is required; otherwise the sudden changes of color during the light source movement are possible. The diffuse part is computed by combining the images of the directions L_i , $i \in \{1 \dots 5\}$, as it is shown in Eq. 9.6, where $\{a_i\}$ are the linear interpolation coefficients.

$$I = K_d I_a + \left(\sum_{i=1}^5 a_i K_d \max(\bar{N} \cdot \bar{L}_i, 0) \right) \frac{I_d}{1 + \beta d^2} = K_d I_a + \left(\sum_{i=1}^5 a_i C_i \right) \frac{I_d}{1 + \beta d^2} \quad (9.6)$$

The K_d term and the $K_d \max(\bar{N} \cdot \bar{L}_i, 0)$ term called as C_i are taken from the BTFs. The both terms are used for the diffuse component computation. For a directional light source (e.g., the Sun), the coefficients are computed once for the whole grass surface. Lighting computation with a random light source position is very expensive due to a point light source requires a computation, using Eqs. 9.5 and 9.6, per each pixel.

9.7 Shaders

Consider the implementation of the shaders on the example of the Unity 3D 3.x package [63]. A scheme of rendering workflow of the Unity3D is depicted in Fig. 9.15.

The shading languages were developed for various libraries, utilities, and environment. Usually the shading languages have an open-access interface, and the end-user can write the own shaders. A list of the popular shading languages is mentioned below (notice that not all abbreviations can be encrypted):

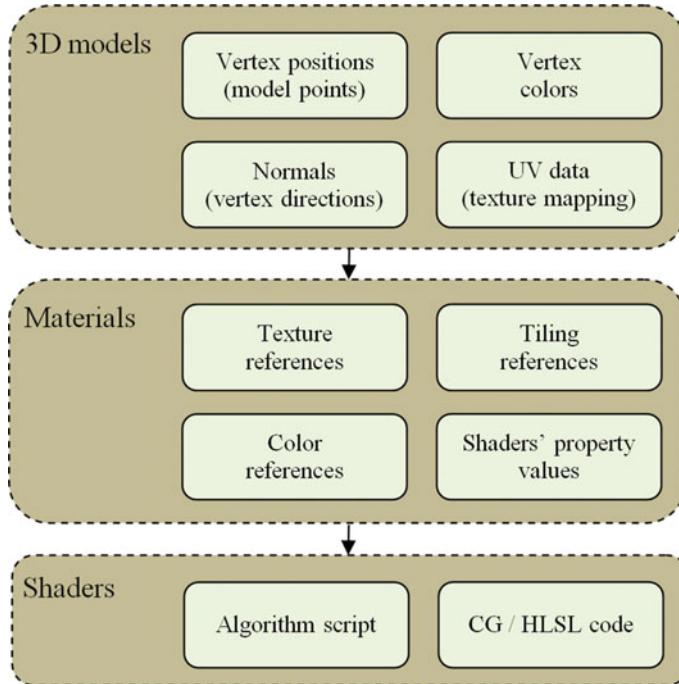


Fig. 9.15 Scheme of rendering workflow of the Unity3D (UV data means data of texture mapping with “U” and “V” axes of 2D texture into 3D object with “X”, “Y”, “Z” axes in 3D model space)

- OpenGL Shading Language (the GLSL or the GLslang).
- High-Level Shading Language for use with the Direct3D (HLSL).
- High-level shading language for programming the vertex and pixel shaders (Cg).
- Low-level shading language, the ARB assembly language.
- Low-level intermediate language introduced by the Gallium3D (TGSI).
- Low-level intermediate language used internally at the AMD, the AMD Intermediate Language (AMDIL).

The 3D models are a collection of the vertices representing by 3D coordinates. Usually they are connected together as the triangles. Each vertex has additional information, such as the color, direction to camera viewpoint (normal), and coordinates of the texture mapping (UV data). The models cannot be rendered without a material. The materials are wrappers, they contain the initial parameters and values of the shaders' properties. In the Unity 3D package, the materials, shaders, and textures are connected closely. The materials define of how a surface should be rendered including the texture, tiling, and color references. The available options for a material depend on that shader is used. The shaders are small scripts that

contain the mathematical calculations and algorithms. The textures are the bitmap images, representing many types of the material surfaces with the reflectivity, the roughness, and other properties.

One material specifies one specific shader to use, and the shader used determines, which options are available in the material. One shader specifies one or more textures variables that it expects to use. For a normal rendering, the standard shader is usually the best choice that is capable of rendering many types of surface in a highly realistic way. In other situations, the different built-in shader (the custom written shader) might be appropriate for illustration of the artistic effects, such as liquids, foliage, refractive glass, particle effects, or others, or special effects like night vision, heat vision, x-ray vision, among others.

Every time a model is drawn to the screen, the Central Processing Unit (CPU) requests the draw calls to the Graphics Processing Unit (GPU) due to one call is a call per a material within a model. Moreover, if a call is a draw call per a light pass or a dynamic reflection, then at least twice potentially doubling draw calls are required. The Unity 3D tries to optimize a number of calls, for example, it combines the models, using the same material, so that they would be drawn with just one draw call. Also a number of draw calls can be reused if the same materials are applied as often as it is possible.

The trees are the objects with different textures due to the tree species. If it is known that some tree species are located closely one another, then their textures can be combined into a texture atlas. (The texture atlas is a large texture made from the multiple textures.) This allows the Unity 3D to extract a material batch of such objects into one draw call. Sometimes the problem to scene performance has nothing to do with graphics. Possibly, there are too many complex operations in the application and the performance is not limited by the GPU but the CPU instead. To control this situation, a profiler can be built that can display the exact processing times of all methods in the application. The reasons may be the following: too much time to calculate the physics, too many vertices to send to the GPU, or anything else. The images are used as the textures for rendering foliage both for the billboards and polygonal plants. The Unity 3D supports all of the most often used formats, such as the PhotoShop Data (PSD) file, the Tagged Image File (TIFF), the Joint Photographic Experts Group (JPEG), the Portable Network Graphics (PNG), the Graphics Interchange Format (GIF), the BitMaP (BMP) file, the Truevision Targa Graphic (TGA) (the TGA is called sometimes a texture file), the Interchange File Format (IFF), and the Macintosh PICTure (PICT) file format. For the billboards and texturing leaves, the transparent textures are required. The image formats that support an alpha channel are the PSD, the TIFF, the PNG, the TGA, and the BMP. However, only the PSD, the TIFF, and the PNG formats are truly usable for saving and using the transparent images within the Unity 3D.

Foliage, whether the plant is geometrically modelled or just a billboard, cannot be rendered to a screen without the assigned material. The material is an instance of a certain shader with the required textures and parameters. The Unity 3D includes many shaders usable for rendering foliage that makes difficult at times to choose the right shader. Some properties of the shaders are situated in Table 9.1 [64].

Table 9.1 Properties of the Unity 3D shaders

Shader group	Shader title	Short description
Normal Shader Family	Vertex-Lit	All lights shining on it are rendered in a single pass and calculated at vertices only. Generally, this shader is very cheap to render
	Diffuse	Diffuse computes a simple (Lambertian) lighting model, when the lighting depends only on this angle and does not change as the camera moves or rotates around. Generally, this shader is cheap to render
	Specular	Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight that is dependent on surface angle, light angle, and viewing angle. Additionally, the alpha channel of the main texture acts as a specular map defining, which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
	Bumped diffuse	Diffuse computes a simple (Lambertian) lighting model. The lighting on the surface decreases as the angle between it and the light decreases. The lighting depends only on this angle and does not change as the camera moves or rotates around. Generally, this shader is cheap to render
	Bumped specular	Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Generally, this shader is moderately expensive to render
	Parallax diffuse	The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is on the more expensive rendering side
	Parallax bumped specular	Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Generally, this shader is on the more expensive rendering side

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
	Decal	This shader is a variation of the vertex-lit shader. All lights that shine on it will be rendered as vertex lights by this shader. In addition to the main texture, this shader makes use of a second texture for additional details. The second “Decal” texture uses an alpha channel to determine visible areas of the main texture. This shader is approximately equivalent to the vertex-lit shader. It is marginally more expensive due to the second decal texture but will not have a noticeable impact
	Diffuse detail	This shader is a version of the regular Diffuse shader with additional data. It allows to define a second “Detail” texture that will gradually appear as the camera gets closer to it. It can be used on terrain. The detail texture is put “on top” of the base texture. Darker colors in the detail texture will darken the main texture and lighter colors will brighten it. This shader is pixel-lit, and approximately equivalent to the diffuse shader. It is marginally more expensive due to additional texture
Transparent shader family	Transparent vertex-lit	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. In the alpha, 0 (black) is completely transparent, while 255 (white) is completely opaque. This shader is vertex-lit and this makes it one of the simplest shaders. All lights shining on it are rendered in a single pass and calculated at vertices only. Generally, this shader is very cheap to render
	Transparent diffuse	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Transparent Specular	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Generally, this shader is moderately expensive to render
	Transparent bumped diffuse	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
	Transparent bumped specular	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Generally, this shader is moderately expensive to render
	Transparent parallax diffuse	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is on the more expensive rendering side
	Transparent parallax specular	This shader can make mesh geometry partially or fully transparent by reading the alpha channel of the main texture. Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Generally, this shader is on the more expensive rendering side
Transparent cutout shader family	Transparent cutout vertex-lit	Cutout shader is an alternative way of displaying transparent objects due to the following: (i) this shader cannot have partially transparent areas (everything will be either fully opaque or fully transparent), (ii) the objects using this shader can cast and receive shadows, (iii) the graphical sorting problems normally associated with Transparent shaders do not occur, while this shader is used. This shader uses an alpha channel contained in the base texture to determine the transparent areas. This shader is vertex-lit and this makes it one of the simplest shaders. All lights shining on it are rendered in a single pass and calculated at vertices only. Generally, this shader is very cheap to render

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
	Transparent cutout diffuse	Cutout shader is an alternative way of displaying transparent objects. This shader uses an alpha channel contained in the base texture to determine the transparent areas. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Transparent cutout specular	Cutout shader is an alternative way of displaying transparent objects. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
	Transparent cutout bumped diffuse	Cutout shader is an alternative way of displaying transparent objects. This shader uses an alpha channel contained in the base texture to determine the transparent areas. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Transparent cutout bumped specular	Cutout shader is an alternative way of displaying transparent objects. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. The normal map is a tangent space type of normal map. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
Self-illuminated shader family	Self-illuminated vertex-lit	This shader allows to define the bright and dark parts of the object. The alpha channel of a secondary texture will define areas of the object that “emit” light by themselves, even when no light is shining on it. In the alpha channel, black is zero light, and white is full light emitted by the object. Any scene lights will add illumination on top of the shader’s illumination. This shader is vertex-lit and this makes it one of the simplest shaders. Generally, this shader is cheap to render
	Self-illuminated diffuse	This shader allows to define the bright and dark parts of the object. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
	Self-illuminated specular	This shader allows to define the bright and dark parts of the object. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
	Self-illuminated normal mapped diffuse	This shader allows to define the bright and dark parts of the object. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. The normal map is a tangent space type of normal map. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Self-illuminated normal mapped specular	This shader allows to define the bright and dark parts of the object. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. The Normal Map is a tangent space type of normal map. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
	Self-illuminated parallax diffuse	This shader allows to define the bright and dark parts of the object. Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is on the more expensive rendering side
	Self-illuminated parallax specular	This shader allows to define the bright and dark parts of the object. Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
		others. Generally, this shader is on the more expensive rendering side
Reflective shader family	Reflective vertex-lit	This shader will simulate reflective surfaces, such as cars, metal objects etc. It requires an environment Cubemap, which will define what exactly is reflected. The main texture's alpha channel defines the strength of reflection on the object's surface. Any scene lights will add illumination on top of what is reflected. This shader is vertex-lit and this makes it one of the simplest shaders. All lights shining on it are rendered in a single pass and calculated at vertices only. Generally, this shader is not too expensive to render
	Reflective diffuse	This shader will simulate reflective surfaces. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Reflective specular	One consideration for this shader is that the Base texture's alpha channel will double as both the reflection map and the specular map. This shader will simulate reflective surfaces. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render
	Reflective bumped diffuse	This shader will simulate reflective surfaces. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is cheap to render
	Reflective bumped specular	One consideration for this shader is that the base texture's alpha channel will double as both the reflection map and the specular map. This shader will simulate reflective surfaces. Like a Diffuse shader, it computes a simple (Lambertian) lighting model. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map, defining which areas of the object are more reflective than others. Generally, this shader is moderately expensive to render

(continued)

Table 9.1 (continued)

Shader group	Shader title	Short description
	Reflective parallax diffuse	This shader will simulate reflective surfaces. Parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Diffuse computes a simple (Lambertian) lighting model. Generally, this shader is on the more expensive rendering side
	Reflective parallax specular	One consideration for this shader is that the base texture’s alpha channel will double as both the reflection map and the specular map. This shader will simulate reflective surfaces. parallax normal mapped is the same as regular normal mapped but with a better simulation of “depth”. The parallax mapping technique is pretty simple, so it can have artifacts and unusual effects. Specifically, very steep height transitions in the height map should be avoided. Specular computes the same simple (Lambertian) lighting as diffuse plus a viewer dependent specular highlight. Additionally, the alpha channel of the main texture acts as a specular map (sometimes called “gloss map”), defining which areas of the object are more reflective than others. Generally, this shader is on the more expensive rendering side
	Reflective normal a mapped unlit	This shader will simulate reflective surfaces. This shader does not use normal-mapping in the traditional way. The normal map does not affect any lights shining on the object because the shader does not use lights at all. The normal map will only distort the reflection map. This shader is special because it does not respond to lights at all. Generally, this shader is quite cheap to render
	Reflective normal mapped vertex-lit	This shader will simulate reflective surfaces. This shader is vertex-lit and this makes it one of the simplest shaders. All lights shining on it are rendered in a single pass and calculated at vertices only. This shader is a good alternative to reflective normal mapped. Generally, this shader is not expensive to render

The Unity 3D supports one of three rendering types: the deferred lighting, vertex lit, and forward rendering. In vertex lit, a rendering of each object is generally drawn just once, while in deferred lighting they are drawn twice, no matter the lights affecting it. Difference in shader performance in these modes mostly depends on the textures used and calculations performed. Notice that shader lighting models are built in conjunction with forward rendering (the default rendering model used in the Unity 3D). In the Forward rendering, the performance of the shader depends on both the shader itself and lights in the scene. Three types of shader lighting models, such as the Unlit, Vertex-Lit, and Pixel-Lit, are used. The Unlit shaders are not affected by lighting. The Vertex-Lit shaders calculate a light based on the mesh vertices, using all lights at once. This means that the object will be drawn just once. The Pixel-Lit shaders are the most expensive because they calculate the final lighting of each pixel that is drawn. This means that the object is drawn once to get the ambient and main directional light and once for each additional light shining on the object. The Pixel-Lit shaders are popular because they are able to display pixel based rendering effects and normal mapping that makes the objects more visually appealing.

The opaque shaders are usable for rendering of the tree trunks, branches, and other plant parts. The transparent shaders are usable for rendering of leaves, flowers and bushes. They either support texture transparency (allowing for different levels of transparency) or they support the alpha clipping that renders the pixels as either completely transparent or completely opaque.

In the Forward rendering path, the performance of a shader depends on both the shader itself and the lights in the scene. There are two basic categories of shaders from a performance perspective, Vertex-Lit, and Pixel-Lit. The Vertex-Lit shaders in the Forward rendering path are always cheaper than the Pixel-Lit shaders. These shaders work by calculating a lighting based on the mesh vertices, using all lights at once. Because of this, no matter how many lights are shining on the object, it will only have to be drawn once. The Pixel-Lit shaders calculate the final lighting at each pixel that is drawn. Because of this, the object has to be drawn once to get the ambient and main directional light, and once for each additional light that is shining on it. This increases the load on the CPU to process and send off commands to the graphics card and on the graphics card to process the vertices and draw the pixels. The size of the Pixel-Lit object on the screen will also affect the speed, at which it is drawn. The larger the object, the slower it will be drawn. Thus, the Pixel-Lit shaders come at performance cost but this cost provides some spectacular effects: the shadows, normal-mapping, good looking specular highlights, and light cookies. Also, the lights can be forced into a pixel (“important”) or a vertex (“not important”) mode. Any vertex lights, shining on the Pixel-Lit shader, will be calculated based on the object’s vertices or whole object and will not add to the rendering cost or visual effects that are associated with the pixel lights.

In the Unity 3D 3.x package, the tree editor based on the L-systems is implemented. The tree creator inspector is split into three different parts: the Hierarchy view, the Editing tools and the Properties. The Hierarchy view shows a schematic representation of a tree, where each box represents a group of nodes. Every node

has properties that can be changed. There are five different types of nodes as mentioned below:

- The Root node is the starting point of the tree holding the global parameters for the tree.
- The Branch node is the first branch node represents the trunk, others create the child branches.
- The Leaf node is a final node; no other nodes can be attached to them. It can be used for the leaves, flowers, seeds, and similar.
- The Frond node is a similar node to branch node but has specific properties to enable easier creation of fronds.
- The Branch & Frond node is a node that has both the branch and frond properties.

The Editing tools allow a manual adjustment of a tree including moving and rotating the nodes in real time. The Editing tools also allow free hand drawing that allows to draw an exact spline for branches to follow. Also there are many properties to adjust and play with until creating the desired tree, e.g., in the leaf node properties, a geometry type can be chosen between the plane, billboard, cross, tree cross, and a custom mesh.

9.8 Conclusions

The conflicting criteria of high realistic representation, low cost implementation, and real-time application make the task of the large scene rendering very difficult. This leads to find the compromise technical solutions, especially during a virtual walkthrough in the forest scene. The lighting and shadow rendering is the necessary conditions of good visibility. To provide fast implementation, the techniques of the texture storage and loading in different types of computer memory were discussed as well as the shaders' possibilities on the example of the Unity 3D 3.x package.

References

1. Duchaineau M, Wolinsky M, Sigeti DE, Miller MC, Aldrich C, Mineev-Weinstein MB (1997) ROAMing terrain: real-time optimally adapting meshes. 8th conference on visualization VIS 1997, pp 81–88
2. Rocchini C, Cignoni P, Montani C, Scopigno R (1999) Multiple textures stitching and blending on 3D objects. 10th Eurographics workshop on rendering EGWR 1999, pp 119–130
3. Tanner CC, Midgall CJ, Jones MT (1998) The clipmap: a virtual mipmap. 25th annual conference on computer graphics and interactive techniques SIGGRAPH 1998, pp 151–158
4. Losasso F, Hoppe H (2004) Geometry clipmaps: terrain rendering using nested regular grids. ACM Trans Graph SIGGRAPH 23(3):769–776

5. de Boer WH (2000) Fast terrain rendering using geometrical MipMapping. http://www.flipcode.com/archives/article_geomipmaps.pdf. Accessed 10 Jan 2016
6. Reeves WT, Blau R (1985) Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Comput Graph* 19:313–322
7. Decoret X, Durand F, Sillion FX, Dorsey J (2003) Billboard clouds for extreme model simplification. *ACM Trans Graph* 22(3):689–696
8. Cohen F, Decaudin P, Neyret F (2004) GPU-based lighting and shadowing of complex natural scenes. *ACM SIGGRAPH 2004 posters*, p 91
9. Decaudin P, Neyret F (2004) Rendering forest scenes in real-time. *Eurographics symposium on rendering, rendering techniques 2004*:93–102
10. Decaudin P, Neyret F (2009) Volumetric billboards. *Comput Graph Forum* 28:2079–2089
11. Zhang H, Hua W, Wang Q, Bao H (2006) Fast display of large-scale forest with fidelity. *Comput Anim Virtual Worlds* 17:83–97
12. Liu F, Hua W, Bao H (2010) GPU-based dynamic quad stream for forest rendering. *Sci China Inf Sci* 53:1539–1545
13. Gumbau J, Chover M, Remolar I, Rebollo C (2011) View-dependent pruning for real-time rendering of trees. *Comput Graph* 35:364–374
14. Bao G, Li H, Zhang X, Che W, Jaeger M (2011) Realistic real-time rendering for large-scale forest scenes. *IEEE Int Symp VR Innov ISVRI 2011*:217–223
15. Deng Q, Zhang X, Gay S, Lei X (2007) Continuous LOD model of coniferous foliage. *Int J Virtual Reality* 6:77–84
16. Hoppe H (1996) Progressive meshes. 23rd annual conference on computer graphics and interactive techniques SIGGRAPH 1996, pp 99–108
17. Kim J, Lee S, Kobbelt L (2004) View-dependent streaming of progressive meshes. *Shape Model Int SMI 2004*:209–220
18. Maglo A, Lee H, Lavoue G, Mouton C, Hudelot C, Dupont F (2010) Remote scientific visualization of progressive 3D meshes with x3d. 15th international conference on web 3D technology web3D 2010, pp 109–16
19. Shopf J, Barczak J, Oat C, Tatarchuk N (2008) March of the froblins: simulation and rendering massive crowds of intelligent and detailed creatures on GPU. *ACM SIGGRAPH 2008 classes SIGGRAPH 2008*, pp 52–101
20. Qin X, Nakamae E, Tadamura K, Nagai Y (2003) Fast photo-realistic rendering of trees in daylight. *Comput Graph Forum* 22(3):243–252
21. Behrendt S, Colditz C, Franzke O, Kopf J, Deussen O (2005) Realistic real-time rendering of landscapes using billboard clouds. *Eurographics* 24(3):507–516
22. Scherzer D, Wimmer M, Purgathofer W (2011) A Survey of real-time hard shadow mapping methods. *Comput Graph Forum* 30(1):169–186
23. Stamminger M, Drettakis G (2002) Perspective shadow maps. 29th annual conference on computer graphics and interaction techniques, SIGGRAPH 2002, pp 557–562
24. Wimmer M, Scherzer D, Purgathofer W (2004) Light space perspective shadow maps. In: Keller A, Jensen HW (eds) *Rendering techniques 2004*, Eurographics symposium on rendering 2004, pp 143–151
25. Martin T, Tan TS (2004) Anti-aliasing and continuity with trapezoidal shadow maps. 15th Eurographics Workshop on rendering techniques EGSR 2004, pp 153–160
26. Lloyd DB, Govindaraju NK, Quammen C, Molnar SE, Manocha D (2008) Logarithmic perspective shadow maps. *ACM Trans Graph* 27(4):1–39
27. Zhang F, Sun H, Xu L, Lee K (2008) Hardware-accelerated parallel-split shadow maps. *Int J Image Graph* 8:223–241
28. Tadamura K, Qin X, Jiao G, Nakamae E (1999) Rendering optimal solar shadows using plural sunlight depth buffers. *IEEE Int Conf Comput Graph CGI 1999*:166–173
29. Lloyd B, Tuft D, Yoon S, Manocha D (2006) Warping and partitioning for low error shadow maps. *Eurographics Symp Rendering 2006*:215–226
30. Engel W (2007) Cascaded shadow maps. In: Engel W (ed) *ShaderX5: advanced rendering techniques*, Charles River Media

31. Franzke O, Deussen O (2003) Accurate graphical representation of plant leaves. In: Hu BG, Jaeger M (Eds) Plant growth modelling and applications PMA 2003, pp 1–16
32. Vogelmann C (1993) Plant tissue optics. *Ann Rev Plant Physiol Plant Mol Biol* 44:231–251
33. Tucker C, Garratt M (1976) Leaf optical system modeled as a stochastic process. *Appl Opt* 16 (3):635–642
34. Yamada N, Fujimura S (1991) Nondestructive measurement of chlorophyll pigment content in plant leaves from three color reflectance and transmittance. *Appl Opt* 30:3964–3973
35. Haberlandt G (1914) Physiological plant anatomy. *Nature* 93(2332):477
36. Allen W, Gausmann H, Richardson A (1973) Willstatter-Stoll theory of leaf reflectance evaluation by ray tracing. *Appl Opt* 12:2448–2453
37. Kumar R, Silva L (1972) Light ray tracing through a leaf cross section. *Appl Opt* 12:2950–2954
38. Govaerts Y, Jaquemoud S, Ustin MVS (1996) Three dimensional radiation transfer modeling in a dicotyledon leaf. *Appl Opt* 35:6585–6598
39. Baranowski G, Rokne J (1997) An algorithmic reflectance and transmittance model for plant tissue. *Comput Graph Forum* 16(3):141–150
40. Williams L (1983) Pyramidal parametrics. *Comput Graph* 17(3):1–11
41. Ewins JP, Waller MD, White M, Lister PF (1998) MIP map level selection for texture mapping. *IEEE Trans Vis Comput Graph* 4(4):317–329
42. Debevec PE, Taylor CJ, Malik J (1996) Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. 23rd annual conference on computer graphics and interactive techniques, SIGGRAPH 1996, pp 11–20
43. Debevec P, Yu Y, Borshukov G (1998) Efficient view-dependent image-based rendering with projective texture-mapping. In: Drettakis G, Max N (eds) *Rendering techniques* 1998. Springer, Wien
44. Asirvatham A, Hoppe H (2005) Terrain rendering using GPU-based geometry clipmaps. In: Pharr M, Fernando R (eds) *GPU Gems*, vol 2. Addison-Wesley, USA, pp 27–46
45. Kobbelt (1996) Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Comput Graph Forum* 15(3):409–420
46. Dick C, Krüger J, Westermann R (2009) GPU ray-casting for scalable terrain rendering. *Eurographics* 50:43–50
47. Dick C, Krüger J, Westermann R (2010) GPU-aware hybrid terrain rendering. IADIS international conference on computer graphics, visualization, computer vision and image processing CGVCVIP 2010, pp 3–10
48. Clasen M, Hege H-C (2006) Terrain rendering using spherical clipmaps. In: Santos BS, Thomas Ertl T, Joy K (Eds) *Eurographics/IEEE-VGTC symposium on visualization EUROVIS 2006*, pp 91–98
49. Feldmann D, Steinicke F, Hinrichs KH (2011) Flexible clipmaps for managing growing textures. International conference on computer graphics theory and applications VISIGRAPP 2011, pp 173–180
50. Taibo J, Seoane A, Hernández L (2009) Dynamic virtual textures. *J WSCG* 17(1–3):25–32
51. Chen B, Swan JE II, Kuo E, Kaufman A (1999) LOD-sprite technique for accelerated terrain rendering. *Visualization* 1999:291–298
52. Meyer A, Neyret F, Poulin P (2001) Interactive rendering of trees with shading and shadows. 12th Eurographics conference on rendering EGWR 2001, pp 183–196
53. Dischler J-M (1998) Efficiently rendering macrogeometric surface structures using bi-directional texture functions. In: Drettakis G, Max N (eds) *Rendering techniques* 1998. Springer, Wien
54. Max N, Deussen O, Keating B (1999) Hierarchical image-based rendering using texture mapping hardware. In: Lischinski D, Larson GW (eds) *Rendering Techniques* 1999. Springer, Wien
55. Max NL (1998) Horizon mapping: shadows for bump-mapped surfaces. *Visual Comput* 4 (2):109–117

56. Schaufler G, Dorsey J, Decoret X, Sillion FX (2000) Conservative volumetric visibility with occluder fusion. 27th annual conference on computer graphics and interactive techniques SIGGRAPH 2000, pp 229–238
57. Boulanger K, Pattanaik S, Bouatouch K (2006) Rendering grass in real-time with dynamic light sources and shadows. Research report PI 1809
58. Hanrahan P, Krueger W (1993) Reflection from layered surfaces due to subsurface scattering. 20th annual conference on computer graphics and interactive techniques SIGGRAPH 1993, pp 165–174
59. Pharr M, Hanrahan P (2000) Monte Carlo evaluation of non-linear scattering equations for subsurface reflection. 27th annual conference on computer graphics and interactive techniques SIGGRAPH 2000, pp 75–84
60. Jensen H, Marschner S, Levoy M, Hanrahan P (2001) A practical model for subsurface light transport. 28th annual conference on computer graphics and interactive techniques SIGGRAPH 2001, pp 511–518
61. Wang L, Wang W, Dorsey J, Yang X, Guo B, Shum H-Y (2005) Real-time rendering of plant leaves. ACM Trans Graph ACM SIGGRAPH'2005 24(3):712–719
62. Justice CO, Holben BN (1979) Examination of Lambertian and Non-Lambertian models for simulating the topographic effect on remotely-sensed data. Goddard Space Flight Cneter, Greenbelt NASA TM 85290
63. Unity|Documentation. <http://docs.unity3d.com/Manual/Shaders.html>. Accessed 12 Jan 2016
64. Unity|Documentation. <http://docs.unity3d.com/Manual/shader-Performance.html>. Accessed 13 Jan 2016

Chapter 10

Scene Rendering Under Meteorological Impacts

Abstract The rendering of the large landscape scenes is impossible without simulation of meteorological impacts and atmospheric phenomena. Four types of the meteorological impacts, such as wind, fog, rain, and snow, are discussed in this chapter. Additionally, the water surfaces and cloud simulation are considered. Great variety of methods can be classified as the physical-based, computer-based, and hybrid approaches. In this chapter, it is shown that many natural impacts are successfully described by the Navier-Stokes equations. The main goal of the computer-based methods is to provide the real-time implementation to the prejudice of the realistic rendering and modelling accuracy. Nowadays, the hybrid methods become popular in virtual reality and computer games, likewise in forest monitoring and inventory.

Keywords Scene rendering · Meteorological impacts · Wind rendering · Fog rendering · Rain rendering · Snow rendering · Water surface simulation

10.1 Introduction

The real-time rendering of the large landscape scenes is employed in many applications, such as video games, serious gaming, landscape and urban scene walk-through, simulators in forest inventory, web applications, among others. Forest contains many natural objects, and its real-time rendering remains the great challenge. On the one hand, the algorithmic implementation of the rich plant details in the large landscape scenes ought to be compact and optimal in accuracy. On the other hand, a computer memory should be enough for the storage and real-time rendering of big modelling data that are loaded in the required part of the whole scene for the end-user viewing.

In the current chapter, four main types of the meteorological impacts, such as wind, fog, rain, and snow, are represented in different models, describing their simulation and rendering in the natural scenes. Also, the water surfaces and cloud simulation are reviewed. Many physical models use the Navier-Stokes equations.

At the same time, the computer implementations prefer to apply the simplified models, using shaders.

This chapter is organized as follows. In the next Sect. 10.2, some related works are reviewed. The models of natural impacts for the wind, fog, rain, and snow rendering are described in Sects. 10.3–10.6, respectively. In Sect. 10.7, some natural objects’ rendering is discussed. Some modelling results are reported in Sect. 10.8. Finally, the concluding remarks are given in Sect. 10.9.

10.2 Related Work

There are many atmospheric phenomena and meteorological impacts that can be simulated in the modelling scenes. However, wind, rain, fog, and snow are the most often conditions for a simulation. Depending on the goal of the simulation, various techniques provide the real-time but non-well realistic procedures or the computational expensive physical-based procedures. In such simulation, two aspects ought to be considered, viz., the modelling that is often represented by millions or thousands of the particles in a scene with the given depth and a point of view, and the rendering of the particles without their occlusions and the right visibility. Consider the overview of wind, fog, rain, and snow rendering techniques in Sects. 2.1–2.4, respectively.

10.2.1 *Overview of Wind Rendering*

A wind presents in outdoor landscape practically always. Thus, the turbulence models of a wind as well as the simulation of its influence on various natural objects like trees [1], shrubs, prairies [2], grass [3], water surface, clouds, among others, attract attention of many researchers since 1990s. Conventionally, a wind is characterized by its force as the weak wind, mid-force wind, and storm wind [4]. The weak wind simulation is enough simple and particularly it can be interpreted as a synthesis of a stochastic video from a single image, sometimes known as “Tour into the Picture” system developed by Horry et al. [5]. Chuang et al. [6] investigated different ways to animate of natural textures. One way is to add the randomness by a noise field generation in spatial domain that is not well and leads to erratic and unrealistic simulations of natural phenomena. A noise imposition can be done in the frequency domain with better visibility results. The spectral method for synthesizing of a stochastic noise field has three steps: the generation of a complex Gaussian random field in the frequency domain, application of a domain-specific spectrum filter, and computation of the inverse Fourier transform to synthesize a stochastic field in the temporal or frequency domain. The main advantage of this method is that the synthesized stochastic field can be tiled seamlessly. Hence, a

preliminary segmentation and separation on the layers are required in order to simulate the full homogeneous texture.

The mid-force wind and storm wind simulation can be implemented by two approaches:

- Physically-based animations use the numerical modelling of the natural objects under physical forces. Armstrong and Green [7] simulated the dynamics of rigid bodies that included the rigid links joined at hinges to form a tree. The segment-based approach of tree animations that is considered the tree branches as the connected springs is very popular approach [8]. During such simulation, the dynamic equations of branch's displacement are solved [9]. The interesting technique of physically-based parallel approach to animate a tree motion in real time and the leaf deformation accelerated on the Compute Unified Device Architecture (CUDA)-based platform was presented by Yang et al. [10]. The real-time simulation, using the physically-based methods, meets the challenges in its implementation till nowadays.
- Procedurally-based animations simulate the consequences of a wind influence. Ota et al. [11] proposed a method to sway individual leaves and branches in a wind field for a simulation of natural motions. Stam [12] analyzed the periodic motions of trees in the Fourier domain using a spectral synthesis algorithm. Chuang et al. [6] considered a tree as a dynamic system and each branch as a harmonic oscillator. Li et al. [13] developed a probabilistic approach for the automatic production of the tree models using a video of a moving tree with a possibility of a wind rendering. Usually the procedurally-based methods can achieve the branch motions but they are lack in physical exactness.

The modelling of a dynamic loading on the trees during a storm wind is the task of special investigations involving a complex interaction of each component of a tree, such as the trunk, main branches, and sub-branches, and interaction of individual trees and shrubs. As James mentioned [14], the loads on the trees and stresses, developing in trunks and branches, are a combination of the following impacts:

- The tension (a pulling force).
- The compression (a pushing force).
- The bending (tension, compression, and shear).
- The shear (a force causing layers to slide over each other).
- The torsion (a twisting force).
- The growth (a circumferential force).

Two types of loads on a tree are known. The static loads are increased under the natural process of a growing, while the dynamic loads are caused by the strong wind. A swaying of a tree trunk can be modelled by interpreting a tree model as a mechanical model considering a set of parameters that describe the sizes and shapes of trunk, branches, sub-branches and a woody material.

Notice that it is important to consider the structure of the Earth's surface during simulation. These types of models are based on the wind speed and wake turbulence that is evaluated by the conducting computational fluid dynamics simulation. At the same time, the simple models of a wind simulation for computer game and virtual reality applications do not require the complex physical models of the wind propagation.

10.2.2 Overview of Fog Rendering

In general, a fog is simulated using two approaches: physical-based methods and simplified methods of computer graphics that prevail. The first approach simulates all radiative transfers between emitting and reflecting surfaces taking into account the interactions with the participating medium [15]. Due to calculations of global illumination, this approach is a time-consuming. The second approach simulates the perturbations caused in a scene by the interactions of light with the scattering medium based on the depth-dependency of fog visual effects [16].

The atmosphere always contains some concentration of sub-micron hygroscopic particles (for example, dust or smoke) that serve as the condensation nuclei of water vapor, when the air temperature is close to a dew point. This happens, when the air cools or the humidity rises. The most stable fogs are caused by an inversion of the air—ground temperature and can be of two types:

- The radiation fog called as a ground fog is the most common type of fog. This usually happens in late fall or early winter, when the land cools by the thermal radiation after sunset in calm conditions of clear sky.
- The advection fog happens, when moist air moves over a cooler surface (for example, a cold marine layer near a coastline).

The less stable fogs include a valley fog (concern to the radiation fog), an evaporation or steam fog (caused by a cold air standing over warm water or moist land), an upslope fog (caused by a moist air cooling under the wind), an ice fog (occurs at extremely low temperatures), among others. Fog contains both water droplets formed around the condensation nuclei and non-active sub-micron particles with different numbers, sizes, shapes, and orientations. These particles have relatively little effect on light propagation. Therefore, it is assumed that a fog contains the spherical water droplets in different numbers and sizes [17].

In the natural world, a fog medium involves many non-homogenous layers moving with the wind influence and undergoing turbulent motion. However, the real-time simulation is limited to depict a homogenous fog or fog with constant density distribution as a volumetric layered effect. Such representation is often used in commercial and open software packages [18, 19]. Existing methods of modelling of a heterogeneous fog can be classified in two groups. Methods from the first group use physical models of a turbulent motion based on the Fast Fourier

Transform or the Navier-Stokes equations [20]. They produce very realistic images but require high computational resources. Methods of the second group approximate physical properties of a participating medium enabling its visualization in real time. In this case, the periodical or fractal functions are used for simulation [21].

10.2.3 *Overview of Rain Rendering*

The techniques for rain simulation can be divided as off-line or on-line methods. The off-line methods are usually focused on the physical phenomena of rain. Many researchers are addressed to the famous database of high quality renders of rain drops for various values of the illumination parameters that was produced by Garg and Nayar [22]. They presented the off-line rain rendering and streak simulation algorithm in still images and animations based a rain drop oscillation model for various atmospheric scenarios. Physical properties of the rain drop splashes were studied by Stow and Stainer in 1980s [23] that later were used by Garg et al. [24] to infer a splash model with development of the image-based rain rendering algorithm with splashes.

The basic approaches of the traditional real-time rain rendering techniques involve a rendering of line primitives, using a translucent white material, in front of the observer or the beforehand created texture of rain streaks, blending and scrolling down at each frame. Wang and Wade [25] proposed four artist-generated rain/snow textures mapped onto a double cone centered about the camera. The algorithm permitted to tilt the cones modelling a camera movement, elongate the textures for motion blurring, and scroll the textures to simulate the gravity. A hardware vertex shader was applied to blend all four textures. The depth of a scene with parallax was created by a tuning of the rain drops' sizes and speed. Wang et al. [26] represented each particle by a sphere model with the refractive index of water, a random rain stroke matte, and the physical attributes, such as position and velocity. The rain color was computed on the fly, and the particles were uniformly distributed in a space. The optical effect was enhanced with homogeneous fog, light glows, and random rain splashes designed by an artist. However, the parallax problems of the image-based techniques restrict the approximate horizontal views to the end-user. Slomp et al. [27] used a straight-forward approach for real-time rendering rain drops in temporal conditions. First, a rain drop mask was generated. Second, the spherical rain drops as screen-aligned billboards with hierarchical reflective and refractive maps were rendered. Puig-Centelles et al. [28] defined the raining area and then performed a suitable management of the particle systems inside them using the multi-resolution techniques. This approach was completely integrated in the GPU as fast, simple, and efficient solution that was employed easily in the virtual-reality environments. Puig-Centelles et al. [29] proposed the CUDA-based algorithm for dynamic scenes including the rain drops' collisions.

10.2.4 Overview of Snow Rendering

Snow as a common natural phenomenon can be observed in three states, including the snowfall that is filled the air with falling and fluttering snowflakes, the snow accumulation placing a white blanket over the landscape that is changing under the wind influence and the snow blanket properties, and the snow melting under the lighting and temperature conditions, when a snow blanket or snowflakes are transferred into the water.

There are two main approaches for a snow simulation in computer graphics. The nonphysical-based models imitate the properties of a phenomenon without using the actual physics behind the phenomenon. The physical-based models provide better visibility with high computational cost. The researchers began to apply the particle systems to simulate water, rain, snow, among others, since 1990s. For example, Sims [30] simulated a waterfall, where the initial color of the particles was blue, but when the particles hit a rock they became white. Fearing [31] modelled the accumulation of snow as the thick layers of the fallen snow using a particle system. Chang and Ryoo [32] simulated a snow accumulation generating an exposure map and a snow melt using a light map. The snow rendering was implemented as a real-time application, computing the vertex displacements. A snow diffuse effect is achieved using the Perlin noise. The snow boundaries were estimated using the stochastic sampling.

The naïve technique for a snow accumulation with a snowflake modelling using billboards was suggested by Haglund et al. [33]. However, they as well as Fearing [34] were one of the pioneers, who simulated a snow blanket by a triangulation of the scattered points. Also, these authors paid much attention for the snowbound surface blending and the edges of a snow cover. Very refined modelling results for that time were achieved.

Langer et al. [35] defined the spectrum of the falling snow texture by a dispersion relation in the image plane derived from a linear perspective. A standard particle system generates a relatively sparse set of the falling snowflakes and then the particles are fulfilled by a synthesized dynamic texture. The image speed, image size, and particle depth can be interpreted as a set of the wavelengths in the frequency domain. This technique provided a good visibility of the snow effects in static and dynamic scenes.

Reynolds et al. [36] claimed that they proposed the first real-time implementation of snow simulation in 3D dynamic and moving scene. The implementation maps use the surface-bound accumulation buffers to each object in a scene forming the height-maps of the accumulated snow cover. The mapping between buffers and texture coordinates are rendered, forming the accumulation height-map. Additional blurring updated frame by frame and re-coloring improved the existing scene geometry.

The majority of the physical-based models use fluid dynamics theory, simplifying the Navier-Stokes equations. Feldman [37] used the reduced version of the Navier-Stokes equations to model the accumulation of a snow. The scene obstacles

were taken into account. Snowflakes were modelled as particles fluttering over a scene. Additionally, these particles were influenced by the wind field until they collide with a surface.

Maréchal et al. [38] simulated the snow fall over the ground through the conductive, convective, and radiative thermal transfers using a finite volume method. The thermal transfer appears in a medium and at the interface between two media, when a temperature gradient $\nabla T \neq 0$ conducting a transfer from the warmest region to the coolest one. There are three types of the thermal transfers:

- The heat transfers by conduction take place inside a medium or between two media in contact. There is no material displacement composing the medium, the heat is diffused gradually by molecular agitation.
- The heat transfers by convection take place in fluid materials and at the interface of a fluid medium and a solid medium. Unlike the conduction, the convection is produced by material displacement. The heat is carried by the fluid move.
- The heat transfers by radiation are emitted as electromagnetic rays from the surface of any material, whose temperature is greater than 0 K.

The variations of air and dew point temperatures, the amount of snow, cloud cover and day-night cycles are considered. Also, this approach took into account the phase changes, such as snow melting into water or water freezing into ice.

The model proposed by Festenberg and Gumhold [39] is derived from a physical snow deposition model. The proposed statistical snow deposition model computes an approximate snow mass probability distribution and is used to derive a geometric snow shape formulation creating the enhanced height span map.

Stomakhin et al. [40] developed the original a semi-implicit Material Point Method (MPM) that was specifically designed to efficiently treat the wide range of material stiffnesses, collisions and topological changes arising in complex snow scenes. The MPM method combines the Lagrangian material particles (points) with the Eulerian Cartesian grids. This approach gives a possibility for the end-user to control a snow behavior. Very impressive dynamic snow scenes are provided by the MPM method.

Gucer and Ozguc [41] presented an approach for simulation of a flowing snow avalanche using the molecular dynamics and the discrete element method. A snow avalanche is formed of dry and liquefied snow that slides down a slope. A particle system was utilized as the basic method for a simulation. The real-time shaders were employed for rendering. The GPU language and multi-threaded programming were utilized for collision detection and resolution that provided the realistic representation of a flowing avalanche.

A balance between the complexity and the visual result has to be found for each task. It is not reasonable to create a physical-based model with high exact results if no significant visual improvement is achieved. Thus, the assumptions are often introduced in order to simplify the physical model.

10.3 Wind Rendering

Wind is the major disturbance impact for forests, and it requires much attention to simulate and predict the damages under possible meteorological phenomena. However, the goals of a wind rendering may be different that defines a wide spectrum of the models available for simulation.

The influence on a tree depends on the tree morphology during growth, the wind duration and intensity, the location of a tree in the forest, the type of the Earth's relief, and other ecological parameters. If a tree is in a young age, then its trunk and branches are elastic. Later a tree becomes increasingly rigid and the plastic deformations occur if a wind constantly acts on the tree skeleton. Thus, the mechanical property of wood expresses a tree response to the loading stress caused by a wind. One can find the slant trees in the space, where a wind has a dominant direction. Forest is an ecological system, and a tree at the edge of the forest will be affected more than a tree that is surrounded by other trees.

The realistic behavior simulation of the broad-leaf trees and plants implies three models:

- The tree model includes the trunks and branches that are represented as the related little cylinders or prism primitives with different lengths and radius. This model simulates the rotations and translations under the internal and external forces.
- The leaves model means a simulation of the parallel venation structure of a leaf as well as leaf shapes, such as oblong, elliptic, rhombic, or spoon shape. This model simulates a tree animation in close scales.
- The animation model adopts two previous models for realistic simulation of the broad-leaf tree behavior under a wind.

Pirk et al. [42] developed very reasonable simulation model of a wind based on the Lagrangian fluid model that uses particles as discrete quantities. This approach provides the complex fluid phenomena, such as turbulence, fusion, and separation. This model employs the forces that cause dynamic behavior of the branches. Pirk et al. [42] proposed the Smoothed Particle Hydrodynamics (SPH) approach for the wind simulation that is able to model the wind loading realistically in real time. The main idea is to add the particle integrators to the tree geometry in order to simulate the wind effect. Many approaches represent a tree model as a skeletal graph $G = \{V, E\}$, where V is a set of the vertices, E is a set of the edges [43, 44]. Such skeletal graph describes fully the hierarchical structure of the L-system representing a tree structure. Each branch is a chain of a varying number of edges $C = \{e_1, e_2, \dots, e_n\}$ within the same branching level, where n is a length of a chain. For a given edge e_i , the corresponding vertices (v_s, v_t) are denoted, where v_s is an ancestor of v_t . The graph representation decreases the computational cost of a simulation in comparison to the full mesh geometry. During rendering, the mesh geometry is produced on the fly using the modern API shaders.

A wind field can be described by the Navier-Stokes equations that can be solved by various algorithms using splitting, grids, or smoothed particle systems [45]. In recent years, the SPH approach became very popular due to the simplicity of the particles displacement in 3D space and a possibility to control a varying density of particles in different regions. The SPH can simulate the objects that move quickly in a wind as well as the processes like turbulence and eddies in a wind field that appear from the swaying trees.

Each SPH particle has its position x and represents the local density or pressure. The Navier-Stokes equations describe the acceleration a_i of the i th particle as the material derivative of the velocity \vec{u} provided by Eq. 10.1, where ρ is a fluid density, p is a pressure, μ is a coefficient of viscosity, \vec{f} is an external force.

$$\ddot{\vec{u}}_i = \frac{d\vec{u}_i}{dt} = \frac{-\nabla p + \mu \nabla(\nabla \cdot \vec{u}) + \vec{f}}{\rho_i} \quad (10.1)$$

$$\nabla \cdot \vec{u} = 0$$

The quantities at a certain location are computed by summing the relevant quantities from contributing particles that lie within a certain radial distance. The contribution of each j th particle is weighted by a distance-based kernel smoothing function (Eq. 10.2), where A is the calculated quantity (density or pressure), x is a particle position, m is its mass, ρ is a particle density, W is a smoothing kernel with kernel radius h .

$$A(x) = \sum_{j=1}^N \frac{m_j}{\rho_j} A_j W(x - x_j, h) \quad (10.2)$$

The fluid development over time is computed using the Euler integration yielding 3D velocity field of the wind [46].

The swaying tree motions are simulated by the transformed graph. The positions of edges are changed under the external wind force that is compensated with the restoration, damping, and propagation forces. Pirk et al. [42] proposed to approximate the tree structure by particles distributed along the edges e_i of the tree graph instead of the input mesh, covering the leaves and branches by triangles with particles distributed with equal density. Also they used a dynamic clustering of smaller branches and leaves that are represented by cluster particles.

Oliapuram and Kumar [47] modelled each branch as a tapered cylinder, while Pirk et al. [42] performed a branch as a chain of edges, where each edge has a rigid link with the corresponding vertices $E = \{v_s, v_t\}$, where a vertex v_t rotates around its parent v_s . The moment of the force \mathbf{N}_f acting on the cylinder has a view of Eq. 10.3, where ω is an angular velocity, m is a mass, l is a length.

$$\mathbf{N}_f = \frac{ml^2}{3} \frac{d\omega}{dt} \quad (10.3)$$

If a force \mathbf{F} is the sum of all forces applied to a segment, then the moment \mathbf{N} can be expressed by $\mathbf{N} = \mathbf{F} \times \mathbf{e}$, where \mathbf{e} is a vector from the fixed end to the center of gravity of the branch between the corresponding vertices v_t and v_s . The force \mathbf{F} can be estimated as a superposition of the external wind force \mathbf{F}_w , the restoration force of the branch to its resting position \mathbf{R} , an axial damping force \mathbf{D} , the back propagation force that propagates the forces acting upon the child branch to the parent branch \mathbf{P} , and a force term representing the drag of leaves \mathbf{L} as in Eq. 10.4.

$$\mathbf{F} = \mathbf{F}_w + \mathbf{R} + \mathbf{D} + \mathbf{P} + \mathbf{L} \quad (10.4)$$

The dynamics computation according to Eq. 10.4 is performed recursively from the outer edges to the root edge.

The wind model provides a velocity vector affecting the tree graph. The computation of a wind force is employed using the aerodynamic drag equation Eq. 10.5, where \mathbf{S}_b is a normal projected area of the surface branch facing towards the wind, σ is a drag coefficient, $\sigma = 0.6$, \mathbf{u} is an external velocity of a wind.

$$\mathbf{F}_w = \mathbf{S}_b \sigma \mathbf{u} \quad (10.5)$$

The restoration force \mathbf{R} moves a deflected branch towards its rest position with respect to the parent edge. It is denoted by Eq. 10.6, where \mathbf{d}_r is a force spherical direction to the rest position, k_r is a branch rigidity, α is an angular displacement.

$$\mathbf{R} = \mathbf{d}_r k_r \alpha \quad (10.6)$$

A branch is suppressed by the axial damping force if the strong binding forces are directed among branch segments. The damping force acts against the edge angular velocity ω and proportional to the edge thickness coefficient η using Eq. 10.7.

$$\mathbf{D} = -(\hat{\omega} \times \hat{e}) \eta \omega |\omega| \quad (10.7)$$

The back propagation force \mathbf{P}_{i-1} models how the forces propagate from the child branch to the parent branch. This force propagates recursively from the outer edges to the root edge in a view of Eq. 10.8, where k_p is a propagation coefficient of the force.

$$\mathbf{P}_{i-1} = - \sum k_p \mathbf{R}_i \quad (10.8)$$

Coefficient k_p is a constant and determined by a ratio of child and parent branch thickness using Eq. 10.9, where k_c is the fixed propagation coefficient, Th_i and Th_{i-1} are the thicknesses of the parent and child vertexes, respectively.

$$k_{p_i} = k_c \frac{Th_i}{Th_{i-1}} \quad (10.9)$$

If a magnitude of the restoration force of child branches has a zero value, then a magnitude of the propagation force is also has zero value.

The leaf force \mathbf{L} for each of the leaf clusters is computed similar to the wind force \mathbf{F}_w using Eq. 10.10, where S_l is an approximate projected area of the leaf surface, σ is a drag coefficient, k_l is a constant leaf coefficient representing the leaf characteristics of different species, \mathbf{u} is an external velocity of a wind.

$$\mathbf{L} = S_l \sigma k_l \mathbf{u} \quad (10.10)$$

Pirk et al. [42] proposed to consider a wind influence on a tree in two different time scales, such as the immediate tree response and the long-term tree response. In the first case, a tree is swaying in a dynamic wind field, when the positions of branches change periodically within the bend range. This process is well-written by the model suggested by Oliapuram and Kumar [47].

The direction of an edge is given by Eqs. 10.11, where θ and θ' are the original and the new angular orientations, respectively, ω and ω' are the current and the new velocities, respectively, a is the angular acceleration, $a = d\omega/dt$, Δt is a time step.

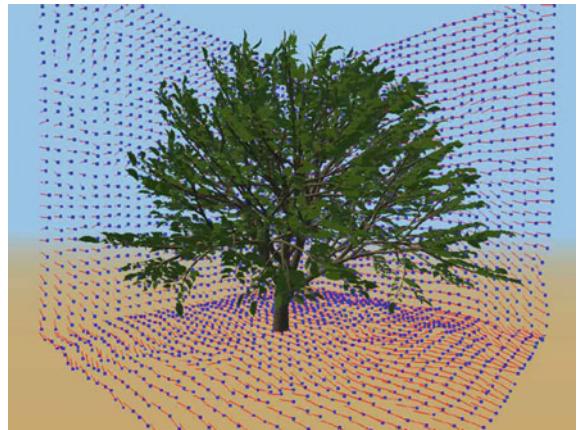
$$\begin{aligned} \theta' &= \theta + \omega(\Delta t) + \frac{1}{2} a(\Delta t)^2 \\ \omega' &= \omega + a(\Delta t) \end{aligned} \quad (10.11)$$

In the second case, a tree is the subject to plastic deformation, when it is exposed to stress of a wind field for a long time. Two types of stabilization modes for capturing the long-term adaptations of branching structures can be introduced. The dynamic stabilization allows the structural changes of a tree to be modelled under a dominant direction of a wind. The growth stabilization affects on the plasticity of the branches.

The breaking of branches is observed, when the acting forces exceed a certain level of stress. The physical model ought to consider the physical properties of wood, such as the rigidity, stiffness, and hardness. The simplified model is built on the Young's approximation and Hook's law, considering the species of a tree and its local growth structure [48]. The current stress σ acts upon an edge of branching structure and is described by Eq. 10.12, where c is the coefficient of Young's modulus E and the radius of the curvature R , $c = E/R$, r is a radius of a branch, \mathbf{M} is the bending moment, $\mathbf{M} = \mathbf{dr} \times \mathbf{F}$, \mathbf{dr} is the direction vector of a branch, \mathbf{F} is the force of the wind.

$$\sigma = \frac{4c|\mathbf{M}|}{3\pi r^2} \quad (10.12)$$

Fig. 10.1 3D wind field simulation



The maximum stress σ_{\max} is a proportional value to the branch thickness d and the material property p (p is defined by the end-user):

$$\sigma_{\max} = d^3 p.$$

A branch breaks, when the current stress $\sigma > \sigma_{\max}$. Note that the particle-based approach allows to model the breaking of individual and upwind-side branches. Example of 3D wind field simulation is depicted in Fig. 10.1.

Some investigations are directed on accurate simulation of wind flows over real terrains based on the grid approach [49–52]. The complexity of the micro-scale flows in the atmospheric environment is determined not only by complex topography but also flow obstacles, such as natural and man-made objects [53]. The Earth's surface contains multiple natural objects and features that interact among themselves under the wind propagation. The complex terrain, such as mountains or hills, requires to compute the mean wind speed and turbulence intensity information at several locations [54]. Many national standards, such as NBCC (Canadian standard), ASCE-7 (American standard), AS/NZS 1170.2 (Australian/New Zealand standard), and EUROCODE 1 (European standard), provide general guidelines to estimate the topographical multiplication factors for a wind speed over the complex terrain.

Jackson and Hunt [55] were the first, who analyzed a wind flow over isolated hills of a low slope using the linear forms of fluid flow equations. Their approach remains the conventional way for large scale wind mapping, requiring the low computational resources. However, many researchers apply the complex turbulence models, such as the Reynolds Averaged Navier-Stokes (RANS) and the Large Eddy Simulation (LES), in order to get accurate information about the turbulence structure in complex terrains [56–58]. The RANS models are often applied for industrial applications due to their relatively low cost of computation with reasonable evaluations. The LES models in spaces of flow separation are increasingly



Fig. 10.2 Photograph of the damaged tree

being used for the simulating flows with moderately high Reynolds numbers. The detailed information about different turbulence models and wall functions one can find in research [59]. The dynamic behavior of the trees is the subject of persistent interest. The original models were proposed in many researches, for example, [9, 48, 60, 61].

The strong wind like storm, tornado, or strong breeze carries the potential damage for a forest ecology. A photograph of the damaged tree is depicted in Fig. 10.2.

The prediction of probability of such events is very difficult and very important task. Hale et al. [62] validated three versions of the hybrid-mechanistic wind risk model and a statistical logistic regression model on the example of a Scottish upland conifer forests using the software tool ForestGALES. The ForestGALES is the decision support tool for estimation of the probability of a wind damage to any conifer stand in Britain. It calculates the wind speed that would be expected to damage a stand, current risk of overturning and trunk breakage, change in risk over the life of the crop, the effect on risk of thinning and creation of brown edges, and risk to any number of stands that is calculated simultaneously. The ForestGALES simulates the wind speed distribution at individual locations in Great Britain using the Weibull distribution with the Weibull shape parameter.

Tree animation in environment of hurricane impacts was proposed by Singh et al. [63]. Usually the consequences of storm rendering are determined by a species of the trees, their heights and state of roots, topography, climate, and soil properties. It is important to consider a tree model not only as a mechanical system that is exposed to the horizontal component of a strong wind force but also as ecological

system having its own shapes of the trunk, branches, and crown. The effect of storm wind may be not only branches break but also uprooting or a trunk break on determined height.

Two scenarios of behavior may be suggested dependently from “vertical” or “horizontal” roots of a tree. The first model (with deep root system) may be considered as a rigid column on the elastic base. The deviation angle θ of such “column” may be determined using Eq. 10.13, where P is a horizontal moment of wind, h is a point ordinate of application of force P , Q is a force moment of column weigh, l is a distance from root to a point ordinate of application of force P , J is a moment of inertia of root system, c is a proportional coefficient.

$$Ph - Ql\theta + cJ\theta = 0 \quad (10.13)$$

A simplifying condition $\theta \approx \sin \theta$ was employed in Eq. 10.13.

A critical load value will be, when $P = 0$ in Eq. 10.13. Thus, a critical load Q_c is defined by Eq. 10.14.

$$Q_c = cJ/l \quad (10.14)$$

Equation 10.14 is right until a root system would not break from the Earth’s surface. If this process begins, then Eq. 10.13 will be a nonlinear dependence.

The second model (with non-deep root system) has another interpretation. Under a wind influence, a tree is tilted. In this case, the gravity and the reaction force of the root system do not located in the same plane. They generate forces that aim to return back a tree into a normal position. A tree tilt is less for the second model and achieves a maximum durability by $\theta = 10\text{--}12^\circ$. In the first model, this parameter may achieve $30\text{--}35^\circ$ approximately.

10.4 Fog Simulation

Fog as a participating medium consists of small water droplets, and a radiance of light entering through such medium is influenced by four factors: absorption, emission, out-scattering, and in-scattering [64]. A fog in a daylight outdoor scene is a composition of two effects, such as a drain of any color received by the eyes and a creation of a veil of white mist. Since daylight is scattered in a fog, an in-scattering can be represented by a constant amount of light L_{fog} and emission can be neglected. Wherein, the integral transport equation proposed by Biri et al. [65] is simplified and provided by Eq. 10.15, where O and P are the points in the ray, $L(O)$ and $L(P)$ are the amount of light in points O and P , respectively, $\tau(\cdot)$ is a transmittance along the ray, $K_r(\cdot)$ is an extinction coefficient defined as a sum of the absorption and diffusion coefficients.

$$L(O) = \tau(O, P)L(P) + L_{fog} \int_0^P \tau(O, u) K_t(u) du = \tau(O, P)L(P) + L_{fog}(1 - \tau(O, P)) \quad (10.15)$$

Equation 10.15 can be interpreted in such manner that each pixel with color C_{in} is blended with the color of the fog C_{fog} to obtain the final color C_{fin} using Eq. 10.16:

$$C_{fin} = f C_{in} + (1 - f) C_{fog}, \quad (10.16)$$

where coefficient f is defined by Eq. 10.17.

$$f = \tau(O, P) = \exp \left(- \int_0^P K_t(x) dx \right) \quad (10.17)$$

The first term of Eq. 10.16 represents the loss of light from the incoming color due to scattering and absorption of light, while the second term is responsible for the drain of color and the white veil simulating the important in-scattering of a fog. Note that approximation based on Eq. 10.17 is generally applied in commercial software tools. Moreover, a density of fog is considered a constant as well as the approximated distance OP , which induces the simplest fog representation in a virtual scene. A coefficient $f \in [0, 1]$ defines the intensity of a fog at the point of a scene represented by the pixel. The smaller the value of f , the higher is the influence of fog on the color of the point. If $f = 0$, then a “fogging” of the point has a total value. If $f = 1$, then there is no “fogging”.

A function $K_t(\cdot)$ can be decompose into a set of chosen functions $\gamma_i(\cdot)$ with the weighted coefficients c_i :

$$K_t(x) = \sum_{i=1}^N c_i \gamma_i(x). \quad (10.18)$$

Functions $\gamma_i(\cdot)$ have different view in various algorithms. In the case of homogeneous fog, the linear, exponential, and squared exponential models are applied. In linear method, a function f is computed using linear interpolation and depends on the distance of point from the observer and on its location in relation to the fogged area in a view of Eq. 10.19, where Fog_{Start} is the beginning of the area influenced by fog, Fog_{End} is the distance (the border of visibility), beyond which nothing can be seen, d is a distance between the considered point and the observer.

$$f = \frac{Fog_{End} - d}{Fog_{end} - Fog_{start}} \quad (10.19)$$

In exponential and squared exponential models, a coefficient f depends on the distance d between the observed point and the observer, and on the fog density g . These models are expressed by Eqs. 10.29–10.21.

$$f = e^{-(d*g)} \quad (10.20)$$

$$f = e^{-(d*g)^2} \quad (10.21)$$

The simplicity of homogeneous fog interpretation is determined by a low computational cost and poor visibility of fog as a realistic phenomenon. These methods are implemented in Direct3D and OpenGL libraries.

In the case of heterogeneous fog, functions $\gamma_i(\cdot)$ can be cosine or polynomial functions. The cosine functions are represented by Eq. 10.22, where Λ_i are special operators on 3D point x like projection on one axis or on a chosen direction, c_i , k_i , and ϕ_i are coefficients chosen by the end-user.

$$K_t(x) = c_0 + \sum_{i=1}^{N-1} c_i \cos(k_i \Lambda_i(x) + \phi_i) \quad (10.22)$$

Note that the extinction function must not be negative:

$$c_0 \geq \sum_{i=1}^N \|c_i\|. \quad (10.23)$$

Functions from Eq. 10.22 are naturally periodic and can easily introduce by a phase difference.

The polynomial functions are represented by Eq. 10.24, where P_i are polynomial functions on 3D point x , c_i are the chosen coefficients.

$$K_t(x) = \sum_{i=1}^N c_i P_i(x) \quad (10.24)$$

The polynomials as the simple functions are easily integrated and combined. They may be used in 2D and 3D variants that have the increased computational time. However, the polynomial functions are not periodic. This requires to restrict their interval of definition to the bounded area in a scene.

Zdrojewska [64] proposed to simulate a fog with complex shapes introducing the equipotential functions. In this case, the extinction coefficient depends on a potential created by a defined virtual object, such as point, line or sphere. Adding and subtracting such virtual objects leads to the complex potential functions, for example:

$$p = \frac{1}{c + dr^2},$$

where r is a distance between the segment OP and the virtual object, c and d two chosen parameters.

In such manner, various functions of mentioned above families can be combined in order to obtain the original shape of a fog. The main problem for such combination is to blend them well.

Another simple approach for the heterogeneous fog simulation is to use so called the Perlin noise model. Perlin [66] was the first, who introduced a functional synthetic turbulence model as the basis for the realistic simulation of many different effects, including the fog, smoke, water, clouds, and fire. The Perlin model can be seen as a particular type of stochastic fractal that is generated as a summation of several appropriately scaled and dilated copies of a continuous noise function. The Perlin model is often called as the Perlin noise. To generate the Perlin noise, it is required to have a reproducible white noise as the layering multiple noise-functions at different frequencies and amplitudes. The frequency of each layer is usually the double of the previous layer. That is why, the layers usually are referred to as the octaves. By making the noise two-dimensional, a texture can be obtained. Equation 10.25 demonstrates a creation of 1D fractal noise from the multiple octaves of the Perlin noise. Notice that all the amplitudes are represented by a single parameter α . Higher values for α will give a rougher looking noise, whereas lower values for α will make the noise smoother.

$$f_{\text{noise}}(x) = \sum_{i=0}^{\text{octaves}-1} \alpha_i \text{noise}(2^i \cdot x) \quad (10.25)$$

A creation of fractal noise as a sum of the multiple octaves of the Perlin noise is depicted in Fig. 10.3.

As a particular case of Eq. 10.25, the displacement of vertices, simulating the height field of the water surface, can be computed using the Perlin turbulence function Eq. 10.26, where α_i is a fractal exponent modelling a roughness of the height field, t is a time in order to simulate dynamic behavior.



Fig. 10.3 Four octaves of the Perlin noise with parameter $\alpha = 0.5$ are summed up to a fractal noise

$$y = \sum_{i=0}^1 \alpha_i \text{noise}(2^i \cdot x, 2^i \cdot z, 2^i \cdot t) \quad (10.26)$$

Nowadays, the main attention is directed on simulation of a heterogeneous fog. One of the extensions based on the Perlin noise generation was developed by Guo et al. [67]. They estimated the transmission map using the Markov Random Field (MRF) model and the bilateral filter. The transmission map is a visual representation of a continuous function of a scene depth. In the case of a single video camera, a transmission map can be built based on a segmentation map. The generation of a transmission map involves three steps, such as an image segmentation using mean-shift algorithm, the initial map estimation based on the MRF model, and the refined map estimation using bilateral filter. The mean-shift algorithm is a conventional image segmentation technique that can significantly reduce the number of basic image entities in an image due to good discontinuity, preserving a filtering characteristic. However, the salient features of the overall image are preserved. In natural scenes, the objects like sky, mountain, forests, lakes, rivers, etc. are the salient objects, whereas other information is often less important and can be neglected [68]. These extracted features are very useful for estimation of the relative depth of scene objects.

The initial map estimation is used the graph-cut based on the α -expansion method to estimate the map $t(\mathbf{x})$. The α -expansion method is able to handle regularization via specific energy functions [69]. Each element t_i of the transmission map is associated with a label l_i , where the set of labels $L = \{0, 1, 2, \dots, l\}$ represents the transmission values $\{0, 1/l, 2/l, \dots, 1\}$. This procedure is applied to gray-level image in order to reduce a number of labels, for example, for $l = 31$. The most probable labeling minimizes the associated energy function $E(x)$ provided by Eq. 10.27, where P is a set of pixels in the unknown transmission t , N is a set of pairs of pixels defined over the standard four-connect neighborhood, $E_i(l_i)$ is a unary function representing a possibility of pixel having transmission t_i associated with label l_i , $E_{ij}(l_i, l_j)$ is a smooth term defining a possibility to join the neighboring pixels with similar depth.

$$E(l) = \sum_{i \in N} E_i(l_i) + \sum_{(i,j) \in N} E_{ij}(l_i, l_j) \quad (10.27)$$

Bilateral filter removes the discontinuities of the transmission map obtained by the MRF model. This filter can smooth the textured regions with the edges preserving. Thus, the initial transition map t_{ini} can improve a fog rendering using Eq. 10.28, where $t_{ini}(\mathbf{u})$ is an initial transition map corresponding to the pixel $\mathbf{u} = (x, y)$, $N(\mathbf{u})$ is a neighbor of \mathbf{u} , $W_c(x)$ and $W_s(x)$ are the Gaussian filter with the standard deviations σ_c and σ_s , respectively. The values of the standard deviations are defined empirically, for example, $\sigma_c = 3$ and $\sigma_s = 0.4$.

$$t(\mathbf{u}) = \frac{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p} - \mathbf{u}\|) W_s(|t_{ini}(\mathbf{u}) - t_{ini}(\mathbf{p})|) t_{ini}(\mathbf{p})}{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p} - \mathbf{u}\|) W_s(|t_{ini}(\mathbf{u}) - t_{ini}(\mathbf{p})|)} \quad (10.28)$$

For fog simulation, an image without fog $I(\mathbf{x})$, the final refined transmission map $t(\mathbf{x})$, and the air light A as a set 0...255 are required in order to obtain the simulated foggy image $J(\mathbf{x})$ according to the atmospheric scattering model, which is described by Eq. 10.29.

$$J(\mathbf{x}) = \frac{I(\mathbf{x}) - A}{\max(t(\mathbf{x})^\lambda, t_0)} + A \quad (10.29)$$

The value of t_0 is set to be 0.1. The transmission map $t(\mathbf{x})$ combines the geometric distance $d(\mathbf{x})$ and the medium extinction coefficient β (the net loss from scattering and absorption) [70], which is defined by Eq. 10.30.

$$t(\mathbf{x}) = e^{-\beta d(\mathbf{x})} \quad (10.30)$$

The different fog density can be adjusted by a coefficient β with the parameter λ such that

$$t^\lambda = (e^{-\beta d})^\lambda = e^{-(\lambda\beta)d}. \quad (10.31)$$

Using Eqs. 10.27–10.31, a homogenous fog can be added in the image. If the heterogeneous fog ought to be generated, then the Perlin noise model is applied that makes a scene more natural for a visual perception. The recommended values of extinction coefficients for various weather conditions are depicted in Table 10.1 [67].

The original approach to compute the extinction function of fog was proposed by Giroud and Biri [71]. First, a fog density is modelled in a B-spline function basis.

Table 10.1 Visibility grades with their medium extinction coefficients

Grade	Weather	Visibility distance	Extinction coefficient β
0	Dense fog	<50 m	>78.2
1	Thick fog	50–200 m	78.2–19.6
2	Moderate fog	200–500 m	19.6–7.82
3	Light fog	500–1 km	7.82–3.91
4	Thin fog	1–2 km	3.91–1.96
5	Haze	2–4 km	1.960–0.954
6	Light haze	4–10 km	0.954–0.391
7	Clear	10–20 km	0.391–0.196
8	Very clear	20–50 km	0.196–0.078
9	Extremely clear	>50 km	0.0141



Fig. 10.4 Photographs of natural fog scenes

Second, the Mallat's wavelet decomposition is applied on the extinction function in order to automatically generate different resolutions for real-time rendering using the GPU. The authors modelled 2D shape of a fog due to it always appears in large outdoor scenes as a horizontal layer of varying thickness.

Photographs of natural fog scenes are depicted in Fig. 10.4.

The time-consuming fog rendering cannot be a problem for dynamic scenes due to the slow fog changes in shape and density in natural scenes.

10.5 Rain Simulation

Rain is a wide distributed weather impact, and its realistic simulation has a great meaning in the modelling scenes. The synthetic methods to simulate weather can be off-line or on-line (e.g., real-time) respect to the desired effects. The off-line methods produce more realistic effects, being used in movie industry, while the on-line methods are used in scene modelling applications, video-games, and virtual reality simulators. The real-time rendering of rain assumes the realistic modelling of streaks and splashes with complex illumination effects with fog, halos, and light glows. This makes the rain simulation very difficult and challenging problem. The amount of small details and particles, such as rain drops, puddles, splashes, ripples, fog, light glows, clouds, rainbows, and lightning, requires the high computational capabilities. The simulation becomes more complex if the optical properties of a water are considered.

Photorealistic rain rendering in the still images or in the captured video sequences remain the challenging problem during last decade. The particle-based

rain rendering using the simple photometric models to render the individual rain streaks is applied in the commercial software products, such as Maya or 3D Studio Max. Moreover, the streaks are often assumed to have simple shapes like rectangles or ellipses with the constant brightness of each streak. Such approach permits to simulate a rain at the large distance from the camera. Many attempts were made to design the oscillation model induced by the aerodynamic forces and the surface tension. The falling rain drops change their shape and produce the motion-blurred streaks in an image. Also, the lighting sources and viewpoint changes influence on the rain streaks representation.

The first attempts may be concerned to 1980s. Thus, Green [72] proposed the simplest model of ellipsoid rain drop shapes as a balance between the surface tension and the gravity effect. Afterwards, Beard and Chuang [73] presented a complex and accurate model, distorting a regular sphere r_0 by a sum of weighted cosines in a view of Eq. 10.32 (e.g., a 10th order cosine distortion of a sphere), where θ is an elevation angle, $\{C_n\}$ is a set of the shape coefficients calculated previously.

$$r(\theta) = r_0 \left(1 + \sum_{n=0}^{10} C_n \cos(n \theta) \right) \quad (10.32)$$

Hereinafter, more complex models were suggested. The spherical shape of a drop at time instant t is denoted by the distance of a point on the drop's surface from its center $r(t, \theta, \phi)$ provided by Eq. 10.33, where θ and ϕ are the elevation and azimuthal angles of the point with respect to the OY axis (the direction of the drop's fall) and OX axis (perpendicular to the OY axis), respectively, r_0 is the undistorted radius (referred to as the drop size), $A_{n,m}$ is the amplitude of the spherical harmonic mode (n, m) , $P_{n,m}(\theta)$ is the Legendre function, which describes the dependence of the shape on the angle θ for the mode (n, m) [74].

$$r(t, \theta, \phi) = r_0 \left(1 + \sum_{n,m} A_{n,m} \sin(\omega_n t) P_{n,m}(\theta) \cos(m\phi) \right) \quad (10.33)$$

The frequencies depend on the order n and a rain drop size r_0 and calculated by Eq. 10.34, where σ is the surface tension, $\sigma = 0.0728 \text{ N/m}^2$, ρ is the density of a water, $\rho = 1000 \text{ kg/m}^3$.

$$\omega_n = 2\pi \left([n(n-1)(n+2) \sigma] / (4\pi^2 \rho r_0^3) \right)^{1/2} \quad (10.34)$$

The empirical studies shown that the oscillations in a rain drop are limited by two cases [75, 76], such as a rotationally-symmetric (oblate-prolate) mode ($n = 2$; $m = 0$) and a non rotationally-symmetric (transverse) mode ($n = 3$; $m = 1$). The shape distortions of a rain drop are depicted in Fig. 10.5.

According to the empirical studies, Eq. 10.33 may be simplified by Eq. 10.35.

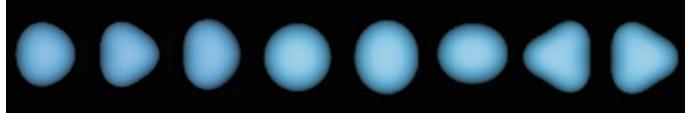


Fig. 10.5 Examples of shadow simulation

$$r(t, \theta, \phi) = r_0 (1 + A_{2,0} \sin(\omega_2 t) P_{2,0}(\theta) + A_{3,1} \sin(\omega_3 t) \cos(\phi) P_{3,1}(\theta)) \quad (10.35)$$

The terms of Eq. 10.35 show that the shape of a rain drop is not rotationally symmetric but depends on the rain drop size r_0 , the amplitudes $A_{2,0}$ and $A_{3,1}$, and the frequencies ω_2 and ω_3 .

The rain drops have a wide size variety. The empirical distribution for rain drop size is the Marshall-Palmer distribution [77], which is expressed by Eq. 10.36, where h is a rain rate, mm/hr, a is a radius of the drop, m, $N(a)$ is a number of rain drops per unit volume.

$$N(a) = 8 \times 10^6 e^{-8200*h^{-0.21}a} \quad (10.36)$$

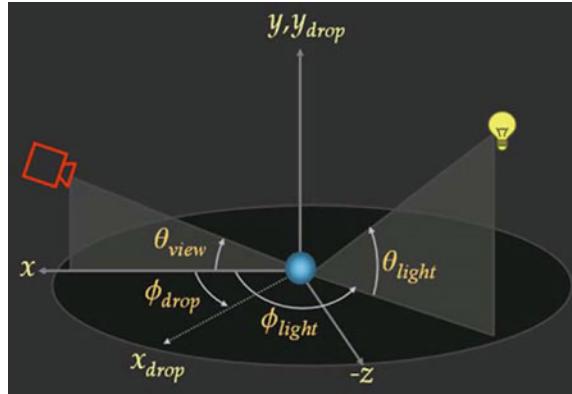
A rain drop is characterized by a constant velocity called the terminal velocity. Gunn and Kinzer [78] obtained that the terminal velocity v of a rain drop can be expressed as a function of its size and is given by Eq. 10.37, where v is measured in m/s.

$$v = 9.40 \left(1 - e^{-3.45 \cdot 10^3 a^{1.31}}\right) \quad (10.37)$$

The efforts of Garg and Nayar [22] led to creation of the database of the rain streaks in off-line mode. In common case, the parameters, such as the lighting and viewing directions, the distances from the source and the camera, the oscillation parameters, and the size of the drop and the camera's exposure time, are required to know. Some of the mentioned above parameters can be efficiently rendered on-line, other parameters, such as the lighting and viewing directions and the oscillation parameters, ought to be pre-computed or extracted from the database of Garg and Nayar. This database included about 6300 patterns of the rendered streaks in 1980s and was extended up to 15,000 patterns in 2010s. It contains textures indexed by three different angles: θ_{view} is an angle from the view vector to its projection onto the plane perpendicular to the direction of a rain particle, θ_{light} is an angle from light vector to its projection, and ϕ_{light} is an angle between the projections of the view and light vectors as it is depicted in Fig. 10.6. The parameters x_{drop} , y_{drop} , and ϕ_{drop} define a direction of a rain drop falling.

When a rain drop falls, its shape is changed by the distortions caused by the aerodynamic forces and the surface tension. Also, they developed the algorithm for rain rendering considering perspective effects (scale and rotation) and camera effects (expose time, defocus blur). The algorithm provides a streak texture with

Fig. 10.6 Model to indexing of the rain textures



different lighting, viewing, and rain drop size from the database or by interpolation of the close patterns. As a result, many day or night scenes with different illumination can be modelled.

Rousseau et al. [79] focused on the properties of the rain drops defined by the geometrical optics. Under this assumption, a light is considered as a set of the monochromatic rays that refract and reflect at interfaces between different propagation media. For a specific ray, a ratio between reflection and refraction is given by the Fresnel factor. The Fresnel factor demonstrates that a reflection has a significant participation to the color of a surface at grazing angles. This means that reflection in a rain drop is visible only on the border of the drop. Rousseau et al. built the model for the case of a close light source, when the external and internal reflections cannot be ignored. During experiments, they observed that the directions of the outgoing rays can be classified in three groups, even after three internal reflections:

- Back in the direction of the original ray (external reflection or third internal reflection).
- Opposite side of the drop, upward.
- Opposite side of the drop, downward.

Based on these assumptions, a formula to illuminate rain drops was derived taking into account the color extracted from the texture, the ambient light color of a scene, the maximum color modification that can be applied to the pixel, whose normal is facing the end-user, and color modifications produced by the external and the internal reflections.

Creus and Patow [80] proposed the solution that simulates the collisions using a depth map and solves the scalability and visibility problems in real time with high visual quality. This algorithm includes three main steps:

- Preprocessing generates the data as the rain particles and two texture atlases for direct illumination and ambient lighting needed for the real-time simulation. Textures for the rain drop billboards were taken from the database created by Garg and Nayar [22]. First, the rain space volume is defined by an orthographic

camera such that its Z-near plane corresponds to the top of the space and the Z-far to its ground level. Second, the particles are distributed using a probability density function, originating at the top and going straight to the ground with uniform droplet terminal velocity. Third, the particle is re-spawned again at the top of the rain space, when it reaches the ground that produces a cyclic movement for each particle saving the computational cost. For a wind with constant speed and direction, a slanted rain space should be used.

- Real-time image-based effects are added prior to the rendering of streaks and splashes. A fog is simulated by blending a uniform color, which opacity is linearly increased with the distance to the end-user. For halos effect, a uniform color is blended onto the horizontal surfaces to mimic the accumulation of the small droplets, modelling the continuous splashes. The light glows are generated by the spheres with the light color rendering at each light position. The fragment shaders are used to these purposes.
- Real-time simulation animates the particles and computes the collisions in a scene, rendering the particles as billboards that represent rain streaks (if no collision is found) or splashes (when a collision has happened). The geometry shader defines the projected particle coordinates on the rain space depth map for the collision checking. The fragment shader computes the fragment depth in a scene depth map to discard the billboard fragments that are below a scene surface. Also, the post-processing is required to consider the defocus effects due to a rain is a volumetric phenomenon. The defocus effects are achieved by a fusion of an image without rain with defocusing effects and an image with the droplets. Depending on a camera position, the rain drops blurring is changed.

Wang et al. [81] presented a physically-based method to model the intersection of fluid free surfaces and solid objects, the water drops in a particular case. The surface (interfacial) tension is caused by the unbalanced molecular cohesive forces in the interfacial region, where two mediums, such as liquid-air, liquid-solid, or solid-air, meet. One of the ways to analyze the surface tension's influence on the liquid motion is to use the surface tension force directly in the incompressible Navier-Stokes equation (Eq. 10.38), where a symbol “.” denotes a dot product between vectors, ∇ is a vector of spatial derivatives, \vec{u} is a velocity field, μ is a viscosity coefficient, ρ is a liquid density, k is a surface mean curvature, \vec{N} is the liquid surface normal vector, \vec{F} is the external force, γ is the surface tension coefficient.

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla) \vec{u} - \frac{1}{\rho} \nabla p + \mu \nabla (\nabla \cdot \vec{u}) / \rho + (\vec{F} - \gamma k \cdot \vec{N}) / \rho \quad (10.38)$$

$$\nabla \cdot \vec{u} = 0$$

A surface tension can be represented in terms of the pressure difference across the surface (the Laplace's Law) in a view of Eq. 10.39, where ΔP_{surf} is the pressure difference across the liquid surface.

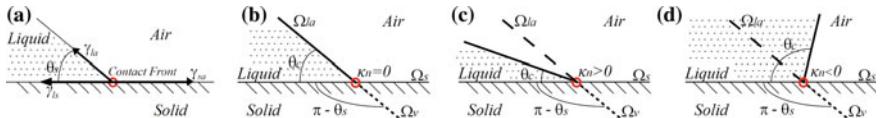


Fig. 10.7 Schemes of the contact fronts in equilibrium: **a** 1D case, **b** 2D case, stable front: $\theta_c = \theta_s$, **c** 2D case, receding front: $\theta_c < \theta_s$, **d** 2D case, advancing front: $\theta_c > \theta_s$ (θ_c is a current angle, Ω_{la} is a liquid-air surface, Ω_s is a solid surface, Ω_v is a virtual surface, k_n is a curvature in a normal plane)

$$\Delta P_{surf} = \gamma \cdot k \quad (10.39)$$

Another approach to analyze the surface tension's influence on the liquid motion is to estimate the contact line at the intersection of mediums using the Young's relation [82] provided by Eq. 10.40, where θ_s ($0 < \theta_s < \pi$) is a stable contact angle, γ_{ls} , γ_{sa} , and γ_{la} are the interfacial tension coefficients for the liquid-solid, solid-air, and liquid-air surfaces, respectively (Fig. 10.7).

$$\gamma_{sa} - (\gamma_{la} \cos \theta_s + \gamma_{ls}) = 0 \quad (10.40)$$

It is difficult to measure surface tension directly. The stable contact angle Ω_s used to quantify the affinity between a liquid and solid material. If θ_s has a small value (close to zero), then it means that the solid surface is a hydrophilic and the liquid surface tends to spread flat. In the opposite case, a solid surface is called hydrophobic and the liquid tends to bead up on the surface.

Equation 10.40 can be modified to 2D surfaces. Photographs of rain and dew drops on the natural surfaces are depicted in Fig. 10.8. The solution of liquid interaction with the curved solid shapes can be obtained for local flat solid surface. Therefore, some boundary conditions are introduced.

Note that a rain simulation depends from a wind-field generation. The open-field wind simulation requires the large-scale grids that makes impossible to apply the Navier–Stokes and the Lattice–Boltzmann models. Usually a primitive-based



Fig. 10.8 Photographs of rain and dew drops on the natural surfaces

procedural modelling technique is adopted in computer graphics. These primitives can be grouped according to the wind propagation:

- The primitive “Source” proposes that a wind blows in all directions from this point.
- The primitive “Source” means that a wind concentrates from all directions towards this point.
- The primitive “Vortex” indicates that a wind spins around an axis.
- The primitive “Uniform wind” denotes that a wind is constant in orientation and strength in every location.
- The primitive “Tornado” is the combination of a sink on the base, a source on the top, and a vortex.

Note that some connected impacts, such as puddles, splashes, ripples, rainbows, and thunderstorms, exist. However, a simulation of these effects is very restricted in experimental and commercial software packages.

10.6 Snow Covering Simulation

The most often type of snowflakes is considered the plates, examples of which are depicted in Fig. 10.9. McClung and Schaerer [83] were the first, who made the classification of ice crystal types like plates, stellar crystals, columns, needles, spatial dendrites, copped columns, irregular particles, soft hail, ice pellets, and hail. Hereinafter, Moeslund et al. [84] investigated how the ice crystals merge into the actual snowflakes, depending on the level of saturation and the temperature in the area between the cloud and the ground.

The shape, size, and density of a snowflake depend on the level of saturation in the air and the temperature. The temperature can affect on a scene, while a modelling of the air saturation is not necessary for rendering. The diameter of snowflakes has been measured as a function of a temperature using Eq. 10.41, where D is a diameter, m, T is a temperature, degree Celsius [84].

$$D = \begin{cases} 0.015 \cdot |T|^{-0.35} & \text{for } T \leq -0.061 \\ 0.04 & \text{for } T > -0.061 \end{cases} \quad (10.41)$$



Fig. 10.9 Examples of snowflakes

Equation 10.41 represents a heuristic estimation of the average diameter based on the experimental data with uncertainties up to $\pm 50\%$. The random number is added with the diameter D in order to limit the uncertainty.

The density of snowflakes $\rho_{snowflake}$ is inversely proportional to the diameter of the snowflakes, $\rho_{snowflake} = C/D$, where C is the proportionality constant, $C = 0.170 \text{ kg/m}^2$ for dry snow and $C = 0.724 \text{ kg/m}^2$ for wet snow [85]. The distinction between dry and wet snow is defined to be minus one degree.

Real snowflakes are constructed by ice crystals colliding. Moeslund et al. [84] modelled a snowflake by combining triangular polygons in a random manner. In order to make 3D snowflake, the concentric spheres are fulfilled by the polygons: 40 polygons for the wet snow and 10 polygons for the dry snow. In such manner, a wet snowflake with a diameter of 6 cm (largest possible) was constructed.

The movement of a snowflake is caused by the four forces:

- The gravitational force $\mathbf{F}_{gravity}$ defines the Earth's gravity. This force has the constant magnitude.
- The force $\mathbf{F}_{buoyant}$ represents the up-drift by the surrounding air. This force has the constant magnitude.
- The lift force \mathbf{F}_{lift} makes the snowflake move in circular and irregular patterns caused by the aerodynamic shape of a snowflake and the turbulence created behind a snowflake.
- The drag force \mathbf{F}_{drag} represents the drag that the air will assert on a snowflake.

The direction of the force $\mathbf{F}_{buoyant}$ is always opposite to the force $\mathbf{F}_{gravity}$. Moreover, the force $\mathbf{F}_{buoyant}$ is relatively small compared to the force $\mathbf{F}_{gravity}$ and it can be ignored as it has no impact on the visual movement result. The forces \mathbf{F}_{lift} and \mathbf{F}_{drag} change according to the wind direction. At high wind speeds the force \mathbf{F}_{lift} is insignificant but in calm weather it cannot be ignored. In this case, a snowflake follows the path of a helix towards the ground, while rotating around its center of mass. These two rotations depend on the shape of a snowflake, the rotational radius, and the ratio between the current speed of the wind and the current speed of a snowflake. The drag force \mathbf{F}_{drag} makes a snowflake follow the wind direction with the magnitude expressed by Eq. 10.42 [86], where m_{snow} is a mass of the snowflake, g is a gravitational acceleration, U_{fluid} is a speed of the air moving by the snowflake, U_{max} is the maximum vertical velocity taking wind resistance into consideration.

$$|\mathbf{F}_{drag}| = \frac{U_{fluid}^2 \cdot m_{snow} \cdot g}{U_{max}^2} \quad (10.42)$$

For dry snow this value lays in the interval [0.5 m/s, 1.5 m/s], and for wet snow it is in the interval [1 m/s, 2 m/s]. While parameters m_{snow} , g , and U_{max} are all constants for a particular snowflake, U_{fluid} is not. The direction of parameter U_{fluid} is also the direction of the drag force \mathbf{F}_{drag} . The parameter U_{fluid} consists of two components: $U_{fluid} = U_{wind} - U_{snowflake}$, where U_{wind} is a wind velocity and $U_{snowflake}$ is a velocity of the air friction. The wind field is a particular instant of

fluid dynamics and can be described by the Navier-Stokes equations. Moreover, the Navier-Stokes equations can be simplified to the incompressible Euler equations in a view of Eqs. 10.43, where ∇ is a divergence, \vec{u} is a vector field of the velocity, p is a pressure.

$$\begin{aligned}\frac{\partial \vec{u}}{\partial t} &= -(\vec{u} \cdot \nabla) \vec{u} - \nabla p \\ \nabla \cdot \vec{u} &= 0\end{aligned}\quad (10.43)$$

The first expression from Eqs. 10.43 states that the fluid conserves momentum, while the second expression states that the fluid conserves mass. The term $-(\vec{u} \cdot \nabla) \vec{u}$ is the convection term that describes how the velocity of the fluid evolves over time. The term ∇p is the acceleration of the fluid caused by the pressure gradient.

Foldes and Beneš [87] focused on the simulation of a falling snow and its accumulation for the large-distance views. This method includes two step process. The first step determines the local occlusion accounting the indirect illumination. The second step defines the global occlusion using a shadowing from the direct illumination of snow. A snow melting is considered as a slow process, depending on an average of sunlight during a day and a weather temperature. The static scene represented as an irregular input mesh is rendered as many times as the number of vertices. The total occlusion is stored as a luminance texture. The simulation of snow distribution is based on the ambient occlusion value of each vertex of the input mesh denoted by $a_{i,j}$, where i and j are the indices of the vertex. The amount of snow $s_{i,j}$ added at each vertex is determined by Eq. 10.44, where τ is a user-defined threshold value, s_{\max} is a height of a snow layer (a snow layer is constant for the entire surface), θ is an angle between the normal vector $\vec{n}_{i,j}$ and the direction to the zenith.

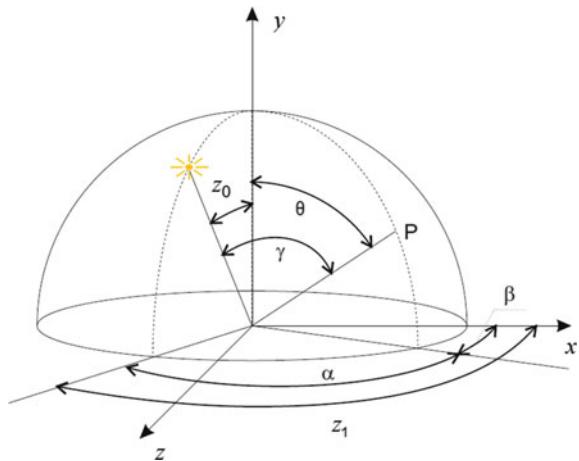
$$s_{i,j} = s_{\max} (\tau - a_{i,j}) \cos \theta \quad (10.44)$$

The value of s_{\max} is defined by the end-user. If $s_{i,j} < s_{\max}$, then the amount of snow is added in a vertex $a_{i,j}$, otherwise no snow is accumulated at this vertex.

During the first step, the basic shape and location of a snow are determined. The second step clarifies how much snow will be retained after a direct sunlight. The skylight illumination $L(\theta, \gamma)$ of a point P with coordinates (θ, γ) of a clear sky with the Sun is given by Eq. 10.45 [88], where L_z is a luminance of zenith, γ is an angle between the Sun and the point P , θ is an angle between the zenith and the point P , z_0 is an angle between the zenith and the Sun, α is an angle γ projected into the plane of horizon.

$$L(\theta, \gamma) = L_z \frac{(0.91 + 10e^{-3\gamma} + 0.45 \cos^2 \gamma) (1 - e^{-0.32 \sec \theta})}{0.274(0.91 + 10e^{-3\gamma} + 0.45 \cos^2 z_0)} \quad (10.45)$$

Fig. 10.10 Scheme of a sky illumination



The angle γ can be calculated using Eq. 46.

$$\gamma = \arccos(\cos z_0 \cos \theta + \sin z_0 \sin \theta \sin \alpha) \quad (10.46)$$

The dispositions of angles used in the sky illumination are depicted in Fig. 10.10.

Luminance of each point on the sky is calculated from Eq. 10.45 and then is averaged over the one day period. Each visible point on the snow cover gets the average daily luminance. The amount of snow melt is computed as the weighted summation of the occlusions of each sun position.

Hinks and Museth [89] presented a physically-based snow modelling approach that handles geometrically complex scenes and arbitrary amounts of accumulated snow. Their transportation model was based on the concept of snow packages including the wind packages (a simplified version of the snowflake motion model ignoring rotations) and the slide packages (deposition of a snow at stable locations). It is interesting that Hinks and Museth investigates a curvature of the surfaces on the subject of snow stability (Fig. 10.11). Also, they computed a temperature-dependent angle of response θ_{AOR} in a view of Eq. 10.47, where T is a temperature in $^{\circ}\text{C}$. This model is valid for $T \in [-35, 0]$.

$$\theta_{AOR} = \begin{cases} 50 - 30(|T + 6|^{-0.25}) & T \in [-35, -8.5] \\ -\frac{26.1418}{8.5}T & T \in [-8.5, 0] \end{cases} \quad (10.47)$$

The photographs of a snow covering and snowfalls are depicted in Fig. 10.12. Note that the snow particles in a snow scene are half-transparency because of motion blurring and defocus blurring. Koenderink and Richards [90] analyzed, why snow is so bright. They pointed three factors that are responsible for this effect:

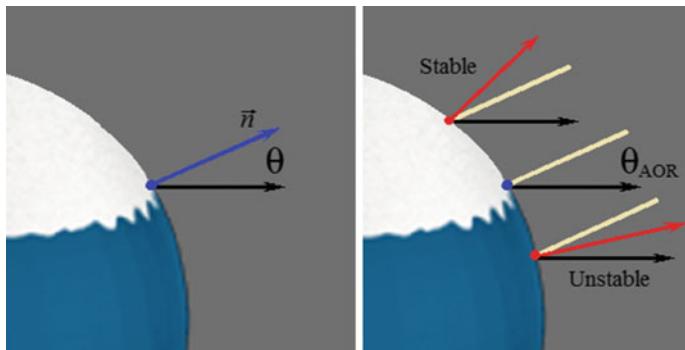


Fig. 10.11 Snow stability: a point on the surface is stable if $\theta \geq \theta_{AOR}$, otherwise a point is unstable



Fig. 10.12 Photographs of snow covering and snowfall

- The law of darkening for the cloud cover.
- The reflectivity of the snow and the average landscape albedo.
- The observer's contrast sensitivity function.

In the most video of a snow scene the snow particles are brighter than background because of high albedo. Also, the color of snow particle is white because it reflects the sun light.

10.7 Natural Objects' Rendering

The natural objects, such as lakes, rivers, seas, clouds, vapor trails, smoke, fire, explosions, among others, are often an integral part of landscape scene, and their realistic simulation and rendering ought to be provided in real time. The main challenge of these phenomena is a complexity of the numerical solutions of known physical models or approximate methods. Consider the water surface and cloud

modelling in details in Sects. 10.7.1 and 10.7.2, respectively. The fairly simple techniques that provide surprisingly realistic images can be used for smoke, vapor trails, fire, and explosions simulation, for example, the billboard techniques or special texture mapping.

10.7.1 *Rendering of the Water Surfaces*

Recently, the real-time rendering of water surfaces using graphics hardware have been proposed in literature [91–94]. Sometimes, the water is assumed to stay in a closed and opaque container with environment mapping by pre-computed texture maps. In general, these techniques rely on hardware support for computing reflection and refraction vectors with a possibility for the automatic generation of texture coordinates used to index into the environment maps. Due to the shader limitations, a simulation of water surfaces requires local evaluations using only a few simple operations. Another functional approach is the simulation by stochastic fractals.

The movement and rendering of the water surfaces can be separated into two tasks. Usually a behavior of fluids is described by the Navier-Stokes equations. These equations allow to represent a fluid by its velocity field and a pressure field, varying both in time. If both fields are known at the initial time, then the state of the fluid over time can be described by Eq. 10.48, where a symbol “.” denotes a dot product between vectors, ∇ is a vector of spatial derivatives, \vec{u} is a velocity field of a fluid, p is a pressure field of a fluid, ρ is a density, μ is a dynamic viscosity, \vec{f} is a vector representing the external forces.

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla) \vec{u} - \frac{1}{\rho} \nabla p + \mu \nabla^2 \vec{u} + \vec{f} \quad (10.48)$$

$$\nabla \cdot \vec{u} = 0$$

The right-hand side of the first equation consists of four parts:

- Advection $-(\vec{u} \cdot \nabla) \vec{u}$ represents the process, by which a fluid's velocity transports itself and other quantities in the fluid.
- Pressure $-\frac{1}{\rho} \nabla p$ causes the regions with a higher pressure to accelerate the molecules away from that area.
- Diffusion $\mu \nabla^2 \vec{u}$ represents the force caused by the viscosity of the fluid.
- External forces \vec{f} represent forces that act on the fluid like gravity.

The second equation is called the continuity equation and means that fluids conserve mass.

The complex movements like breaking waves or splashing water require a volumetric representation of a water that leads to physically accurate but computationally intensive simulations. The separate task is a modelling of ocean surfaces, when the Navier-Stokes equations are transformed to the Gerstner and Biesel swell

model. However, the realistic results had been obtained by combining the swell model with a modelling of waves' propagation as a simulation of amplitude, frequency, direction, and refraction near coasts [93, 95]. These approaches compute a height field representation of the water surface in real time. The spectral approaches simulate the water surface height field using the Fast Fourier Transform (FFT).

The rendering techniques of animated water surfaces are classified into different categories:

- A wave propagation can be described by the linear hyperbolic wave equation that defines the displacement of the surface points along OY direction under tension during small vibrations. This equation has a view of Eq. 10.49, where c specifies a wave speed.

$$\frac{\partial^2 y}{\partial t^2} - c^2 \left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2} \right) = 0 \quad (10.49)$$

The general solution involves expensive computations of trigonometric function series but can be simplified using only the dominant frequencies. A closed form solution can be found by specifying the initial conditions on the surface boundaries and on the displacement of the surface.

- The ocean waves can be simulated using statistical models. The water surface height field is modelled by the FFT algorithm. One can receive good simulation results but not in real time due to the necessity of high frame rates, in general.
- Particle models are built upon the separate displacement of particles with respect to physics based constraints. Shoreline waves or wave loops can be modeled by simulating the circular motion of particles. B-splines are employed for smooth wave simulation.
- Stochastic fractals are well suited for the limited types of wave phenomena. However, their ease and efficient implementation makes them a very appealing alternative decision.

In general, the simplified models of water simulation prevail due to their intensive use in game industry, when the demand of real-time simulation is over-balanced by physical realism.

Usually refraction, reflection and the Fresnel term are simulated in order to model the optical characteristics of water surfaces [96] and ocean waves [93]. Refraction and reflection are simulated easily using the ray optics based on the Snell's law. The Fresnel term models the dependence of the reflectance from the incoming light direction and the surface's index of refraction. The Fresnel term is approximation of physically accurate Bidirectional Reflectance Distribution Function (BRDF). Additionally, caustics are simulated based on the diverging refractions of wave front at water surfaces.

The use of time-varying fractals for the visual simulation of natural phenomena like clouds, fire, smoke or water has been studied extensively during the last decades. Not all water surfaces can be exhibit in fractal shape. However, the

simulation of calm water or water in narrow containers can be successfully implemented using this approach.

The rendering of water surfaces is difficult task due to complex interactions with light. Light absorption, refracted shadows, and refracted caustics are the main effects simulating the water volumes. The foundations for modelling of the water surfaces were created in 1990–2000s in the researches of Fournier and Reeves [97], Premoze and Ashikhmin [98], Enright et al. [99], among others. Later the work was directed on the graphics hardware implementation in a view of the GPU-based rendering algorithms.

The simplified light absorption model was proposed by Premoze and Ashikhmin [98]. For a given wavelength λ , the outgoing radiance $L_\lambda(p_w, \vec{\omega})$ from a point p_g under water to a point p_w on the surface in the direction $\vec{\omega}$ is provided by Eq. 10.50, where d is a distance between p_g and p_w , z is a depth difference between p_g and p_w , $\alpha_\lambda(d, z)$ is an exponential attenuation, depending on traveled distance and depth, $L_{d\lambda}$ is a constant diffuse radiance computed from the scattering measurements involving sky and sun colors.

$$L_\lambda(p_w, \vec{\omega}) = \alpha_\lambda(d, 0) L_\lambda(p_g, \vec{\omega}) + (1 - \alpha_\lambda(d, z)) L_{d\lambda} \quad (10.50)$$

An exponential attenuation $\alpha_\lambda(d, z)$ is defined by Eq. 10.51, where a_λ is an attenuations coefficient of traveled distance, b_λ is an attenuations coefficient of diffuse scattering, depending on depth.

$$\alpha_\lambda(d, z) = e^{-a_\lambda d - b_\lambda z} \quad (10.51)$$

Due to the precise coordinates of points p_g and p_w are known, it is easy and costless to compute the traveled depth and distance. The light attenuation varies with wavelength, therefore, the color spectrum ought to be discretized and the computations must be done per a wavelength. In practice, the term $b_\lambda z$ from Eq. 10.51 is rejected, and this simplification is sampled in a 1D color texture, which makes Eq. 10.51 the simple linear blending between $L_\lambda(p_g, \vec{\omega})$ and L_d with respect to $\alpha(d)$.

As Baboud and Décoret mentioned in their research [92], the reflections on the water surface can be separated in three groups: the reflections of distant objects, reflections of near objects, and reflections of the ground relief emerging from water. The reflections of distant objects are handled using an environment map, whereas the reflections of near objects are usually simulated using the heuristics only. The reflections of the ground relief are useful for low height field values near coasts.

Caustics are caused by the convergence of light due to reflections and refractions. They depend from global illumination like photon mapping. Thus, caustics rendering is computationally expensive process, the simplicity of which leads to appearance of noisy or blurred caustics in a scene [100, 101]. Most existing caustics algorithms are based on backward ray tracing, for example, two-pass photon-mapping-like algorithm. The first pass renders the water surface from the light source into a photon texture. A random jittering to light rays reduces the aliasing artifacts. Then an illumination texture (an illumination map) is created. The



Fig. 10.13 Photographs of water surfaces

second (final) render pass considers the illumination texture. Shadows of other objects in a scene are simulated as a coarse approximation of the shadow map approach.

The development of the Navier-Stokes-based methods deals with 3D modelling of water, air, or water-air (water surfaces) using 3D grids [102]. Many issues are devoted to the additional speedup provided by the CUDA implementation [103]. Very interesting research was provided by Yu [104], where the river simulation was introduced as a multi-scale system with procedural river flow (macro-scale), high-quality rendering of wave propagation (meso-scale), and spatio-temporal wave sprites (micro-scale). The original algorithm for real-time wave simulation in large-scale water environments with physics-based object interaction like boats and ships was provided by Cords and Staadt [105]. This algorithm was implemented efficiently on the GPUs achieving high frame rates on a desktop personal computer.

Photographs of water surfaces and natural scenes with rivers, lakes, and creeks are depicted in Fig. 10.13. The realistic rendering of a water surfaces is the task for future investigations.

10.7.2 *Rendering of Clouds*

Clouds like a smoke can be concerned to dynamic non-rigid textures without well defined surfaces and boundaries. In computer graphics, the physical modelling techniques are replaced by the simplified mathematical models with appropriate aesthetical visibility. There are five general methods that have been used to model and render clouds, such as the particle systems, metaballs, voxel volumes, procedural noise, and textured solids. Often several techniques have been combined in order to receive better visual results. The main propositions in this issue were made in 1990s–2000s. Thus, Gardner [106] proposed to generate a realistic 2D or 3D texture function $t(x, y)$ using a fractal or spectral based function (a Fourier-like sum of sine waves with phase shifts) provided by Eq. 10.52:

$$t(x, y) = k \sum_{i=1}^n (c_i \sin(fx_i x + px_i) + t_0) \sum_{i=1}^n (c_i \sin(fy_i y + py_i) + t_0) \quad (10.52)$$

with the relationships

$$\begin{aligned} fx_{i+1} &= 2fx_i & fy_{i+1} &= 2fy_i & c_{i+1} &= 0.707 c_i \\ px_i &= \frac{\pi}{2} \sin(fy_{i-1}y) & i > 1 & py_i &= \frac{\pi}{2} \sin(fx_{i-1}x) & i > 1. \end{aligned}$$

Gardner also suggested to simulate 3D cloud structures by ellipsoids, when the texture was generated using a 3D extension of the Fourier synthesis method with increasing transparency near the boundary of the ellipsoid. These 3D textures can be combined with the volume rendering techniques to produce 3D cloud images. In order to improve the fast performance of the rendering, 3D volumes are replaced by 2D structures with elliptical and opaque shape at the center. Therefore, the interior of a cloud can be represented as a polygonal shell and the volume rendering techniques applied for the outer edges only.

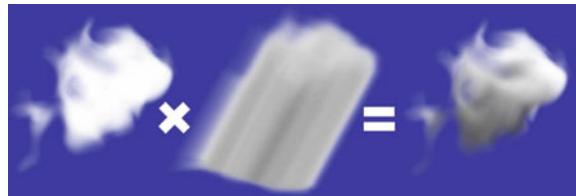
Liao et al. [107] developed the framework for dynamic cloud rendering. They proposed the simplified lighting model based on the designed shadow relation table and metaball lighting texture database. Thereby, the features, such as self-shadowing and light scattering, were computed quickly using table lookup at the run time.

Before rendering, the initial distribution of the cloud and vapor should be determined. The phase transition from vapor to water is controlled by the probability distributions. The vapor and phase transition probabilities are set higher in central portions than in the boundary regions. Also the cloud extinction probability is set higher in the boundary regions than in the central portions. Therefore, the clouds would appear thinner in the surroundings and thicker in the middle. Instead of ellipsoids used to control a cloud motion [108], Liao et al. [107] utilized the meatball function applying the run-time horizontal filter to smooth the density distribution.

Cellular automata for the clouds simulation is a conventional approach that was proposed by Dobashi et al. [108] and extended by some authors. Thus, Liao et al. [107] modified the cellular automata for run-time simulation by adding the following features:

- The cloud-to-vapor probability was added to transform the clouds to vapor rather than just disappear.
- The constraint was introduced for the cloud growing and disappearing only in the boundary regions.
- The growing ability of clouds was reduced to have the balance between the growing and extinction abilities.
- The upper and lower bound for the number of cloud voxels was added to avoid of too much cloud growing or disappearing.

Fig. 10.14 Example of oriented light volume illumination of 3D cloud: the cloud density volume without illumination is combined with oriented light volume illumination [110]



The parameters' tuning permit to simulate various types of the clouds, for example, cirrus or stratocumulus clouds, with different color due to the weather conditions and day period.

Hu et al. [109] modified the original method of Dobashi et al. [108] in order to avoid the “space regularity” and “texture regularity” problems. The solving of the first problem was achieved by a jittering of the centers of billboards after cloud shading process and before the cloud rendering process. To resolve the second texture regularity problem they “polluted” the pure pattern and perturb the symmetry in the basis function used in texture generation applying the Perlin noise technique. As a result, very realistic images with clouds were generated at midday and dawn.

Correct lighting and shading produce the visually convincing cloud images and their dynamic animations. The clouds are partially translucent and refractive. Also, a light diffusion through clouds causes the reflection, refraction, absorption, and scattering. Additionally, a reflection of light between clouds creates an interdependence between separate clouds. The ray-tracing techniques produce accurate lighting but they are remained very time-consuming with problems in real-time animation [108]. The result of the cloud density volume illuminated via modulation by the oriented light volume illumination is depicted in Fig. 10.14.

Wind can change the structure of a cloud completely by altering different parts of the cloud in different directions. Dobashi et al. [108] introduced a new set of transition rules for the cellular automaton to simulate a simple wind with linear velocity along the global OX axis. Afterward, a cloud simulation under wind was extended in more promising algorithms.

Photographs of clouds in natural scenes are depicted in Fig. 10.15. As one can see, the daily illumination influences significantly on the color of the clouds. The shapes of the clouds are very different according to many ecological and atmospheric phenomena.

10.8 Experimental Results

Software tool “REWELS” (REndering of Wind Effects in Landscape Scenes) contains three main modules: the module of a tree generation based on the laser data and the L-system, the module of 3D scene generation, and the rendering module



Fig. 10.15 Photographs of clouds in natural scenes

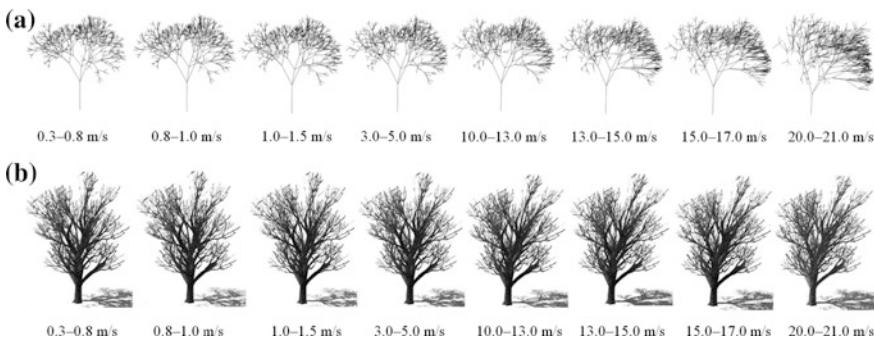


Fig. 10.16 Modelling results of a mid wind rendering: **a** fractal models of the trees, **b** models of the trees without foliage

that includes three developed algorithms of a wind simulation in dependence of a wind speed [111]. The designed software tool has a possibility to tune the wind parameters and also to move, rotate, and scale the modelling scene. The maximum tilts of a tree model (on the assumption that a wind direction is along axis OX) under rendering of a mid-wind are depicted in Fig. 10.16. The simplified fractal models of the trees and the models of the trees without foliage under the influence of a wind speed 0.3–21.0 m/s are represented in Figs. 10.16a and b, respectively.

Modeling results of storm wind rendering are shown in Fig. 10.17.

For program realization, the graphical jet GLScene of the library OpenGL was applied. The standard components of jet GLScene, such as TGLCamera (a camera object), TGLSceneViewer (a 3D viewer object), TGLMaterialLibrary (a library of materials), TGLFreeForm (a static 3D model), etc., were used. The program interface was designed in Embarcadero RAD Studio 2010 Architect.

Fig. 10.17 Modelling results of a storm wind rendering:
a maximum tilt of a tree,
b uprooting modeling

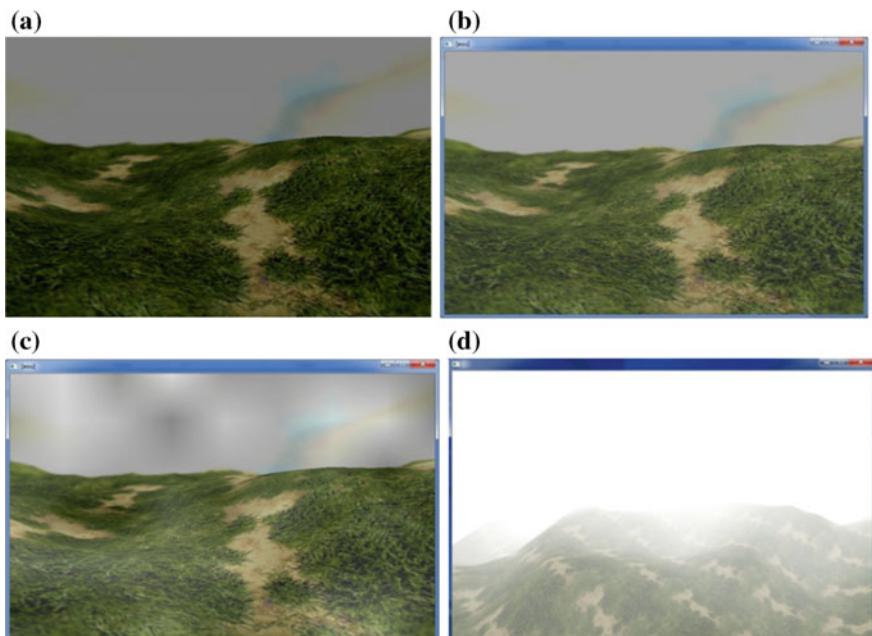
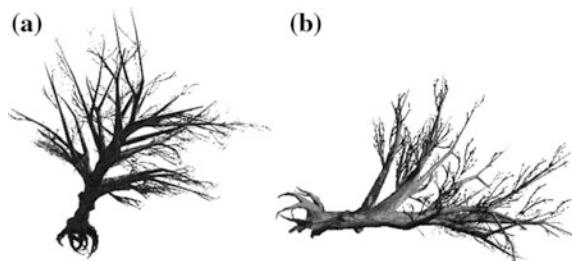


Fig. 10.18 Fog rendering: **a** initial image, **b** color bleaching, **c** low degree of a fog, **d** large degree of a fog

The fog, rain and snow simulation was implemented using the designed software tool “NaturalEffects” based on the library OpenGL. The rendering results of fog, rain, and snow simulation are depicted in Figs. 10.18, 10.19, and 10.20, respectively. The modelling scene can be shifted or rotated by the end-user control. The natural impacts are simulated in the time scale using the hand-tuned parameters.

The designed software tool demonstrates the fast rendering using different texture images of rain drops and snowflakes. The scale location and lighting of particles were considered.

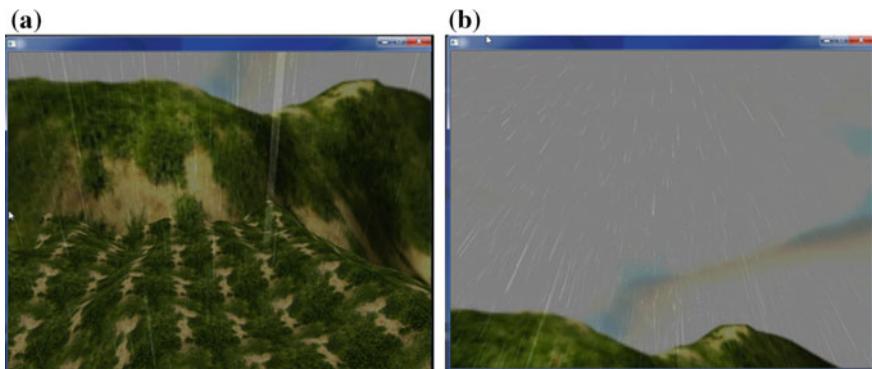


Fig. 10.19 Rain rendering: **a** low degree of a rain, **b** middle degree of a rain

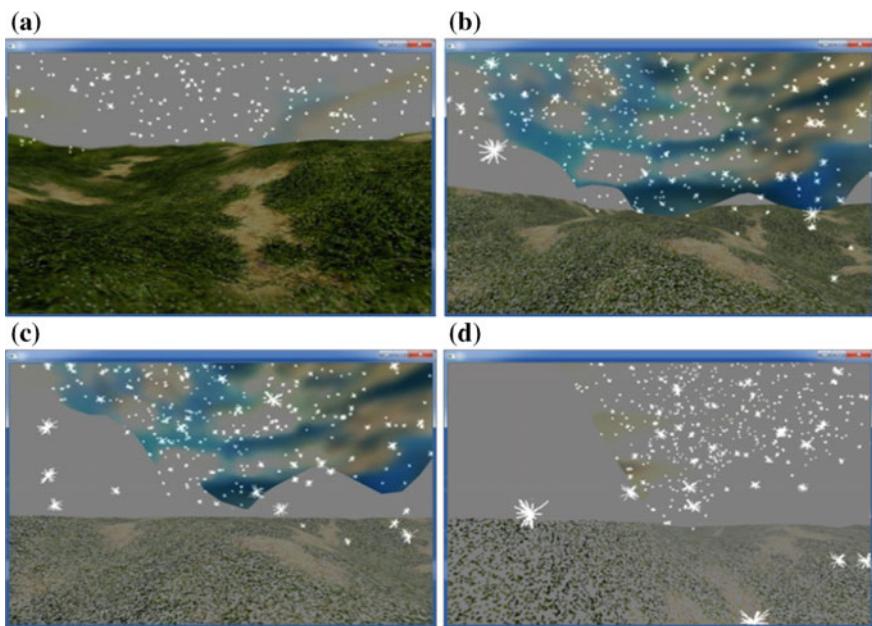


Fig. 10.20 Snow rendering: **a** initial image, **b** image after 19 s of a snowfall beginning, **c** image after 1 min of a snowfall beginning, **d** image after 2 min of a snowfall beginning

10.9 Conclusions

The accurate simulation of meteorological impacts over the real environment is a challenging task due to the great amount of ecological, geometrical, and physical parameters that sometimes cannot be estimated. At the same time, the excess of detailed information is a barrier for real-time implementation, especially when a

human vision does not perceive the details. Nowadays, many models are created to simulate and rendering the common meteorological impacts. However, many additional effects take place in natural environment that do not supported in experimental and commercial software packages. Such simulation may be concern to the future investigations.

References

1. Diener J, Rodriguez M, Baboud L, Reveret L (2009) Wind projection basis for real-time animation of trees. *Comput Graphics Forum* 28(2):533–540
2. Perbet F, Cani MP (2001) Animating prairies in real-time. The 2001 Symposium on Interactive 3D Graphics I3D 2001, pp 103–110
3. Neyret F (1995) Animated texels. In: Terzopoulos D, Thalmann D (eds) Computer animation and simulation 1995, Eurographics 1995. Springer-Verlag, Wien
4. Favorskaya M, Tkacheva A (2013) Rendering of wind effects in 3D landscape scenes. *Procedia Comput Sci* 22:1229–1238
5. Horry Y, Anjyo KI, Arai K (1997) Tour into the picture: using a spidery mesh interface to make animation from a single image. In: 24th Annual conference on computer graphics and interactive techniques SIGGRAPH 1997, pp 225–232
6. Chuang YY, Goldman DB, Zheng KC, Curless B, Salesin DH, Szeliski R (2005) Animating pictures with stochastic motion textures. *ACM Trans Graph* 24(3):853–860
7. Armstrong WW, Green MW (1985) The dynamics of articulated rigid bodies for purposes of animation. *Visual Comput* 1(4):231–240
8. Yang M, Sheng B, Wu E, Sun H (2009) Multi-resolution tree motion synthesis in angular shell space. In: 8th International conference on virtual reality continuum and its applications in industry VRCAI 2009, pp 47–52
9. Pivato D, Dupont S, Brunet Y (2014) A simple tree swaying model for forest motion in windstorm conditions. *Trees* 28:281–293
10. Yang M, Huang MC, Wu EH (2011) Physically-based tree animation and leaf deformation using CUDA in real-time. In: Pan Z, Cheok AD, Müller W (eds) Trans edutainment VI. Springer-Verlag, Berlin Heidelberg
11. Ota S, Tamura M, Fujimoto T, Muraoka K, Chiba N (2004) A hybrid method for real-time animation of trees swaying in wind fields. *Visual Comput* 20(10):613–623
12. Stam J (1997) Stochastic dynamics: simulating the effects of turbulence on flexible structures. *Comput Graph Forum* 16(3):159–164
13. Li C, Deussen O, Song YZ, Willis P, Peter Hall P (2011) Modeling and generating moving trees from video. *ACM Trans Graph* 30(6):Article No. 127
14. James K (2003) Dynamic loading of trees. *J Arboric* 29(3):165–171
15. Pérez F, Pueyo X, Sillion FX (1997) Global illumination techniques for the simulation of participating media. In: Julie Dorsey J, Slusallek P (eds) Rendering techniques 1997. Springer-Verlag, New-York
16. Sang ZQ, Ding MY, Tian-xu Zhang TX (2000) Imaging for outdoor scene in rain and fog. *Acta Electronica Sinica* 28(3):131–133
17. Kokhanovsky AA (1999) Optics of light scattering media—problems and solutions. Wiley, New York
18. Engel WF (2002) Direct3D shaderX: vertex and pixel shader tips and tricks. Wordware Publishing Inc, Plano Texas
19. Shreiner D (2009) OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1, 7th ed The Khronos OpenGL ARB Working Group

20. Stam J, Fiume E (1993) Turbulent wind fields for gaseous phenomena. *Comput Graph* 27:369–376
21. Zhou K, Hou Q, Gong MM, J, B, Shum HY (2007) Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. *Pac Graph* 116–128
22. Garg K, Nayar SK (2006) Photorealistic rendering of rain streaks. *ACM Trans Graph* 25 (3):996–1002
23. Stow CD, Stainer RD (1977) The physical products of a splashing water drop. *J Meteorol Soc Jpn* 55:518–531
24. Garg K, Krishnan G, Nayar SK (2007) Material based splashing of water drops. In: The 8th eurographics conference on rendering techniques EGSR 2007, pp 171–182
25. Wang N, Wade B (2004) Rendering falling rain and snow. *SIGGRAPH* 2004, sketches_0186
26. Wang L, Lin Z, Fang T, Yang X, Yu X, Kang SB (2006) Real-time rendering of realistic rain. Technical Report MSR-TR-2006-102
27. Slomp M, Johnson MW, Tamaki T, Kaneda K (2011) Photorealistic real-time rendering of spherical rain drops with hierarchical reflective and refractive maps. *Comput Animation Virtual Worlds* 22(4):393–404
28. Puig-Centelles A, Ripples O, Chover M (2009) Creation and control of rain in virtual environments. *Visual Comput* 25(11):1037–1052
29. Puig-Centelles A, Sunyer N, Ripples O, Chover M, Sbert M (2011) Rain simulation in dynamic scenes. *Int J Creative Interfaces Comput Graph* 2(2):23–36
30. Sims K (1990) Particle animation and rendering using data parallel computation. *Comput Graph* 24(4):405–413
31. Fearing P (2000) The computer modelling of fallen snow. Ph.D. Thesis, University of British Columbia
32. Chang JK, Ryoo ST (2015) Real time rendering of snow accumulation and melt under wind and light. *Int J Multimedia Ubiquitous Eng* 10(12):395–404
33. Haglund H, Andersson M, Hast A (2002) Snow accumulation in real-time. *SIGRAD* 2002:11–15
34. Fearing P (2000) Computer modelling of fallen snow. In: 27th Annual conference on computer graphics and interactive techniques SIGGRAPH 2000, pp 37–46
35. Langer MS, Zhang L, Klein AW, Bhatia A, Pereira J, Rekhi D (2004) A spectral-particle hybrid method for rendering falling snow. In: 5th Eurographics conference on rendering techniques EGSR 2004, pp 217–226
36. Reynolds DT, Laycock SD, Day AM (2015) Real-time accumulation of occlusion-based snow. *Visual Comput* 31(5):689–700
37. Feldman B, O'Brien J (2002) Modeling the accumulation of wind-driven snow. *Conf Abstr Appl SIGGRAPH* 2002:218–218
38. Maréchal N, Guérin E, Galin E, Mérillou S, Mérillou N (2010) Heat transfer simulation for modeling realistic winter sceneries. *Comput Graph Forum* 29(2):449–458
39. Festenberg NV, Gumhold S (2009) A geometric algorithm for snow distribution in virtual scenes. In: 5th Eurographics conference on natural phenomena NPH 2009, pp 17–25
40. Stomakhin A, Schroeder C, Chai L, Teran J, Selle A (2013) A material point method for snow simulation. *ACM Trans Graph* 32(4):10.1–10.9
41. Gucer D, Ozguc HB (2014) Simulation of a flowing snow avalanche using molecular dynamics. *Turkish J Electr Eng Comput Sci* 22:1596–1610
42. Pirk S, Niese T, Hadrich T, Benes B, Deussen O (2014) Windy trees: computing stress response for developmental tree models. *ACM Trans Graph* 33(6): article No 204
43. Palubicki W, Horel K, Longay S, Runions A, Lane B, Mech R, Prusinkiewicz P (2009) Self-organizing tree models for image synthesis. *ACM Trans Graph* 28(3):58:1–58:10
44. Livny Y, Pirk S, Cheng Z, Yan F, Deussen O, Cohen-Or D, Chen B (2011) Texture-lobes for tree modelling. *ACM Trans Graph* 30(4):53:1–53:10
45. Bridson R (2015) Fluid simulation for computer graphics, 2nd ed. A K Peters/CRC Press, Taylor & Francis Group, LLC

46. Liu G, Liu M (2003) Smoothed particle hydrodynamics: a meshfree particle method. World Scientific Pub Co, New Jersey
47. Oliapuram NJ, Kumar S (2010) Realtime forest animation in wind. In: 7th Indian conference on computer vision, graphics and image processing ACM ICVGIP 2010, pp 197–204
48. Ennos AR, van Casteren A (2010) Transverse stresses and modes of failure in tree branches and other beams. *Proc Roy Soc B* 277:1253–1258
49. Zhang X, Ni S, He G (2008) A Pressure-correction method and its applications on an unstructured chimera grids. *Comput Fluids* 37(8):993–1010
50. Diebold M, Higgins C, Fang J, Bechmann A, Parlange M (2013) Flow over hills: a large-eddy simulation of the Bolund case. *Bound Layer Meteorol* 148(1):177–194
51. Vuorinen V, Chaudhari A, Keskinen J (2015) Large-eddy simulation in a complex hill terrain enabled by a compact fractional step OpenFOAM solver. *Adv Eng Softw* 79:70–80
52. Mirkov N, Rasuo B, Kenjeres S (2015) On the improved finite volume procedure for simulation of turbulent flows over real complex terrains. *J Comput Phys* 287:18–45
53. Kenjeres S, ter Kuile B (2013) Modelling and simulations of turbulent flows in urban areas with vegetation. *J Wind Eng Ind Aerodyn* 123(Part A):43–55
54. Rasouli A, Hangan H (2013) Micro-scale computational fluid dynamics simulation for wind mapping over complex topography terrains. *J Sol Energy Eng* 135(4):1–18
55. Jackson P, Hunt J (1975) Turbulent wind flow over a low hill. *J Roy Meteorol Soc* 101:929–955
56. Castro F, Palma J, Silvia A (2003) Simulation of the Askervein flow. Part 1: reynolds averaged Navier–Stokes equations (k–e Turbulence Model). *Bound Layer Meteorol* 107 (3):501
57. Tsang C, Kwok K, Hitchcock P, Hui D (2009) Numerical study of turbulent wake flow behind a three-dimensional steep hill. *Wind Struct* 5(2–4):317–328
58. Feng W, Fernando P (2011) Large Eddy simulation of stably stratified flow over a steep hill. *Bound Layer Meteorol* 138(3):367–384
59. Abdi DS, Bitsuamlak GT (2014) Wind flow simulations on idealized and real complex terrain using various turbulence models. *Adv Eng Softw* 75:30–41
60. Theekees B, de Langre E, Xavier Boutillon X (2011) Damping by branching: a bioinspiration from trees. *Bioinspiration Biomimetics* 6:1–11
61. James KR, Haritos N (2014) Branches and damping on trees in winds. In: 23rd Australasian conference on the mechanics of structures and materials ACMSM23 2014, pp 1011–1016
62. Hale SE, Gardiner B, Peace A, Nicoll B, Taylor P, Pizzirani S (2015) Comparison and validation of three versions of a forest wind risk model. *Environ Model Softw* 68:27–41
63. Singh PA, Zhao N, Chen SC, Zhang K (2005) Tree animation for a 3D interactive visualization system for hurricane impacts. *IEEE Int Conf Multimedia Expo ICME 2005*:598–601
64. Zdrojewska D (2004) Real time rendering of heterogenous fog on the graphics hardware acceleration. In: 8th Central European seminar on computer graphics, pp 95–101
65. Biri V, Michelin S, Arquès D (2002) Real-time animation of realistic fog. In: 13th Eurographics workshop on rendering, pp 67–72
66. Perlin K (1985) An image synthesizer. *Comput Graph* 19(3):287–296
67. Guo F, Tang J, Xiaoming Xiao X (2014) Foggy scene rendering based on transmission map estimation. *Int J Comput Games Technol* 2014: article ID 308629
68. Tao WB, Jin H, Zhang YM (2007) Color image segmentation based on mean shift and normalized cuts. *IEEE Trans Syst Man Cybern Part B Cybern* 37(5):1382–1389
69. Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. *IEEE Trans Pattern Anal Mach Intell* 23(11):1222–1239
70. van Rossum MCW, Nieuwenhuizen TM (1999) Multiple scattering of classical waves: microscopy, mesoscopy, and diffusion. *Rev Mod Phys* 71(1):313–371
71. Giroud A, Biri V (2010) Modeling and rendering heterogeneous fog in real-time using B-spline wavelets. In: 18th International conference on computer graphics, visualization and computer vision WSCG 2010, pp 145–152

72. Green AW (1975) An approximation for the shapes of large rain drops. *J Appl Meteorol* 21:1578–1583
73. Beard KV, Chuang C (1987) A new model for the equilibrium shape of rain drops. *J Atmos Sci* 44(11):1509–1524
74. Frohn A, Roth N (2000) Dynamics of droplets. Springer-Verlag, Berlin Heidelberg
75. Kubesh RJ, Beard KV (1993) Laboratory measurements of spontaneous oscillations of moderate-size raindrops. *J Atmos Sci* 50:1089–1098
76. Andsager K, Beard KV, Laird NF (1999) Laboratory measurements of axis ratios for large raindrops. *J Atmos Sci* 56:2673–2683
77. Marshall JS, Palmer WMK (1948) The distribution of rain drops with sizes. *J Meteorol* 5:165–166
78. Gunn R, Kinzer GD (1949) The terminal velocity for water droplet in stagnant air. *J Meteorol* 6:243–248
79. Rousseau P, Jolivet V, Ghazanfarpour D (2006) Realistic real-time rain rendering. *Comput Graph* 30(4):507–518
80. Creus C, Patow GA (2013) R4: Realistic rain rendering in realtime. *Comput Graph* 37(1–2):33–40
81. Wang H, Mucha PJ, Turk G (2005) Water drops on surfaces. *ACM Trans Graph* 24(3):921–929
82. de Gennes PG (1985) Wetting: Statics and dynamics. *Rev Mod Phys* 57(3):827–863
83. McClung D, Schaerer P (1993) The avalanche handbook. The Mountaineers, Seattle
84. Moeslund TB, Madsen CB, Aagaard M, Lerche D (2005) Modeling falling and accumulating snow. *Vis Video Graph VVG* 2005:61–68
85. Rasmussen R, Vivekanandan J, Cole J, Karplus E (1998) Theoretical considerations in the estimation of snowfall rate using visibility. Tech. rep. The National Center for Atmospheric Research, Canada
86. Aagaard M, Lerche D (2004) Realistic modelling of falling and accumulating snow. Master's thesis, Aalborg University, Denmark
87. Foldes D, Beneš B (2007) Occlusion-based snow accumulation simulation. In: 4th Workshop in virtual reality interactions and physical simulation VRIPHYS, pp 35–41
88. Cohen M, Wallace J (1993) Radiosity and realistic image synthesis. Academic Press Professional, Boston
89. Hinks T, Museth K (2009) Wind-driven snow buildup using a level set approach. *Eurographics Ireland Workshop Ser* 9:19–26
90. Koenderink JJ, Richards WA (1992) Why is snow so bright? *J Opt Soc Am A*: 9(5):643–648
91. Johanson C (2004) Real-time water rendering: introducing the projected grid concept. Master of Science thesis, Lund University
92. Baboud L, Décoret X (2006) Realistic water volumes in real-time. In: 2nd Eurographics conference on natural phenomena NPH 2006, pp 25–32
93. Hu Y, Velho L, Tong X, Guo B, Shum H (2006) Realistic, real-time rendering of ocean waves. *Comput Animation Virtual Worlds Spec Issue Game Technol* 17(1):59–67
94. Zhang W, Zhou H, Tang L, Zhou X (2010) Realistic real-time rendering for ocean waves on GPU. *IEEE Int Conf Prog Inform Comput PIC* 2:743–747
95. Gamito MN, Musgrave FK (2002) An accurate model of wave refraction over shallow water. *Comput Graph* 26(2):291–307
96. Schneider J, Westermann R (2011) Towards real-time visual simulation of water surfaces. *Vis Model Vis Conf VMV* 2001:211–218
97. Fournier A, Reeves WT (1986) A simple model of ocean waves. *Comput Graph* 20(4):75–84
98. Premoze S, Ashikhmin M (2001) Rendering natural waters. *Comput Graph Forum* 20(4):189–199
99. Enright D, Marschner S, Fedkiw R (2002) Animation and rendering of complex water surfaces. *ACM Trans Graph* 21(3):736–744

100. Ernst M, Akenine-Möller T, Jensen HW (2005) Interactive rendering of caustics using interpolated warped volumes. *Graph Interface GI 2005*:87–96
101. Yuksel C, Keyser J (2009) Fast real-time caustics from height fields. *Visual Comput* 25 (5):559–564
102. Amador GNP, Gomes AJP (2011) A simple physically-based 3D liquids surface tracking algorithm. *Int J Creative Interfaces Comput Graph* 2(2):37–48
103. de la Asuncion M, Mantas JM, Castro MJ (2010) Programming CUDA-based GPUs to simulate two-layer shallow water flows. In: D’Ambra P, Guaracino M, Talia D (eds) *Euro-Par 2010—parallel processing*. Springer-Verlag, Berlin Heidelberg
104. Yu Q (2008) Models of animated rivers for the interactive exploration of landscapes. Thèse de doctorat, Institut National Polytechnique de Grenoble
105. Cords H, Staadt O (2009) Real-time open water environments with interacting objects. In: *5th Eurographics conference on natural phenomena NPH 2009*, pp 35–42
106. Gardner GY (1985) Visual simulation of clouds. *ACM SIGGRAPH Comp Graph* 19 (3):297–304
107. Liao HS, Ho TC, Chuang JH, Lin CC (2005) Fast rendering of dynamic clouds. *Comput Graph* 29(1):29–40
108. Dobashi Y, Kaneda K, Yamashita H, Okita T, Nishita T (2000) A simple, efficient method for realistic animation of clouds. In: *27th Annual international conference on computer graphics and interactive techniques*, pp 19–28
109. Hu X, Sun B, Liang X, He J, Xiao Y, Xiao R, Ren W (2009) An improved cloud rendering method. In: *5th International conference on image and graphics ICIG 2009*, pp 853–858
110. Harris MJ (2003) Real time cloud simulation and rendering. Ph.D. Thesis, North Carolina at Chapel Hill
111. Favorskaya M, Tkacheva A (2016) Wind rendering in 3D landscape scenes. In: Tweedale JW, Neves-Silva R, Jain LC, Phillips-Wren G, Watada J, Howlett RJ (eds) *Intelligent decision technology support in practice*. Springer International Publishing, Switzerland

Part IV
Forest Ecosystem Modelling

Chapter 11

Lighting and Shadows Rendering in Natural Scenes

Abstract In natural scene rendering, the light is the most important factor. The global lighting models are based on the reflection and diffusion of light on the surfaces of objects. The lighting environment can be considered as a composition of the outgoing radiance due to the direct lighting from the sky and the reflection of skylight of the ground, the outgoing radiance due to the direct Sun lighting, and the outgoing radiance due to the indirect lighting from the Sun through neighbour leaves. Also, the bidirectional reflectance distribution functions for different types of materials can be designed using physically-based approach. In this chapter, some models, such as Lambertian surfaces, Phong's reflectance model, Blinn-Phong's reflectance model, and microfacet models, are discussed. In spite of a shadowing is still an expensive component, the virtual environment without shadows cannot be realistic. Some hard and soft shadow techniques, suitable for the natural scene rendering, are represented in the chapter.

Keywords Lighting rendering · Light source · Ray tracing · Natural scene · Shadow rendering · Shadow mapping · Hard shadows · Soft shadows

11.1 Introduction

The simulation of the natural scenes ought to include the lighting and shadows rendering in any time of day and any season. The walkthrough of the natural landscapes means the providing of the best visual quality of a scene with the imposed lighting and shadows conditions. The computation of the lighting may be executed using very complex and high cost physical models as the off-line application. However, the most approaches are based on the simplified models, providing more rapid applications. The scale of rendering plays the significant role during the choice of methods. The rendering of tree covered landscape is based on shading of a huge amount of small shapes (trees) with shadows and sky lighting. The large-scale rendering of the individual trees and shrubs is especially demanding because of the total amount of foliage details. In this case, the model ought to

consider not only a light propagation and a view point of camera but also the interactive processes of light with a foliage material. A great variety of woody species is counterbalanced by the natural hierarchical structures of the trees, when the instances can be represented as a collection of similar and simple objects. However, the requirement of realistic implementation remains.

The organization of this chapter is as follows. Section 11.2 describes related work. Section 11.3 presents the background of the lighting theory that is appropriate for the computer graphics' applications. Section 11.4 is devoted to the simulation of the lighting in modelling natural scenes, while the simulation of the lighting changes is discussed in Sect. 11.5. Some aspects of implementation are mentioned in Sect. 11.6. Finally, this chapter is concluded by Sect. 11.7.

11.2 Related Work

A simulation of the lighting depends strongly from the trees' modelling and the selected way of lighting like as local or global. The simple tree models are based on a billboard rendering based on the Bidirectional Texture Functions (BTF). Thus, Meyer et al. [1] proposed to render the trees using a hierarchy of the BTFs. The BTF textures describe leaves, small branches, main branches, and trunk paying a particular attention on the leaves viewed from different points with various light directions. During rendering, three nearest sampled view vectors are considered. A blending is implemented using the nine selected images taken from the pre-computed textures.

For realistic scenes, the trees should be rendered using the expensive global lighting [2]. Light from the environment is scattered inside the foliage with inter-reflections between leaves. An ambient occlusion is a cheap alternative to global lighting. It provides the convincing visual results, even if they are not accurate really. Reeves and Blau [3] proposed a simple model of ambient occlusion, where an attenuation of the environmental light is based on the distance from the currently rendered leaf to the border of a tree. The farther from the border, the higher the number of leaves hiding the environment. Hegeman et al. [4] modelled a tree by a sphere and the ambient occlusion term was computed per leaf based on its normal. Such approximation is based on a statistical argument in a simple geometric configuration (sphere or other geometrical shapes like ellipsoids) but with several simplifying assumptions. The goal of these authors was not a physical accuracy but a plausible and convincing effect at acceptable computational cost.

In order to increase a visibility of a tree outline under global lighting, Luft et al. [5] defined the envelope of the tree using the implicit surfaces, when each leaf was considered as a metaball. The distance of each leaf to the border of the tree was computed as a part of an envelope model. The normal of each leaf was replaced by the normal of the closest point on the implicit surface. This means that the lighting of the leaves became equivalent to the lighting of the implicit surface. The similar approach based on the gradients of the scalar functions was proposed by Peterson

and Lee [6]. In this case, a tree branches were represented by a sparse set of points, for which the scalar functions were built. However, these two approaches are empirical and do not handle an indirect lighting. The indirect lighting can be estimated using the off-line radiosity-based solutions as it was proposed by Soler et al. [7] or by Chelle et al. [8]. Unfortunately, these methods are not adapted to the real-time rendering.

The light sources are generally modelled as either points or directions. The lighting of a landscape scene can be interpreted as a scene with a single light source with a finite size – the Sun (a sunny day) and as a scene with the multiple light sources that are distributed uniformly and have a reduced lighting (a cloudy day). The last case is a simulation of one infinitely far removed light source (specified by a direction only). The objects cast the shadows, involving an umbra and penumbra. A space under umbra does not receive light from a source, while a space under penumbra obtains a light partially. Sometimes, a space under shadow has a dark central area surrounded by a border area with penumbra. Sometimes, the shadow boundaries determined by a projecting of the silhouette of object/objects are precise without penumbra. The light source in a landscape scene changes its position slowly, and shadows also move and alter their shapes. The position of the light source determines a type of shadow projection, for example, an orthographic projection for infinitely far removed light source or a perspective projection, when a light source position lies outside the field of a view.

The Monte Carlo ray tracing [9], shadow volume [10], and shadow mapping [11] are the traditional real-time shadow techniques for a shadow generation on a non-flat surface. The main ideas of a shadow volume simulation were introduced by Crow in 1977 [10] but they were not realized due to a weak hardware at that time. Crow reviewed three types of shadow algorithms as mentioned below:

- Shadow computation during a raster-scan means that the edges of shadows are projected onto the image plane as the polygon edges onto the surface being scanned. The shadowed surfaces are determined during a scanning procedure that generates the “cutting” planes through a view point. A set of scan segments is defined by the intersection of visible lines and the cutting planes. The color of a scan segment is changed upon crossing a shadow edge.
- The two-passed algorithms involve two steps. First, the shadowed, partially shadowed and non-shadowed surfaces are determined using the polygons, clipping, or “contour” edges techniques. Second, the colors of shadowed surfaces are modified based on the augmented data from the observer’s viewpoint.
- The shadow algorithms compute a projection of the edges onto the surfaces or the surface enclosing the volume of space under a shadow of an object. The umbra surface is treated as an invisible surface that causes a transition into or out of an object shadow.

McCool introduced a combination of shadow mapping and shadow volume [12]. First, a shadow map is generated. Second, a shadow volume is defined using a depth and the light sources in a scene. An excellent overview of the rendering

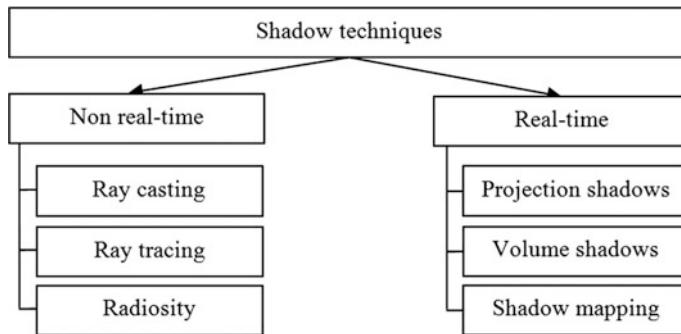


Fig. 11.1 Classification of shadows techniques

shadows in real time can be found in [13]. The classification of shadow techniques is represented in Fig. 11.1. Generally, the non real-time techniques have a high quality and visibility impact. The ray casting is a faster version of the ray-tracing algorithm that uses the ray-surface intersection tests to solve a diversity of problems in computer graphics [14]. The ray tracing is a convenient technique for simulation of the optical effects like reflection and refraction, scattering, and chromatic aberration [15, 16]. It makes a shadow really realistic. The radiosity is a technique to create the soft shadows using a surface representation as a set of elemental patches [17, 18].

The real-time shadowing is more important for implementation of the virtual environment. The projection shadows are the fast rendering method to create a shadow [19, 20]. However, this method can project shadow only on a single flat surface, such as a wall or the ground. A volume shadow method is based on the following idea: a line should be draw from a point of the light source to each vertex and these rays continue infinite [12]. The shadow mapping is very famous technique, building a depth map rendered from a point of view and a light source [21, 22]. Notice that the real-time shadow mapping can be executed with the sub-pixel precision [23]. In spite of the shadow mapping has always been a popular algorithm for fast hard shadow generation since 1978, this approach has the disadvantages, concerning to the principal errors. As Scherzer et al. [24] mentioned, there are several types of errors:

- *Undersampling*. This occurs, when the shadow map samples, projected to the screen, have a lower sampling rate than the screen pixels.
- *Oversampling*. This happens, when the shadow map samples, projected to the screen, have a higher sampling rate than the screen pixels.
- *Reconstruction error or staircase artifacts*. This takes place due to nearest neighbour reconstruction.
- *Temporal aliasing or flickering artifacts*. This derives if a rasterization of the shadow map changes each frame.

The elimination of such errors is the main direction of current shadow mapping algorithms. Two main types of buffers known as Z-buffer [25] and stencil buffer [26] help to create the real-time shadows. The shadow maps were suggested by Fernando et al. [27] as an extension to the traditional shadow mapping techniques.

11.3 Background of Lighting

The scenes in a wild contain a huge amount of small details that are hard to model, render, and shade under the dynamic global lighting. The fundamentals of a lighting are expounded in monograph of McCluney [28]. Some explanations of lighting base, influence of the material properties, and a principle of volume rendering are discussed in Sect. 11.3.1–11.3.3, respectively.

11.3.1 Lighting

The main definitions are introduced in [28].

Definition 1 Radiant energy Q is the quantity of an energy propagating onto, from, or through a surface of a given area in a given period of time. According to the particle nature of light, a radiant energy can be expressed in terms of the number of photons that are propagating onto, from, or through the surface of a given area in a given period of time. The radiant energy is measured in Joules (J).

Definition 2 Radiant flux (or power) Φ is the rate of flow of radiant energy (a quantity of energy transferring) per unit time t . It is expressed in Watts (W) or Joules per second ($J \cdot s^{-1}$) and is defined by Eq. 11.1.

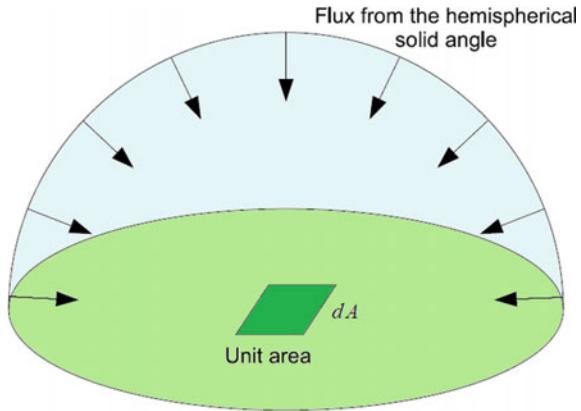
$$\Phi = \frac{dQ}{dt} \quad (11.1)$$

Definition 3 Irradiance E is the radiant flux per unit area dA of a given surface, which is incident on or passing through a point in the given surface. All directions in the hemisphere around the surface point have to be included as it is shown in Fig. 11.2. Irradiance is expressed in W/m^2 ($W \cdot m^{-2}$) and is defined by Eq. 11.2.

$$E = \frac{d\Phi}{dA} \quad (11.2)$$

Definition 4 Radiant exitance B is the same notion as irradiance, except that the energy leaves the surface rather than being incident to it.

Fig. 11.2 The radiant flux per unit area incident at the point of a surface coming from a hemispherical solid angle



Definition 5 Radiant intensity I is the radiant flux per unit solid angle $d\omega$, which is incident on, emerging from, or passing through a point in the space in a given direction (Fig. 11.3). It is expressed in Watts per steradian ($\text{W} \cdot \text{sr}^{-1}$) and is defined by Eq. 11.3.

$$I = \frac{d\Phi}{d\omega} \quad (11.3)$$

Definition 6 Radiance L is the radiant flux per unit solid angle and per projected unit area, which is incident on, emerging from, or passing through a given point of a surface in a given direction (Fig. 11.4). It is expressed in Watts per square meter per steradian ($\text{W} \cdot \text{m}^{-2} \cdot \text{sr}^{-1}$) and is defined by Eq. 11.4, where $dA_p = \cos \theta dA$ is the projected area.

$$L = \frac{d^2\Phi}{d\omega dA_p} = \frac{d^2\Phi}{d\omega \cos \theta dA} \quad (11.4)$$

Fig. 11.3 The radiant flux per unit solid angle $d\omega$ emerging from a point

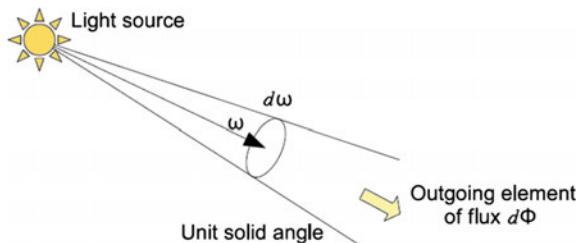
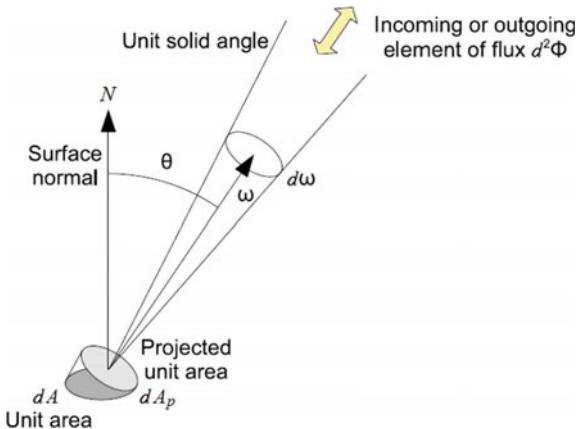


Fig. 11.4 The radiant flux per unit solid angle $d\omega$ and per projected unit area dA_p , which is incident on, emerging from, or passing through a point of a surface in the direction ω



The projected area is the area of a surface that is orthogonal to the solid angle direction ω and projected from the original element of area dA . The angle between the surface normal and the solid angle direction is called θ (Fig. 11.4).

The radiance is the main radiometric quantity that is used to describe a light distribution in an environment. The other basic radiometric quantities can be defined in terms of radiance, which is integrated over a solid angle or an area. The irradiance $E(P)$ at the point P is a summing of the contribution of the radiance from every direction in the hemisphere $\Omega_+(N)$ oriented by normal N provided by Eq. 11.5, where $L_i(P, \omega_i)$ is a given incident radiance from a direction ω_i at the point P on a surface of normal N .

$$E(P) = \int_{\Omega_+(N)} L_i(P, \omega_i) \cos \theta_i d\omega_i \quad (11.5)$$

The cosine of the incidence angle, $\cos \theta_i$, is often defined as the dot product $(\omega_i \cdot N)$.

The spherical coordinates describe the direction vectors $\omega_i \in \Omega_+(N)$ using two angles: the azimuthal angle $\phi_i \in [0, 2\pi)$ and the elevation angle $\theta_i \in [0, \pi/2)$ between ω_i and N . The unit solid angle is expressed as $d\omega_i = \sin \theta_i d\theta_i d\phi_i$. Thus, Eq. 11.5 can be rewritten in a view of Eq. 11.6.

$$E(P) = \int_0^{\pi/2} \int_0^{\pi/2} L_i(P, \phi_i, \theta_i) \cos \theta_i \sin \theta_i d\theta_i d\phi_i \quad (11.6)$$

If $L_i(P, \phi_i, \theta_i)$ is constant for every direction of the hemisphere and equal to $L_i(P)$, then the irradiance is simplified:

$$E(P) = \pi L_i(P). \quad (11.7)$$

Let a radiance in direction ω_o from a small surface around point P of normal N emitting light be $L_o(P, \omega_o)$. The intensity $I(P_l, \omega_o)$ at the center point P is computed by summing a contribution of the radiance for each point of the emitting surface:

$$I(P_l, \omega_o) = \int_{A_l} L_o(P, \omega_o) \cos \theta_o dA. \quad (11.8)$$

Note that the intensity is normally applied to the points. Parameter A_l has very small value compared to the distance of the observer from the light source.

The Bidirectional Reflectance Distribution Function (BRDF) characterizes the reflectance of a surface at a point and is defined as the ratio of outgoing radiance over the incoming irradiance provided by Eq. 11.9.

$$f_r(p, \omega_o, \omega_i) = \frac{dL_o(P, \omega_o)}{dE(P, \omega_i)} \quad (11.9)$$

The light transport (or rendering) equation describes the equilibrium distribution of radiance in an environment. It gives the amount of reflected (or outgoing) radiance $L_o(P, \omega_o)$ from a unit surface dA centered at point P along reflection direction ω_o given the irradiance dE from the environment at this point. It is obtained by integrating the product of the BRDF $f_r(P, \omega_o, \omega_i)$ with the differential irradiance dE provided by Eq. 11.10, where $L_e(P, \omega_o)$ is a radiance due to self-emission, $L_i(P, \omega_i)$ is an incident radiance.

$$\begin{aligned} L_o(P, \omega_o) &= L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) dE(P, \omega_i) \\ &= L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) L_i(P, \omega_i) \cos \theta_i d\omega_i \end{aligned} \quad (11.10)$$

Vector and angles definitions for the BRDF model are located in Fig. 11.5.

The incident light from the direction ω_i is reflected on a point P of surface dA to the direction ω_o , N is a normal to the surface, ω_r is the ideal reflection direction.

The light sources can be the surfaces that emit a light by themselves and are defined by their outgoing radiance. The outgoing radiance $L_o(P_l, \omega_o)$ of a point P_l of a light source is defined by Eq. 11.11.

$$L_o(P_l, \omega_o) = L_e(P_l, \omega_o) \quad (11.11)$$

If there is no participating media causing scattering along the ray P_l , then the radiance is constant along this ray. Therefore, the following Eq. 11.12

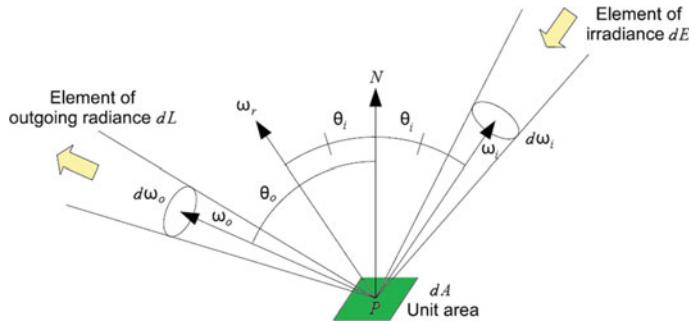


Fig. 11.5 Vector and angles definitions for the BRDF model

$$L_i(P_l, \omega_i) = L_o(P_l, -\omega_i) \quad (11.12)$$

can be used in Eq. 11.5 of the irradiance and Eq. 11.10 of the light transport equation.

The light sources can be classified as the area light sources, the omnidirectional point light sources, the directional light sources, and the environmental lighting.

Area light sources. An area light source is a surface that emits a light. An area light source is integrated over the light source area rather than the subtended solid angle in order to account its contribution in the irradiance and light transport equations. A differential solid angle $d\omega'$ is expressed using the unit area dA' by Eq. 11.13, where dA' is a unit area of the light source, θ' is an angle between the light source surface normal N' and the vector from the light source to the surface being lit, $\widehat{P_l P} = -\omega_i$ (Fig. 11.6a).

$$d\omega_i = \frac{\cos \theta_i dA'}{\|P P_l\|^2} \quad (11.13)$$

An incident light from direction ω_i in unit solid angle $d\omega_i$ in Fig. 11.6a is emitted from the unit area dA' at distance $\|P P_l\|$ from the surface centered in a point $P.N'$ is a normal to the area light surface.

The irradiance at point P due to area light source is expressed by Eq. 11.14

$$E(P) = \int_{A'} L_i(P, \omega_i) \cos \theta_i \frac{\cos \theta' dA'}{\|P P_l\|^2} = \int_{A'} L_o(P_l, -\omega_i) \cos \theta_i \frac{\cos \theta' dA'}{\|P P_l\|^2}, \quad (11.14)$$

while the light transport equation is transformed to Eq. 11.15 with A' the area of the light source surface.

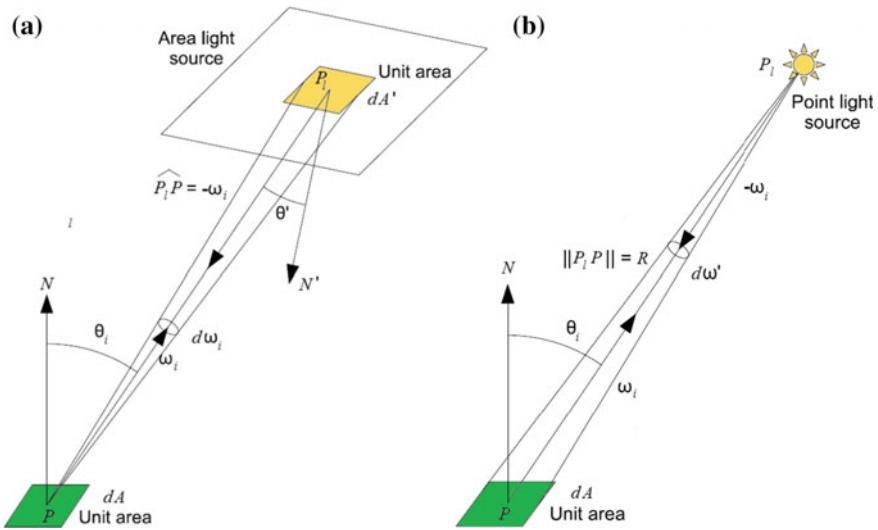


Fig. 11.6 Vector and angles definitions: **a** for an area light source, **b** for a point light source

$$\begin{aligned}
 L_o(P, \omega_o) &= L_e(P, \omega_o) + \int_{A'} f_r(P, \omega_o, \omega_i) L_i(P, \omega_i) \cos \theta_i \frac{\cos \theta' dA'}{\|PP_l\|^2} \\
 &= L_e(P, \omega_o) + \int_{A'} f_r(P, \omega_o, \omega_i) L_o(P_l, -\omega_i) \cos \theta_i \frac{\cos \theta' dA'}{\|PP_l\|^2}
 \end{aligned} \quad (11.15)$$

Omnidirectional point light sources. The point light sources are the light emitters with a zero area. Such approximation greatly simplifies the light transport equation evaluation. The omnidirectional point light sources emit a constant flux in every direction (Fig. 11.6b). A unit area dA centered in a point P receives the flux emitted by the light source in the unit solid angle $d\omega'$ starting from the point light source position P_l and subtended by dA . Since the radiant intensity I represents the flux emitted by the light source per unit solid angle, the irradiance of the unit area dA due to this light source is defined by Eq. 11.16.

$$E(P) = \frac{I \cos \theta_i}{R^2} \quad (11.16)$$

Since flux is received by dA in only one direction, $\omega_i = \widehat{P_l P}$, the light transportation equation becomes Eq. 11.17

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \delta(\omega_i - \widehat{P_l P}) dE(P, \omega_i), \quad (11.17)$$

where $\delta(x)$ is the Dirac operator defined by Eqs. 11.18.

$$\begin{aligned}\delta(x) &= 0 && \text{if } x \neq 0 \\ \int_D \delta(c-x)f(x)dx &= f(c) && \text{if } c \in D\end{aligned}\quad (11.18)$$

Thus, the light transport equation becomes simple (Eq. 11.19).

$$L_o(P, \omega_o) = L_e(P, \omega_o) + f_r(P, \omega_o, \widehat{PP_l}) \frac{I \cos \theta_i}{R^2} \quad (11.19)$$

Directional light sources. The directional light source can be represented as a collimated beam. In this case, the light reaches the surfaces in parallel beams. The irradiance E can be expressed for any virtual surface inside the beam. If the normal N of a surface makes an angle θ_l with the beam direction ω_l , then the irradiance E (P) at any point P of this surface is $E(P) = E_l \cos \theta_l$, where E_l is the irradiance associated with the light beam. In this case, the light transport equation has a view of Eq. 11.20.

$$L_o(P, \omega_o) = L_e(P, \omega_o) + f_r(P, \omega_o, \omega_l)E_l \cos \theta_l \quad (11.20)$$

Environmental lighting. The environment lighting uses a light source at an infinite distance from the rendered scene and surrounding it completely. In this case, the light transport equation (Eq. 11.10) is used to compute a contribution of this light source, except if the incoming radiance from every direction is constant. In the last case, the light transport equation becomes trivial, and the radiance for each incoming direction has to be determined. The most common approach is an environment mapping. It uses an image (environment map or irradiance map), defining the radiance for a finite set of directions in the sphere of incoming light directions. Each pixel of the environment map can be represented as an area light source at an infinite distance, for which the constant outgoing radiance equals the value of the corresponding pixel. The light transport equation can be written using a sum of the contributions of each light source, where each light source k has a subtended solid angle Ω_k from point P (Eq. 11.21).

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \sum_{k=1}^{N_{lights}} \int_{\Omega_k} f_r(P, \omega_o, \omega_i) L_i(P, \omega_i) \cos \theta_i d\omega_i \quad (11.21)$$

The hypothesis can be accepted that the outgoing radiance is constant for each light source. Hence, Eq. 11.21 can be rewritten in a view of Eq. 11.22.

$$\begin{aligned}
L_o(P, \omega_o) &= L_e(P, \omega_o) + \sum_{k=1}^{N_{lights}} f_r(P, \omega_o, \omega_{i_k}) L_i(P, \omega_{i_k}) \cos \theta_{i_k} \int_{\Omega_k} d\omega_i \\
&= L_e(P, \omega_o) + \sum_{k=1}^{N_{lights}} f_r(P, \omega_o, \omega_{i_k}) L_i(P, \omega_{i_k}) \cos \theta_{i_k} \Delta\omega_k
\end{aligned} \tag{11.22}$$

The solid angle $\Delta\omega_k$ for each light source depends on the transformation from image space to world space of the environment map.

A common representation for environment lighting using an image is a spherical environmental mapping (usually latitude-longitude mapping), when a conversion from a spherical to the Cartesian coordinates is performed to convert pixel positions to world space. For an image of N_c columns and N_r rows, the azimuthal angle ϕ is mapped to the horizontal axis and the elevation angle to the vertical axis. The first and last rows represent the poles of the sphere. For a pixel of coordinates (i, j) with $i \in 0 \dots N_c - 1$ and $j \in 0 \dots N_r - 1$, the corresponding spherical angles are defined by Eq. 11.23.

$$\phi(i) = (i + 0.5) \frac{2\pi}{N_c} \quad \theta(j) = (j + 0.5) \frac{\pi}{N_r} \tag{11.23}$$

The solid angle associated with each pixel (i, j) is defined by Eq. 11.24.

$$\Delta\omega(i, j) = \frac{2\pi^2}{N_c N_r} \sin(\theta(j)) \tag{11.24}$$

The irradiance and lighting equations become discrete sums over the light sources and represented by Eqs. 11.25–11.26, where $Env(i, j)$ is a value of a pixel with coordinates (i, j) .

$$E(P) = \frac{2\pi^2}{N_c N_r} \sum_{j=0}^{N_r-1} \sum_{i=0}^{N_c-1} Env(i, j) \max(\omega_i(i, j) \cdot N, 0) \sin(\theta(j)) \tag{11.25}$$

$$\begin{aligned}
L_o(P, \omega_o) &= L_e(P, \omega_o) \\
&+ \frac{2\pi^2}{N_c N_r} \sum_{j=0}^{N_r-1} \sum_{i=0}^{N_c-1} f_r(p, \omega_o, \omega_i(i, j)) Env(i, j) \max(\omega_i(i, j) \cdot N, 0) \sin(\theta(j))
\end{aligned} \tag{11.26}$$

The $\max(\cdot)$ function is used to take into account only one side of the current surface. In this case, the dot product between the incident direction $\omega_i(i, j)$ and the surface normal N is positive.

The environment lighting can be represented using the spherical harmonics [29]. The outgoing radiance of the environment is projected into an orthogonal basis,

giving a set of coefficients. The light transport equation (Eq. 11.10) is a scalar product between the surface BRDF and the irradiance. If both of them are expressed in the spherical harmonics basis, then their scalar product is performed by the product of their spherical harmonics coefficients under condition that one of the BRDF or the irradiance spherical harmonics representations has to be rotated in the same space. Ramamoorthi and Hanrahan [29] claimed that nine coefficients are enough making the evaluation of the light transport equation very efficient.

In previous discussion, the light transport equation was evaluated considering no obstacle between light sources and the receiver surface. However, in normal scenes the obstacles present reducing a contribution of the light sources for a set of incidence directions, creating the shadows. Let us rewrite the light transport equation taking into account the obstacle locations. Equation 11.27 describes the light transport equation, where $V(P, \omega_i)$ is a visibility function equal to 1, when a light source is visible, and equal to 0, when an obstacle prevents in the direction, along which a light reaches the surface unit area.

$$L_o(P, \omega_o) = L_e(P, \omega_o) + \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) V(P, \omega_i) L_i(P, \omega_i) \cos \theta_i d\omega_i \quad (11.27)$$

The integral of the light transport equation is expensive to compute. In order to reach a real-time rendering, a crude approximation can be made to give the illusion of environment lighting with soft self-shadowing of objects. This approximation assumes that incident radiance is constant for every direction, in other words, $L_i(P, \omega_i) = L$ and the BRDF is constant (the Lambertian surface, $f_r(P, \omega_o, \omega_i) = \rho_d / \pi$). Thus, the light transport equation has a view of Eq. 11.28.

$$L_o(P, \omega_o) = L_e(P, \omega_o) + L \frac{\rho_d}{\pi} \int_{\Omega_+(N)} V(P, \omega_i) \cos \theta_i d\omega_i \quad (11.28)$$

An ambient occlusion $AO(P)$ is defined by Eq. 11.29.

$$AO(P) = \frac{1}{\pi} \int_{\Omega_+(N)} V(P, \omega_i) \cos \theta_i d\omega_i \quad (11.29)$$

An ambient occlusion $AO(P) = 1$, when a visibility is equal 1 for every incidence direction. Thus, the light transport equation using constant radiance of the environment, the Lambertian surface and the ambient occlusion can be written by Eq. 11.30.

$$L_o(P, \omega_o) = L_e(P, \omega_o) + L \cdot \rho_d \cdot AO(p) \quad (11.30)$$

Equation 11.30 is inexpensive to compute during rendering and is adopted in the movie industry and video games. An ambient occlusion can be stored per vertex

and interpolated per fragment or can be stored in texels of a texture that is mapped on the lit object. The storage of occlusion data in a volumetric field around lit objects was proposed by Kontkanen and Laine [30].

Global lighting. In previous discussion, the direct lighting (or lighting due to light sources with/without occlusion) was considered. However, the surfaces in any scene are illuminated not only by the light sources but also by light bouncing on neighbour surfaces. The contribution of neighbour surfaces is important for realism in rendered scenes but is very computationally expensive. If N surface unit areas are present in a scene, then $N(N - 1)$ interactions between surfaces have to be considered for a single bounce of light.

For a given surface of the scene, all other surfaces are considered as light sources under the simplifying assumption that the radiance along a ray is constant. This means that the outgoing radiance at a point of a surface in a given direction contributes to the irradiance of a point of another surface. Thus, the outgoing radiance from these surfaces depends on the incoming radiance from the light sources and the other surfaces in a recursive view.

The light transport equation (Eq. 11.10) can be written in a view of Eq. 11.31, where L is the equilibrium light in the scene, L_e is the light due to emission of the light sources, T is the light transport operator.

$$L = L_e + T \cdot L \quad (11.31)$$

A solving Eq. 11.31 for L gives Eq. 11.32.

$$L = L_e(1 - T)^{-1} \quad (11.32)$$

Expansion of Eq. 11.32 to a Neuman series gives Eq. 11.33.

$$L = L_e + T \cdot L_e + T^2 \cdot L_e + T^3 \cdot L_e + \dots \quad (11.33)$$

Expression $T \cdot L_e = L_{direct}$ represents the first bounce of light after leaving the light sources. Therefore, Eq. 11.33 can be rewritten in term of direct lighting (Eq. 11.34).

$$L = L_e + L_{direct} + (T^2 \cdot L_e + T^3 \cdot L_e \dots) \quad (11.34)$$

The content of the parenthesis, representing the second and higher order bounces of light, is considered as $L_{indirect}$. This part is usually the most expensive term to evaluate. Often the evaluation of the first bounce of indirect light L_{direct} is enough, other bounces are rejected.

11.3.2 Material Properties

Each surface in the wild has its own material properties affecting incident light scattering and reflection. The BRDFs $f_r(P, \omega_o, \omega_i)$ provide a description of the reflected light distribution at a point P of a surface given an incident light flux direction ω_i and a reflection direction ω_o (Fig. 11.5). These functions depend on the local micro-geometry but a surface material does not vary spatially. Therefore, the BRDF parameters are the same everywhere on the surface.

The BRDFs have two important properties: reciprocity and energy conservation. The reciprocity rule (Helmholtz reciprocity) gives Eq. 11.35

$$f_r(P, \omega_o, \omega_i) = f_r(P, \omega_i, \omega_o) \quad (11.35)$$

for all pairs of directions ω_o and ω_i in $\Omega_+(N)$. The energy conservation rule requires that the total energy of reflected light is less than or equal to the energy of the incident light. For every direction ω_o in $\Omega_+(N)$ Eq. 11.36 is met.

$$\int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos \theta_i d\omega_i \leq 1 \quad (11.36)$$

Reflectance (also known as albedo) is defined by Eq. 11.37, where Ω_i and Ω_o can be a single direction, a solid angle or the whole hemisphere.

$$\rho(\Omega_i, \Omega_o) = \frac{\int_{\Omega_o} \int_{\Omega_i} f_r(\omega_o, \omega_i) \cos \theta_o \cos \theta_i d\omega_i d\omega_o}{\int_{\Omega_i} \cos \theta_i d\omega_i} \quad (11.37)$$

If Ω_i is a whole hemisphere and Ω_o a single direction, then a hemispherical-directional reflectance is obtained and gives the total reflection in direction ω_o due to the constant lighting defined by Eq. 11.38.

$$\rho_{hd}(\omega_o) = \frac{1}{\pi} \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos \theta_i d\omega_i \quad (11.38)$$

If Ω_i and Ω_o are the whole hemisphere, then a hemispherical-hemispherical reflectance is obtained as a single constant value that represents the fraction of incident light reflected by a surface, when the incident light is constant from every direction (Eq. 11.39).

$$\rho_{hh}(\omega_o) = \frac{1}{\pi} \int_{\Omega_+(N)} \int_{\Omega_+(N)} f_r(P, \omega_o, \omega_i) \cos \theta_o \cos \theta_i d\omega_o d\omega_i \leq 1 \quad (11.39)$$

The BRDF is called isotropic, when only one azimuthal angle is necessary (ϕ_i , ϕ_o , or the difference between these two angles). The BRDF models for different types of materials can be designed with less or more physically-based approach.

Lambertian surfaces. The Lambertian surfaces are the perfect diffuse surfaces that scatter an incident light uniformly in every direction. This model is a very simple analytical model and well-adapted to matte surfaces. The BRDF of a Lambertian surface is defined by Eq. 11.40, where $\rho_d \leq 1$ is the diffuse albedo of the surface (hemispherical-hemispherical reflectance in this case), k_d is a diffuse BRDF, $k_d \leq 1/\pi$.

$$f_r(P, \omega_o, \omega_i) = \frac{\rho_d}{\pi} = k_d \quad (11.40)$$

The hemispherical-directional reflectance can be analytically determined by Eq. 11.41.

$$\rho_{hd}(P, \omega_o) = \frac{\rho_d}{\pi} = k_d \quad (11.41)$$

Phong's reflectance model. The Phong's reflectance model [31] is an isotropic ad hoc model, using simple and intuitive parameters to be defined. It is adapted to plastic-like shiny materials and is defined as the sum of a diffuse reflection term (as for Lambertian surfaces) and a specular term. Afterwards, the Phong's model was modified [32] and now has a view of Eq. 11.42, where ρ_s is the specular albedo (with $\rho_d + \rho_s \leq 1$), k_s is a specular reflectance coefficient, s is a shininess giving an information about the size of the specular highlights (the higher, the smaller are the highlights), ω_r is the ideal reflection vector.

$$f_r(P, \omega_o, \omega_i) = k_d + k_s(\omega_r \cdot \omega_o)^s = \frac{\rho_d}{\pi} + \frac{\rho_s(s+2)}{2\pi} (\omega_r \cdot \omega_o)^s \quad (11.42)$$

The ideal reflection vector ω_r is defined by Eq. 11.43.

$$\omega_r = 2(\omega_i \cdot N)N - \omega_i \quad (11.43)$$

Blinn-Phong's reflectance model. The Blinn-Phong's model [33] is a modification of the Phong's reflectance model. Rather than use of the dot product between the ideal reflection direction ω_r and the reflection direction ω_o , the dot product between the surface normal N and the halfway vector H is applied. The H vector is halfway between ω_i and ω_o . It can be computed by Eq. 11.44.

$$H = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|} \quad (11.44)$$

The reflectance model is defined by Eq. 11.45.

$$f_r(P, \omega_o, \omega_i) = k_d + k_s(N \cdot H)^s \quad (11.45)$$

The shininess s in Eq. 11.45 is different from the Phong's model since the halfway angle is smaller than the angle used by Phong. The value s can be scaled to achieve similar results between the two models. The model provided by Eq. 11.45 is used in the fixed pipeline of 3D graphics cards, through interfaces as OpenGL and DirectX.

Microfacet models. The microfacet models make the assumption that some rough surfaces can be modelled a large set of microfacets. The microfacets are considered very small compared to dA . Therefore, the aggregate behavior of the facets determines the scattering. The most important parameters are the statistical distribution of the facets and the reflection model of the individual facets.

Torrance and Sparrow [34] believed that each facet is a perfect mirror and used the Fresnel's law $F_r(\omega_o)$. The Torrance-Sparrow's model is adapted to metals and brushed metals and is defined by Eq. 11.46, where $D(H)$ is a probability, which normal equals H for a microfacet, $G(\omega_o \omega_i)$ is a geometric term taking self-shadowing into account.

$$f_r(P_l, \omega_o, \omega_i) = \frac{D(H)G(\omega_o \omega_i)F_r(\omega_o)}{4 \cos \theta_o \cos \theta_i} \quad (11.46)$$

Parameter $F_r(\omega_o)$ can be approximated using the Schlick's formula [35] provided by Eq. 11.47, where f_o is the Fresnel factor at normal incidence.

$$F_r(\omega_o) = f_o + (1 - f_o)(1 - (\omega_o \cdot N))^5 \quad (11.47)$$

The Oren-Nayar's model [36] considered that the microfacets are the Lambertian surfaces and that their orientation distribution is based on a Gaussian with σ as the standard deviation of the orientation angle. The BRDF model is defined by Eqs. 11.48.

$$f_r(P, \omega_o, \omega_i) = \frac{\rho}{\pi}(A + B \max(0, \cos(\phi_i - \phi_o)) \sin \alpha \sin \beta), \quad (11.48)$$

where

$$A = 1 - 0.5 \frac{\sigma^2}{(\sigma^2 + 0.33)} \quad B = \frac{0.45\sigma^2}{\sigma^2 + 0.09} \quad \alpha = \max(\theta_i, \theta_o) \quad \beta = \min(\theta_i, \theta_o)$$

However, many materials are not uniform. This means that the reflection properties change for each point of the surface. If surfaces are flat enough, then the Spatial Bidirectional Reflectance Distribution Functions (SBRDFs) that apply the BRDF model for each point of a surface (creating a 6D function) can be used. McAllister et al. [37] proposed the Lafontaine BRDF model [38] for each BRDF. This representation enabled a compact storage of the SBRDF using textures for hardware-accelerated

rendering. McAllister [39] explained how to implement the rendering step efficiently using fragment shaders that compute the local BRDF for each fragment.

The BTFs are adapted to spatially varying surfaces with the meso-structures that are larger than micro-structures modelled by the BRDFs and smaller than the macro-structures modelled using geometry. The BTF can be considered as the SBRDF, where the BRDF for each point of the surface is multiplied by a visibility function to take into account self-occlusion. Also, the BTF is a set of Apparent BRDFs (ABRDF).

The BTFs are 6D functions that require large storage space. Many storage and compression schemes exist for the BTFs [40]. An intuitive storage scheme is a set of images, one for each reflection and incident light direction. An approximation of the surface reflectance is obtained by interpolation of the images for arbitrary reflection and incident light directions. The sets of ABRDFs can be stored using BRDF models adapted to measured data or using compression.

11.3.3 Volume Rendering

2D textures have often been used to represent reflection parameters of flat surfaces. However, many complex surfaces meet in the wild, for example, fur that reflectance properties depend on the position in 3D space. In order to describe such surfaces, Kajiya and Kay [41] introduced the volumetric textures. They introduced a reference volume that is repeated over an underlying surface, consisting of the bilinear patches. They proposed the rendering method called as a ray tracing. When a ray encounters an instance of the reference volume, it is projected inside the reference volume space and approximated by a straight line. Then, an optical model determines a color of the rendering pixel. The ray is point sampled from front to back, and the intensities of the traversed voxels (volume elements) are multiplied by a transparency value cumulated along the ray. An area with density ρ (portion of space occupied by geometry) crossed on a length L has a transparency $e^{-\tau\rho L}$, where τ converts density to attenuation. The final intensity for a ray is $I = \sum_{near}^{far} (I_{loc} \prod_{near}^{current} e^{-\tau\rho L})$ (optical equation). The local lighting I_{loc} is the product of the incident light, the intrinsic reflectance and the albedo (reflected intensity, depending on local density).

Neyret [42] introduced a multi-scale representation for more efficient rendering. During rendering, a coarse representation of data is used, when a viewpoint is far away, while a finer one is considered for a close viewpoint. The reference volume data are stored into an octree.

The main drawback of the ray tracing method is a high rendering time, if a dedicated hardware is not used. By discretizing the optical equation using slices, the classical hardware texturing capabilities can be employed. According to Ikits et al. [43], the accumulated color and opacity (C and A) can be computed using Eq. 11.49, where C_i and A_i are the color and opacity of the i th sample along a ray.

$$C = \sum_{i=1}^n C_i \prod_{j=1}^{i-1} (1 - A_j) \quad A = 1 - \prod_{j=1}^n (1 - A_j) \quad (11.49)$$

Operation A_i approximates the absorption. The opacity-weighted color C_i is an approximation of the emission and the absorption between samples i and $(i + 1)$. These expressions can be iteratively evaluated for a rendering of the samples from back to front using Eqs. 11.50.

$$\hat{C}_i = C_i + (1 - A_i)\hat{C}_{i+1} \quad \hat{A}_i = A_i + (1 - A_i)\hat{A}_{i+1} \quad (11.50)$$

For front to back rendering, Eq. 11.51 is used.

$$\hat{C}_i = (1 - \hat{A}_{i-1}) C_i + \hat{C}_{i-1} \quad \hat{A}_i = (1 - \hat{A}_{i-1}) A_i + \hat{A}_{i-1} \quad (11.51)$$

Operations \hat{C}_i and \hat{A}_i are the accumulated color and opacity from the front of a volume. These operations are efficiently performed using hardware texturing and blending in a real time. With recent hardware, the ray tracing approach can be used in pixel shaders in conjunction with 3D texturing. Engel et al. [44] implemented many approaches for the real-time volume rendering with a lighting using these hardware features.

11.4 Simulation of Lighting in Modelling Scene

The rendering of full ecosystems is a heavy task due to the huge amount of elements to process and the complexity of light interactions between surfaces in a scene. The conventional approach of a computer simulation is to perform the terrain surface of the Earth, trees, shrubs, plants, grass, and other vegetation as the geometric elements having the hierarchical architectures like the L-systems. The light rendering of a nature scene has already been achieved using a ray tracing in spite of that the main drawback is the required computing power. At present time, it is impossible to render a whole scene with the highest level of details (and it is unnecessary due to limitations of a human vision!) and only the close level of details ought to be rendered with the highest realistic in the real-time application.

In previous chapters, several rendering approaches, such as the geometry-based, image-based, and volume-based rendering methods, were discussed. Herein, a study of lighting in the framework of these rendering methods is represented on examples of trees and foliage in Sects. 11.4.1 and 11.4.2, respectively.

11.4.1 Trees Lighting

According to the fundamentals mentioned above in Sect. 11.3, a lighting of trees is always difficult due to the light passes through the multi-layered structures of branches and foliage that have the complex reflectance properties. According to Boulanger [2], a direct lighting $L_{o_{sky}}(P, \omega_o)$ due to the sky light and its reflection off the ground can be estimated as a variable radiance from a low-frequency environment in a view of Eq. 11.52, where P is a center point of a tree envelope, N_{dir} is a sampling of directions ω_j , $s_{\max}(P, \omega_j)$ is a thickness between a lit point P and the light source of radiance $L(\omega_j)$ in direction ω_j , τ is an optical thickness, $L_{sky}(\omega_j)$ is a radiance from the sky in direction ω_j .

$$L_{o_{sky}}(P, \omega_o) \approx \frac{4\pi}{N_{dir}} \sum_{j=1}^{N_{dir}} F(P, \omega_o, \omega_j) e^{-\tau \cdot s_{\max}(P, \omega_j)} L_{sky}(\omega_j) \quad (11.52)$$

This model supposes that a reflection of the sky light off the ground can be computed as the radiance estimation for the hemisphere. The occlusions by other trees or man-made large objects are simulated by a modulating of the incoming radiance. The animations of leaves smooth the variations of an outgoing radiance.

The Sun lighting can be simulated using a directional light ω_l and irradiance E_{sun} , while the outgoing radiance relies on the angle between the light incidence direction and the normal to the lit surface. The visibility term is handled applying different shadowing techniques. The attenuation function provides the overall behavior of shadows. The distance $s_{\max}(P, \omega_l)$ for an arbitrary light direction ω_l can be evaluated respect to the sample directions ω_j . A whole shape of a tree for any direction ω_l is estimated by interpolating the $s_{\max}(P, \omega_j)$ values or by assigning the distance $s_{\max}(P, \omega_l)$, where ω_l is the closest sample direction to ω_l , alternatively.

When the Sun moves, the sudden changes of lighting are possible. The Eq. 11.53

$$\tilde{s}_{\max}(P, \omega_l) = \frac{\sum_{j=1}^{N_{dir}} M(\omega_l \cdot \omega_j) \cdot s_{\max}(P, \omega_j)}{\sum_{j=1}^{N_{dir}} M(\omega_l \cdot \omega_j)} \quad (11.53)$$

approximates the envelope shape using a set of cosine lobes contained in the hemisphere directed by ω_l (non linear due to the M function, $M(\cdot) = \max((\cdot), 0)$). Notice that the computation of a smooth function for \tilde{s}_{\max} using spherical harmonics is a high cost computational procedure due to the evaluation of polynomials and cosine functions.

The final equation of outgoing radiance due to the Sun light has a view of Eq. 11.54.

$$L_{o_{\text{sun}}}(P, \omega_o) = F(P, \omega_o, \omega_l) e^{-\tau \cdot \tilde{s}_{\max}(P, \omega_l)} E_{\text{sun}} \quad (11.54)$$

A contribution of the indirect lighting that includes the huge number of the reflections and transmissions of light within a set of the leaves can be estimated using the probabilistic models. In the simplest case, the lighting contribution of all neighbour leaves is estimated as a sum of the outgoing radiance of each neighbour leaf. This approach does not apply in practice due to a huge amount of computation. Some approximations ought to be executed [45]. At the same time, the indirect lighting due to the sky light requiring the expensive computation can be ignored (its contribution is close to 6%). Also, an offset to the resulting outgoing radiance can be added. However, it is not necessary because even in the overcast days the light is still reaching the bottom of the set of leaves since the ground reflects the light, coming from the sky.

Shadows are the inherent attribute of any modelling scene. Respect to semi-transparent trees, shrubs, and other similar vegetation, the shadows can be classified as the low and high frequency shadows. The soft (low frequency) shadows are known as the indirect lighting. These soft shadows appear due to the lighting from the sky and the light interactions between leaves and branches. The types of hard (high frequency) shadows were introduced by Boulanger [2] as mentioned below:

- From leaves to leaves.
- From leaves to branches.
- From leaves to external objects.
- From branches to leaves.
- From branches to branches.
- From branches to external objects.
- From external objects to leaves.
- From external objects to branches.
- From external objects to external objects.

The traditional methods are based on the shadow mapping or its improvements [46–49] that are applied in many tasks of a scene modelling. The special environment texture based method to project shadows between leaves was proposed by Boulanger [2] as well as a variation of shadow mapping that approximates the soft shadows from natural objects.

The accurate realistic shadowing of leaves and branches is too expensive for the real-time rendering. Thus, some hypotheses, such as the uniform density of leaves, the uniform distribution of leaf normals, and the constant area of leaves, are introduced. This task can be solved using a shadow mask of a texture. This shadow mask contains a visibility function (accepting 0 if a neighbour leaf hides the Sun and 1 if the Sun is visible) for each incoming light direction. Such map can be built for each leaf that is too computationally expensive. Therefore, a single generic mask for a whole tree is a preferred decision. The Sun is considered as a directional light source, and it is evident that the number of neighbour leaves casting shadows varies

based on the Sun direction. Real shadows depend on the location of the leaves and the thickness of the leaf layer to be traversed that can be considered using a gray shadow mask of a texture. The content of a shadow mask is procedurally generated by the parameters of the leaf distribution and the opacity channel of a leaf texture. The multi-layer shadow mask respect to the leaves' positions on the Sun direction can simulate the shadows' cast from the farthest layer that appear naturally larger than those of the closest layer. Notice that sometimes the lighter shadows are required by filtering of the black and white shadow masks.

The shadows cast by trees onto leaves and external objects are a popular scope of discussion. Trees projected on the ground should not be of uniform darkness due to various natural materials of leaves and branches. Also the shadows are getting smoother farther from the trees. Usually the shadow rendering is executed in two passes, such as shadow map building based on the DTM and a rendering of a scene in a whole. The shadow map is recomputed each time, when the light direction changes, and is based on the sparse projection matrix. The shadow maps ought to support a variable opacity using the alpha channel. The leaves area is rendered with a transparency factor of [1...0], while the branches are rendered with a transparency factor of 0.

11.4.2 Foliage Lighting

Boulanger [2] developed the method simulating an indirect lighting, shadows and direct lighting of leaves of a tree. It is considered that the leaves are spatially distributed and oriented according to the probabilistic laws. These leaves are contained in an envelope defined by a discrete function of direction. Such model permits to consider a diffuse inter-reflection within trees, the direct Sun and sky lighting. Also, this method included the material analysis of leaves that influences on the attenuation of light through the leaf layers. Consider this method in details.

Boulanger performs a tree as an envelope of volume V that encloses its leaves. The envelope includes a set of uniformly distributed leaves with a density ρ , a constant area A , and uniformly distributed normals. A leaf approximation, describing by the distances $s_{\max}(P, \omega_j)$ between each leaf's center point P and the envelope for N_{dir} sampling directions ω_j , is a cornerstone of this approach. A leaf shape is approximated by six sectors, more a set of the sampling directions ω_j is the same for all leaves in order to provide the efficient real-time algorithm. A number of the leaves in each sub-volume V_j is provided by Eq. 11.55.

$$N_{leaves_j} = \rho \cdot V_j = \rho \frac{4\pi}{N_{dir}} \frac{(s_{\max}(P, \omega_j))^3}{3} \quad (11.55)$$

A whole volume of a leaf is a sum of sub-volumes V_j for each sampling direction ω_j with a solid angle Ω_j :

$$V = \sum_{j=1}^{N_{dir}} V_j.$$

Lighting environment is a composition of the outgoing radiance due to the direct lighting from the sky and the reflection of skylight of the ground $L_{o_{sky}}$, the outgoing radiance due to the direct Sun lighting $L_{o_{sun}}$, and the outgoing radiance due to the indirect lighting from the Sun though neighbour leaves $L_{o_{ind}}$. Thus, the radiance reaching the viewer $L_o(P, \omega_o)$ is provided by Eq. 11.56, where P is a leaf center, ω_o is a direction (normalized vector) from the leaf to the end-user.

$$L_o(P, \omega_o) = L_{o_{sky}}(P, \omega_o) + L_{o_{sun}}(P, \omega_o) + L_{o_{ind}}(P, \omega_o) \quad (11.56)$$

Boulanger [2] supposed that an object of modelling is a single leaf (not a pixel) and the leaf reflectance properties are Lambertian for both sides for the low-frequency lighting. Thus, a variation of the irradiance along a leaf is assumed negligible. The reflectance is modelled as the texture variations along a leaf. A per-pixel specular component and the transmitted light were modulated using a normal map and a thickness map, respectively. The leaf material model is described by four parameters such as the reflectance ρ_{r_f} of the front face (the same side as N_P), the reflectance ρ_{r_b} of the back face, the transmittance ρ_{t_f} for light entering the front face, and the transmittance ρ_{t_b} for light entering the back face.

The outgoing radiance L_o in direction ω_o from the current leaf centered at P , due to the incident radiance L , is given by Eq. 11.57

$$L_o(P, \omega_o) = \sum_{j=1}^{N_{dir}} \int_{\Omega_j} F(P, \omega_o, \omega) P(P, \omega) d\omega, \quad (11.57)$$

where $F(\cdot)$ is a cosine-weighted bidirectional function

$$F(P, \omega_o, \omega) = \begin{cases} \frac{\rho_{r_f}}{\pi} M(N_p \cdot \omega) + \frac{\rho_{r_b}}{\pi} M(-N_p \cdot \omega) & \text{if } N_p \cdot \omega_o \geq 0 \\ \frac{\rho_{t_f}}{\pi} M(N_p \cdot \omega) + \frac{\rho_{t_b}}{\pi} M(-N_p \cdot \omega) & \text{if } N_p \cdot \omega_o < 0 \end{cases}$$

and $M(x) = \max(x, 0)$.

Leaves of a tree are occluded by other leaves. These shadows are considered as hard shadows because they are sharp and numerous inside a tree. The leaf uniform spatial distribution is described by an attenuation function, which depends on the thickness $s_{max}(P, \omega_j)$ of a leaf layer between a lit point P and the light source of radiance $L(\omega_j)$ in the direction ω_j . The attenuated radiance $L(P, \omega_j)$ at a point P is defined by Eq. 11.58, where τ is an optical thickness, ρ is a density of leaves, A is an area of each leaf.

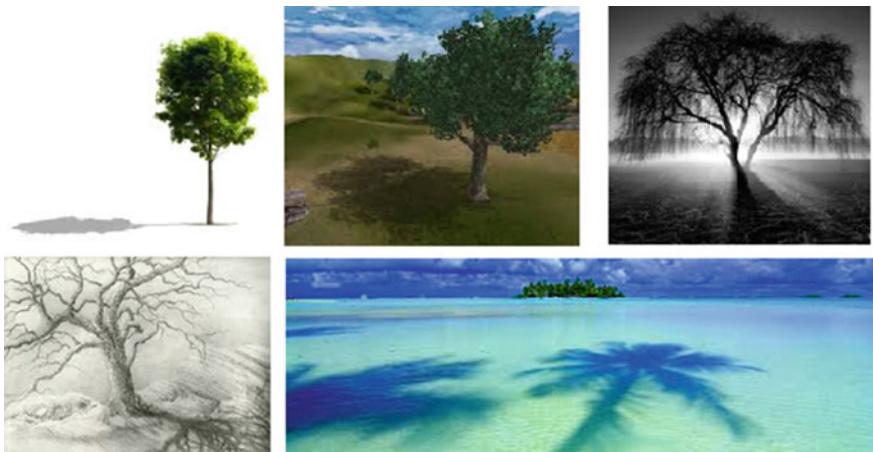


Fig. 11.7 Examples of shadow simulation

$$L(P, \omega_j) = e^{-\tau \cdot s_{\max}(P, \omega_j)} L(\omega_j) \quad \text{with} \quad \tau = \frac{A}{2} \rho \quad (11.58)$$

Equation 11.58 presents a uniform orientation distribution of randomly oriented leaves in area A ($A = 2$ for calculating an optical thickness τ). Max and Ohsaki [50] proposed a similar approach, when a medium contains a high number of spherical particles (Fig. 11.7).

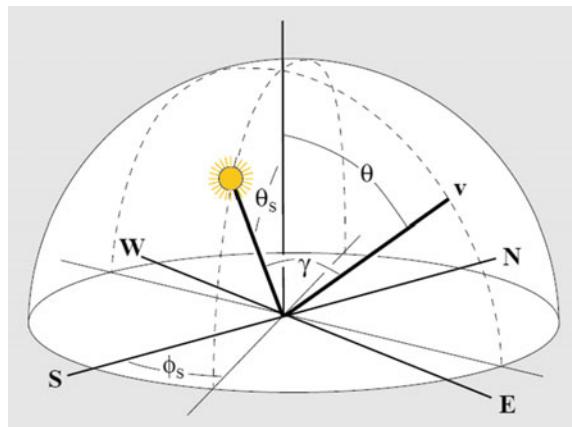
Examples of shadow simulation are depicted in Fig. 11.7.

11.5 Simulation of Lighting Changes in Modelling Scene

For simulation of the lighting changes, the Sun's position ought to be known in order to recalculate the shadow maps. The Earth's oriented North—South line is not exactly perpendicular to the orbit. It has about 23.5° deviation. The Sun's position can be estimated using the longitude and the latitude and the Greenwich Mean Time (GMT). Usually a dome is approximated by a hemisphere, in which a view point is located inside it. The Sun's position in such dome is determined by the zenith and the azimuth. The hemisphere as a half of a sphere by a plane passing thought its center is described in the spherical coordinates by Eq. 11.59, where r is a hemisphere radius, θ is a zenith angle, $\theta \in [0, 2\pi]$, ϕ is an azimuth angle, $\phi \in [0, \pi/2]$.

$$\begin{aligned} x &= r \cos \theta \sin \phi \\ y &= r \sin \theta \sin \phi \\ z &= r \cos \phi \end{aligned} \quad (11.59)$$

Fig. 11.8 The coordinates for specifying the sun position and the direction v on the sky dome



Iqbal [51] proposed a formula to calculate the Sun's position in 1983. Afterwards, Preetham et al. [52] improved it. The Sun's position is given by the angle from zenith θ_s and azimuth angle ϕ_s , which depend on the time of the day, the latitude, and the longitude (Fig. 11.8).

The solar time can be calculated from the standard time using Eq. 11.60, where t is a solar time in decimal hours, t_s is a standard time in decimal hours, SM is a standard meridian for the time zone in radians, L is a site longitude in radians.

$$t = t_s + 0.170 \sin\left(\frac{4\pi(J - 80)}{373}\right) - 0.129 \sin\left(\frac{2\pi(J - 8)}{355}\right) + \frac{12(SM - L)}{\pi} \quad (11.60)$$

The solar declination is approximated by Eq. 11.61, where δ is a solar declination in radians, J is the Julian date (the day of the year as an integer in the range 1–365).

$$\delta = 0.4093 \sin\left(\frac{2\pi(J - 81)}{368}\right) \quad (11.61)$$

The solar position (θ_s, ϕ_s) can be computed from the solar declination angle, the latitude, and the longitude provided by Eq. 11.62, where θ_s is a solar angle from zenith in radians, ϕ_s is a solar azimuth in radians, l is a site latitude in radians, δ is a solar declination in radians, t is a solar time in decimal hours.

$$\begin{aligned} \theta_s &= \frac{\pi}{2} - \arcsin\left(\sin l \sin \delta - \cos l \cos \delta \cos \frac{\pi t}{12}\right) \\ \phi_s &= \arctan\left(\frac{-\cos \delta \sin \frac{\pi t}{12}}{\cos l \sin \delta - \sin l \cos \delta \cos \frac{\pi t}{12}}\right) \end{aligned} \quad (11.62)$$

Due to Eqs. 11.60–11.62, one can estimate the Sun’s position and recalculate the lighting and shadowing parameters in a modelling scene. The issues of shadow rendering using the precise longitude, latitude, and elevation of the actual location of the object as well as the Sun’s position for a given time and day are the subject of discussion provided by Isaza et al. [53].

The building of shadow maps is a traditional approach to quick shadow generation with conventional graphical hardware. However, the shadow maps suffer from aliasing artifacts near the shadow boundaries because of the discretization errors, low resolution, and projection distortions. Wang et al. [54] developed the algorithm that combines the super-sampling filter, geometry-aware reconstruction kernel, and semi-regular sampling filter. This approach permits to generate the real-time subpixel alias-free shadows. The main idea is to use not only the depth values but also the geometry information about scenes for generation of the shadow geometry map. The final shadow map is generated as a convolution of the shadow maps in the layers. This algorithm was implemented using CUDA techniques for a modelling of the plants and trees.

11.6 Implementation

In computer graphics, the Z-buffer called as a depth buffer or shadow buffer is one of the best tools to control 3D environment. Williams was the first, who used Z-buffer in his algorithm to simulate the shadows [11]. Values along OZ axis are stored in this buffer that is located in the main memory. Using the Z-coordinates, a shading of the occluded objects can be computed. Sometimes, this procedure is called as the Z-sorting. The Williams’s algorithm included three phases. First, a whole scene is rendered from the light source as a viewpoint to calculate the depths. Second, a whole scene is rendered using the Z-buffer algorithm. Third, the coordinates of visible points are transformed to the light source coordinate system. Then the checking procedure is applied. If the Z-component is greater than the Z-component that is stored in the Z-buffer, then this point is considered as a shading point, and vice versa in the opposite case. Afterwards, some improving algorithms were proposed.

Crow [10] was the first, who invented the shadow volume approach in 1977. He investigated the front- or back-facing orientations of the consistently rendered shadow volume polygons with respect to the end-user position—inside or outside of the shadow volume. This approach was developed in theoretical and hardware implementations successfully. Fournier and Fussell [55] developed a frame buffer computation model, when each pixel in a frame buffer maintains the depth value and shadow depth count. This computation model lays the theoretical fundamentals for the subsequent hardware stencil buffer-based algorithms.

The stencil buffer as a part of a memory in 3D accelerator controls the rendering of each pixel. In 2002, the stencil test was proposed by Everitt and Kilgard [56]. It is necessary to detect the silhouettes of the occluded objects in order to implement a

shadow volume and it is evident that to form a volume shadow point by point is unreasonable due to a lot of calculations. Kolivand et al. [57] used the silhouettes of the objects and/or silhouette edges of the occluded objects and obtained good results of a shadow volume simulation. If the light source changes its position, then a shadow volume is recalculated. The stencil and the Z-buffer are in a close relationship. The Z-buffer controls the selected pixels with stencil values, when the end-user enters or leaves the shadow volume. The stencil buffer will increase, when an eye's ray enters a shadow volume by passing of a front face of a shadow volume, and it will decrease, when a ray leave a shadow volume by passing of a back face of a shadow volume.

The projected planar shadow techniques are not precise but very simple in computation. A family of simple planar algorithms includes a drawing of a dark shapes similar the occluders under the occluder on the plane, a real-time shadow algorithm [10], a projective algorithm shadowing in the flat surfaces [58], among others. Fast computation and a possibility to manage a shadow as an abstract object with its own properties are the main advantages on these algorithms. The difficulties to have a shadow onto the object's surface differing from a flat surface and a dark shadow without lighting are concerned to the disadvantages. A shadow volume simulation, using the stencil buffer and the depth buffer, is the following step in the history of development of the shadow techniques.

11.7 Conclusions

The lighting and shadowing aspects are one of the main issues in natural scenes' modelling. The promising algorithms can be developed based on the physical fundamentals of lighting. At the same time, the great problem of huge amount of computations appears that makes such algorithms not suitable for the real-time virtual applications. Unfortunately, this contradiction cannot be overcome at the present time. The traditional implementations using Z-buffer technique, CUDA, and parallel processing need in the breakthrough enhancements in order to increase a visibility of the modelling scenes according to the high end-users' requirements.

References

1. Meyer A, Neyret F, Poulin P (2001) Interactive rendering of trees with shading and shadows. In: 12th Eurographics conference on rendering EGWR 2001, pp 183–196
2. Boulanger K (2005) Real-time realistic rendering of nature scenes with dynamic lighting. Ph. D. dissertation, INRIA
3. Reeves WT, Ricki Blau R (1985) Approximate and probabilistic algorithms for shading and rendering structured particle systems. Comput Graph 19(3):313–322

4. Hegeman K, Premoze S, Ashikhmin M, Drettakis G (2006) Approximate ambient occlusion for trees. In: Proceedings of 2006 symposium on interactive 3D graphics and games I3D 2006, pp 87–92
5. Luft T, Balzer M, Deussen O (2007) Expressive illumination of foliage based on implicit surfaces. In: Proceedings of 3th eurographics conference on natural phenomena NPH 2007, pp 71–78
6. Peterson S, Lee L (2006) Simplified tree lighting using aggregate normals. ACM SIGGRAPH 2006 Sketches SIGGRAPH 2006, article No 47
7. Soler C, Sillion F, Blaise F, Dereffye P (2003) An efficient instantiation algorithm for simulating radiant energy transfer in plant models. ACM Trans Graph 22(2):204–233
8. Chelle M, Andrieu B, Bouatouch K (1998) Nested radiosity for plant canopies. Vis Comput 14(3):109–125
9. Cook RL, Porter T, Carpenter L (1984) Distributed ray tracing. In: 11th Annual conference on computer graphics and interactive techniques ACM SIGGRAPH 1984, pp 137–145
10. Crow FC (1977) Shadow algorithms for computer graphics. Comput Graph 11(2):242–248
11. Williams L (1978) Casting curved shadows on curved surfaces. In: 5th Annual conference on computer graphics and interactive techniques ACM SIGGRAPH 1978, pp 270–274
12. McCool MD (2000) Shadow volume reconstruction from depth maps. ACM Trans Graph (TOG) 19(1):1–26
13. Eisemann E, Schwarz M, Assarsson U, Wimmer M (2011) Real-time shadows. An A K Peters book. CRC Press, Boca Raton
14. Maxwell GM, Bailey MJ Goldschmidt VW (1986) Calculations of the radiation configuration factor using ray casting. Comput-Aided Des 18(7):371–379
15. Parker S, Parker M, Livnat Y, Sloan P-P, Hansen C, Shirley P (1999) Interactive ray tracing for volume visualization. IEEE Trans Vis Comput Graph 5(3):287–296
16. Dietrich A, Schmittler J, Slusallek P (2006) World-space sample caching for efficient ray tracing of highly complex scenes. Technical Report TR-2006-01, Computer Graphics Group, Saarland University
17. Ashdown I (1995) Radiosity: a programmer's perspective. Wiley, New York
18. Soler C, Sillion F (1998) Automatic calculation of soft shadow textures for fast, high quality radiosity. In: Drettakis G, Max N (eds) Rendering techniques 1998
19. Stamminger M, Drettakis G (2002) Perspective shadow maps. ACM Trans Graph 21(3):557–562
20. Lloyd DB, Govindaraju NK, Quammen C, Molnar SE, Manocha D (2008) Logarithmic perspective shadow maps. ACM Trans Graph 27(4):1–39
21. Kolic I, Mihajlovic Z (2012) Camera space shadow maps for large virtual environments. Virtual Reality 16(4):289–299
22. Gumbau J, Sbert M, Szirmay-Kalos L, Chover M, Gonzalez C (2013) Smooth shadow boundaries with exponentially warped Gaussian filtering. Comput Graph 37(3):214–224
23. Lecoq P, Marvie J-E, Sourimant G, Gautron P (2014) Sub-pixel shadow mapping. In: 18th meeting of the ACM SIGGRAPH symposium on interactive 3D graphics and games I3D 2014, pp 103–110
24. Scherzer D, Wimmer M, Purgathofer W (2011) A survey of real-time hard shadow mapping methods. Comput Graph Forum 30(1):169–186
25. Catmull EE (1974) A subdivision algorithm for computer display of curved surfaces. Doctoral dissertation, The University of Utah
26. Heidmann T (1991) Real shadows real time. IRIS Universe 18:28–31. Silicon Graphics Inc
27. Fernando R, Fernandez S, Bala K, Greenberg DP (2001) Adaptive shadow maps. In: 28th Annual conference on computer graphics and interactive techniques ACM SIGGRAPH 2001, pp 387–390
28. McCluney WR (2014) Introduction to radiometry and photometry, 2nd edn. Artech House Inc, Norwood, MA, USA

29. Ramamoorthi R, Hanrahan P (2001) An efficient representation for irradiance environment maps. In: 28th annual conference on computer graphics and interactive techniques SIGGRAPH 2001, pp 497–500
30. Kontkanen J, Laine S (2005) Ambient occlusion fields. In: Proceedings of the 2005 symposium on interactive 3D graphics and games SI3D 2005, pp 41–48
31. Phong BT (1975) Illumination for computer generated pictures. Commun ACM 18(6): 311–317
32. Lewis RR (1994) Making shaders more physically plausible. Comput Graph Forum 13(3): 1–13
33. Blinn JF (1977) Models of light reflection for computer synthesized pictures. In: 4th Annual conference on computer graphics and interactive techniques SIGGRAPH 1977, pp 192–198
34. Torrance KE, Sparrow EM (1967) Theory for off-specular reflection from roughened surfaces. J Opt Soc Am 57:1105–1114
35. Schlick C (1994) An inexpensive BRDF model for physically-based rendering. Comput Graph Forum 13(3):233–246
36. Oren M, Nayar SK (1994) Generalization of Lambert's reflectance model. In: 21st annual conference on computer graphics and interactive techniques ACM SIGGRAPH 1994, pp 239–246
37. McAllister DK, Lastra A, Heidrich W (2002) Efficient rendering of spatial bi-directional reflectance distribution functions. SIGGRAPH/EUROGRAPHICS Conf Graph Hardware HWWS 2002:79–88
38. Lafortune EPF, Foo SC, Torrance KE, Donald P, Greenberg DP (1997) Non-linear approximation of reflectance functions. In: 24th Annual conference on computer graphics and interactive techniques SIGGRAPH 1997, pp 117–126
39. McAllister D (2004) Spatial BRDFs. In: GPU gems. Addison-Wesley, Reading
40. MÄuller G, Meseth J, Sattler M, Sarlette R, Klein R (2004) Acquisition, synthesis and rendering of bidirectional texture functions. In: Schlick C, Purgathofer W (eds) Eurographics 2004, state of the art reports, INRIA and Eurographics Association
41. Kajiya JT, Kay TL (1989) Rendering fur with three dimensional textures. Comput Graph 23 (3):271–280
42. Neyret F (1995) A General and multiscale model for volumetric textures. Graph Interface 1995:83–91
43. Ikits M, Kniss J, Lefohn A, Hansen C (2004) Volume rendering techniques. In: GPU gems. Addison-Wesley, Reading
44. Engel K, Hadwiger M, Kniss JM, Rezk-salama C, Weiskopf D (2006) Real-time volume graphics. A. K. Peters, Ltd., Natick
45. Sloan P-P, Kautz J, Snyder J (2002) Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: 29th annual conference on computer graphics and interactive techniques ACM SIGGRAPH 2002, pp 527–536
46. Annen T, Mertens T, Bekaert P, Seidel H-P, Jan Kautz J (2007) Convolution shadow maps. In: 18th Eurographics conference on rendering techniques EGSR 2007, pp 51–60
47. Annen T, Mertens T, Seidel H-P, Flerackers E, Kautz J (2008) Exponential shadow maps. Graph Interface GI 2008:155–161
48. Lauritzen A, McCoolM (2008) Layered variance shadow maps. Graph Interface GI 2008, pp 139–146
49. Kolivand H, Sunar MS (2014) Covering photo-realistic properties of outdoor components with the effects of sky color in mixed reality. Multimedia Tools Appl 72(3):2143–2162
50. Max N, Ohsaki K (1995) Rendering trees from precomputed Z-buffer views. In: Hanrahan PM, Purgathofer W (eds) Rendering techniques '95. Springer-Verlag, Wien
51. Iqbal M (1983) An introduction to solar radiation. Academic Press, Toronto
52. Preetham AJ, Shirley P, Smith B (1999) A practical analytic model for daylight. Comput Graph SIGGRAPH 1999:91–100
53. Isaza C, Salas J, Raducanu B (2013) Rendering ground truth data sets to detect shadows cast by static objects in outdoors. Multimedia Tools Appl 70(1):557–571

54. Wang R, Wu YQ, Pan MH, Chen W, Hua W (2012) Shadow geometry maps for alias-free shadows. *Inf Sci* 55(11):1–12
55. Fournier A, Fussell D (1988) On the power of the frame buffer. *ACM Trans Graph* 7(2): 103–128
56. Everitt C, Kilgard MJ (2003) Practical and robust stenciled shadow volumes for hardware-accelerated rendering. <http://arxiv.org/abs/cs/0301002>. Accessed 02 Jun 2016
57. Kolivand H, Sunar MS, Jusoh NM, Folorunso OA (2011) Real-time shadow using a combination of stencil and the Z-Buffer. *Int J Multimedia Appl* 3(3):27–38
58. Xiao Y, Jin Yicheng J, Yong Y, Zhuoyu W (2006) GPU based real time shadow research. *Int Conf Comput Graph Imaging Visualisation CGIV* 2006:372–377

Chapter 12

Modelling of Forest Ecosystems

Abstract The main goal of the forest ecosystems' modelling is to develop such approaches that can merge the general nature of processing models with the precision and predictive power of the empirical models. The long-term prediction of forest growth plays the significant role in a legal forest management. Some attempts to develop the prototype systems for data analysis and decision-making at forest enterprise level are implemented. However, this complicated scope requires the coordinated efforts of biologists, mathematicians, and programmers. The current situation is that the good biological and rendering models exist separately. In future, the generalized modelling ought to combine the influence of soil resources, also atmospheric, meteorological, and human impacts on the forest life-cycle with the realistic modelling and rendering of large-scale forests in the short-term and long-term frameworks.

Keywords Forest ecosystems · Forest scene · Access to resources · Soil's moisture and minerals · Climate · Solar radiation · Forest fire

12.1 Introduction

The modelling and rendering of forest scenes depend on their application strongly. On the one hand, the industry of games requires the real-time rendering of large-scale forest landscape scenes with quality and fast transmission through a network in the on-line games [1]. In this case, the methods of computer graphics are applied using the techniques of realistic and highly detailed forest scenes with real-time shadows. The traditional Level Of Details (LODs) approach in conjunction with the compact mathematical L-system representation permits to solve this task. The main goal is to provide a high quality and real-time visualization in a short time period, for example, daytime or nighttime. On the other hand, forests, woodlands, and shrubs formations are very important ecosystems. They are required a deep learning in seasons, years, and decades including a competition for access to natural resources and influence of various external factors like forest fires,

natural phenomena, and a human intervention in ecological environment. The modelling of forest ecosystems' development and degradation is the main subject of interest in this case. The airborne, terrestrial, and mobile laser data as well as the digital photographs are the objective source of information. It can be expected that a good virtual visualization will be more and more claimed and evident function, converting the ecological modelling into the high quality serious games.

The interaction between trees as well as the influence of the environment had been neglected by the early models. At present, any garden, architectural design, or virtual reality applications reveal the need of such interactions. The goal in modelling of the individual trees is to obtain the natural look during a rendering. At the same time, the goal in rendering of several trees still persists enforcing by several environmental factors, such as locations of trees, their sizes, and interactions. Weber and Penn [2] were the first, who visualized the forests and shrubs, using the procedural geometrical models and ray-tracing rendering. The approach proposed by Weber and Penn included the featured level of detail adaptation. This means that a few thousand trees can be visualized in a landscape scene. Mech and Prusinkiewicz [3] were the first, who classified these interactions into three different categories, as mentioned below:

- Global properties of the environment define the daily maximum and minimum temperatures, affected the growth rate, and the day length controls a flowering.
- Local properties of the environment include the obstacles (walls, buildings, etc.) that have a great effect on the rooting direction and the supporting constructions for the climbing plants.
- Bi-directional effects between the environment and plants mean the competition for space, resources in minerals, moisture, and light.

The space is fulfilled by trees, shrubs, and other vegetation according to their phototropism property, when the trees try to spread their leaves as wide as it is possible to receive a light for the photosynthesis. The interaction between individual plants of the same type can be easily modelled by the systems using the open L-systems. The external forces, such as a wind or rain, act on the branching structures and can modify a tree shape. The bending of branches is simulated by rotation of the turtle in the directions of external forces.

The practical significance of forest modelling is very high in the framework of the close-to-nature forestry. At last decades, this technique has been practiced successfully in Europe. The achievements in computer graphics facilitate the process of forest monitoring, including the real walkthrough the woods and the manual measurements of many trees. The development cycles of natural forests can be used successfully in forest management planning [4]. The basic principles of close-to-nature silviculture are directed on the high-quality timber production, protecting of trees' regeneration, maintaining of a biodiversity, and ensuring of a continuous forest cover. Notice that the unified optimal strategy for close-to-nature forestry cannot be provided due to a wide range of ecological, economic, social, and cultural aspects of the territories.

De Turckheim [5] formulated the recommendations for the management strategy applied to a tree level:

- Individual treatment of each tree according to the functions assigned at the ecosystem level.
- Long-lasting maintenance of the qualities of the ecosystem, including the use of species adapted to the site and restricted use of exotic species, a cutting only in the case of natural disasters, and a preserving the greatest biomass possible without a compromising regeneration in order to increase the stability of the ecosystem.
- Promoting natural regeneration under big trees across large areas.
- A short rotation time: 6–12 years.
- Restricted cutting of trees with sufficient diameter that is determined by the maximum economic timber value for the species.
- Removal of the diseased trees and storage of the dead trees having a great ecological meaning.

As it is mentioned by Martin-Fernandez and Garcia-Abril [6], the possible functions taken on by the tree include the following ones:

- Productive on their own or protecting of the productive adjacent trees.
- Biodiversity including the rare species, a dead or dying tree, and a protection of the ecosystem (e.g. growing on a steep slope or around rivers or lakes, harbouring nests or wildlife).
- Recreational involving historical trees and trees in a recreational area.

The close-to-nature management ensures that the regeneration occurs at a sufficient rate, the large trees contribute to the self-thinning process, and natural pruning and intervention is not required. The single decision in a forest management at the tree level is whether to cut the best quality trees or not. Following this concept, all trees can be classified as “main” or “complementary”. The “main” trees in the forestry are represented in one of three categories [6]:

- The stabiliser (a dominant and co-dominant tree) concerns to two sub-types: a tree that has not reached its maximum productive state and continues to grow and a tree that has reached its maximum productive state and/or has a lost vitality.
- The sprinter is an outstanding tree because of its height. This is an intermediate state between regeneration and stabiliser.
- The regeneration is a tree that increases the stability of the ecosystem.

The “complementary” trees are the remaining trunks that are needed to support the “main” trees provide a soil cover or protect other trees. All trees are classified according to their dendrometric variables, e.g., diameter, height, crown diameter and crown height. The descriptive variables control a health condition and harvesting impact, while the environmental variables describe an altitude, slope, and orientation of a tree.

The chapter begins from the summarizing of related work in Sect. 12.2. Section 12.3 describes the study area of forest visualization. The main proposition of simulation a forest as an ecological system is introduced in Sect. 12.4. The modelling of living conditions is explained in Sect. 12.5. Section 12.6 concluded the chapter.

12.2 Related Work

For individual tree or plant modelling, the well-known approaches, such as the point/line-based rendering [7, 8], image-based rendering [9], polygon-based rendering, and volume-based rendering, are suitable. The forest model representation complicates the modelling by the competition between individual trees into space, lighting, water, and mineral resources.

Zhang et al. [10] proposed to use the hierarchical layered depth images for forest modelling. The discrete textured quadrilaterals form a hierarchy structure using the sampled depth images of polygonal tree models that provides a fast mapping of a large-scale forest. However, the real-time shadow and dynamic lighting require high storage cost. Liu et al. [11] adopted a dynamic quad stream based on the GPU buffers for a view-dependent rendering. Preliminary, a set of hierarchical layered billboards accordingly to a scene's depth are generated. In a run-time stage, the geometry tessellation is used in order to progressively refine the tree models. The rendering speed and detailed quad meshes provide a high visual quality but the main disadvantage of a billboard approach, such as a non-real-time shadowing, is inherent to Liu et al. methodology.

At present time, a volume-based rendering is very attractive approach. It implies a set of boxes, containing one or more instances to be rendered. Cohen et al. [12] introduced the GPU-based lighting and shadowing of complex natural scenes that added the realism to volume-encoded forest scenes but required the precomputing filtering textures and horizon maps. Decaudin and Neyret [13] utilized the volumetric textures to generate the plants' geometry as a series of parallel image slides and tile over the ground to represent the forests. Herewith, a volume representation offered a full parallax effect from any viewing direction that cannot be possible using billboards. However, a walkthrough and into forest modelled in such manner is not yet possible. Afterwards, the same authors [14] improved their previous method using the GPU geometry shader, when a set of the volumetric billboards presented the plant model foliage. As a result, a walkthrough and into the forest became possible but the memory cost was so high that few trees can be rendered only.

Colditz et al. [15] presented the algorithm to optimally approximate plant models using the extended billboards that contain not only color information inside the textures but also information about the normals to a view point into a separate normal texture. This permitted to update the direction of the normals at the billboards similar to the normals of the geometry model. The volumetric billboards approach is depicted in Fig. 12.1 [14].

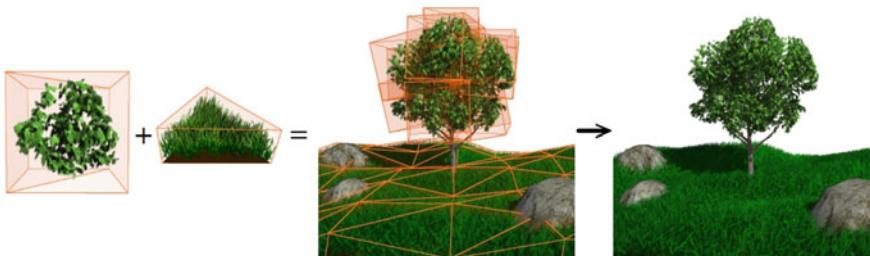


Fig. 12.1 Scheme of the billboards approach from [16]

Notice that the multi-resolution modelling and the LOD algorithms are frequently used methods for the polygon decimation and geometry compression. However, these methods cannot be applied to forest' modelling due to the complex topologies of the trees and other vegetations.

The close scales require the conceived foliage representation. The most methods, devoting to foliage modelling and rendering, deal with the broad-leaves trees. Gumbau et al. [17] developed a foliage pruning method that cuts the non-visible vegetation, depending on the end-user's location. This real-time algorithm has two stages. In a preprocessing stage, the LOD tree models' foliage is divided into a cloud of cells, and each cell is associated with a set of the external viewpoints. In a run-time stage, each cell evaluates the viewpoint position to calculate a percentage value, then a list of polygons for the visible leaves is generated based on this value, and, finally, the size and color of the remaining geometry are rendered. This method is efficient for the GPU implementation but cannot efficiently achieve the real-time shadowing effects. The shadows do not recalculate every time, only on the first frame. Bao et al. [18] suggested the approach of a leaf geometry modelling based on different texture images. Afterwards, Bao et al. [1] developed the LOD shadow maps to render the large-scale forest shadows. Deng et al. [19] proposed a LOD-based method for coniferous leaves that simplifies the coniferous leaves in a view of cylinders and lines. Generally, the automatic generation of the multiple LODs for trees is a non-trivial task because foliage contains many isolated surfaces.

The competition between trees and plants to access for resources like space, light, and below-ground nutrients is the basis for the evolution. One of the aspects of such competition is the strangler trees that germinate and grow on other trees. Okamoto [20] introduced the model that characterized the dynamics of a mutant stranger lineage invading a clean forest community. The model describes the individual trees as a set of finite discrete patches. Each patch is characterized into the following states:

- Patch is empty.
- Patch is occupied by a non-strangling tree.
- Patch is occupied by a non-strangling host and an epiphytic stranger.
- Patch is occupied by a stranger.
- Patch is occupied by a stranger hosting other epiphytic stranger.

The stranglers are modelled as sending out their seeds that germinate on a non-strangling host. The average number of successfully germinating strangler seeds per unit time describes the colonization of non-strangling hosts by stranglers by the parameter γ . Upon germination, the stranglers are in symbiosis with their hosts during an epiphytic life stage. The parameter ω describes, how quickly stranglers develop their germination. Principally, the stranglers can establish on other stranglers with a rate v . When $v = \gamma$, the strangler propagules can colonize non-strangling hosts and other strangling host alike. An inequality $v = \gamma$ implies a different rate of colonization of both host types. Some stranglers can grow without the host trees, e.g., the bare rock surfaces. The goal of this modelling is to estimate how many empty patches remain. The forest is modelled by the k patches considering a direction by use of the system of equations proposed by Okamoto.

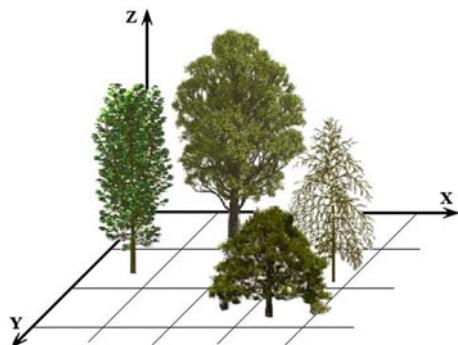
The Dynamic Vegetation Models (DVMs) permit to simulate the forest properties at large spatial scales using a forward modelling of ring-width growth of trees. A drought impacts on a variety of plant physiological processes modifying the structure, functioning, and vitality of individual trees at both the short and the long term. Thus, some models of a tree-ring growth, such as the full Vaganov–Shashkin model [21], the VS-Lite forward model [22], the modified VS-Lite forward model [23], among others, were developed.

Some investigations are devoted to the Digital Elevation Model (DEM) that has been used to derive the hydrological features of territories. This elevation data are available from several sources and at different spatial resolutions: the National Elevation Dataset (NED), the Shuttle Radar Topography Mission (SRTM) data, and the Light Detection and Ranging (LiDAR) data. Li and Wong [24] studied the outcomes of two hydrologic applications, namely river network extraction and flood simulation that may influence on natural environment.

12.3 Forest Scene Modelling

A forest scene simulates the current state of a forest community that includes several tiers usually. All changes during several hours are not significant excluding the ecological catastrophes, such as the forest fire, flood, deforestation, or rarely falling of an old tree. In the case of density forest, it is enough difficult to wait the plausible LiDAR data from the low tiers of a forest. A priori data about type of forest, main species of trees are strongly required. Using the LiDAR data and digital photography, one can simulate the locations of the shrubs and trees under the assumption that the crowns of close trees are often overlapped. The simplest forest model supposes that a space is divided into 3D cellular structure and a crown of a single tree is located in a single 3D cell or, as an exception, in the neighbor cells in horizontal or vertical planes [25]. The sizes of such cells are chosen from the experimental data of a wild forest area. As a result, each tree is associated with some cell and has its space coordinates that permits to simplify essentially a forest model and decrease an execution time of modelling and rendering. The day lighting

Fig. 12.2 Example of a spatial structure of forest scene



for each tree is well modelled by a space division in 3D cells. A spatial structure of forest scene is depicted in Fig. 12.2.

The basic structure unit is a tree, and a forest modelling accumulates the results of dynamic modeling of the individual trees. According to this assumption, an individual development of some tree relative to the spatial distribution of other trees is considered. In this model, the local barriers (as a part on planes under different angles) can be set near a growth tree if an old tree with a big crown is located closely. The L-system with geometrical limitations is used, when a tree cannot develop well in these restricted cells.

Decaudin and Neyret [13] proposed a method of large forest scene rendering. This method used the LODs and the Graphic Processor Unit (GPU) in order to achieve a real-time performance. The main assumptions of this method are the following. Forest coverage is represented by a volumetric texture layer. Decaudin and Neyret introduced a new term “texcell” (unlike the term “texel”, texture pixel that was introduced by Kajiya and Kay [26]) as a volumetric instance of the pattern that has a base triangle on the ground. Two different types of texcells were introduced. The regular texcells with good quality and efficiency except at the silhouette are sliced parallel to the ground. The silhouette texcells (more expensive) are sliced parallel to the screen. This representation permitted to transform a simple vertex shader into the vertices of the GPU-resident mesh. The LODs were created by adapting the slicing rate to 3D mipmap level. The mapping of the texcells is aperiodic. Shading and shadows are not considered in this real-time application. The patterns cannot be rotated; at least eight patterns are required during rendering. However, Decaudin and Neyret received the good rendering result at that time.

Candussi et al. [27] improved the approach of Decaudin and Neyret adding the lighting and shadowing in the large forest scene. Three types of geometric primitives composed a tree: the deformed cylindrical meshes (roots, trunk, major branches, and sub-branches), the grid meshes (minor branches), and the billboards (leaves). Leaves were grouped into clusters, each cluster was rendered as a separate billboard, and the billboards ought to be as few as possible for realistic representation. Roots, trunk, and major branches were rendered with bump mapping and shadow mapping. Grid meshes had shadow mapping and per vertex lighting. Leaf

lighting was obtained as the product of four factors: per vertex lighting, density factor, shadow map factor, and occlusion factor.

Pirk et al. [28] developed a method for a dynamic tree modelling based on the L-systems that allows to interact with the environment, including the obstacles of various shapes. If a tree has been grown close to an obstacle, then it changes its shape, bending or shedding. This model involves two parts. First part changes the main branching skeleton bending or pruning according to the light distribution. Second part calls the procedural content of the leaf clusters that is modified under the local light distribution. Small branches and twigs bend towards the light and can be pruned if they interact with a solid obstacle. The morphological parameters of the tree branches are estimated under the assumption that the light distribution is the most important factor for a tree growth. The achieved results are very impressive. It is interesting that the L-systems can be applied to the root modelling providing the parameters for the root branching order (main axis, its daughter axes, etc.) [3].

12.4 Forest Ecosystems

Deussen et al. [29] created the first ecosystem simulator used in computer graphics according to some natural laws of spontaneous afforestation. The tree models had the structures of the L-systems and were rendered using ray casting or ray tracing. Two opposite approaches from global to local and from local to global were applied in order to estimate a credible distribution of trees. The global to local (first) approach was semi-automatic and required a standard paint program. The local to global (second) approach calculated the distribution automatically and included the natural laws of spontaneous afforestation considering the interaction among the trees themselves and the environment.

Zamuda and Brest [30] elaborated the laws of interaction of Deussen et al. included in their second approach using the bottom-up software agents. The first agent had a tendency (for the same species of trees) to grow in clusters emerging the effects of clustering [31], clumping [32], and under-dispersion [33]. The second agent was a competition among the individual trees, where one tree influences on the growth of others in an area of an ecologic neighborhood. The intersection of two ecological neighborhood areas leads to the trees interaction, when one tree becomes dominated. When a tree is being the dominated, its growth becomes slower or it may even die. Such competition results in a phenomenon of a tree distribution known as self-thinning: when a density of trees increases, the dominated trees start dying off. The competitive ability depends on the age of the tree, domination tolerance, water concentration at its location, and its preference for wet or dry areas.

Zamuda and Brest [30] simulated the life of trees within an ecosystem as a three sub-steps: the creation of new trees, their growth, and the elimination of the dead trees. The height above sea level, slope, moisture, and the intensities of the sun and wind were the main factors of living conditions. The simulation step interval was one year. This algorithm simulated the lives of several hundred thousand trees over

several hundred years starting from the empty DTM. New trees were added using the under-dispersion principle, when a probability of chosen locations increased for new trees close to existing trees of the same species. The model of the tree growth considered the height of a tree, the persistent clinging to life, and the suitability of living conditions for a tree. The first factor was the growth obstruction, caused through domination by more powerful trees within the neighborhood. The second factor is the non-uniform growth intensity throughout a tree life, when the young trees grow faster than the old ones. Three causes of death, such as the natural life-cycle, intolerableness of living conditions, and suffocation, were modelled in this simulation system.

The competitive ability $a_{i,s}$ of i th tree from the tree species s can be expressed by Eq. 12.1, where h is a height of a tree, pc is a persistent clinging to life (this parameter is used to show an influence between the forest's tiers), lc is a suitability of living conditions for a tree.

$$a_{i,s} = f(h_{i,s}, pc_{t,s}, lc_{i,s}) \quad (12.1)$$

It does not seem possible to find the dependences between a tree height, a persistent clinging, and a suitability of living conditions for each tree species in the analytical view. The reasonable recommendation is to use the stored data of natural measurements during many years' observations of forests. Notice that it is interesting to simulate a growth of the real forest area based on the data obtained from the airborne/terrestrial/mobile laser shooting.

One of the important components of the forest ecosystems is the lying dead wood. Lying dead wood provide a habitat for various plants and animals [34], facilitate a tree regeneration [35], contribute to soil organic material through the forest nutrient cycles [36]. Also, a dead wood is a significant element of the total carbon stock in forests [37]. Polewski et al. [38] studied the spatial distribution of a dead wood in forests using qualitative and quantitative data. They developed an integrated, data driven method for detecting single tree trunks from the unstructured ALS point clouds given by 3D coordinates of their constituent points under the assumption that the reference data in the form of the accurate fallen trunk positions, lengths, and diameters are available. This method involves two-step procedure: a detection of trunk segments in the point cloud and a merger of the segments to produce the individual fallen trees. The output of this method is a list of subsets of the original points that correspond to the individual detected fallen trees.

12.5 Modelling of Living Conditions

A tree growth is the most durable sub-step in a life-cycle of a tree and the living conditions are a decisive factor of this period. Generally, the living conditions are determined by a set of parameters, including the geographical location (type of a

climate), ecological factors, and influence of a human activity. The multi-valued dependencies between these parameters make the modelling of the living conditions very complicated. Thus, the qualitative descriptions of soil's moisture and minerals, wind, temperature, and solar radiation influences are represented in Sects. 12.5.1–12.5.4, respectively. Forest fire risk evaluation is discussed in Sect. 12.5.5.

12.5.1 Soil's Moisture and Minerals

The moisture intensity of a soil is determined by two main factors: the water absorption under influence of a snow covering, flood, rain, fog, or dew and the water outflow. The water absorption depends on the climate, terrain topology, a slope in particular, and a soil absorption rate that is defined by the mineral components of a soil. The water outflow is caused by the high moisture levels of territories, lakes, rivers, seas that accumulate the water. It is reasonable to assume the uniformly distributed precipitation through a territory, however, due to the mentioned above factors, a non-uniform soil moisture distribution can be expected.

The probability map of a soil moisture distribution can be built by estimating of the water flow and outflow for each patch of a territory, considering the heights of neighbor patches through every season. The good model of soil moisture can be geographically dependent based on the multi-years observation only. The same is possible to tell about the map of a minerals' distribution.

If the modelled resources are considered independent of species specific interactions with the environment, then the models are simple and understandable intuitively. If the models are based on data obtained from the durable observations, then these models lose a generality. Here, the simple models, suitable for many types of territories, will be discussed.

The foundations of climatological scale processes of precipitation and hillslope scale processes of drainage were introduced by Beven and Kirkby [39]. Thus, the water input to plants was formalized by multiplying the annual precipitation by the wetness index so that the moisture input m_i (m^3 per meter contour length per year) was determined by Eq. 12.2, where w is a wetness index, p is a precipitation.

$$m_i = wp \quad (12.2)$$

Notice that the moisture input is a physically based index correlated to the actual moisture available to the plants. The moisture input can be calculated on an annual basis because the relationship between the precipitation and elevation in the study area is known for an average year. (More complex models consider the climate data across the elevation range in a study area.)

The precipitation (m/year) is calculated by Eq. 12.3, where E is the elevation from the DEM, m.

$$p = -415.9 + 0.636E \quad (12.3)$$

Equation 12.3 was empirically determined for a sub-basin of the Kananaskis [40]. The 8-year study used the eleven precipitation gauges at elevations, ranging from 1780 to 2285 m.

The local hillslope profiles distribute the precipitation provided by Eq. 12.3. The wetness is the amount of water draining into a point divided by the rate water would be flowing through that point. For example, when two regions have the same catchment area, the steeper one will be drier. The model of the hillslope wetness can be computed by Eq. 12.4, where B is a slope angle in a region, A is the catchment area, m^2/m , showing the size of the area, which collects the water flows through 1 m of a contour length [39].

$$w = \ln(A / \tan B) \quad (12.4)$$

Even such simple model of hillslope processes has been shown to be effective, despite its simplicity and the assumption of uniform ground water loss [41]. More detailed models include a soil texture and variation in the water flux within the drainage area [42].

The catchment area values for the wetness model, Eq. 12.4, are based on an accumulation algorithm, where the water from each pixel on the DEM flows into the lowest elevation pixel of its eight neighboring pixels. First, the accumulation algorithm is used to determine the upslope area of each pixel by calculating how many drops are drained into each pixel. This can be done by tracing the flow of water upslope from the pixel until the point, where the pixels in the path are no longer the lowest in a neighborhood, and water ceases to flow into them. Second, the median of a 3×3 pixels' neighborhood is calculated in order to smooth out some of the high variance in accumulation. Third, the check of the minimum upslope area, at least one pixel, is implemented. Also, the pixels are converted to meters in order to receive the result of the area drained per meter contour length. The angle of the slope B can be calculated as the angle of the plane formed by the vector connecting the left and right neighbor pixels, and the vector connecting the upper and lower neighbors of the pixel.

12.5.2 Wind

The wind rendering in natural scenes is based on two types of models. As shown in survey of Zhang and Pang [43] the models for separate types of multi-level vegetation and co-influence models, connecting with growing and stagnation of some plants under their life-cycle, received the most propagation. Some researches connect with very complex biological modelling that requires many initial parameters and their values. Thus, in original research of Qu et al. [44], an agent-based functional-structural model for orange tree growth simulation was

developed based on 42 various input parameters considering a photon flux density, an air temperature, a soil humidity, the state transitions, etc. This is a prediction model for orange tree growth in several years duration. Such modeling is based on very complex probability dependences.

Another type of wind rendering models is focused on a geometry reduction technique in 2D or 3D spaces. Three basic models of a wind rendering concerning to the weak wind, mid-force wind, and storm rendering were developed in [45]. The weak and mid-force wind models find more often applications into a landscape scenes modelling. The storm hazard is a rarely natural effect that may occur once during several years. The landscape-scale interactions on the relation between the storm damage and forest management have been discussed in a few studies, for example, in [46–48]. The relationship between several factors, such as forest management alternatives, climate change impact, and disturbances caused by storms, was studied by Ray et al. [49]. Seidl et al. [50] proposed to combine a process-based model of wind-disturbance with a forest growth model in order to analyze the impact of alternative silvicultural treatment regimes on storm damage using some scenario simulations.

In the aspect of a tree growth, the preferred direction of a wind ought to be defined for any season. This leads to the slope of the trees along the preferential direction of the winds with a relative height loss and a non-uniform lighting of trees.

12.5.3 Temperature

For the North hemisphere, the mean July maximum temperature in degrees Celsius T_J can be estimated by empirical Eq. 12.5, where E is an elevation, m.

$$T_J = 32.7 - 0.00771 E \quad (12.5)$$

The linear regression relationship in a view of Eq. 12.5 was determined over an 8-year period for a sub-basin of the Kananaskis watershed. The study used seven weather stations located from 1370 to 2285 m [51]. This study also found that the highest temperatures occurred in July, therefore, this represents the optimal growing month.

12.5.4 Solar Radiation

The Sun intensity distribution can be estimated based on the astronomical models for the Earth's rotation around the Sun. The different territories receive different amounts of the Sun's radiation on a surface that directly influenced on a degree of

photosynthesis and is defined a degree of vegetation growth. In literature, some models of the Sun's radiation can be found, such as a prediction model of the amount of solar energy falling on inclined absorbing surfaces [52], a general expression for the solar radiation incidence angle in terms of the slope and surface azimuth [53], a comprehensive account of solar energy sources and conversion methods [54], a solar radiation model based on defining incidence angle by computing normal-to-the-surface tangent plane and direction of the Sun [55], a meteorological radiation model [56]. Sometimes it is reasonable to consider a hemisphere model, when the Sun is never directly in its zenith at noon but rather some degrees lower to the South of the North (for the northern hemisphere).

The radiation measurements on the horizontal surfaces for hourly interval can be estimated easily. As mentioned Braun and Mitchell [53], the hourly incident radiation on the surface of any orientation is evaluated by the ratio of incident radiation on the tilted surface to that on a horizontal surface considering beam, sky diffuse, and ground reflected radiation separately. Evaluating beam radiation on a tilted surface requires knowledge of the Sun position relative to both the surfaces, normal and vertical. The incident sky diffuse and ground reflected radiation depend upon the view factors between the surface and the sky and the surface and the ground. These both factors are described by functions of the position of the surface relative to the horizontal.

Clouds and fog can be considered as a filter that strengthens scattering of an incoming radiation in the atmosphere. Thus, a number of cloudy or foggy days is one of the most significant climate data. Among climate parameters, the duration of the solar radiation is the most important data for the meteorological model. This parameter is commonly recorded with the Campbell–Stokes heliograph.

If the surface is not exposed to the Sun, it is in the shadow. There are two types of shadow—hill shadow and cast shadow. The hill shadow occurs, when the surface is oriented away from the light source. The cast shadow occurs only on surfaces that are orientated towards the Sun, when there is a topographic obstacle between the Sun and the surface. The hill shadow can be easily determined if the incidence angle is previously calculated, while the evaluation of the cast shadow is more complex. Solar radiation is divided into direct and diffuse. The surfaces in the shadows receive only the energy of diffuse radiation that is normally at least twice less than the energy of a direct solar radiation. Therefore, shadow (hill and cast) determination is an important part of the model.

Three types of data are usually used in such models:

- Astronomical data (the Sun declination, the Earth–Sun distance).
- Surface data (the DEM).
- Climate data (zenith angle of the Sun, duration of solar radiation, type of clouds, transmission coefficients relative to absorption and dispersion, temperature, relative humidity, and atmospheric pressure).

The details can be studied from [53, 55].

Muneer et al. [57] remarked that the most common sources of errors arise from the sensors and their construction and declared the general types of errors as mentioned below:

- Cosine response.
- Azimuth response.
- Temperature response.
- Spectral selectivity.
- Stability.
- Non-linearity.
- Shade-ring misalignment.
- Dark offset (nocturnal) long-wave radiation error.

In addition to the above sources of equipment-related errors, the operational errors may be caused by various sources:

- Operation related problems and errors.
- Complete or partial shade-ring misalignment.
- Dust, snow, dew, water-droplets, bird droppings, etc.
- Incorrect sensor leveling.
- Shading caused by building structures.
- Electric fields in the vicinity of cables.
- Mechanical loading of cables.
- Orientation and/or improper screening of the vertical sensors from ground-reflected radiation.
- Station shut-down.

Notice that the meteorological radiation models are based on the regressions between the ratio of diffuse irradiance to beam irradiance, diffuse to beam ratio, and beam clearness index. The following detailed information about these issues is outside of this book.

12.5.5 Forest Fire Risk Evaluation

Climate impacts on forests directly and indirectly providing not only the growth of trees but also affecting on the disturbance regimes, such as a wildfire frequency. The wildfire risk is one of the most important indirect factors, strongly interlinked with climate changes [58–60]. However, the increased temperatures alone do not necessarily mean that more fires will occur. Several other climatic and non-climatic factors are also involved, such as ignition sources, fuel loads, vegetation characteristics, rainfall, humidity, wind, topography, landscape fragmentation, and management policies. These factors may lead to the increased fire hazard and depleted soil moisture [61].

Some meteorological indices help to estimate a wildfire risk. Thus, Angström [62] developed a simple instantaneous meteorological index relating fire risk to relative humidity Rh and temperature T from field experiments in Sweden. The Angström index AI is calculated using Eq. 12.6.

$$AI = \frac{Rh}{20} + \frac{29 - T}{10} \quad (12.6)$$

The AI provides lower values, when fire risk is higher. Fire is generally considered “very likely” at values less than 2.0, and “unlikely” at values over 4.0.

Nesterov (1949) constructed a cumulative index, where each day’s calculated value is added to that from the previous day. Each daily increment N_d is calculated by Eq. 12.7, where T_{\max} and T_{dew} are the daily maximum and dew-point temperatures, respectively.

$$N_d = T_{\max}(T_{\max} - T_{\text{dew}}) \quad (12.7)$$

The temperature T_{dew} is equal to the daily minimum temperature T_{\min} . The accumulated Nesterov index value NI is provided by Eq. 12.8, where W is a number of days since 4 mm of rainfall was collected.

$$NI = \sum_{i=1}^W N_{d_i} \quad (12.8)$$

The Nesterov index (Table 12.1) is currently used as an official indicator of fire risk in Russia [63] and in Austria [64].

The Keetche–Byram Drought Index (KBDI) [66] was specifically designed as a fire weather warning system. The metric version, showing a daily addition to moisture deficiency dK [67], has a view of Eq. 12.9, where KBDI_{t-1}^* is a previous day’s moisture deficiency less the net rainfall, R_{av} is a local annual average precipitation.

$$dK = 10^{-3} \frac{(203.2 - \text{KBDI}_{t-1}^*) (0.968 e^{(0.0875 T_{\max} + 1.5552)} - 8.3)}{1 + 10.88 e^{-0.001736 R_{av}}} \quad (12.9)$$

The KBDI value increases daily according to temperature and humidity, and decreases by the net rainfall, when a rainfall P over consecutive days exceeds

Table 12.1 Risk classes of the Nesterov index [65]

Index range	Class	Description
<300	1	No fire risk
301–1000	2	Low fire risk
10,001–4000	3	Medium fire risk
4001–10,000	4	High fire risk
>10,000	5	Very high fire risk

5.1 mm. Net rainfall is the depth of the rainfall minus the 5.1 mm buffer. The threshold values have a view of Eq. 12.10 [68].

$$\begin{aligned} \text{KBDI}_t &= \text{KBDI}_{t-1} + dK \quad \text{if } P_t = 0 \\ \text{KBDI}_t &= \text{KBDI}_{t-1} + dK \quad \text{if } P_t > 0 \quad \text{and} \quad \sum P = 5.1 \text{ mm} \\ \text{KBDI}_t &= \text{KBDI}_{t-1}^* + dK \quad \text{if } P_t > 0 \quad \text{and} \quad \sum P > 5.1 \text{ mm} \end{aligned} \quad (12.10)$$

The KBDI values are intended as a direct indicator of soil water deficit. In its original form, it estimated the amount of net precipitation necessary to bring the soil to saturation. In the metric form, this index is measured in mm and ranges from zero (saturated soil) to a maximum of 203.2 mm. Keetch and Byram [66] divided the index into eight “stages” but point out that the significance of each stage will depend on local climatological conditions. Geographical variations in the KBDI values across Austria were studied by Petritsch and Hasenauer [69].

The variables “soil water” and “labile litter carbon” are used in the ecophysiological model BIOME-BGC for predicting of a fire ignition [70]. Afterwards, the BIOME-BGC model was modified by Eastaugh and Hasenauer [71].

12.6 Conclusions

In this chapter, some models of forest ecosystems are presented. The basic concept of such modelling involves three sub-steps, such as the creation of new trees, their growth, and elimination of dead trees. The modeling complexity is caused by the environmental interactions of individual trees, shrubs, and other vegetation under the main factors of living conditions: territorial (the height above sea level, slope), soil’s properties (moisture, minerals), climatic (wind, precipitation, solar radiation, temperature), human (industrial development, legal/non-legal cutting, forest fire), among others. The reliable forest ecosystems’ modelling is possible only using the analysis of big data obtained from many years of observations of forests around the world.

References

1. Bao G, Li H, Zhang X, Dong W (2012) Large-scale forest rendering: real-time, realistic, and progressive. *Comput Graph* 36(3):140–151
2. Weber J, Penn J (1995) Creation and rendering of realistic trees. 22nd annual conference on computer graphics and interactive techniques SIGGRAPH 95, pp 119–128
3. Mech R, Prusinkiewicz P (1996) Visual models of plants interacting with their environment. 23rd annual conference on computer graphics and interactive techniques SIGGRAPH 1996, pp 397–410

4. FAO, ECE, ILO Committee on Forest Technology, Management and Training (2003) Report of the seminar on close-to-nature forestry, Zvolen, Slovakia (2016). <http://www.unece.org/fileadmin/DAM/timber/joint-committee/facts.htm>. Accessed 11 July 2016
5. De Turckheim B (1999) Bases Economicas de la Selvicultura Proxima a la Naturaleza. Technical Meeting of Pro Silva, Valsain, Spain
6. Martin-Fernandez S, Garcia-Abril A (2005) Optimisation of spatial allocation of forestry activities within a forest stand. *Comput Electron Agric* 49(1):159–174
7. Gilet G, Meye A, Neyret F (2005) Point-based rendering of trees. 1st Eurographics conference on natural phenomena NPH 2005, pp 67–73
8. Simons L, He S, Tittmann P, Amenta N (2014) Point-based rendering of forest LiDAR. Workshop on visualisation in environmental sciences EnvirVis 2014. [10.2312/enviris.20141105](https://doi.org/10.2312/enviris.20141105)
9. Fuhrmann AL, Umlauf E, Mantler S (2005) Extreme model simplification for forest rendering. 1st Eurographics conference on natural phenomena NPH 2005, pp 57–66
10. Zhang H, Hua W, Wang Q, Bao H (2006) Fast display of large-scale forest with fidelity. *Comput Anim Virtual Worlds* 17(2):83–97
11. Liu F, Hua W, Bao H (2010) GPU-based dynamic quad stream for forest rendering. *Sci China Inf Sci* 53(8):1539–1545
12. Cohen F, Decaudin P, Neyret F (2004) GPU-based lighting and shadowing of complex natural scenes. ACM SIGGRAPH 2004 posters SIGGRAPH 2004, p 91
13. Decaudin P, Neyret F (2004) Rendering forest scenes in real-time. Eurographics symposium on rendering techniques 2004, pp 93–102
14. Decaudin P, Neyret F (2009) Volumetric billboards. *Comput Graphics Forum* 28(8):2079–2089
15. Colditz C, Coconu L, Deussen O, Hege C (2005) Real-time rendering of complex photorealistic landscapes using hybrid level-of-detail approaches. In: Buhmann E, Paar P, Bishop I, Lange E (eds) Trends in real-time landscape visualization and participation. Wichmann Verlag
16. Volumetric billboards. <http://phildec.users.sourceforge.net/Research/VolumetricBillboards.php>. Accessed 07 July 2016
17. Gumbau J, Chover M, Remolar I, Rebollo C (2011) View-dependent pruning for real-time rendering of trees. *Comput Graph* 35(2):364–374
18. Bao G, Li H, Zhang X, Che W, Jaeger M (2011) Realistic real-time rendering for large-scale forest scenes. In: IEEE international symposium on VR innovation ISVRI 2011, pp 217–223
19. Deng Q, Zhang X, Gay S, Lei X (2007) Continuous LOD model of coniferous foliage. *Int J Virtual Reality* 6(4):77–84
20. Okamoto KW (2015) The dynamics of strangling among forest trees. *J Theor Biol* 384:95–104
21. Vaganov EA, Hughes MK, Shashkin AV (2006) Growth dynamics of conifer tree rings: images of past and future environments. *Ecological Studies*. Springer, Berlin
22. Tolwinski-Ward SE, Evans MN, Hughes MK, Anchukaitis KJ (2011) An efficient forward model of the climate controls on interannual variation in tree-ring width. *Clim Dyn* 36 (11):2419–2439
23. Marco Mina M, Martin-Benito D, Bugmann H, Cailleret M (2016) Forward modeling of tree-ring width improves simulation of forest growth responses to drought. *Agric For Meteorol* 221:13–33
24. Li J, Wong DWS (2010) Effects of DEM sources on hydrologic applications. *Comput Environ Urban Syst* 34(3):251–261
25. Favorskaya M, Zotin A, Chunina A (2011) Procedural modeling of broad-leaved trees under weather conditions in 3D virtual reality. In: Tsihrintzis GA, Virvou M, Jain LC, Howlett RJ (eds) Intelligent interactive multimedia systems and services. Springer, Berlin
26. Kajiya J, Kay T (1989) Rendering fur with three dimensional textures. *Comput Graphics* 23 (3):271–280

27. Candussi A, Candussi N, Höllerer T (2005) Rendering realistic trees and forests in real time. *Eurographics* 2005:73–76
28. Pirk S, Stava O, Kratt J, Said MAM, Neubert B, Mech R, Benes B, Deussen O (2012) Plastic trees: interactive self-adapting botanical tree models. *ACM Trans Graphics* 31(4):Article No. 50
29. Deussen O, Hanrahan P, Lintermann B, Mech R, Pharr M, Prusinkiewicz P (1998) Realistic modeling and rendering of plant ecosystems. 25th annual conference on computer graphics and interactive techniques SIGGRAPH 1998, pp 275–286
30. Zamuda A, Brest J (2013) Environmental framework to visualize emergent artificial forest ecosystems. *Inf Sci* 220:522–540
31. Benes B, Guerrero JMS (2004) Clustering in virtual plant ecosystems. In: International conferences in Central Europe on computer graphics, visualization and computer vision WSCG 2004, pp 9–16
32. Lane B, Prusinkiewicz P (2002) Generating spatial distributions for multi-level models of plant communities. *Graphics interface 2002 conference GI 2002*, pp 69–80
33. Hopkins B (1954) A new method for determining the type of distribution of plant individuals. *Annals Botany* XVIII:213–226
34. Siitonen J, Martikainen P, Punttila P, Rauh J (2000) Coarse woody debris and stand characteristics in mature managed and old-growth boreal mesic forests in Southern Finland. *For Ecol Manage* 128(3):211–225
35. Weaver JK, Kenefic LS, Seymour RS, Brisette JC (2009) Decaying wood and tree regeneration in the acadian forest of Maine, USA. *For Ecol Manag* 257(7):1623–1628
36. Harmon ME, Franklin JF, Swanson FJ, Sollins P, Gregory SV, Lattin JD, Anderson NH, Cline SP, Aumen NG, Sedell JR, Lienkaemper GW, Cromack K Jr, Cummins KW (2004) Ecology of coarse woody debris in temperate ecosystems. *Adv Ecol Res* 34:59–234
37. Woodall C, Heath L, Smith J (2008) National inventories of down and dead woody material forest carbon stocks in the United States: challenges and opportunities. *For Ecol Manage* 256 (3):221–228
38. Polewski P, Yao W, Heurich M, Krzystek P, Stilla U (2015) Detection of fallen trees in ALS point clouds using a normalized cut approach trained by simulation. *ISPRS J Photogramm Remote Sens* 105:252–271
39. Beven KJ, Kirkby MJ (1979) A physically based, variable contributing area model of basin hydrology. *Hydrological Sci Bull* 24:43–69
40. Storr D, Ferguson HL (1972) The distribution of precipitation in some mountainous Canadian watersheds. Geilo symposium—distribution of precipitation in mountainous areas (WMO/OMM), vol 326(1), pp 243–263
41. Wolock DM, Hornberger GM, Beven KJ, Campbell WG (1989) The relationship of catchment topography and soil hydraulic characteristics to lake alkalinity in the North-eastern United States. *Water Resour Res* 25:829–837
42. O'Loughlin EM (1986) Prediction of surface saturation zones in natural catchments by topographical analysis. *Water Resour Res* 22(5):794–804
43. Zhang Q-L, Pang M-Y (2008) A survey of modeling and rendering trees. In: Pan Z, Zhang X, Rhalihi AE, Woo W, Li Y (eds) Technologies for E-learning and digital entertainment. Springer, Berlin
44. Qu H, Wang Y, Cai L, Wang T, Lu Z (2012) Orange tree simulation under heterogeneous environment using agent-based model ORASIM. *Simul Model Pract Theory* 23:19–35
45. Favorskaya M, Tkacheva A (2016) Wind rendering in 3D landscape scenes. In: Tweedale JW, Neves-Silva R, Jain LC, Phillips-Wren G, Watada J, Howlett RJ (eds) Intelligent decision technology support in practice. Springer, Switzerland
46. Zeng H, Talkkari A, Peltola H, Kellomaki S (2007) A GIS-based decision support system for risk assessment of wind damage in forest management. *Environ Model Softw* 22(9):1240–1249
47. Gardiner BA, Byrne KE, Hale S, Kamimura K, Mitchell SJ, Peltola H, Ruel J-C (2008) A review of mechanistic modelling of wind damage risk to forests. *Forestry* 81(3):447–463

48. Albrecht AT, Fortin M, Kohnle U, Ningre F (2015) Coupling a tree growth model with storm damage modeling—conceptual approach and results of scenario simulations. *Environ Model Softw* 69:63–76
49. Ray D, Bathgate S, Moseley D, Taylor P, Nicoll B, Pizzirani S, Gardiner B (2015) Comparing the provision of ecosystem services in plantation forests under alternative climate change adaptation management options in Wales. *Reg Environ Change* 15(8):1501–1513
50. Seidl R, Rammer W, Blennow K (2014) Simulating wind disturbance impacts on forest landscapes: tree-level heterogeneity matters. *Environ Model Softw* 51:1–11
51. Janz B, Storr D (1977) The climate of the contiguous mountain parks. Project no. 30, Applications and Consulting Division, Environment Canada, Edmonton
52. Norris D (1966) Solar radiation on inclined surfaces. *Sol Energy* 10(2):72–76
53. Braun JE, Mitchell JC (1983) Solar geometry for fixed and tracking surfaces. *Sol Energy* 31 (5):439–444
54. Sen Z (2004) Solar energy in progress and future research trends. *Prog Energy Combust Sci* 30(4):367–416
55. Zaksek K, Podobnikar T, Ostir K (2005) Solar radiation modelling. *Comput Geosci* 31 (2):233–240
56. Kambezidis HD, Psiloglou BE, Karagiannis D, Dumka UC, Kaskaoutis DG (2016) Recent improvements of the meteorological radiation model for solar irradiance estimates under all-sky conditions. *Renew Energy* 93:142–158
57. Munee T, Younes S, Munawwar S (2007) Discourses on solar radiation modeling. *Renew Sustain Energy Rev* 11(4):551–602
58. Bowman DMJS (2005) Understanding a flammable planet—climate, fire and global vegetation patterns. *New Phytol* 165(2):341–345
59. Taylor AH, Beaty RM (2005) Climatic influences on fire regimes in the Northern Sierra Nevada mountains, Lake Tahoe Basin, Nevada, USA. *J Biogeography* 32:425–438
60. Lynch AH, Beringer J, Kershaw P, Marshall A, Mooney S, Tapper N, Turney C, Van Der Kaars S (2007) Using the paleorecord to evaluate climate and fire interactions in Australia. *Annu Rev Earth Planet Sci* 35:215–239
61. Flannigan MD, Stocks BJ, Wotton BM (2000) Forest fires and climate change. *Sci Total Environ* 262:221–230
62. Angström A (1949) Swedish meteorological research 1939–1948. *Tellus* 1(1):60–64
63. McRae DJ, Conard SG, Ivanova GA, Sukhinin AI, Baker SP, Samsonov YN, Blanke TW, Ivanov VA, Ivanov AV, Churkina TV, Hao WM, Koutzenogi KP, Kovalev N (2006) Variability of fire behaviour, fire effects, and emissions in Scotch Pine forests of Central Siberia. *Mitig Adapt Strat Glob Change* 11:45–74
64. Arpacı A, Eastaugh CS, Vacik H (2013) Selecting the best performing fire weather indices for Austrian Ecozones. *Theoret Appl Climatol* 114:393–406
65. Skvarenina J, Mindas J, Holecy J, Tucek J (2003) Analysis of the natural and meteorological conditions during two largest forest fire events in the Slovak Paradise National Park. International Bioclimatological Workshop, pp 29–36
66. Keetch JJ, Byram GM (1968) A drought index for forest fire control. Research paper SE-38. U.S. Department of Agriculture, Forest Service, Southeastern Forest Experiment Station, Asheville, North Carolina, p 32 (revised Nov 1988)
67. Crane WJB (1982) Computing grassland and forest fire behaviour, relative humidity and drought index by pocket calculator. *Australian For* 45(2):89–97
68. Janis MJ, Johnson MB, Forthun G (2002) Near-real time mapping of Keetch-Byram Frougt index in the South-Eastern United States. *Int J Wildland Fire* 11(3–4):281–289
69. Petritsch R, Hasenauer H (2014) Climate input parameters for real-time online risk assessment. *Nat Hazards* 70(3):1749–1762
70. Pietsch SA, Hasenauer H, Thornton PE (2005) BGC-model parameters for tree species growing in Central European forests. *For Ecol Manage* 211(3):264–295
71. Eastaugh CS, Hasenauer H (2014) Deriving forest fire ignition risk with biogeochemical process modelling. *Environ Model Softw* 55:132–142