Université Aix-Marseille, France
Institut Français de Pondichéry, India
École Doctorale en Mathématiques et Informatique de Marseille
UFR des Sciences
LSIS UMR 7296

Thèse présentée pour obtenir le grade universitaire de docteur

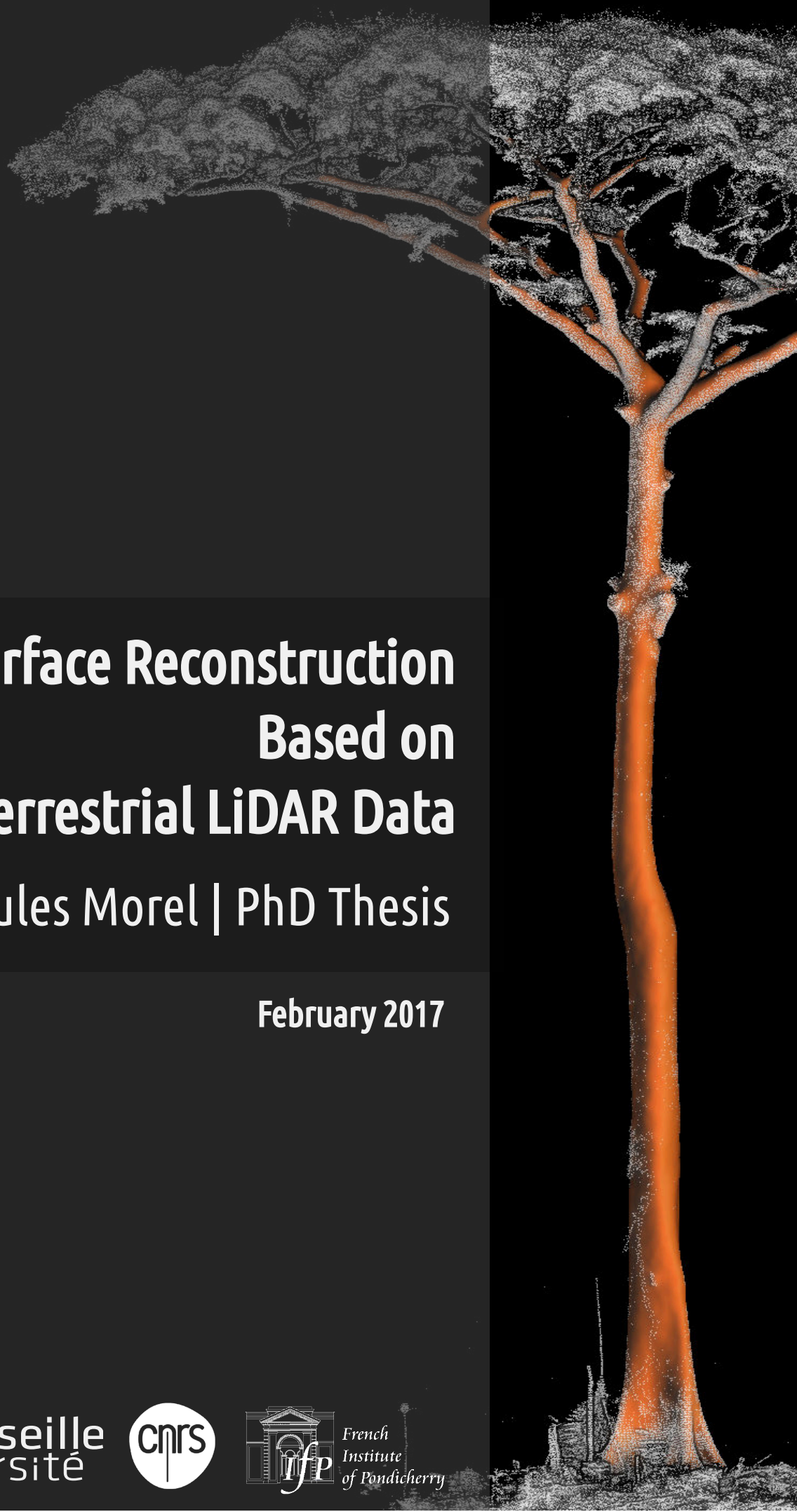
Discipline : MATHEMATIQUES ET INFORMATIQUE
Spécialité : Informatique

Jules Morel

Reconstruction de surface à partir de données LiDAR terrestre acquises en forêt

Soutenue le 17/02/2017 devant le jury :

Takashi Kanai	*University of Tokyo*	Rapporteur
Bruno Lévy	*INRIA Nancy Grand-Est*	Rapporteur
Stéfanie Hahmann	*INRIA Grenoble*	Examinatrice
Loïc Barthe	*IRIT, Toulouse*	Examinateur
Alexandra Bac	*Université Aix-Marseille*	Co-directrice
Cédric Véga	*IGN, Nancy*	Co-directeur
Marc Daniel	*Université Aix-Marseille*	Directeur de thèse



Surface Reconstruction Based on Forest Terrestrial LiDAR Data

Jules Morel | PhD Thesis

February 2017

Aix*Marseille université
cnrs
IFP French Institute of Pondicherry

Surface Reconstruction Based on Forest Terrestrial LiDAR Data

a dissertation presented
by
Jules Morel
to
The Department of Mathématique et informatique

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Computer science

Université Aix-Marseille
Marseille, France
February 2017

Doctoral advisor: Professor Marc Daniel
Coadvisors: Dr. Alexandra Bac, Dr. Cédric Véga

Surface Reconstruction Based on Forest Terrestrial LiDAR Data

Abstract

In recent years, the capacity of LiDAR technology to capture detailed information about forests structure has attracted increasing attention in the field of forest science. In particular, the terrestrial LiDAR arises as a promising tool to retrieve geometrical characteristics of trees at a millimeter level.

This thesis studies the surface reconstruction problem from scattered and unorganized point clouds, captured in forested environment by a terrestrial LiDAR. We propose a sequence of algorithms dedicated to the reconstruction of forests plot attributes model: the ground and the woody structure of trees (i.e. the trunk and the main branches). In practice, our approaches model the surface with implicit function build with radial basis functions to manage the homogeneity and handle the noise of the sample data points.

Our first focus is on the reconstruction of the ground surface whose level of detail is based on local complexity, through alternation between scale refinement, filtering and reconstruction. The result arises from the polygonization of the implicit function expressed as the merging of local approximations by compactly supported radial basis function used as partition of unity.

Once the ground is modeled, the topology effects can be ignored in the following computation steps that focus on the modeling of trees. Traditionally, the processing of the woody part is achieved by a discrete reconstruction in the form of a stack of independent building blocks. From such a model, our approach developed for the ground is adapted to approximate the woody part of the tree by a more flexible continuous surface.

Expressed as an implicit function, the tree model can be refined by an additional computational step in order to describe precisely the geometry. With this in mind, we propose a method dedicated to the fine reconstruction of occluded objects: from 3D samples presenting occlusions, we use the previously described continuous model to guide a Poisson surface reconstruction. Thus, we guarantee the production of a watertight surface that approximates sharply the point cloud in the visible areas and extrapolates consistently the tree shape in the occlusions.

Key-words: Surface reconstruction, Interpolation, Approximation, Implicit function, Radial basis functions, Partition of unity, Poisson surface, Terrestrial LiDAR

Directeur de thèse: Professeur Marc Daniel
Co-encadrants: Dr. Alexandra Bac, Dr. Cédric Véga

Reconstruction de surface à partir de données LiDAR terrestre acquises en forêt

Résumé

Au cours des dernières années, la capacité de la technologie LiDAR à capturer des informations détaillées sur la structure des forêts a attiré une attention croissante de la part de la communauté des écologues et des forestiers. Le LiDAR terrestre, notamment, apparaît comme un outil prometteur pour recueillir les caractéristiques géométriques des arbres à une précision millimétrique.

Cette thèse étudie la reconstruction de surface à partir de nuages de points épars et non structurés, capturés en environnement forestier par un LiDAR terrestre. Nous proposons une suite d'algorithmes dédiés à la reconstruction de modèles d'attributs de placettes forestières : le sol et la structure ligneuse des arbres (*i.e.* troncs et branches principales). En pratique, nos approches modélisent le problème par des surfaces implicites construites à partir de fonctions à base radiale pour faire face à la forte hétérogénéïté spatiale du nuage de points Lidar terrestre.

Tout d'abord, nous proposons une méthode de reconstruction de la surface du sol dont le niveau de détails est basé sur la complexité locale des données, reposant sur une séquence de raffinement d'échelle, de filtrage et de reconstruction. La fonction implicite résultante, exprimée comme un ensemble d'approximations locales fusionnées par partition de l'unité, est polygonisée pour extraire sa surface de niveau zéro.

Une fois le sol modélisé, les effets de la topologie peuvent être ignorés dans les étapes de calcul suivantes qui se concentrent sur la modélisation des arbres. Traditionnellement, le traitement des parties ligneuses est réalisé à partir d'une reconstruction discrète des arbres sous la forme d'un empilement de composantes indépendantes. Reposant sur un tel modèle, l'approche développée pour le sol est adaptée afin de proposer une technique de modélisation qui estime la partie boisée de l'arbre par une fonction implicite continue.

Exprimé sous cette forme, le modèle d'arbre peut être affiné par une étape de calcul supplémentaire pour décrire précisément la complexité de la géométrie. Dans cette optique, nous proposons une méthode adaptée à la reconstruction fine d'objets occlus : à partir de nuages de points 3D présentant des occlusions, nous utilisons le modèle continu précédemment décrit pour guider une reconstruction de surface de Poisson. Ainsi, nous garantissons la production d'un modèle surfacique fermé qui extrapole de façon cohérente la forme de l'arbre dans les occlusions.

Mots-clés : Reconstruction de surface, Interpolation, Approximation, Fonction implicite, Fonction de base radiale, Partition de l'unité, Surface de Poisson, LiDAR terrestre

Acknowledgments

This PhD work was carried out between the Laboratory of Geomatic and Applied Informatics of the French Institute of Pondicherry (IFP), India, UMIFRE 21 CNRS-MAEE, and the team of geometric modeling of the Laboratoire des Sciences de l'Information et des Systèmes (LSIS), UMR 7296. I would like to thank IFP for providing financial assistance for my research and LSIS for providing research facilities.

I express my most sincere thanks to Alexandra Bac and Cédric Véga, first, for managing to convince me to start this academic career, then for their guidance, their insightful comments and encouragement during those three years, but also for the very pleasant time shared and all the interesting discussions we had. This work would not have been possible without their advice and support.

I would like to thank Marc Daniel, head of the geometric modeling team of LSIS, who kindly agreed to lead this work and has given me the opportunity to work in his laboratory.

I am very grateful to the members of my PhD committee, Takashi Kanai, Bruno Levy, Stéphanie Hahmann and Loïc Barthe. Their academic support is greatly appreciated. Thank you.

I profoundly thank all the members of the French Institute of Pondicherry for having contributed to the development of my research, for their kindness and hospitality and for having taken part in my discovery of India and its culture. I am also grateful to the members of the G-Mod team for providing a stimulating and fun filled environment.

It is also my pleasure to express my appreciation to my friends for their advice and feedback on my research, for their unconditional support, for being sources of inspiration and for creating a warm and pleasant atmosphere wherever I was staying, in France or in India.

My sincere thanks also go to Nelly & Jo who kindly welcomed me, motivated and encouraged me during my stays in Marseille.

Last but not least, I take this opportunity to deeply thank my sisters Charlotte and Jeanne, and my parents Sabine and Patrick, who have always been supportive of my career choices despite the geographical remoteness entailed.

Contents

List of figures

List of tables

Abbreviations

LiDAR	Light Detection And Ranging
ALS	Airborne LiDAR Scanning
TLS	Terrestrial LiDAR Scanning
RBF	Radial Basis Function
CSRBF	Compactly Supported Radial Basis Function
TIN	Triangulated Irregular Network
NURBS	Non-Uniform Rational B-Spline
MC	Marching Cubes
PCA	Principal Component Analysis
PU	Partition of Unity
MPU	Multi level Partition of Unity
MLS	Moving Least Squares
PSR	Poisson Surface Reconstruction
DTM	Digital Terrain Model
QSM	Quantitative Structure Modeling
DBH	Diameter at Breast Height
VR	Virtual Reality

Notation

We denote:

Ω	an open subset of $\mathbb{R}^d$, $\Omega \subseteq \mathbb{R}^d$
$\mathcal{C}^k(\Omega)$	the set of k-times continuously differentiable functions on Ω
$\Pi_m(\mathbb{R}^d)$	the space of d-variate polynomials of absolute degree at most m
$\lambda \times A$	the scalar multiplication of the matrix A with the scalar λ
$A.B$	the matrix product of A and B, n x m and m x p matrix respectively, which is given as the $n \times p$ matrix of elements $AB_{(i,j)} = \sum_{k=1}^{m} A_{ik}B_{kj}$
$< f,g >$	the inner product of $f,g : \Omega \to \mathbb{R}$, which is given by $< f,g >= \int_\Omega f(\mathbf{p}) \cdot g(\mathbf{p}) d\mathbf{p}$
∇	the gradient operator in $\mathbb{R}^d$
$\nabla\cdot$	the divergence operator in $\mathbb{R}^d$
Δ	the Laplacian operator in $\mathbb{R}^d$, such as $\Delta = \nabla \cdot (\nabla)$
$*$	the convolution, *i.e.* $[f * g](\mathbf{x}) = \int_\Omega f(\mathbf{x}) \cdot g(\mathbf{x} - \mathbf{y}) d\mathbf{y}$
$\Phi_{r_0}(r)$	the radial basis function such as $\Phi_{r_0}(r) = \Phi(r/r_0)$
$(.)_+$	the cut-off function such as $(x)_+ = x$ if $x \geq 0$, 0 otherwise
$\partial\mathcal{M}$	the surface boundary of the solid $\mathcal{M}$

Moreover, we fix some convention about indices:

- Considering a vector $\mathbf{p} \in \mathbb{R}^d$, its components are given as $p_0, ..., p_{d-1}$.
- We denote $\|\mathbf{p}\|_l$ the discrete l-norm $\|\mathbf{p}\|_l = \sum_{j=0}^{d-1} |p_j|^l$. In this document, we simplify as $\|\mathbf{p}\|$ the writing of the squared norm, *i.e.* $\|\mathbf{p}\|_2 = \sum_{j=0}^{d-1} p_j^2$.

1 Introduction

Data, information and knowledge are key-worlds and fundamental concepts in information science and knowledge management. As presented by Ackoff[1], they are linked and affect each other through distinct relationships: data are composed of basic, unrefined, and unfiltered sets of measures. Information is refined data that has evolved to the point of being useful for some form of analysis. Knowledge is information with meaning; it is used by the user to understand, explain and decide. The field of surface reconstruction from 3D point clouds stands at the interface between data and information: using the context, surface reconstruction algorithms analyze, organize, formalize and package the data stored in the point samples, and present them as a surface. Applied to forest terrestrial LiDAR scanning (TLS) data, surface reconstruction provides information on forest objects, further combined with the user's insight and experience to increase knowledge and understanding of forest environments.

1 The challenge of 3D point clouds in the field of geometric modeling

The research topic of surface reconstruction from 3D point samples has been explored for decades. However it is still problematic in the case of incomplete, noisy and sparse data and many questions remain open. Many challenges remain in the field of *shape detection* that reduces the complexity based on context knowledge; in *classification* that separates permanent structure informa-

tion and clutter; and finally in *reconstruction* that generates geometric models faithful to the real physical structure and thus provides a plausible completion of missing data while maintaining a low complexity.

The purpose of this thesis is to filter, classify and organize data to extract information from 3D point clouds in order to reconstruct surface models. Our first focus was on the reconstruction of relatively flat surfaces whose level of detail is based on local complexity, through alternation between scale refinement, filtering and reconstruction. Then, based on a classic discontinuous model build from a stack of independent building blocks, we introduced a specific modeling technique that approximates the point by a more flexible continuous surface over the whole scene. Finally, we focused on the reconstruction of objects heavily occluded: from 3D data-set presenting large holes, we computed a surface approximation as a watertight mesh, guided by a model known a priori.

Our main objective was to develop general methods and algorithms, and implement those approaches in order to contribute to the development of a complete processing line for terrestrial LiDAR scanning (TLS) data acquired in forests. LiDAR, standing for Light Detection And Ranging, is a surveying technology that measures distance by illuminating a target with a laser light. The LiDAR technology and its forestry-related terrestrial application are presented in the next chapter.

Thus, the research project of this PhD was to design new approaches for the reconstruction of forest object surfaces from TLS point clouds. While dealing with the reconstruction of flat surfaces of the ground, the drop in point density with the distance to the scanner and the large occlusions created by the trees and by the ground topology have to be addressed. In respect of the approximation of the woody part of the trees by a continuous surface, algorithms must rise to the challenge of handling complex cylinder-like shapes that could be occluded in various proportions according to the scene geometry.

1.1 Geometric modeling choices

To cope with the characteristics and imperfections of 3D point clouds acquired with TLS in forests, specific modeling choices have emerged as convincing techniques. The huge size of those data-sets impelled us to avoid mesh models that are too costly, but rather to choose a modeling

as an implicit function. More precisely, we based our approaches on modeling with radial basis functions, a well-known technique to interpolate scattered data, in order to tackle the noise, the non-uniform sampling produced by the scanner and also to extrapolate the resulting surface in the occlusion holes. Finally, in order to reduce the computational complexity, we chose radial basis function with finite support size.

1.2 Application to Terrestrial LiDAR scanning (TLS) in forest

Based on those modeling choices, we developed a complete processing line to process TLS data acquired over forests plots with an emphasize on the major geometrical components, the ground and the woody part of the trees. The complete processing is made of three independent computation steps. First, the ground is modeled as a continuous surface on the scanned area. Once the ground surface is extracted, the topology effects can be ignored thus allowing trees to be detected in the point cloud. Finally, the detected trees are isolated and processed individually.

Following these statements, we first focused on the computation of digital terrain model. To do so, we developed a new surface reconstruction method dedicated to the processing of anisotropic and noisy data through alternation between scale refining, filtering and reconstruction.

Then we addressed the automatic detection of trees in a forest environment. On TLS data acquired in a forest, cylinder instances are detected through an efficient voting process to model trees following a Hough transform approach proposed by Bac[14] and further improved by Ravaglia[129].

The model developed for digital terrain model (DTM) has been adapted to process tree-like shapes. From a set of discrete cylinders generated from current reconstruction approaches, we improved the shape reconstruction of the woody parts of the trees by computing a continuous surface.

Finally we designed a Poisson surface reconstruction approach based on compactly supported radial basis function (CSRBF) for incomplete data where the reconstruction is guided by the previously described continuous model in occluded areas. Thus, we guarantee the production of a watertight surface that approximates precisely the point cloud in the visible areas and extrapolates consistently the tree shape in the occlusions.

Our research pipeline is illustrated Figure 1.1 over a 20 meters by 20 meters forest plot described by 61 millions of points. Figure 1.1a) highlights the complexity of forest scenes. Figure 1.1b) shows

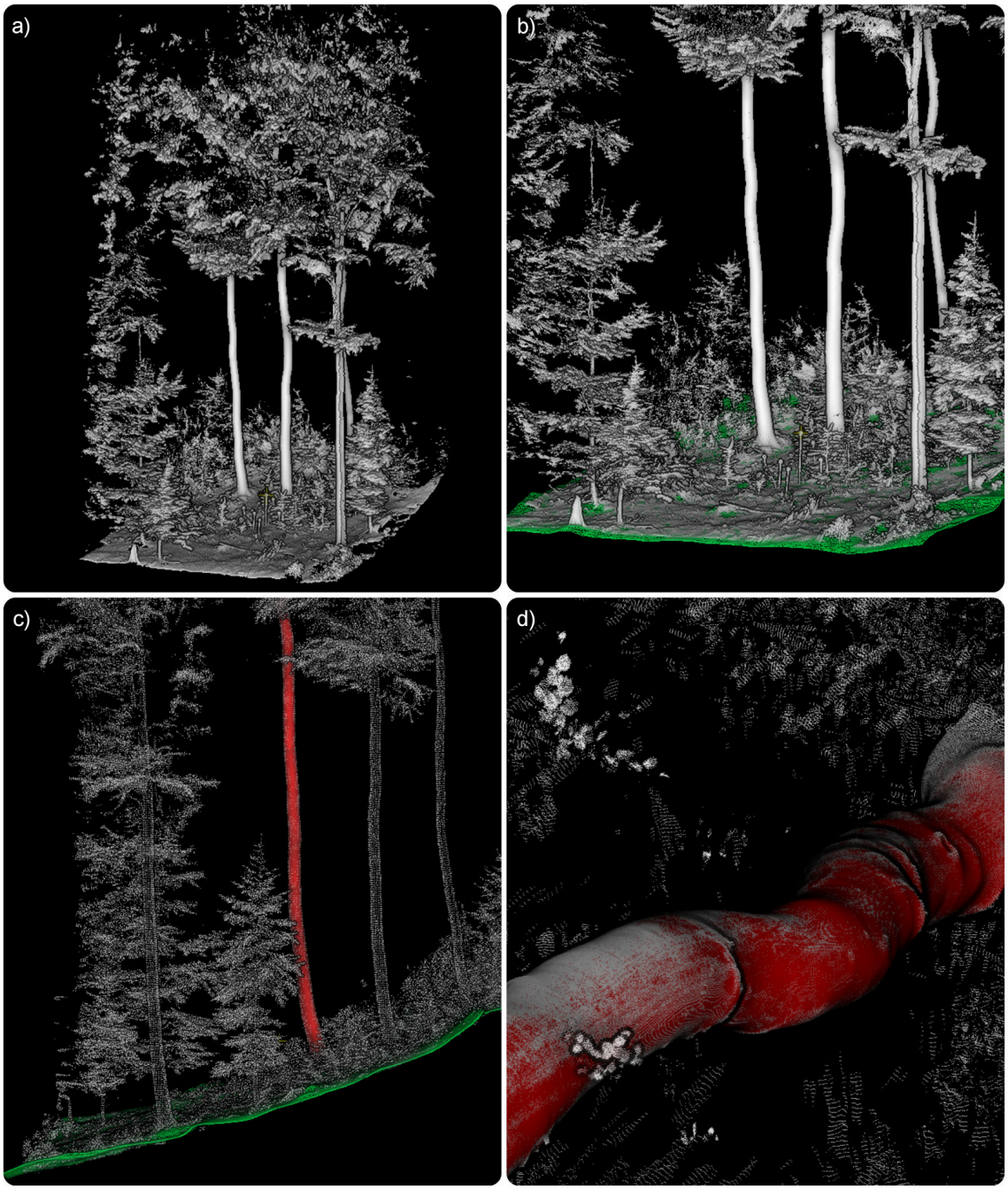


Figure 1.1: Illustration of the processing line in a preview example on a TLS point cloud (a). (b) The digital terrain model is reconstructed after vegetation filtering (see Chapter 2). (c) The reconstructed surface of the trunk of the tree is shown as a red mesh with a decimated point cloud for visual purpose. (d) Close up on a bottom view of the tree model along with the raw point cloud.

the resulting terrain model. Finally Figure 1.1c) and d) point out a stem detection and the details of the stem shape respectively.

In practice, we implemented the algorithms developed in the context of this research as a module of the open source platform Computree (computree.onf.fr). This collaborative platform originated in 2010 from ANR EMERGE project and is specialized in the processing of TLS data to extract forest information. It is nowadays a well known and powerful software which aspires to both scientific and operational development. The Computree phylosophy, which promotes modularity and openness by packaging the computation steps as independent plug-ins, has seduced scientists from prominent institutions (Among others: Office national des forêts (ONF), École nationale supérieure d'arts et métiers (ENSAM), Université de Sherbrooke, Laboratoire d'étude des ressources forêts-bois (Lerfob), Centre de coopération internationale en recherche agronomique pour le dêveloppement (Cirad)). Today, Computree lists hundreds of contributors across the world and is available on Windows, Linux and Mac OS.

Our algorithms are implemented in C++ and rely on several well known libraries. We used the *Eigen library* (included in Computree) for linear algebra: matrices, vectors, numerical solvers and 3D geometric operations (rotations and projective or affine transformations). Then, We used the 3D point clouds and the tree data structures of the *Point Cloud Library* (PCL)[131]. Some parts of the code have been parallelized using the *Boost library*[84]. We also used Boost to set up graph structures and to build Hash maps. Finally, we relied on the *GNU scientific library* (GSL)[69] to perform least squares fits to the 3D data and to find minima of arbitrary multidimensional functions.

2 Thesis outline

This thesis addresses the reconstruction of surface from unorganized and scattered point cloud acquired from TLS. While the methods have been designed to be generic and flexible, the thesis focuses on forest plot reconstruction because of their inherent complexity.

The thesis is written and mainly organized in an article-based style. Some articles are accepted or published, others are still submitted in view of the time constraints for writing this manuscript. Chapter 2 and 3 provide literature reviews on TLS data acquisition and forest parameters estima-

tion, and surface reconstruction from point sample respectively. Chapter 4 to 6 introduce key topics of our modeling pipeline and are based on the following papers:

- Morel J., Bac A., Véga C. 2016. **Digital terrain model construction with compactly supported radial basis functions**. *IEEE Computer Graphics and Applications*, in press
- Morel J., Bac A., Véga C. 2016. **An innovative tree modeling: from terrestrial LiDAR point clouds to continuous implicit surface**. *9th International Symposium on Mobile Mapping Technology, MMT2015, Sydney, Australia*, accepted. As recommended by the MMT2015 organizing committee, it has been submitted in *Photogrammetric Engineering & Remote Sensing* as a peer-reviewed paper.
- Morel J., Bac A., Véga C. 2016. **Poisson surface reconstruction with CSRBF**. *Computer Graphics Forum*, submitted.

Finally, Chapter 7 discusses the major results and conclusions drawn regarding the potential of our approaches to support forest inventories.

Note that each chapter provides a short introduction to the overall topic, which contains essential background information and additional literature review.

During the development of the algorithms, we encountered difficulties in the estimation of the quality of the surface approximation: because of the size and the complexity of the input data-set, and the intricacy of the mesh surface reconstructed, the proximity between both input and output was often hard to evaluate on a classic 2D display. Based on this observation, immersive visualization emerged as an interesting alternative, especially since virtual reality (VR) became accessible and affordable with simple devices such as Google Cardboard. In this context, we developed an Android application to display point clouds and surface in VR. Using virtual reality head-mounted device and an Android smart-phone, the user immerses himself into the data and navigates in it by the mean of a blue-tooth controller. This application, «**LiDAR VR Viewer**», is free on the Google Play Store. It relies on a strong optimization of OpenGL code and GLSL shaders to maintain a smooth experience with heavy dataset. The features of the application are presented in Appendix A in the form of a short-paper: Morel J., Bac A., Véga C. 2017. **An Android application to visualize point clouds and meshes in VR**. *WSCG 2017*, submitted.

2

Terrestrial LiDAR scanning in forests

In the last few years, 3D acquisition has become a routine action in several activities: the study of forests, medicine, archaeology, architecture and urban engineering are among the most important fields of application. While both the performance and ease of use of 3D acquisition techniques are continuously increasing, several 3D scanning solutions have been proposed by the research and industrial community. For instance, on the one hand, volumetric techniques developed for medicine (Computed tomography or Magnetic resonance imaging) produce a discrete 3D representation of the surface and interior of an object. On the other hand, surface techniques provide a digitization of the surface of a 3D object. Several possibilities are available based on budget, accuracy and time constraints. Currently, 3D digitization techniques can be classified in two major groups: passive methods and active ones. Passive methods only use information contained in images of the scene. *Stereoscopic* methods[80], measuring range by triangulation of selected locations in a scene imaged from two different points of view, and *Shape from Shading* methods[79] employing photometric stereo techniques to produce depth measurements, are two well known passive methods. Active methods, on the other hand, use a controlled source of light to retrieve 3D information. *Laser Ranging* systems, such as those embedded in *Light Detection And Ranging* (LiDAR) systems, are based on the following principle: the object surface reflects laser light back towards a receiver which then measures the time (or phase difference) between transmission and reception in order to calculate the depth. While ranging systems dominate the

market of 3D optical measurements, other techniques have been developed, such as *the Structured Light*[134], that projects a pattern of light onto the subject and studies its deformation to assess the depth.

Among all these acquisition techniques, LiDAR is especially appropriate to the study of forests. Applied to large-area forest characterization, air-borne LiDAR scanner (ALS) and space-borne LiDAR focus on vertically distributed forest attributes, such as height, volume and biomass. At a more local scale, very dense measurements such as those obtained from terrestrial sensors enable the capturing of very detailed information on the object shape surrounding the sensor, making it possible to address surface reconstruction problems. The application framework of this thesis is the reconstruction of 3D models from data acquired in a forest environment with terrestrial LiDAR scanner. The following sections introduce this recent technology that allows for recording detailed information on the forests 3D structure.

1 The TLS technology

Light detection and ranging (LiDAR) technology is an active remote sensing technique enabling the rapid acquisition of dense and accurate 3D information on object surfaces by laser range finding. Terrestrial LiDAR scanning (TLS) is the ground-based application of LiDAR. It sweeps a laser beam over a scene and analyzes the back-scattered light to produce a 3D point cloud. It produces accurate measurements with very little time and expense compared to traditional manual methods. Indeed, the TLS instruments currently marketed are extremely precise (see Table 2.1), portable, easy to operate, and have been used successfully in a variety of environments to support a wide range of geoscience investigations including detailed mapping of geologic outcrops and imagery of topography, cultural objects, and trees and vegetation. Figure 2.1 presents examples of point clouds acquired in forested areas.

TLS devices are generally separated into three categories according to their measurement principle: phase-shift (**PS**), time-of-flight (**TOF**) or time-of-flight using waveform digitizing (**WFD**).

- **PS** scanners make use of continuous laser illumination and amplitude modulation of the beam to discern the range at high frequency. Only one return is recorded for each direction. Such instruments allow wide fields of view, very high point densities and fast acquisition speeds. Due to their range, phase-shift based scanners are well suited for high precision and detailed measurements of relatively near scenes.

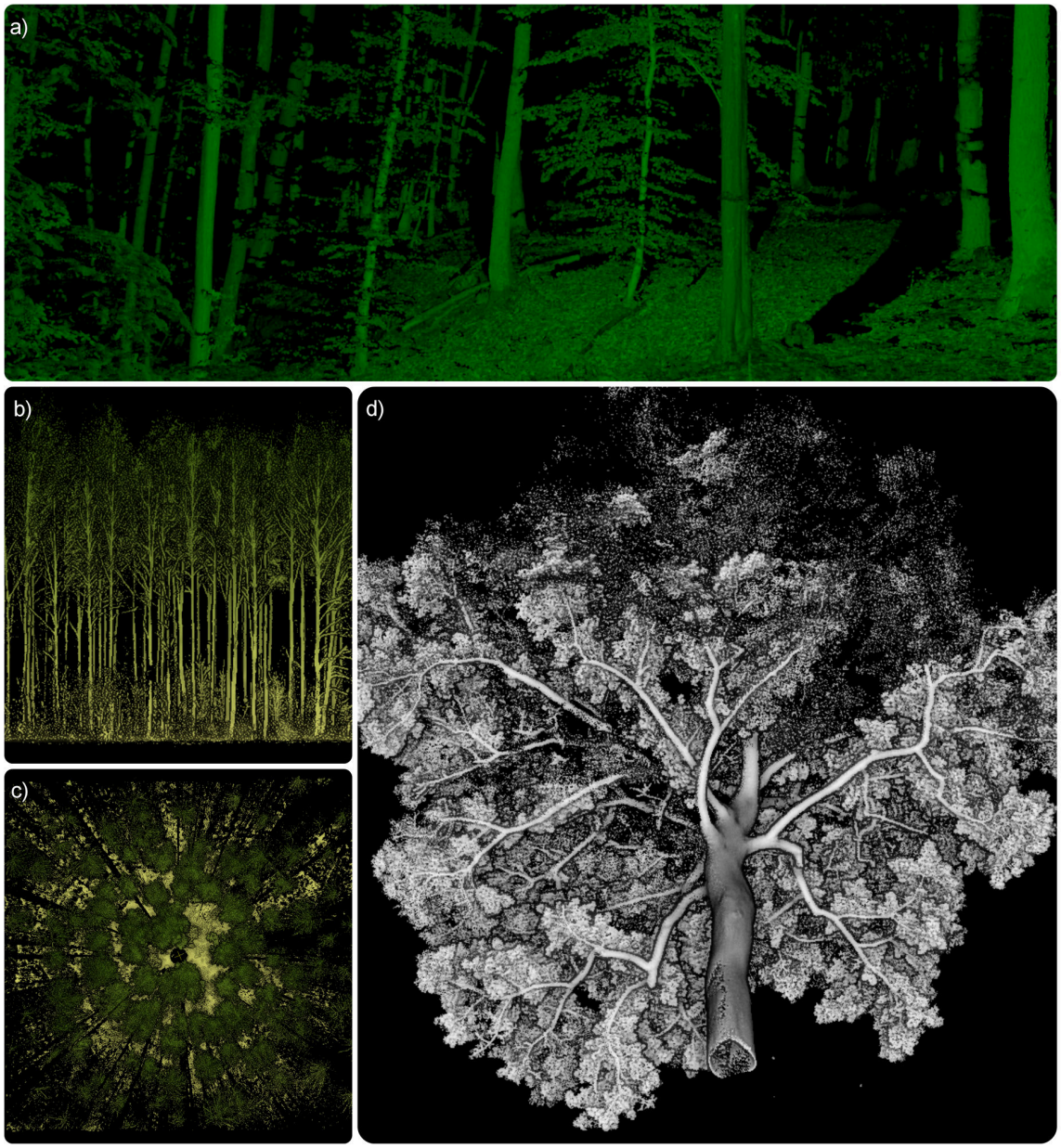


Figure 2.1: Visualization of TLS point clouds acquired in forest environment: a) Forest plot near Nancy, France, scanned by ONF. The green color scale is linked to the signal intensity. b) and c) show side and top view of a forest plot scanned in the context of the EuroSDR benchmarking project. The color ramp scales with the altitude of the points. One should note the occlusion as shadow strides created by the tree trunks, especially visible on the top view c). d) Bottom view of an isolated tree. This unpublished data from Cameroon were collected in collaboration with Alpicam company within the IRD project PPR FTH-AC. On b), c) and d), the point clouds have been shaded with eye-dome lightning for display purposes.

- **TOF** scanners utilize precise timing for determining the range from the pulse time of flight and the speed of light. These characteristics allow very long measurement distances but lower acquisition speeds. These types of scanner are well suited for the 3D reconstruction of scenes at greater distances. According to the LiDAR sensor capabilities, two acquisition methods can be distinguished: (i) single return recording which describes the first objects hit by the laser beam and (ii) multiple return recording, which produces denser point clouds by providing multi-depth information.
- **WFD** scanners, introduced by Leica in 2012, are based on a combination of time-of-flight range measurement and modern waveform digitizing technologies to boost the scan rate. Such devices conserve the TOF scanners precision but lose in action range. Those devices are able to record the full waveform by digitizing completely the back-scattered laser signal. They can potentially be used to retrieve additional information with respect to discrete returns.

Sensor	Faro Focus 3D X330	Riegl VZ-1000	Leica ScanStation P40
Category	PS	TOF	WFD
Range	0.6 m - 330 m	2.5 m - 1400 m	0.4 m - 270 m
Scan rate (pts/sec)	up to 976,000	up to 122,000	up to 1,000,000
Precision	2 mm @ 10 m	5 mm @ 100 m	6 mm @ 100 m

Table 2.1: Caracteristics of current TLS sensors available in the market.

2 Forestry and Terrestrial LiDAR scanning (TLS)

In the recent years, the capability of TLS devices to capture detailed information about the structure of the environment surrounding the sensor has attracted increasing attention in the field of forest science. Indeed, TLS data provide a unique way to retrieve precise information about tree attributes that are usually difficult to measure in the field because of time and cost constraints. Indeed, TLS nowadays complements other remote sensing techniques (aerial LiDAR and photography) in the analysis of the forests by providing millimeter-level of detail from the surrounding area of a forest plot[47]. By facilitating the studies of forest structural information, TLS is considered as a promising tool to complement (or assist) field work and retrieve additional tree or plot parameters[94,114].

For foresters, it is interesting as it allows for precise tracking and management of timber, while for ecologists the interest lies in tree population growth and dynamics study. Even though TLS represents an initial investment compared to traditional methods, it is establishing itself as an efficient complementary tool in the field. With respect to the traditional manual techniques of collecting tree parameters, measurements with TLS are cheaper in time and workforce, are more accurate and non-destructive and allow for varied treatment a posteriori. Nevertheless, TLS data acquired in a forest present three flaws: (i) the occlusion that hides parts of the plot because of vegetation elements[136]; (ii) the decreasing resolution with the distance to the sensor because of the spherical geometry of the device[103]; and (iii) the large amount of raw data produced, which make the interpretation difficult[58].

The following section proposes a non-exhaustive survey on the methods developed to extract information in order to propose descriptors of the features of the forests. It introduces historical methods based on destructive field measurement, and newer ones that use TLS as source of information.

3 Literature review on tree modeling

Traditionally, forest plot parameters are either measured (diameter, height, wood density) or predicted (volume, biomass) using allometric theory, that focuses on the scaling of plant attributes with plant size[59]. Huxley[81] defined the allometric equation as a relationship between a structural or functional variable Y of an organism and a power function of its mass M (or any measure of its size):

$$Y = Y_0 \cdot M^b \tag{2.1}$$

where Y_0 is a constant that varies with the type of variable and b is the allometric exponent. In forest science, allometric equations are widely used to estimate forest parameters that are difficult to assess directly from field measurement, as for example the amount of biomass held in ecosystems[118,64]. Considering that tree biomass is strongly correlated to tree diameter[28] and to its height[38], ecologists establish empirical allometric equations that relate those parameters via destructive field campaign. In practice, trees are felled at ground level and their biomass is estimated by directly weighting small branches (diameter below 20 cm) and by approximating the bigger parts from geometrical considerations[77,63]. With a sufficient tree sample, one can build

an allometric model by regressing the biomass against independent parameters such as the trunk diameter D, the wood specific gravity ϱ and the total tree height h. For instance, Chave[39] proposes the following model:

$$\ln(AGB) = \alpha + \beta \cdot \ln(\varrho \cdot D^2 \cdot h) + \varepsilon \tag{2.2}$$

where AGB is the amount of dried above-ground organic living matter in the tree, α and β are model coefficients derived from least-squares regression and ε is an error term. One should note here the quasi-cylindrical approximation made on the shape of the tree. Whereas this allometric model performs to measure the biomass at the plot scale, Chave[39] have estimated approximately 50% errors on the biomass assessment at the individual tree scale. That error incorporates volume and wood density errors.

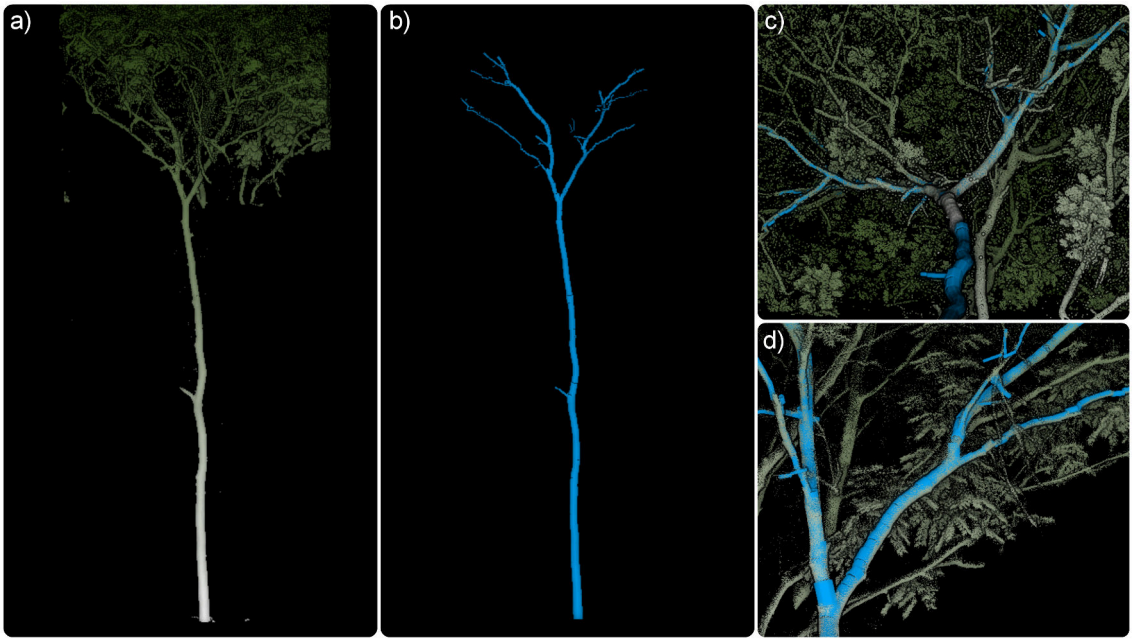


Figure 2.2: TLS scan of an *Erytrophleum fordii* colored with height ramp along with the QSM computed by Simpletree (http://www.simpletree.uni-freiburg.de/). a) presents the input point cloud colored by height, b) presents the QSM represented as polygons, c) and d) show the superposition of the point cloud and the model. Those data have been published in [72] under the Creative Commons 4.0 license. The point clouds and surfaces have been shaded with eye-dome lightning for display purpose.

Setting out from this premise, TLS proved to be a powerful tool to refine the volume estimation at the tree scale in a non-destructive manner. Moreover, it is efficient in overcoming the challenge of capturing the fine structure and architecture of trees on a forest plot[126,83,123]. Indeed, recent studies have shown that TLS is efficient in recovering trees' characteristics[47,99,149] such as

diameter and volume, in an unambiguous, objective, and reproducible manner. Based on TLS data, several approaches have been explored to retrieve the complete architecture of trees as an hierarchically connected set of geometric components. For instance, Pfeifer[123] extracts the stem modeled as a set of cylinders and the outer hull of crowns. Xu[156] produces polygonal models of trees by using allometric theory to ensure appropriate dimension. Cote[44] introduces an interesting approach producing realistic architecture of multi-view scanned conifer: based on the intensity of the scattered signal, the points are classified as wood points that define skeletal curves, or as foliage points that play the role of attractors to grow a complete branching structure. Then, the model expands its volume using allometric theory. Recently, Raumonen[128] has summarized all the geometric and hierarchical information about a tree in the concept of quantitative structure modeling (QSM): such a model describes the tree structure, branching organization and the branch size distribution as a connected sequence of cylinders. Following this model, Raumonen[128] considers a single tree and segments its components based on *a priori* knowledge about the data, then approximates the segments as a sequence of cylinders. As for Hackenberg[72], the set of cylinders arise from the division of the point cloud into local subsets by the browsing of the point cloud through a sphere, and then from a local fitting (see Figure 2.2).

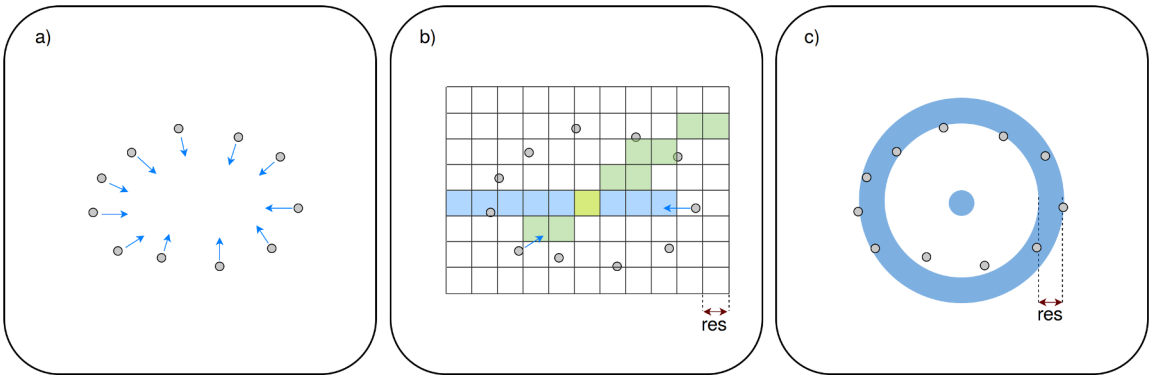


Figure 2.3: Illustration of the Hough transform in 2D: (a) computation of the point normals, (b) definition of a 4D Hough space (x,z,y,r) of "res" resolution. Vote of two points along their normals, and (c) after the vote of the whole sample points, extraction of the most probable circle approximating the points with a given uncertainty linked to the resolution of the Hough space.

Ravaglia[129] uses a multi-scale combination of an original Hough transform and open growing active contours to evaluate such a model. This original approach not only adjusts cylinders on the point samples without the costly least squares fitting of cylinders, but also proceeds to the tree detection in the raw 3D point cloud. Indeed, while the previous QSM methods perfom on isolated trees, the Hough transform, illustrated in Figure 2.3, detects the trees in the whole

scanned scene, a crucial step in the processing of TLS data.

In practice, the Hough transform recovers position and radius of cylinders using the formulation given in Kimme[89]: for a given radius r, each point votes along its normal in the accumulator space. The most probable cylinder corresponds to the bin of maximum score. The precision on the position and radius are related to the resolution of the Hough accumulator space. In order to bring coherence to the scroll of cylinders by estimating their orientation and also to fill missing cylinders due to occlusion, Ravaglia[129] considers the skeleton of cylinders as an active contour, and smooths its sequence through an energy minimization process. This method naturally arises as an interesting way to detect trees in the point cloud and to set up cylinders on the trunk and the branches.

While QSM has proven efficient to represent the structural parameters, it proposes a discontinuous model and relies on local cylindrical approximations of the tree which has proven to be the best local approximation[101] but can lead to substantial errors.

Indeed, the cylindrical approximation is questionable, especially on tropical trees presenting buttresses for instance (see Figure 2.4), in particular on the older individuals as pointed out by Olagoke[117]. The acknowledgment of the tree cylindrical approximation limits, as shown by Chave[39], highlights the need of a sharper 3D modeling in order to better describe the shape and improve the geometric model, in order to precisely assess tree properties.

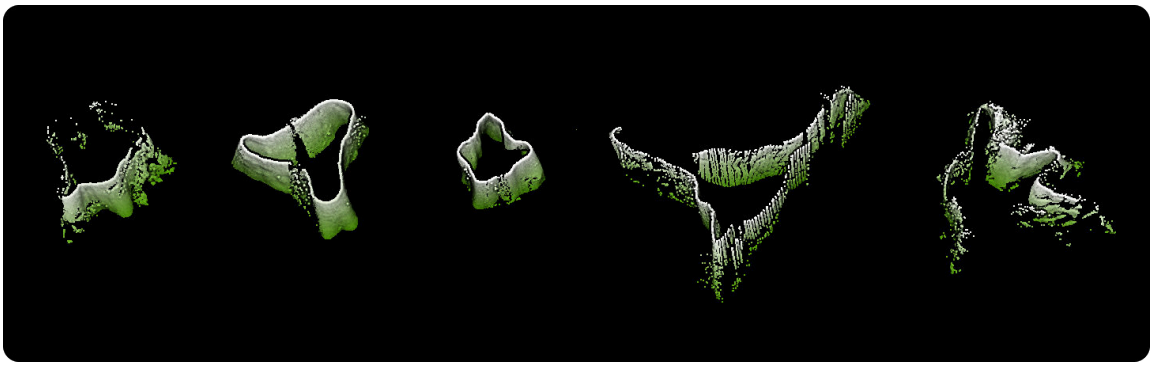


Figure 2.4: Top view of a cross section of 3D point clouds captured on the lower part of five tropical trees. This unpublished data from Cameroon was collected in collaboration with Alpicam company within the IRD project PPR FTH-AC. The point clouds have been shaded with eye-dome lightning for display purposes.

On the basis of these findings, we propose to improve tree modeling by leaving out the cylindrical approximation: by relying on specific surface reconstruction methods based on 3D point clouds to provide an accurate modeling, we propose to retrieve the complex geometrical shape

of trees as a continuous surface without using rigid building blocks.

Retrieving continuous surface from TLS point clouds is an open issue: TLS sensors produce 3D images of the scene structured as point clouds, *i.e.* a collection of (x,y,z) coordinates, without connectivity. While a 3D point cloud is the simplest representation of 3D-information, it actually does not provide any continuous surface information and thus topology, neighborhood relation and local properties of the surface cannot be assessed. As a consequence, it lacks the ability of a direct smooth rendering, and does not propose an intrinsic simplification or subdivision scheme. Moreover, TLS scanners available nowadays produce data-sets that present inherent processing difficulties. First, such data-sets are huge: for instance, a recent terrestrial LiDAR scanning (TLS) sensor can capture millions of points in a few minutes. Then, they present noise, in-homogeneity and outliers. Depending on the complexity of the scene, they are also subject to occlusions: objects behind other objects are occluded or hidden from view. The challenge of 3D point cloud processing consists of tackling those difficulties by analyzing and filtering the data, smoothing the noise and filling the holes in order to segment, classify and finally reconstruct surface models of scanned objects. Because of the complexity of such datasets, processing often relies on the semantic to enrich geometrical models. Indeed, analysis, segmentation and reconstruction need to be adapted to the context: according to the application framework, specific guides are implemented and employed to ease the retrieval of underlying 3D models.

The next chapter presents our modeling and methodological choices by proceeding to a literature review on surface modeling, multidimensional reconstruction and surface reconstruction methods and by resituating those concepts in the context of TLS data processing in forested environments.

3

Survey on surface reconstruction

Digitization is the process of converting information into a digital format. It is crucial in data processing, storage and transmission, because it «allows information of all kinds in all formats to be carried with the same efficiency and also intermingled» [105]. While the quality of analog data worsens with each copy, digital data can be propagated with no degradation. Sampling is the process of transforming analog into digital, *i.e.* recording the values $y \in \mathbb{R}$ of a signal $f\colon \Omega \subseteq \mathbb{R}^d \to \mathbb{R}$ at given points $\mathbf{x} \in \mathbb{R}^d$. Reconstruction is the process that rebuilds the initial information from the samples. In our application context, the terrestrial LiDAR sensor records an intensity value for each 3D point it samples. Based only on the position of this 3D points, our objective is to retrieve the surface of ground and trees with this sampling. This chapter first presents an overview of the different surface models and then, the reconstruction methods associated.

1 Geometric description of 3D surface

In computer graphics, we use various surface model representations: explicit surface models define explicitly the set of points of the surface through functions (they include *polygonal meshes* and *parametric surfaces*), and *implicit surface models*, that define the surface as the zero level-set of a continuous function. Figure 3.1 illustrates these three kinds of models for the rendering of a cylinder surface.

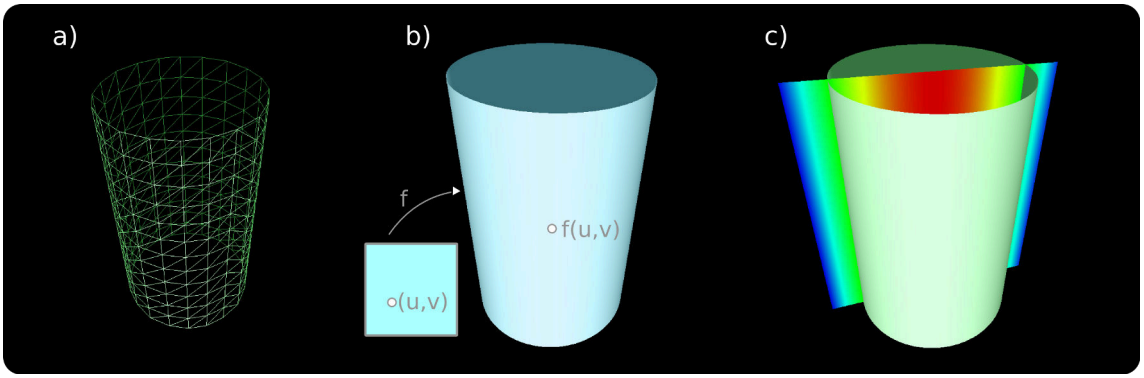


Figure 3.1: Representation of a cylinder by: a) a triangle mesh, b) the image of a parametric surface and c) the level-set of an implicit function.

On the left, Figure 3.1a) represents a cylinder as a mesh, that is, roughly speaking, a set of polygons that are connected by a common edge or corners. Given a domain $\mathbb{U}$ of $\mathbb{R}^2$, Figure 3.1b) represents the cylinder as the image $S = \{f_{parametric}(x, y), (x, y) \in \mathbb{U}\}$ of the parametric function $f_{parametric} : \mathbb{U} \to \mathbb{R}^3$. Then on the right, Figure 3.1c) represents the cylinder as the zero level-set of an implicit function $f_{implicit} : \mathbb{R}^3 \to \mathbb{R}$. Only a planar cross section of this function $f_{implicit}$ is represented and is colored according to $f_{implicit}$ values.

While those mathematical primitives for representing solid-appearing objects are broadly used in the literature, they also constitute important concepts in our research. In the next sections, we thus introduce each of those three surface models, along with their specific features and characteristics.

1.1 Discrete mesh model

Mesh models are disctete models representing objects as a connected set of elementary polygons (or simplices) in dimension n. Among such models, surface meshes (in dimension 2 or 3) play a major role. Quadrangular meshes consist of quadrangular patches connected by their edges or vertices, whereas triangular meshes are composed of triangles (and are an essential class of models directly handled by most graphic devices). However, let us point out that meshes can be defined more generally and in higher dimension. For instance, let us mention volume mesh models (consisting of tetrahedra or tree-dimensional polyedra). In the context of this thesis, we model only surfaces in the 3D space. Hence, we introduce only surface mesh models (and refer the reader to «Polygon mesh processing» [26] and «Delaunay Mesh Generation» [40] for general

presentations of mesh models). A mesh $\mathcal{M}$ is a collection of vertices, edges and faces, defined as a pair of ordered lists:

$$\begin{aligned} \mathcal{M} &= \langle \mathcal{V}, \mathcal{F} \rangle \\ \mathcal{V} &= \{v_1, v_2, \ldots, v_{n_v}\} \\ \mathcal{F} &= \{f_1, f_2, \ldots, f_{n_f}\} \end{aligned} \tag{3.1}$$

where $\mathcal{V}$ is a list of n_v three dimensional vertices v_i and $\mathcal{F}$ is a list of polygons each specified as a list of vertex indices $f_i = \{v_{(i,1)}, \ldots, v_{(i,n_{v_i})}\}$. If $n_{v_i} = 3$, the polygon is a triangle. It is the most common representation in the literature. The convexity of the triangle simplifies the rendering and reduces the memory cost. Indeed, the triangles rasterization is faster and also natively supported in hardware. This makes triangle discrete mesh the most efficient model to represent a surface: even when a surface is represented by a parametric or implicit model, it is eventually converted to a discrete mesh at least for the rendering. Because triangle meshes are extensively used in Computer Graphics, and in particular during the rendering of the forest objects we model in our research, we introduce in next sub-sections some key geometric properties.

1.1.1 Estimation of the local geometry of meshes

One of the strength of meshes is actually that they offer a local control on the surface. Many works showed that local geometric attributes such as normal and curvature can be efficiently and rather accurately estimated. Those attributes are widely used in applications such as remeshing[3], smoothing[49], segmentation[145], visualization[82], but also in algorithms dedicated to the ground surface extraction of aerial LiDAR scanner (ALS) such as the TIN-iterative method proposed by Axelsson[13].

As pointed out by Meyer[108], since meshes are piecewise linear surfaces, the notion of continuous normal vectors or curvatures is non trivial. Actually, assuming that the mesh approximates a continuous surface, the following question arises: can we define geometric attributes for the mesh that converge towards those of the continuous surface when the mesh density increases. Thus, several expressions have been designed to estimate those first and second order differential properties on discrete surfaces. Commonly, for a given vertex $\mathbf{v}_i$, the vector $\mathbf{n}_i$ that approximates the true normal is computed as a weighted average of faces normals in a one-ring neighborhood of the vertex (see Equation 3.2 and Figure 3.2a)).

$$\mathbf{n}_i = \sum_{j=1}^{m_i} w_j \times \mathbf{N}_j \tag{3.2}$$

where m_i is the number of edges adjacent to the vertex $\mathbf{v}_i$ and $\mathbf{N}_j$ is the normal of the j^{th} face. Stressing that the normal vector should only be defined very locally, the weights $\{w_j\}$ are computed by Thürmer and Wüthrich[143] using the angle between both edges of face adjacent to $\mathbf{v}_i$. However, this normal remains consistent only if the faces are subdivided linearly, introducing vertices which are not on a smooth surface. Max[102] proposed a different approach and derived the weights $\{w_j\}$ assuming that the surface locally approximates a sphere. These weights are therefore exact if the object is a tiling of a sphere. However, it is unclear how this approximation behaves on more complex meshes, since no error bounds are defined. Regarding the approach proposed by Meyer[108], it proposes to link the weights to the area of the local Voronoi cell (see Section 1.1.2 for the definition of the Voronoi diagram).

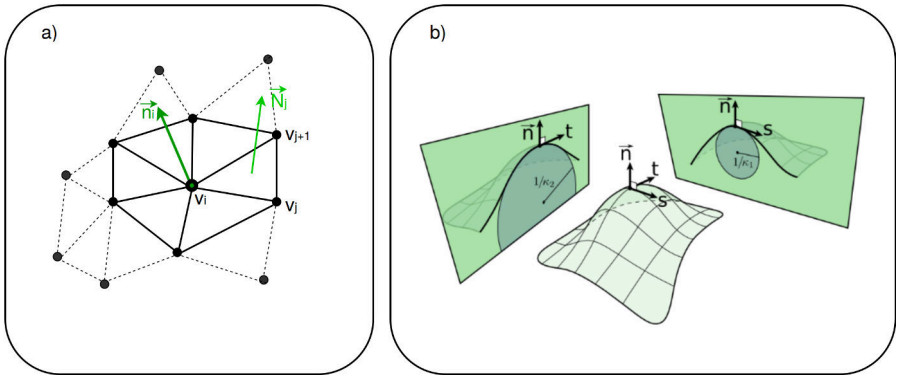


Figure 3.2: Illustration of normal (a) and curvature (b) on a surface mesh. (b) is inspired from an illustration of the course CS 177: Discrete differential geometry of the California Institute of technology under Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Surface curvature, which is a second order differential property, is essential for many techniques in analysis and rendering. The normal curvature $\mathcal{K}_n$ of a surface at a vertex $\mathbf{v}$ is the reciprocal of the radius of the circle that best approximates a normal slice of surface along that direction. For a smooth surface, $\mathcal{K}_n$ expressed in the tangent plane, is a quadratic form, called curvature tensor. Let then exist a basis of the tangent plane $(\mathbf{d}_1, \mathbf{d}_2)$ orthogonal for $\mathcal{K}_n$. In this basis:

$$\mathcal{K}_n(\mathbf{s}, \mathbf{t}) = \mathcal{K}_1\mathbf{s}^2 + \mathcal{K}_2\mathbf{t}^2 \tag{3.3}$$

where $\mathcal{K}_1$ and $\mathcal{K}_2$ are called the principal curvatures at $\mathbf{v}$ and $(\mathbf{d}_1, \mathbf{d}_2)$ its principal directions, which are actually the directions along which the normal curvature reaches its minimum and maximum. The two closely related quantities $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$ called the Gaussian curvature and $\mathcal{H} = (\mathcal{K}_1 + \mathcal{K}_2)/2$ called the mean curvature, play a very important role in the theory of surfaces.

Figure 3.2b) illustrates principal curvatures and directions at a given vertex of normal $\mathbf{n}$. Several methods have been proposed for estimating principal curvatures and directions. They can be grouped into three categories: (i) Patch fitting methods that fit an analytic surface to points in a local region and then compute analytically curvatures on the resulting smooth surface[34]; (ii) Normal curvature-based methods that first estimate the normal curvature along the edges adjacent to a vertex then approximate the curvature tensor to obtain principal curvatures and directions $\mathcal{K}_1$ and $\mathcal{K}_2$ [142]; and (iii) mean and Gaussian curvatures approximating methods based on Gauss's Theorema Egregium. Lightweight and easier to implement, the last group of methods are broadly used. They are based on the estimation of the Gaussian curvature through the angular defect (several studies[108,24] refine the computation and study the convergence towards the smooth surface curvature).

1.1.2 Voronoi diagram and Delaunay triangulation

As opposed to the point we just covered, we can approach discrete meshes globally. Surface reconstruction methods based on discrete meshes focus on estimating a polygonal mesh interpolating the 3D samples. However, such a mesh is clearly not unique and according to the neighborhood choice, the resulting mesh can be more or less regular or contain stretched or degenerate polygons or rectangles. The quality and regularity of the mesh is thus a key issue. Delaunay triangulations and its dual, namely the *Voronoi diagram*, are essential both from a theoretical and applied perspective. In this context, they stand as a fundamental data structures[12].

For sake of simplicity, we will introduce Delaunay triangulations and Vononoi diagrams in 2D. Given a finite set of points in the plane, the idea is to assign to each point a region of influence in such a way that these regions partition the plane. Considering $\mathcal{P} \subseteq \mathbb{R}^2$ a set of n points, V_p the Voronoi region of $\mathbf{p} \in \mathcal{P}$ is defined as the set of points $\mathbf{x} \in \mathbb{R}^2$ that are as close to p as to any other points in $\mathcal{P}$.

$$V_p = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x} - \mathbf{p}\| \leq \|\mathbf{x} - \mathbf{q}\|, \forall \mathbf{q} \in \mathcal{P}\} \tag{3.4}$$

The Voronoi diagram of $\mathcal{P}$ is the graph whose vertices are the Voronoi regions and whose edges

are given by the adjacency between them.

Considering a Voronoi diagram of $\mathcal{P}$, the dual graph connecting $\mathbf{p}, \mathbf{q} \in \mathcal{P}$ if and only if their Voronoi regions intersect along a common line segment, is referred to as the *Delaunay triangulation*[67].

A restricted Delaunay triangulation is a subcomplex of the three-dimensional Delaunay triangulation. It is defined as follows: considering a surface $\mathcal{S}$ and $\mathcal{P} \subset \mathcal{S}$ a finite set of n points sampled from $\mathcal{S}$, $D(\mathcal{P})$ and $V(\mathcal{P})$ denote respectively the Delaunay triangulation and the Voronoi diagram of $\mathcal{P}$. In that case, the restricted Delaunay triangulation $D_{\mathcal{S}}(\mathcal{P})$ is the set of facets of the Delaunay triangulation whose dual edges intersect the surface (see Cohen and Morvan[42] for more details). Figure 3.3 illustrates this concept on a $2D$ point set sampled on a curve.

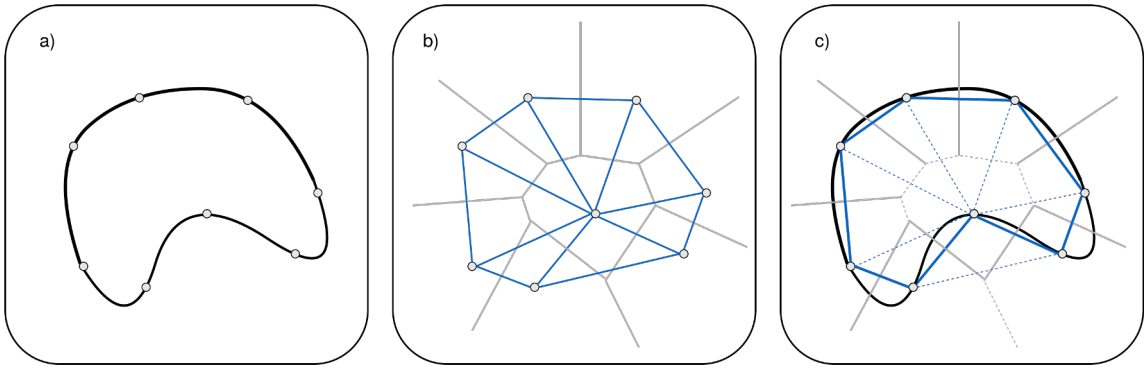


Figure 3.3: Illustration of the restricted Delaunay triangulation for a 2D point set sampled on a smooth curve: a) the point set, b) Delaunay triangulation and Voronoi diagram of point set and c) The Delaunay triangulation restricted to the curve is the set of Delaunay faces whose dual voronoi faces have a non-empty intersection with the curve (dashed edges are Delaunay but not restricted Delaunay).

The definitions and illustrations given here refer to a 2D problem. In that case, Shamos and Hoey[138] were the first to introduce the Voronoi diagram for a finite set of points in the Euclidean space $\mathbb{E}^2$ into the field of computational geometry, providing the first efficient algorithm for its computation. For higher dimensional problems, by using a dual correspondence to convex hulls discovered by Brown[27], Voronoi diagram can be obtained following the method proposed by Seidel[137].

In the context of 3D data processing, Delaunay triangulation and Voronoi diagram have proven to be mathematically powerful tools for surface reconstruction. Indeed, while 3D laser scanners generate noisy point clouds, the resulting raw meshes are also subject to noise and may contain degenerate triangles. Thus, several algorithms[75,4] have been proposed to approximate the raw mesh with another mesh of better quality. Isotropic remeshing receives much attention in the

current research, in particular the Centroidal Voronoi tesselation proposed by Du[51].

1.1.3 Strengths and weaknesses of meshes

As pointed out previously, triangle meshes are natively supported in hardware, thus they stand as the model of choice for computer graphics. Because of their simple representation as a collection of vertices, edges and faces, such models appear very effective for processing and refinement. Indeed, the control on the surface is very local: local deformations and manipulations during processing operations affect only a limited amount of elements.

However, in the context of surface modeling based on TLS point clouds, the choice of discrete mesh modeling is questionable. Indeed, data sets are too heavy, noisy and occluded to be reliable for the construction of the mesh, and meshing remains a tedious task when the size of the sample data-set increases. The successive re-meshings needed to obtain an adequate model, multiply the cumbersome computations. Moreover, the in-homogeneity of sampling leads to an unbalanced mesh: with larger polygons far from the sensor, tiny polygons close to it (where the point density can reach 1 millions points/m^2) and stretched polygons in occluded areas. Such problem, which depends on the point cloud by nature, can hardly be managed through meshes.

1.2 Parametric surfaces

Let us now briefly introduce parametric surfaces in order to justify our modeling choices. Parametric models have been largely studied and used in computer assisted modeling and reconstruction. They can be expressed as:

$$\mathcal{S}(u,v) = \begin{cases} x = f(u,v) \\ y = g(u,v) \\ z = h(u,v) \end{cases} \tag{3.5}$$

where u and v are surface parameters varying in a domain $D \subset \mathbb{R}^2$, typically a unit square $[0,1]^2$. Parametric representation provides a direct access to the local geometry of the surface. While parametric surfaces can be very generic, the next subsections introduce the Basis Splines (B-Splines) and the Non-Uniform Rational Basis Splines (NURBS), that stands as the most common parametric formulation.

1.2.1 B-Splines and NURBS

Given the following information:

- a set of $m+1$ rows and $n+1$ control points $\mathbf{p}_{i,j}$, where $0 \leq i \leq m$ and $0 \leq j \leq n$
- a knot vector of $h+1$ knots in the u-direction, $\mathcal{U} = \{u_0, u_1, \ldots, u_h\}$
- a knot vector of $k+1$ knots in the v-direction, $\mathcal{V} = \{v_0, v_1, \ldots, v_k\}$
- the degree p in the u-direction
- the degree q in the v-direction

the B-spline surface defined by these information is the following:

$$\mathcal{S}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j} \cdot \mathcal{N}_{i,p}(u) \cdot \mathcal{N}_{j,q}(v) \tag{3.6}$$

The coefficient of control point $\mathbf{p}_{i,j}$ is the product of two 1D B-spline basis functions, one in the u-direction, $\mathcal{N}_{i,p}(u)$, and the other in the v-direction, $\mathcal{N}_{j,q}(v)$ resulting in a 2D B-spline function. The i^{th} basis function $\mathcal{N}_{i,p}(u)$ defined on a knot vector $\mathcal{U} = \{u_1, \ldots, u_{m+p+1}\}$ is recursively defined by Cox[46] - De Boor[48] recursion formula as:

$$\mathcal{N}_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} \cdot \mathcal{N}_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} \cdot \mathcal{N}_{i+1,p-1}(u) \tag{3.7}$$

where $\mathcal{N}_{i,0}(u) = \begin{cases} 1 & \text{if} \quad u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}$

The NURBS, introduced by Versprille[151], are a form of generalization of B-splines allowing to assign a weight to the control points thus improving the modeling flexibility. They are defined as a tensor product form:

$$\mathcal{S}(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} \cdot \mathbf{p}_{i,j} \cdot \mathcal{N}_{i,p}(u) \cdot \mathcal{N}_{j,q}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} \cdot \mathcal{N}_{i,p}(u) \cdot \mathcal{N}_{j,q}(v)} \tag{3.8}$$

where $\{w_{i,j}\}$ are the weights, $\{\mathbf{p}_{i,j}\}$ are the control points, p and q are the degrees in the u and v directions and $\{\mathcal{N}_{i,p}\}$ and $\{\mathcal{N}_{j,q}\}$ are the normalized B-spline basis functions defined on the non-

periodic knot vectors $\mathcal{U} = \{0, 0, \ldots, 0, u_{p+1}, \ldots, u_m, 1, 1, \ldots, 1\}$ and $\mathcal{V} = \{0, 0, \ldots, 0, v_{q+1}, \ldots, v_n, 1, 1, \ldots, 1\}$ respectively.

A NURBS surface is completely determined by its control points $\{\mathbf{P}_{i,j}\}$, its weights $\{w_{i,j}\}$ and the knot vectors $\mathcal{U}$ and $\mathcal{V}$. The surface changes in a predictable way according to control points movement. The weighting factors $\{w_{i,j}\}$ determine how much a control point influences the shape locally (*note*: if the weighting factors are all assigned to one, the NURBS model is reduced to the particular case of B-Spline surfaces).

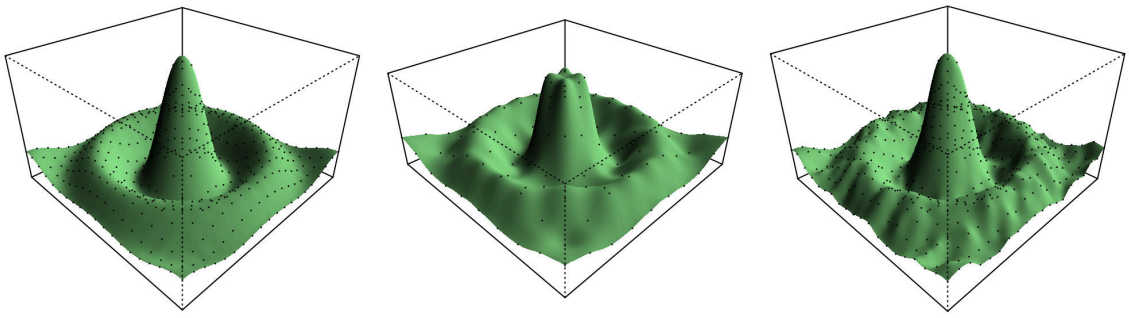


Figure 3.4: Influence of sample characteristics on interpolating B-Spline: (left) 400 points sampling the function $z = sin(r)/r, r = \sqrt{x^2 + y^2}$ and the approximating B-Spline in green, (center) idem with 100 points only, (right) 400 point sampling the same function with an additional white noise of amplitude equal to one.

Figure 3.4 illustrates the effect of sample characteristics on the B-Spline interpolation: the modeling requires a descent point density to retrieve the initial shape, plus, it is highly sensitive to noise.

1.2.2 Strengths and weaknesses of parametric surfaces

NURBS and parametric surfaces in general are suitable for smooth and continuous surface modeling. Hence, they are broadly used in computer aided geometric design. However, NURBS prove difficult to control when the model becomes more and more complex. Indeed, as the shape is completely defined by the control points, the knots vector and associated weights, deformation and refinement require heavy computation to re-estimate the geometry. Actually, the modeling of complex objects using one NURBS patch leads to the creation of extremely complex knot vectors that become impractical to handle. Moreover, this sort of modeling requires a regular network of control points. In the context of TLS processing, heavy data-sets and the consecutive refinements due to noise filtering would lead to prohibitive computations. Moreover, this model

requires ordered control points in order to be linked to a regular network of 2D parameters. In practice, it is impractical for TLS point clouds.

1.3 Implicit surface

Let us now come to the last class of models presented in this thesis, namely implicit surfaces. Such models describe the surface as a levelset of an implicit function, which is a function from $\mathbb{R}^3$ to $\mathbb{R}$. The implicit function can be in the form of a signed distance field[78] that quantifies the distance of a point to the surface. Other models that are not implicit *a priori*, an indicator function that discriminates between the interior and the exterior of a solid shape for instance, are sometimes brought back to the implicit surfaces class of model (by smoothing the border through convolution as proposed by Kazhdan[86] in its Poisson surface reconstruction).

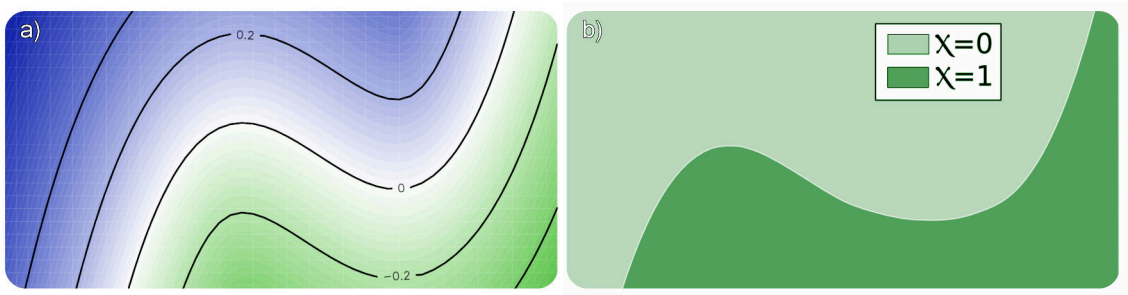


Figure 3.5: Example of implicit functions in 2D: (a) Signed distance function with its corresponding level-sets as black lines (the color ramp scales from green for the negative values to white for the zero-values then to blue for the positive values) and (b) Indicator function segments the plane in two areas: the light green area where the function is 0 and the darker green area where the function is 1.

The reconstructed surface is usually rendered as a discrete mesh. The Marching Cubes (MC) algorithm is probably the most popular method currently for iso-surface extraction and rendering. The implicit function is sampled on a 3D grid and the iso-surface is then extracted from this scalar field. Regular 3D grids can be used, as proposed by the first Marching Cubes (MC) algorithm introduced by Lorensen[97]. Since the MC was proposed, many researchers have focused on extending the basic approach, resolving its ambiguities, and improving its performance (see the survey proposed by Newman[113] for more details).

Among implicit functions, radial basis functions (RBF) stand out as major class of functions with solid mathematical properties and prove efficient to smooth noise and limit complexity.

Hardy[74] with his decomposition on a multiquadric functions basis, and Duchon[52] with his thin plate spline based model, opened the door of such surface modelings.

1.3.1 Radial basis functions

In the present work, we use RBF functions as a modeling tool and we will hence introduce them as a family of functions used as basis functions to approximate the surface. However, RBF can be studied from a more general point of view (reproducing kernels Hilbert spaces in functional analysis which are a fundamental theory in statistics, artifical intelligence and recognition). The reader can refer to Berlinet and Thomas-Agnan[19] for a more detailed presentation.

In a basic way, modeling with RBF consists of the decomposition in a finite linear combination of translations of a given basis function, as illustrated in $1D$ on Figure3.6. These basis functions (see definition below) are radially-symmetric functions.

Definition 1: Radial functions

A function $\Phi : \mathbb{R}^d \to \mathbb{R}$ *is called a radial basis function if there exists a univariate function* $\varphi : \mathbb{R}^+ \to \mathbb{R}$ *such that* $\Phi(\boldsymbol{p}) = \varphi(r)$*, with* $r = \|\boldsymbol{p}\|$*, where* $\|\cdot\|$ *is some norm on* $\mathbb{R}^d$*, usually the Euclidean norm.*

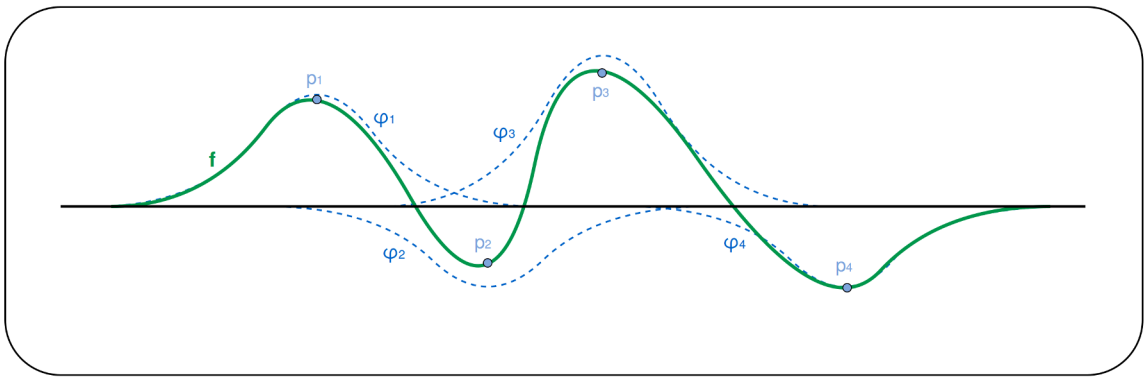


Figure 3.6: Illustration of the RBF interpolation on four points in 1D. Blue hashed curves represent the contribution of each RBF. The green curve is the resulting curve interpolating sample points.

1.3.1.1 Some elements on interpolation theory

RBF have been widely studied through interpolation theory. The results then obtained actually provide the tools for surface reconstruction with RBF. Considering $\Omega \subseteq \mathbb{R}^d$ and $\mathcal{C}(\Omega)$, the

space of continuous function on Ω, the interpolation problem can be stated as follows: Given the data $\mathcal{D} = \{(\mathbf{x}_i, s_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i=0,\dots,N-1}$, a function $f : \mathbb{R}^d \mapsto \mathbb{R} \in \mathcal{C}(\Omega)$ interpolates $\mathcal{D}$ if $f(\mathbf{x}_i) = s_i, \forall i \in \{0, \dots, N-1\}$. Given a set of basis functions $\{u_k\}_{k=0,\dots,N-1}$, a major question is: can we find an interpolant of $\mathcal{D}$ in the subspace of finite dimension generated by this basis? In other words, does any linear combination of such functions $f = \sum_{k=0}^{N-1} c_k \cdot u_k, c_k \in \mathbb{R}$ interpolates $\mathcal{D}$?

Given this linear decomposition, resolving the interpolation problem consists in solving a linear system of equation of the form:

$$\mathcal{A} \cdot \mathbf{c} = \mathbf{s} \tag{3.9}$$

where $\mathcal{A}_{i,j} = u_j(p_i)$, $i, j \in \{0, \dots, N-1\}$, $\mathbf{c} = [c_0, \dots, c_{N-1}]^T$ and $\mathbf{s} = [s_0, \dots, s_{N-1}]^T$. Now this system has a solution if and only if the matrix $\mathcal{A}$ is non-singular.

In 1D, considering the basis of polynoms $\Pi_m(\mathbb{R})$, it is well known that a polynomial of degree $N-1$ can interpolate N data points. However for higher dimensional problems, Mairhuber[100] brought the following negative result:

Definition 2: Haar space

Let the finite dimensional linear function space $\mathcal{B} \subseteq C(\Omega)$ have a basis $\{u_k\}_{k=0}^{N-1}$. Then $\mathcal{B}$ is a Haar space on Ω if for any set of distinct $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{N-1}$ in Ω, given the matrix $\mathcal{U} = u_j(\boldsymbol{x}_k)_{j,k \in \{0,\dots,\mathbb{N}-1\}}$, $det(\mathcal{U}) \neq 0$.

Theorem 1: Maihuber-Curtis

If $\Omega \in \mathbb{R}^n$, $n \geq 2$, contains an interior point, then there exist no Haar spaces of continuous functions except for the one-dimensional ones.

In other words, if we want to have a well-posed scattered data interpolation problem in dimension $D \geq 2$, we cannot fix in advance the set of basis functions we plan to use. Instead, the basis needs to depend on the data locations. Actually, in order to obtain such data dependent approximation spaces, we consider positive definite functions. The following paragraphs summarize the different properties of RBF in this context (see Wendland[155], Buhmann[29] and Fasshauer[60] for more details).

1.3.1.2 Strictly positive definite radial basis functions

A RBF interpolation is a function f defined as:

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} \alpha_i \cdot \varphi_i(\mathbf{x}) \tag{3.10}$$

where $\{\varphi_i\}_{i=0,\dots,N-1}$ is a set of functions frequently chosen as the shift of a single RBF : $\varphi_i(\mathbf{x}) = \varphi(\|\mathbf{x} - \mathbf{x}_i\|)$, as stressed out by Fasshauer[62].

A major issue is then to guarantee interpolation with such linear combination. To do so, we consider strictly positive definite functions. The definition of (strictly) positive definite function is given as:

Definition 3: Positive definite function (respectively strictly positive definite function)
A function $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ is positive definite (respectively strictly positive definite function) on $\mathbb{R}^d$ if and only if for all finite subset $\mathcal{P} = \{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{n-1}\}$ of distinct points of Ω the matrix $\Phi_\mathcal{P}$ with entries $\varphi(\boldsymbol{p}_i - \boldsymbol{p}_j)$, $0 < i, j < n-1$ is positive definite (respectively strictly positive definite function):

$$\boldsymbol{c}^T \cdot \Phi_\mathcal{P} \cdot \boldsymbol{c} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_i \cdot c_j \cdot \varphi(\boldsymbol{p}_i - \boldsymbol{p}_j) \geq 0, \forall \boldsymbol{c} \in \mathbb{R}^n \tag{3.11}$$

(respectively $\boldsymbol{c}^T \cdot \Phi_\mathcal{P} \cdot \boldsymbol{c} > 0, \forall \boldsymbol{c} \in \mathbb{R}^n$)

The interpolation problem, expressed in its matrix form becomes :

$$\Phi \cdot \alpha = \mathbf{s} \tag{3.12}$$

where $\alpha = [\alpha_0, \dots, \alpha_{N-1}]^T$, $\mathbf{s} = [s_0, \dots, s_{N-1}]^T$ and $\Phi_{i,j} = \varphi(\mathbf{x}_i - \mathbf{x}_j)$.

Let us assume that φ is a strictly positive definite RBF, it is well known that the matrix Φ (which is symmetric definite positive) is invertible. Hence this guarantees the existence and uniqueness of an interpolation function of the form 3.10.

Several RBF found in the literature are strictly positive definite. Some common strictly positive definite RBF are presented in Table 3.1.

Name	$\Phi(r)$	Condition
Gaussian	e^{-r^2}	
Matern	$r^u K_u(r)$, K_u spherical Bessel function	$u > 0$
Inverse multiquadric	$(1 + r^2)^{b/2}$	$b < 0$

Table 3.1: Common strictly positive definite radial basis functions. ($r = \|\mathbf{x} - \mathbf{x}_i\|$).

1.3.1.3 Conditionally positive definite radial basis functions

While the interpolation problem leads to the idea of using strictly positive definite RBF, an important class of RBF does not verify this property. However, a weaker property is defined (conditionally positive definite RBF) allowing to extend the interpolation results. As stressed by Schaback[133], in such cases, one has to add polynomials of a certain maximal degree in the linear decomposition Equation 3.10.

Let $\Pi_m(\mathbb{R}^d)$ denote the space spanned by all d-variate polynomials of degree up to m, and pick a basis $\pi_0, \ldots, \pi_{q-1}$ of this space. The dimension q comes out to be $q = \binom{m+d}{d}$ and Equation 3.10 is then augmented to:

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} \alpha_i \cdot \varphi(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=0}^{q-1} \beta_j \cdot \pi_j(\mathbf{x}) \tag{3.13}$$

Table 3.2 provides a selection of the most useful conditionally positive definite functions.

Name	$\Phi(r)$	Parameters	Order m
Multiquadrics	$(-1)^{b/2}(1 + r^2)^{b/2}$	$b > 0, b \notin 2\mathbb{N}$	$b/2$
Polyharmonic splines	$(-1)^{b/2} r^b$	$b > 0, b \notin 2\mathbb{N}$	$b/2$
Polyharmonic splines	$(-1)^{1+b/2} r^b . log(r)$	$b > 0, b \in 2\mathbb{N}$	$b/2 + 1$
Thin-plate spline	$r^2 . log(r)$		2

Table 3.2: Common conditionally positive definite radial basis functions ($r = \|\mathbf{p} - \mathbf{p}_i\|$)

The q additional degrees of freedom induced are then removed by q additional homogeneous equations, restricting the coefficients $\alpha_0, \ldots, \alpha_{n-1}$ in Equation 3.13:

$$\sum_{i=0}^{n-1} \alpha_i \cdot \pi_0(\mathbf{x}_i) = \ldots = \sum_{i=0}^{n-1} \alpha_i \cdot \pi_{q-1}(\mathbf{x}_i) = 0 \tag{3.14}$$

In this setting, RBF and polynomial coefficients for interpolation, α and β respectively, are calculated by solving the following linear system, expressed in its matrix form as:

$$\left[\begin{array}{c|c} \Phi & P^T \\ \hline P & 0 \end{array}\right] \cdot \left[\begin{array}{c} \alpha \\ \hline \beta \end{array}\right] = \left[\begin{array}{c} \mathbf{s} \\ \hline 0 \end{array}\right] \tag{3.15}$$

where $\Phi_{i,j} = \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$ and $P_{i,j} = \pi_j(\mathbf{x}_i)$.

Some profile of useful strictly and conditionally positive definite radial basis functions are illustrated on Figure 3.7.

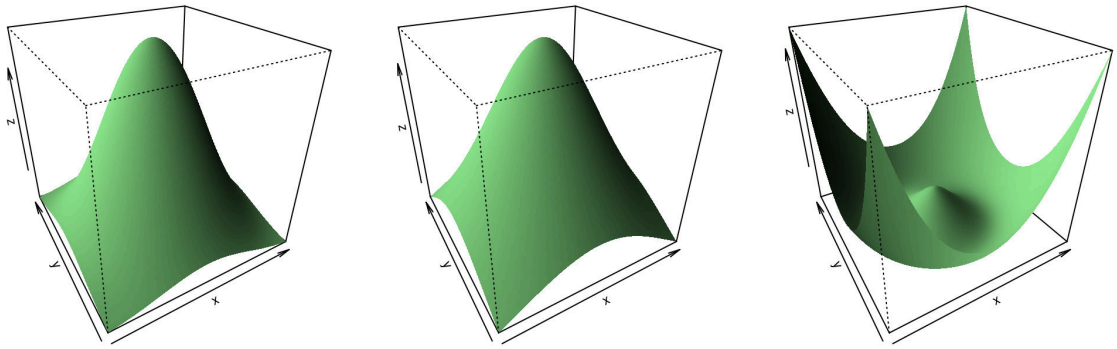


Figure 3.7: Commonly used types of radial basis functions: (left) Gaussian $\Phi(r) = e^{-r^2}$, (center) inverse multiquadric $\Phi(r) = \frac{1}{\sqrt{1+r^2}}$, and (right) thin plate spline $\Phi(r) = r^2 ln(r)$.

1.3.1.4 Compactly supported radial basis functions

Using globally supported radial basis functions allows the reconstruction of both non uniform sampling and samples affected by missing data. However, as it entails inverting the matrix Φ previously defined, the computational cost becomes prohibitive while dealing with a heavy dataset.

To make RBF methods suitable for larger point clouds, defining compactly supported radial basis functions (CSRBF) is a key issue, thus leading to sparse interpolating matrices and reducing the computational complexity. In this context, the most useful example is Wendland's function[152]: a minimum-degree polynomial solution for locally supported radial basis functions that guarantees positive-definiteness of the interpolation matrix. These functions are defined iteratively through pseudo integral and differential operators (see Wendland[152] for a detailed presen-

tation). Basically, for various dimensions d and required continuities C^k, those functions have the form:

d	1	3	5
C^0	$(1-r)_+$	$(1-r)_+^2$	$(1-r)_+^3$
C^2	$(1-r)_+^3(3r+1)$	$(1-r)_+^4(4r+1)$	$(1-r)_+^5(5r+1)$
C^4	$(1-r)_+^5(8r^2+5r+3)$	$(1-r)_+^6(35r^2+18r+3)$	$(1-r)_+^7(16r^2+7r+1)$

Table 3.3: Wendland's CSRBF for $d \in \{1,2,3\}$ and continuity $C^k, k \in \{0,2,4\}$. $(\cdot)_+$ is the cut-off function such as $(x)_+ = x$ if $x \geq 0$, 0 otherwise. These functions have radius of support equal to 1. Scaling of the basis functions, *i.e.* $\varphi(r/\sigma)$, allows any desired radius of support σ.

In particular, the function $(1-r)_+^4(4r+1)$ ranks amongst the most usefull RBF; it is positive definite in $\mathbb{R}^d$ for $d \leq 3$ and twice differentiable in x when $r = \|x\|$.

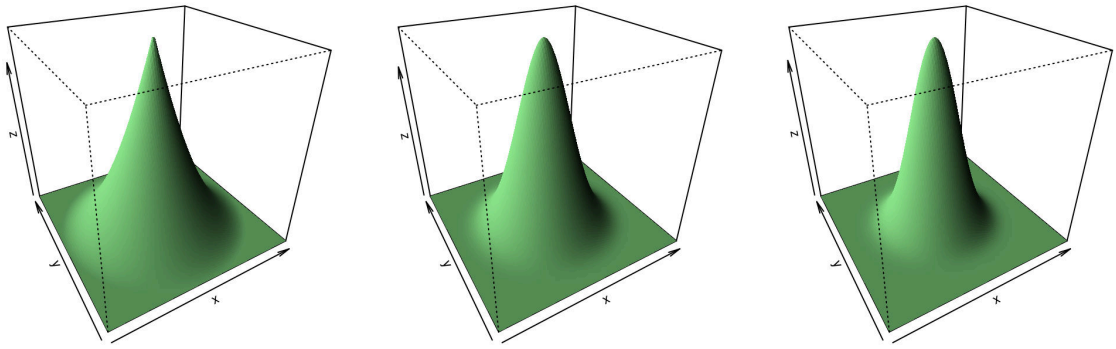


Figure 3.8: Wendland's compactly supported radial basis function for $d = 3$: (left) C^0, (center) C^2 and (right) C^4.

The finite support of such functions allows faster filling of the interpolation matrix[154], this matrix becoming sparse at the same time. The inversion of the matrix is then speeded up by using a direct sparse matrix solver.

Morse[111] summarizes the advantage of CSRBF over classical RBF as follows: considering the complexity of the interpolation of n points, $O(n\log(n))$ for CSRBF and $O(n^2)$ for RBF computation is required to set-up the interpolation problem, *ie* to build the system of equations. $O(n^{1.5})$ computation is required to solve the system of equations with CSRBF ($O(n^2)$ with RBF). And finally, $O(\log(n))$ computation is required to evaluate the function ($O(n)$ with RBF).

Nevertheless let us point out that CSRBF may have a drawback. Indeed, the choice of the support radius is a delicate question. Small support radii lead to an efficient way to solve the system but also to a poor representation (see Figure3.9). Actually, the resulting surface may even

fail to be continuous as RBF are too localized. On the contrary, large support radii lead to a good representation but also to worse computational performance, because of the densification of the interpolation matrix.

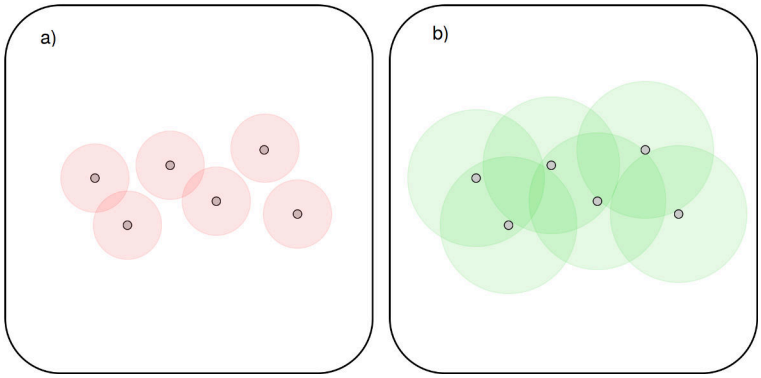


Figure 3.9: Effect of the support size of the CSRBF: too small supports lead to non-continuous surface (level-set of the function) (a). Larger supports leads to smoother reconstructed surface (b) but at higher computational cost. Indeed, the size of the overlapping zones is directly linked to the density of the interpolation matrix.

1.3.2 Strengths and weaknesses of implicit surfaces

Implicit surfaces appear to be an efficient framework for surface reconstruction as such models allow to smooth the noise by approximating the input points. Moreover, because the surface is extracted as a levelset of an implicit function, the resulting mesh is guaranteed to be a watertight manifold. However, spurious components can be produced as unwanted level-set and would need to be filtered out. In the context of TLS processing, modeling with CSRBF stands as a powerful option: the required resolution of the shape geometry is determined by a wise distribution of the centers and the control on the shape remains very local by limiting the support radius of the basis functions.

2 The challenge of surface reconstruction

Surface reconstruction is concerned with recovering a digital representation of real world objects sampled by a 3D point cloud. Those point clouds, resulting from different 3D acquisition methods, have intrinsic characteristics and artifacts, as illustrated on Figure 3.10.

As pointed out in Chapter 2, the scattering characteristics of the scanned material and the precision of the scanner itself induce *noise in the data*. The goal of the reconstruction algorithm

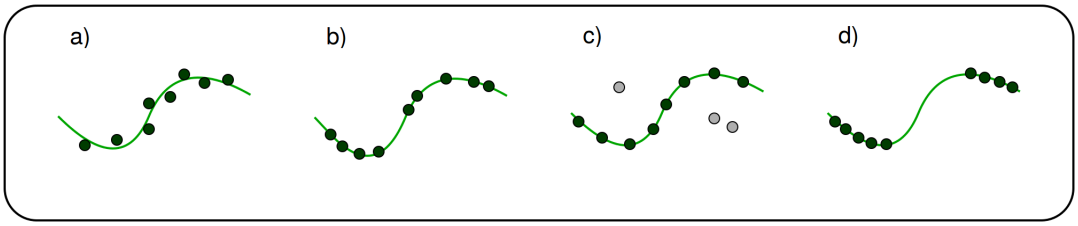


Figure 3.10: Different forms of point cloud artifacts, shown in the case of a curve in 2D: (a) noisy data, (b) non-uniform sampling, (c) outliers, and (d) missing data or hole.

is to produce a surface approximating the points without over-fitting to the noise[35]. The scanner orientation, and the variation of the distance to the scanned surface produce an in-homogeneous distribution of points on the surface. Algorithms have to tackle this *non-uniform sampling* by computing a sampling density at every point to estimate correctly the surface. Because of the structural complexity of the scene or because of the acquisition process itself, points lying far from the surface can arise. Those points, called *outliers* should be filtered out and should not be used to infer the surface. *Missing data or holes* are due to occlusions in the scanning process where large portions of the shape are not sampled. Missing data differ from non-uniform sampling, as the sampling density is zero in such regions.

The objective of surface reconstruction is to tackle such point clouds imperfections in order to produce a geometric model, faithful reproduction of the shape initially scanned. More precisely, the surface reconstruction problem can be stated as follows: given a set of points S sampled from a surface $\mathcal{M}$ embedded in $\mathbb{R}^3$, construct a surface $\mathcal{F}$ such that the points of S lie on or are close to $\mathcal{F}$, and $\mathcal{F}$ approximates $\mathcal{M}$ geometrically and is topologically equivalent to $\mathcal{M}$. The reconstruction problem is divided in *interpolation*, when the function goes through all the points, and *approximation*, when the function goes close to the sample points. Strides have been made in geometric modeling by developing specific algorithms. Those surface reconstruction algorithms can be roughly classified in three categories, each of them relying on specific modeling choices: (i) Delaunay based methods approximate the 3D sample points by a *discrete mesh model*, (ii) Piecewise smooth methods represent and approximate the samples by a *parametric form*, and (iii) Implicit Function methods fit an *implicit function* on the point cloud, then use a marching-cube-like algorithm to extract a level-set of the function into a mesh. As underlined previously, implicit surfaces appear to be particularly appropriate in the context of TLS processing. Therefore, in the sequel of Section 3 we present surface reconstruction methods and emphasis is given to implicit function methods.

3.1 Delaunay based methods

Gopi[68] exploits the local characteristics of the Voronoi diagram to build a mesh. By assuming that the normal to the tangent plane at the point $\mathbf{p}_i$ can be approximated by the direction of the Voronoi cell $\mathcal{V}(\mathbf{p}_i)$, this algorithm derives a local triangulation around each point. Also based on tangent planes, Cohen-Steiner & Fa[41] start with a seed triangle and grow an oriented surface by selecting triangles out of the Delaunay triangulation $\mathcal{D}(\mathcal{P})$. The methods that compute a subset of the Delaunay triangulation are usually referred to as restricted Delaunay-based methods. Among them stand two famous algorithms: the *Crust* and the *Cocone* algorithm. The *Crust* algorithm was designed by Amenta and Bern[6] from the Crust algorithm for curve reconstruction[7]. The principle of this algorithm, illustrated in 2D on Figure3.11, is the following: considering a set of sample points $\mathcal{P}$, the algorithm first computes the Voronoi diagram $\mathcal{V}(\mathcal{P})$. Then, a set of poles $\mathcal{Q}$ of the Voronoi diagram are selected and the Delaunay triangulation of $\mathcal{P} \bigcup \mathcal{Q}$ is computed. Finally, the algorithm discards all triangles having not all vertices in $\mathcal{P}$ to extract the candidate triangles of the reconstructed surface.

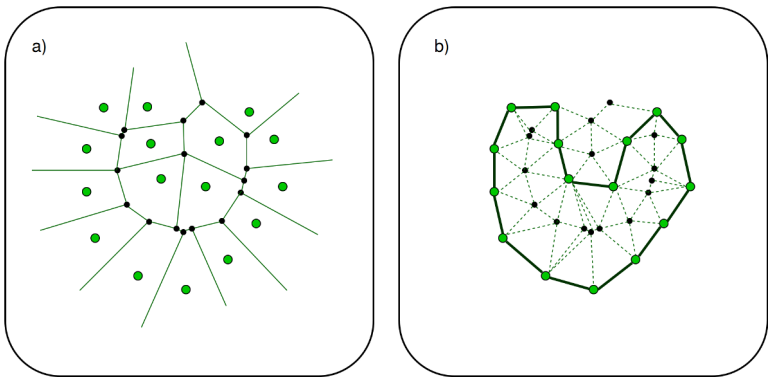


Figure 3.11: Illustration of Crust algorithm in 2D: a) Voronoi diagram and Voronoi poles (black dots) of the input points (green dots), b) Delaunay triangulation of the union of the sample points and their Voronoi poles. The crust is indicated by the solid line.

From the Crust algorithm, Amena[8] designed the *Cocone* algorithm. The Cocone algorithm demands in contrast to the Crust algorithm only the computation of the Delaunay triangulation $\mathcal{D}(\mathcal{P})$, thus reducing the runtime and memory complexity. Two important methods extending those two approaches previously described and resulting in watertight meshes were Powercrust and Tight Cocone introduced by Amenta[9] and Dey[10] respectively.

While those methods prove efficient on regular and clean sampling, they face difficulties while dealing with noise and in-homogeneity. In order to handle noise and outliers, Kolluri[90] introduces the Eigen crust algorithm which uses a spectral graph partitioning. However, in areas where data are missing, the quality of the approximating surface is questionable because of the coarse underlying Delaunay triangulation.

Another category of approaches is inspired by the alpha shape reconstruction proposed by Edelsbrunner[55]. Those methods build a spatial subdivision according to the sampling $\mathcal{P}$, then classify the Delaunay tetrahedra as inside or outside the domain bounded by $\mathcal{S}$ depending on the radius of their circumscribing spheres. However, building such alpha shapes is computationally expensive. Bernardini[20] proposed a ball pivoting algorithm to construct efficiently the boundary of an alpha shape (see Figure 3.12).

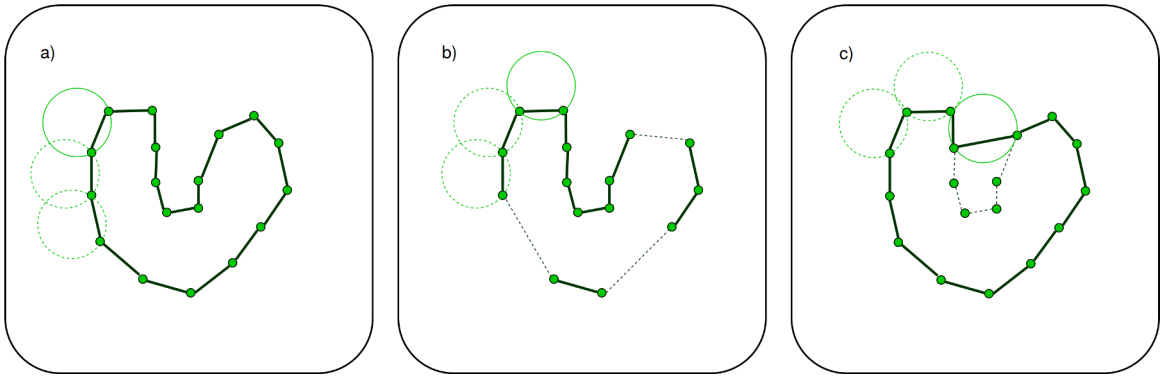


Figure 3.12: Pivoting ball algorithm in 2D: a) A circle pivots from sample point to sample point, connecting them with edges, b) when the sampling density is too low, some of the edges will not be created, leaving holes and c) when the curvature of the manifold is too large, some of the sample points will not be reached by the pivoting ball, and features will be missed.

3.2 Piecewise smooth surface methods

In order to distance from noise and in-homogeneity issues, surface fitting problem has been investigated through the adjustment of local parametric smooth patches subsequently stitched together. In this context, NURBS is one of the most used models[125] and is supported by modern standards like OpenGL. In addition, NURBS surface models are stable and flexible. These schemes are usually designed to interpolate sparse data, rather than directly fit dense, noisy point sets similar to those obtained from TLS devices. Indeed, because of the sensitivity to the parameters, *i.e.* the control points and the weights, those models turn out to be unstable for surface re-

construction problems. Some spatial regularity in the data is also required given the tensor product form of NURBS. Moreover, inherent characteristics of B-Spline and NURBS complicate the extrapolation in missing data area, and thus the filling of holes. To fit an unorganized and scattered point cloud using parametric patches, several approaches have been explored: among others, Eck[54] and Gregorski[70] fit a network of B-Spline patches while ensuring some degree of continuity between them, Dokken[50] introduces the concept of locally refined splines, and Ohtake[115] approximates locally the points with quadratic parametric patches and later merges them using radial basis functions to produce a single implicit function. While NURBS and spline based model are too flexible to handle noise and outliers of TLS scan, the relative stiffness of quadratic patches model have proven efficient to handle those imperfections in order to produce smooth and accurate surface reconstruction[116]. Moreover, because this method reconstructs the final model as an implicit surface, it stands at the verge of the implicit function methods, introduced in the following Section.

3.3 Implicit function methods

In view of the strengths and weaknesses of the mesh and parametric approaches, implicit surfaces prove suitable for the reconstruction from dense point clouds, thanks to their noise tolerance and in-homogeneity and because they are adapted to holes filling by using the neighborhood. For those reasons, numerous works focus on surface reconstruction with implicit surface from point clouds.

3.3.1 Normal estimation for reconstruction

Reconstruction methods have different prerequisites as input: while some methods rely only on the points position, many of them need an estimate of points normal as additional data. The normal, defined at every point is the direction orthogonal to the tangent plane, *i.e.* it is related to the local surface approximation at the given point. The challenge is to estimate this normal directly from the point cloud (as it will be used to further reconstruct the surface).

Hoppe[78] presentes a simple method to compute an oriented normal field by analyzing the point distribution. In practice, it first assesses for each point $\mathbf{p}_i$ an un-oriented normal by performing a principal component analysis (PCA) on the *xyz* coordinates of its neighbors N_p, *i.e.* computing the spectral decomposition of the covariance matrix C_p.

$$\mathcal{C}_p = \sum_{\mathbf{p}_i \in N_p} (\mathbf{p} - \mathbf{p}_i)(\mathbf{p} - \mathbf{p}_i)^T \tag{3.16}$$

Since the eigenvectors of $\mathcal{C}_p$ associated with the two largest values capture most of the variance in the data, the eigenvector associated with the smallest eigenvalue thus defines the un-oriented normal. In order to obtain a consistent global orientation of the normals, the algorithm wanders on the data-set modeled as a graph, and flips normals to keep them geometrically close (see Figure 3.13). Liu[96] extends this method and focuses on preserving the surface connectivity in sparse regions.

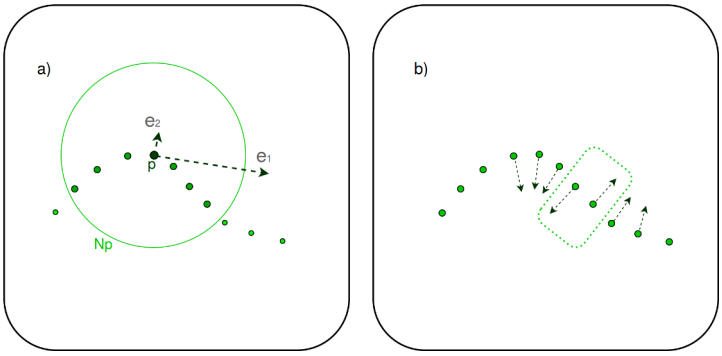


Figure 3.13: Illustration of the normal estimation in 2D. a) e_1 and e_2 are the eigen-vectors of the PCA on the *xyz* coordinates of the neighbors N_p of the point p. e_2 is then assigned as p normal, b) highlight a zone of normal inconsistency. Those normal will be re-oriented in a further step to keep a consistent global orientation.

3.3.2 Surface reconstruction as an approximate of the signed distance function

In 1992, Hoppe[78] opens a door in the field of surface reconstruction with implicit functions. Considering $\mathcal{S} = \{\mathbf{s}_i\}_{i=0}^{N}$ a dataset of 3D points, where each sample contains the position $\mathbf{p}_i = [p_{ix}, p_{iy}, p_{iz}]^T$ of the point and a normal $\mathbf{n}_i = [n_{ix}, n_{iy}, n_{iz}]^T$, this method approximates a signed distance function $f\colon \mathbb{R}^3 \to \mathbb{R}$ such that:

$$f(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_i)^T \cdot \mathbf{n}_i, \forall \mathbf{x} \in \mathbb{R}^3 \tag{3.17}$$

where $\mathbf{p}_i$ is the closest point from $\mathbf{x}$ among the sample points $\mathcal{S}$. While being straightforward to implement, this method suffers from several issues. Indeed, it is very sensitive to the quality of the pre calculated normal field. Noisy or inverted normals can affect badly the consistency of

the signed function. Furthermore, non uniform sampling can pollute the choice of the closest tangent plane and produce unstable results. The methods introduced below actually address these issues. In particular, the next sub-sections present (3.3.3) the Moving Least Squares, (3.3.4) the RBF reconstruction methods, (3.3.5) the patch stitching through partition of unity approach and finally the (3.3.6) Poisson surface reconstruction method.

3.3.3 Moving least squares (MLS)

The MLS method was first proposed by Lancaster and Salkauskas[92] in 1981 for smoothing and interpolating data. Given a set of sample points $\mathbf{x}_i$ and the corresponding f_i of a function f as $\mathbf{x}_i$, MLS method aims at approximating f. The idea is to start for an arbitrary fixed point in $\mathbb{R}^d$ with a local weighted least squares formulation and then move this point over the entire parameter domain. More formally, if we consider a set of N points $\mathbf{x}_i$ where $i \in [1, \ldots, N]$ and f_i the value of the function f sampled at $\mathbf{x}_i$, the MLS approximant g of $\{(\mathbf{x}_i, f_i), i = 1, \ldots, N\}$ is defined as:

$$g \in \Pi_m(\mathbb{R}^d), min\{\sum_{i=0}^{N-1} \vartheta(\|\mathbf{x} - \mathbf{x}_i\|) \cdot \|f_i - g(\mathbf{x}_i)\|^2\} \tag{3.18}$$

where $\vartheta : \mathbb{R}^d \to \mathbb{R}$ is a weight function whose values decrease with the distance to the positions of data points $\mathbf{x}_i$. Due to the fact that for every evaluation point $\mathbf{x}$ a linear least squares problem needs to be solved, those versions of MLS are set aside for problems with no more than few thousands of samples. However, by using compactly supported weight functions only a few terms are considered in the matrix computation (see Equation 3.19), or it is even possible to completely avoid solving the linear system[61].

In order to cope with this problem, Levin[93] pointed out that the MLS surface is actually the set of stationary points of the function $g : \mathbb{R}^3 \to \mathbb{R}^3$. A point $\mathbf{x} \in \mathbb{R}^3$ belongs to the MLS surface exactly when $g(\mathbf{x}) = \mathbf{x}$. In this context, Amenta[11] expresses the MLS surface approximation as an energy minimization problem (see Figure3.14). Considering a set of points $\mathbf{p}_i = [p_{ix}, p_{iy}, p_{iz}]^T, i \in [1, \ldots, N]$, a local fitting plane of normal $\mathbf{n} = [n_x, n_y, n_z]^T$, for each $\mathbf{q} = [q_x, q_y, q_z]^T \in \mathbb{R}^3$, the distance t between $\mathbf{q}$ and the plane, the energy of the plane E_{MLS} passing through the point $\mathbf{p} = \mathbf{q} - t \times \mathbf{n}$ is :

$$\mathcal{E}^{\mathbf{q}}_{MLS}(t, \vec{n}) = \sum_{i=0}^{N-1}((\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n})^T \cdot \mathbf{n})^2 \cdot \vartheta(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\|) \tag{3.19}$$

Let $(t_q, \mathbf{n}_q)$ be a local minima of $\mathcal{E}^{\mathbf{q}}_{MLS}(t, \vec{n})$, then $\mathbf{p}_q = \mathbf{q} - t_q \times \mathbf{n}_q$ is the nearest point to $\mathbf{q}$ and we define $f(\mathbf{q}) = \mathbf{p}_q$. The stationary points of this map f form the MLS surface. Figure 3.14 summarizes MLS method on a set of points.

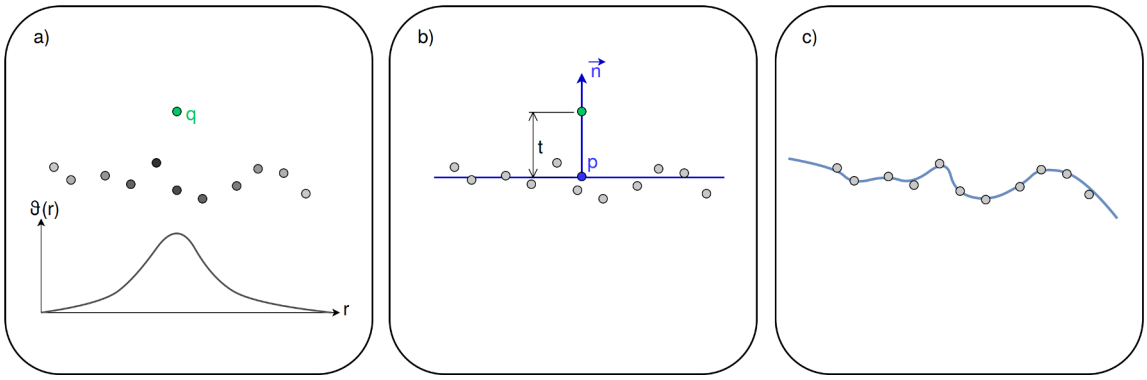


Figure 3.14: Illustration of the moving least squares principle. a) Considering input points $\mathbf{p}_i \in P$ and a Gaussian-like weight function ϑ whose values at each $\mathbf{p}_i$ are denoted by its shade of gray, b) we evaluate the energy of the plane E_{MLS} passing through the point $\mathbf{p} = \mathbf{q} - t \times \mathbf{n}$. c) The minimization of E_{MLS} leads to the construction of the MLS surface.

A major characteristic of MLS is the use of the weighting function ϑ: it balances the influence of the points in both estimating the tangent space as well as constructing the polynomial. The adjustment of the spatial influence of ϑ also helps to smooth the noise. In the case of a non uniform sampling, the support size of the weighting function needs to be adjusted to the point density. In this context, Lipman[95] introduced a new data-dependent error formula for the MLS approximation. However, MLS methods still struggle in presence of missing data[18]: actually the large spatial support size required near the holes spoils the reconstruction.

3.3.4 RBF RECONSTRUCTION METHOD

To reconstruct surfaces from point-clouds derived from 3D range scanners and simultaneously to smooth the data noise, Carr[32] proposes an unified framework that is simple and elegant based on the implicit representation of object surfaces with RBF. As pointed out by Carr, an RBF model offers a compact functional description of a set of surface data. Interpolation and extrapolation are inherent in the functional representation. The RBF model associated with a surface can be evaluated anywhere to produce a mesh at the desired resolution. The benefits of modeling surfaces with RBFs have been broadly recognized[132,33,148,147].

Given a set of sample points $\mathbf{x}_i$ lying on the surface, the methods looks for a function f satisfying: $f(\mathbf{x}_i) = 0, i = 1, \ldots, N$. To avoid the trivial solution that f is zero everywhere, extra points are added and are given non-zero values. The interpolation problem then becomes:

$$\begin{array}{lll} f(\mathbf{x}_i) = 0 & i = 1, \ldots, N & \text{on-surface points} \\ f(\mathbf{x}_i) = d_i \neq 0 & i = (N+1), \ldots, M & \text{off-surface points} \end{array} \tag{3.20}$$

Usually, off-surface points are generated by projecting along surface normals[148] and d_i are chosen to be the distance to the closest on-surface point.

3.3.5 Partition of unity (PU)

The theory of partitions of unity is an important tool that allows one to pass from local to global. A PU is a decomposition of the constant function equal to 1 into a sum of continuous functions. More precisely it is defined as :

Definition 4: Partition of unity
Let $\mathcal{X}$ be a topological space, $\mathcal{U} = \{\mathcal{U}_i : i \in [1, \ldots, N]\}$ an open cover of $\mathcal{X}$. A partition of unity subordinated to $\mathcal{U}$ is a locally finite family of functions $\eta_i : \mathcal{X} \to [0, 1]$ satisfying:
$\sum_i \eta_i = 1, supp(\eta_i) \subset \mathcal{U}_i$, *where* $supp(\eta_i)$, *the support of* η_i *is defined as the closure of* $\{\eta_i \neq 0\} \subset \mathcal{X}$

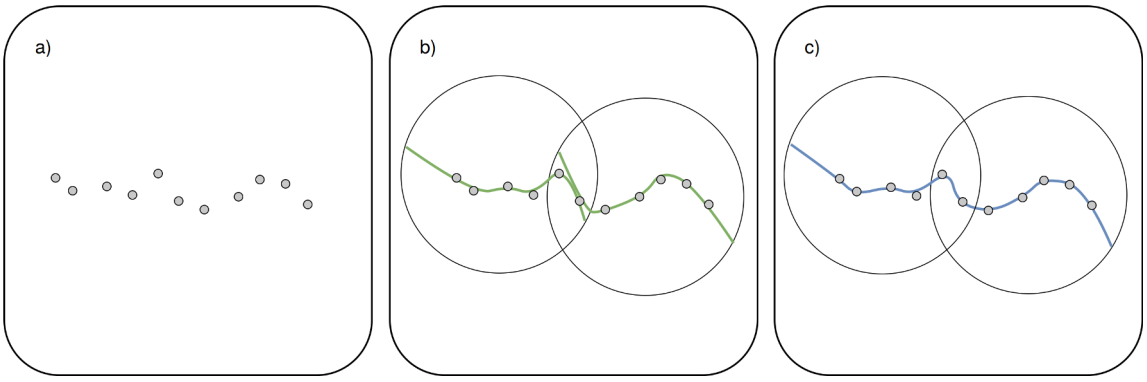


Figure 3.15: Illustration of PU principle. a) Point data set and global topological space are first divided in sub domain according to the weight function η_i. On each sub-domains an approximation or an interpolation is computed (b). The global solution is finally obtained by merging all local solutions (c).

As illustrated in Figure 3.15, for each sub-domain $\mathcal{U}_i$, a local approximation g_i is assessed on the subset of points $\mathcal{P}_i = \{\mathbf{p} \in \mathcal{U}_i\}$. Then a global function f can be computed as a combination of the local functions weighted by η_i :

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} \eta_i(\mathbf{x}) \cdot g_i(\mathbf{x}) \tag{3.21}$$

Note that the partition of unity process respects an interpolation property, *i.e.* if all g_i are interpolants at $\mathcal{P}_i = \{\mathbf{p} \in \mathcal{U}_i\}$ then f is an interpolant at the entire point set $\mathcal{P}$.

Based on this principle and using partition of unity defined through CSRBF, Ohtake[115] proposed the multi-level partition of unity (MPU), an innovative approach dividing the point cloud according to its complexity, then adjusting piece-wise quadratic functions to capture the local shape and further integrating those locally defined functions into a global approximation with CSRBF partition of unity. At a certain scale, a local shape fit is considered as adequate if its residual error is sufficiently small, otherwise the space is locally refined and the overall procedure is repeated. As proposed by Wendland[153], he uses an octree division of space as a support for a CSRBF basis used as partition of unity. For a N leaves octree, the approximating implicit function f is given by:

$$f(\mathbf{x}) = \sum_{i=1}^{N} g_i(\mathbf{x}) \cdot \varphi_i(\mathbf{x}) \tag{3.22}$$

where φ_i is the partition of unity function related to the domain $\Omega_i \in \Omega$, and g_i is a quadratic function fitting the data samples in the domain Ω_i. For each φ_i the radius of the domain Ω_i depends on the local octree subdivision.

By encoding part of the local structural information in quadric functions, this method tends to limit the number of functions in the basis and hence to reduce the computational cost while producing sharp and detailed models. Moreover, MPU is robust to non uniform samplings[115] as it does not require an estimate of the sampling density: a shape fits only if it is below a certain residual error. The level of smoothness and hence the robustness can be easily adjusted by a threshold on this residual error. Holes in these data can be filled by the extrapolation and subsequent merging of the shapes in the neighborhood. In order to handle errors arising from the extrapolation on big holes, Nagai[112] extended the approach by the direct smoothing of PU implicit surfaces.

3.3.6 Poisson surface reconstruction

The so-called Poisson surface reconstruction methods are based on a different approach. A closed surface is actually modeled through its indicator function χ (which discriminates the interior from the exterior of a solid shape: $\chi = 1$ in the interior of the shape, 0 elsewhere). Actually the gradient of χ is null away from the surface whereas it coincides with the inwards normals close to its border (see Figure 3.16). Based on this idea, Kazhdan[86] introduced a surface reconstruction method based on the resolution of a Poisson equation. This method considers a set of points $\mathbf{s} \in \mathcal{S}$, each consisting of a point sample $\mathbf{p}_s$ and a normal $\mathbf{n}_s$ describing a solid $\mathcal{M}$ with surface boundary $\partial\mathcal{M}$. The objective of the method is to find $\chi_\mathcal{M}$, a piecewise constant function, indicator function of $\mathcal{M}$.

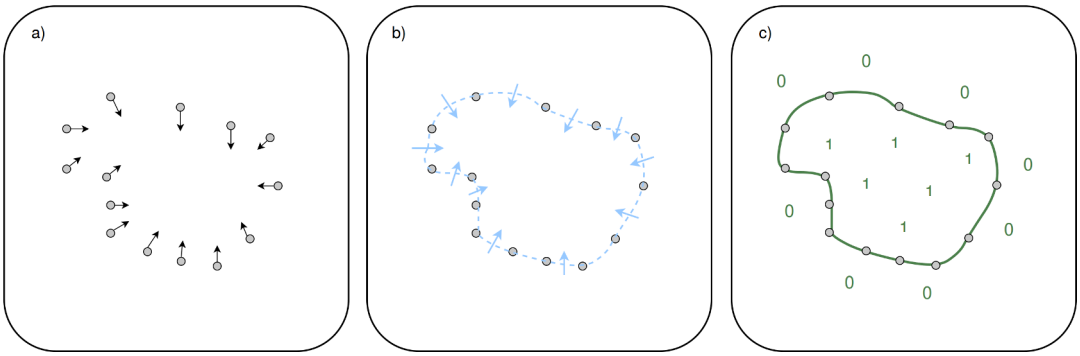


Figure 3.16: Illustration of Poisson reconstruction in 2D. a) Considering oriented input points $\mathbf{s} \in \mathcal{S}$, b) we evaluate a vector that approximates the gradient $\nabla\chi_\mathcal{M}$. c) We recover the indicator function $\chi_\mathcal{M}$ after solving the Poisson equation.

As the indicator function is a piece-wise constant function, its gradient field would diverge at the surface boundary. To avoid this, the indicator function is convoluted with a smoothing filter $\mathcal{F}$. The gradient of the smoothed indicator at $\mathbf{q} \in \mathcal{P}$ is equal to the vector field obtained by smoothing the surface normal field.

$$\nabla(\chi * \mathcal{F})(\mathbf{q}) = \int_{\partial\mathcal{M}} \mathcal{F}(\mathbf{q} - \mathbf{p}) \cdot \vec{N}(\mathbf{p}) \cdot dp \tag{3.23}$$

where $\vec{N}(\mathbf{p})$ is the normal at $\mathbf{p} \in \partial\mathcal{M}$.

Considering that a sample point $\mathbf{s} \in S$ describes an elementary patch of surface $\mathcal{P}_\mathcal{S} \subset \partial\mathcal{M}$ of surface $\mathcal{A}_{\mathcal{P}_\mathcal{S}}$, Equation 3.23 is approximated as:

$$\nabla(\chi * \mathcal{F})(\mathbf{q}) \approx \sum_{s \in \mathcal{S}} \mathcal{A}_{\mathcal{P}_\mathcal{S}} \cdot \mathcal{F}(\mathbf{q} - \mathbf{p}_s) \cdot \mathbf{n}_s = \vec{\mathcal{V}}(\mathbf{q}) \tag{3.24}$$

where $\vec{\mathcal{V}}$ is the vector field induced by the samples normal. The variational problem is then turned into a standard Poisson problem by applying the divergence operator to Equation 3.24:

$$\Delta\chi = \nabla.\vec{\mathcal{V}} \tag{3.25}$$

This equation is then solved on a finite basis of functions in the space of continuous functions. In order to build this basis and hence discretize the problem, the space is divided as an octree according to the position of the sample points. A function F_o is centered in each node of the octree, and stretched by the size of the node. After expressing χ in the space of function as $\chi = \sum_o x_o \cdot F_o$ and projecting Equation 3.25 in $Span\{F_o\}$, $\{x_o\}$ is computed by a least square minimization:

$$\sum_{o'} \| \sum_{o} x_o \cdot < \Delta F_o, F_{o'} > - < \nabla.\vec{\mathcal{V}}, F_{o'} > \|^2 \tag{3.26}$$

The reconstructed surface is the iso-surface of $\chi_{\mathcal{M}}$, whose iso-value is the average of the evaluations of χ at the sample points.

For surface reconstruction, such a gradient-domain formulation results in robustness to non-uniform sampling, noise, outliers, and to a certain extent missing data. However, fitting directly the gradient of χ leads often to over-smoothing. Kazhdan[87] proposes to use the position of the points as constraints in the optimization to overcome this issue. Calakli[30] extends the approach to improve the surface extrapolation in the regions of missing data by adding a Hessian constraint.

4 Summary

Whereas discrete mesh models and parametric surfaces prove to be inappropriate for the processing of such huge, in-homogeneous and noisy datasets produced by TLS, modeling through implicit functions arises as a judicious and efficient way to handle the problem. Among the literature on implicit surface reconstruction techniques, the methods based on CSRBF, in particular the ones following a multiscale approach, are empowered to tackle characteristics and imperfections of TLS point cloud. Poisson surface reconstruction is also a robust and efficient method to handle the noise and the non-uniformity of forest TLS scans even if it is designed for surface reconstruction from well segmented data-sets (the presence of numerous outliers and large holes may compromise the outcome).

4

Reconstruction of open surface

This chapter focuses on the reconstruction of an open surface based on 3D point samples. An open surface is defined as a surface with boundary, topologically equivalent to the disk surface: a non-closed surface without hole. While dealing with open surface reconstruction based on 3D samples, the boundaries are fixed at the edges of the space containing the points, *i.e.* the limit of the data points bounding box. By applying this concept to the reconstruction of digital terrain model (DTM) from TLS data, the objective is to recover a curved surface model describing the terrain model with boundaries attached to the limits of the area scanned. This chapter introduces our specific method adapted to the characteristics of TLS point cloud.

1 Introduction

Digital terrain model (DTM) extraction is an open issue in the field of LiDAR remote sensing of the Earth surface. Many filtering algorithms have been proposed in the last two decades to solve this problem using Airborne LiDAR sensors (ALS) data[139]. They operate either on the raw data point cloud, on interpolated surfaces, or on a combination of both[106]. They can be classified into two groups: virtual deforestation algorithms, progressively filtering out above ground objects[150] and densification algorithms, progressively enriching the initial surface according to geometrical criteria (ie. distance and angle) such as the triangulated irregular network iterative approach pro-

posed by[13]. Until now, the later approach is considered as one of the best methods for ground filtering and terrain modeling from ALS data[139],[106].

The development of terrestrial Laser scanning (TLS) technologies was initiated while ALS processing algorithms were already available, and those algorithms were largely applied to TLS data. However ALS and TLS point clouds differ in many ways, making it interesting to develop dedicated processing chains for TLS-based DTM modeling[2]. Compared to ALS point clouds, TLS ones provide very dense sampling rates at the ground level, describing the micro-topography around the sensor. However, the presence of vegetation and the terrain topography itself generate strong occlusions causing large data gaps at the ground level, and a risk of integrating objects above the ground within the DTM. Additionally, the scanning intensity of TLS devices depends by nature on the distance to the sensor, resulting in spatial variations in point density. These locally extreme differences in point density at ground level and the potential noise created by low vegetation are difficult to handle using algorithms designed for ALS as far as a fine description of the ground surface is required.

Due to these sampling properties, one might consider that DTM extraction from TLS data related more to a surface reconstruction problem than a classification issue. To the best of our knowledge, reconstruction approaches have never been considered in this context. Based on this statement, we propose an innovative approach designed to reconstruct detailed DTMs from TLS data under forest canopies. Our approach is based on techniques developed for surface reconstruction from scattered data and relied on an original multi-scale data interpolation framework and a surface approximation with implicit surfaces.

1.1 Context of digital terrain model extraction from aerial LiDAR data

Since the emergence of aerial LiDAR scanning (ALS) in the 90's, several research teams have explored the field of terrain model extraction based on 3D point clouds. Sithole and Vosselman[139] provided a review of ground filtering challenges along with a benchmarking of state of the art algorithms in the early 2000. Most of those algorithms operated on a local neighborhood and used geometrical rules to discriminate points belonging to the ground surface. Among others, Elmqvist[57] float a deformable surface upwards from beneath the point cloud by minimization of the model energy function. Through an iterative process, Axelsson[13] increases the density of a triangular mesh by including point meeting certain angle and distance criteria. Véga[150] introduces a discriminant function based on neighboring points slope and height difference. In order to segment the point cloud and so retrieve the ground points every approach delineate a portion

of the 3D space where bare-Earth points are expected to reside. The input points falling in this portion of space are tagged as ground points. Sithole[139] points out that the performance of such algorithm is related to the use of context: algorithm strategies that use more context are better able to separate ground from other points.

1.2 Comparison of aerial and terrestrial LiDAR data

The context of TLS data in forest areas is extremely different (see Figure 4.1). The point of view and the distance to the sensor produce unlike characteristics in the recorded point cloud.

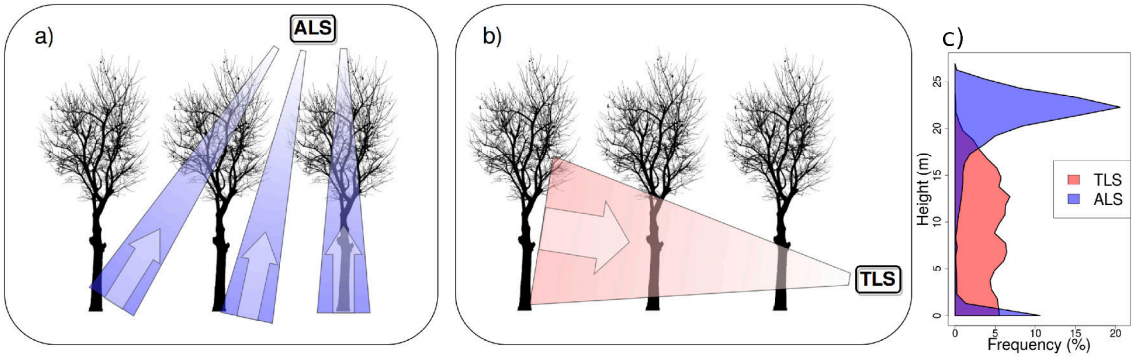


Figure 4.1: Illustration of laser pulse interception for (a) ALS and (b) TLS. (c), adapted by Chasmer[37], represents the average laser pulse frequency (percentile) distributions for ALS (blue) and TLS (red) obtained for a survey of a mixed deciduous forest.

For instance, Chasmer[37] analyzed for the two systems, ALS and TLS, the frequency of laser returns within defined slice of space divided along the *Oz* axis. The results, shown on the Figure 4.1 highlight a continuous distribution of the points for the TLS, decreasing with the altitude. Whereas for the ALS data, the distribution presents two clear modes: one corresponding to the signal interception by the canopy, the other one being the reflection of the ground. While occlusion is a problem affecting both TLS and ALS data, the size of the occlusion compared to point cloud size is much higher for TLS: TLS dedicated methods cannot rely only on points neighborhood to extrapolate and fill the holes. Even if the context of TLS data differ strongly from ALS, to our knowledge there has not been any method developed exclusively for the processing of TLS data. Our motivation was to fill this lack by developing a specific method which adapts the level of detail according to the distance to the sensor and to the occlusions, and that builds upon surface reconstruction with CSRBF to produce accurate digital terrain model.

As described in previous section, our goal is to reconstruct digital terrain model surfaces from dense 3D point clouds; actually such data is simultaneously inhomogeneous, noisy, locally highly

occluded or overdescribed. Hence, we aim at simultaneously segmenting points (ground / non ground) and reconstructing a terrain model. Therefore, chosing an appropriate model is a key issue. Taking the holes and the noise into account, mesh models appear unsuitable. As the point clouds contain both vegetation and ground points, filtering and reconstruction must be implemented simultaneously, what also rejects parametric models. Thus implicit models arise as a natural solution with the required properties of continuity and curvature controls. Among them, the radial basis functions, as described in Chapter 3, have strong approximation properties. Hence, we have chosen this model in our work.

2 Related work

The pioneering works of Turk and O'Brien[147] demonstrated that implicit surfaces such as RBF were efficient to repair incomplete or noisy data sets. And several approaches were built on the variational nature of global RBF to obtain interpolation or approximation results[33]. However, such approaches entail inverting large dense matrices as the approximation is not local. Starting from the work of Morse *et al.*[111], the class of compactly supported RBF (CSRBF) has been largely studied for data approximation: it provides a local control over the reconstruction and entails only the inversion of extremely sparse matrices. Another way to repair incomplete and noisy data sets is to use a support vector machine approach[121]. However such methods are not local enough to be used as a filtering method.

Because TLS data might show extreme variations in density due to occlusions, we hypothesize that CSRBF models are appropriate in order to both fill holes and to produce a continuous surface as an approximation of the ground. Our idea originates in Ohtake's work on surface reconstruction from unstructured point clouds using so-called functional RBF (see[115] and previous work). This approach differs from other RBF approaches in the sense that it combines local parametric surface patches and compactly supported Wendland's RBF functions[152]. The latter are actually used as partitions of unity in order to merge local patches.

Based on this background work, we propose an innovative method to overcome the flaws of TLS data. We manage the high point density and sudden shifts in this density by using a quadtree division of space to adapt the scale locally. In order to handle the holes created by occlusion we chose to model the surface by the combination of implicit patches merged with compactly supported radial basis functions.

3 Algorithm

3.1 Overview of the approach

The workflow of the proposed method is presented in Fig. 5.1. It relies on 3 main steps: 1) a quadtree construction allowing a dynamic multi-scale local approximation of the ground surface, 2) a refinement step dedicated to identify and correct approximation errors in the quadtree cells and 3) the computation of the implicit function and its polygonization.

The quadtree subdivision of space is used to provide the supporting neighborhood for CSRBF and parametric patches. The quadtree is iteratively built according to the quality of the local reconstruction with respect to the data. Practically, the quadtree structure is driven by the quality of the parametric patches thus providing an optimal scale structure for digital terrain modeling using CSRBF.

Hence, we first compute an adapted quadtree subdivision of space. Simultaneously, for each cell, we compute by least square fitting an approximating local quadratic elevation patch[2] . Then, the splitting process is guided by the quality of the fit: if the data contained in a cell cannot be properly approximated by such a quadratic patch, then the node is subdivided (unless the maximum level is reached). At the end of this process, the fitting error can remain high in some cells: Such a fitting problem may be associated with the presence of non-ground points in the quatree cell. Our next steps filter such points. Indeed, we added 2 sequential filters to address this issue. The first filter identifies and removes non-ground points based on a statistical analysis of the vertical distribution of points in a given quadtree cell. The second filter compared the local quadratic patches to its neighbors to detect and replace any local inconsistencies by computing a new surface approximation using the neighboring patches. Following these steps, a global model is computed by merging the local implicit patches into an implicit function using CSRBF[2] . Finally, the DTM is computed through a polygonization of the zero level set of the global implicit function.

3.2 Input points

A driving hypothesis of LiDAR point classification algorithms is that ground points are local elevation minima. Unfortunately, in TLS point clouds, such local minima may often correspond to non-ground points as illustrated on Figure 4.3. This is either due to occlusions generated by surrounding objects (*ie.* vegetation and trees in forest environment), or to fluctuations of the ground itself, owed to the oblique view of TLS. In addition, due to the relation between point

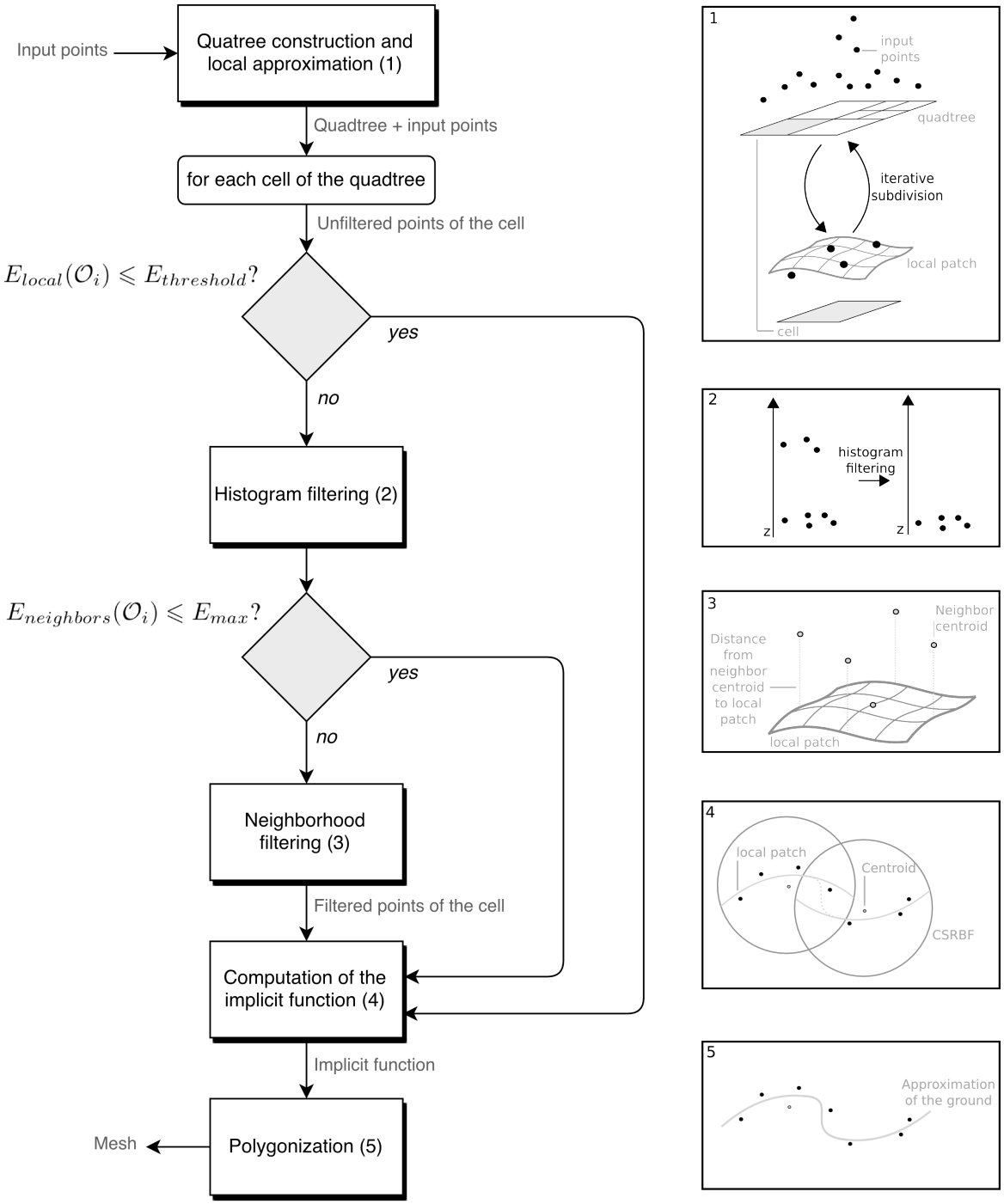


Figure 4.2: Overview of the method : (1) iterative build of the quadtree, (2) filtering of points using histogram, (3) filtering of points with the distance of neighbor centroids to the local patch, (4) merge of local patches using CSRBF and (5) polygonization of the surface.

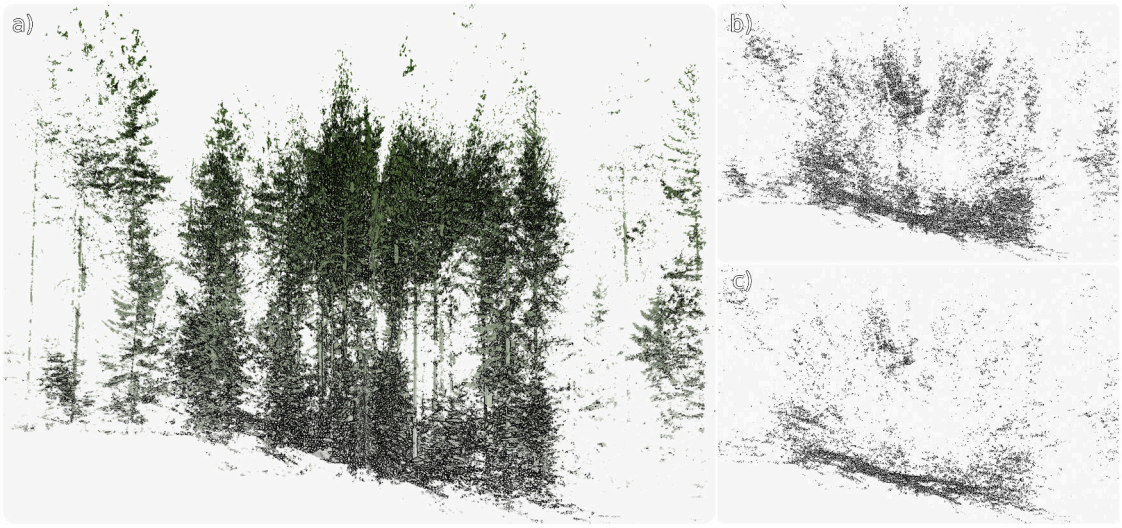


Figure 4.3: Effect of the resolution of the 2D grid on the selection of the local minima: a) raw point cloud (26 millions points), b) local minima with a grid of 5cm (120k points), c) local minima with a grid of 10cm (38k points)

density and distance to the sensor, fixing an appropriate scale to identify those local minima is challenging. Besides these limitations, the use of local minima remains interesting, especially because it drastically reduces the amount of data. Under this hypothesis, the first approximation of the ground surface consists in projecting the raw point cloud into a fine regular $2D$ (x, y) grid and further selecting the points of minimum elevation in each cell. The resulting set of points denoted by $P = \{p_i\}_{i=1\ldots N} \subset \mathbb{R}^3$ serves as input data for the proposed algorithm. Generally speaking, the size of the $2D$ grid should be selected according to the expected DTM accuracy. The smaller the grid size, the higher the point density and the finer the DTM.

3.3 Local approximation of data contained in a cell

Given $\{\mathcal{O}_1, \ldots, \mathcal{O}_N\} \subseteq \mathcal{P}(P)$ a partition of P (in our case, a quadtree subdivision of space), for each subset $\mathcal{O}_i$ (or cell), we denote by c_i the centroid of $\mathcal{O}_i$ (we will call it *center* of $\mathcal{O}_i$).

In each cell $\mathcal{O}_i$, an approximate tangent plane is computed using least square fitting. Let (u, v, w) denotes a local orthonormal coordinate system such that the direction w is orthogonal to the fitting plane. We approximate $\mathcal{O}_i$ by a local implicit quadric function

$$g_i(u, v, w) = w - h_i(u, v) \tag{4.1}$$

where, in the local coordinate system, h_i is a quadratic parametric surface of the following form:

$$h(u, v) = Au^2 + Buv + Cv^2 + Du + Ev + F \tag{4.2}$$

Coefficients A, B, C, D, E, F are determined by the least square minimization of:

$$\mathcal{E}_i = \sum_{p_j=(u_j,v_j,w_j)\in\mathcal{O}_i} (w_j - h(u_j, v_j))^2 \tag{4.3}$$

where $\{p_j\}$ is the set of points contained in the cell $\mathcal{O}_i$. Once g_i is defined, the weighted sum of squared residuals quantifies the quality of the fit. We denote it by $E_{local}(\mathcal{O}_i)$ and use it as a metric to guide and adjust the quadtree division of the xy plane (see section 3.4).

In a further step, we use compactly supported Wendland's RBF functions[?] as a partition of unity to merge these local implicit functions (issued from parametric patches) and compute a global implicit model. More precisely, we use $\phi(r) = (1 - r)_+^4(1 + 4r)$, where $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise, which is $\mathcal{C}^2$ and radial on $\mathbb{R}^3$. Hence in each cell $\mathcal{O}_i$, we use, as a partition of unity, the function $\Phi_i(\|x - c_i\|) = \phi(\frac{\|x-c_i\|}{\sigma^i})$, where the parameter σ^i controls the size of the zone of influence centered around c_i. Computation of the global implicit model is presented in section 2.4.

3.4 Quadtree division

As mentioned in section 2, the quadtree subdivision process is guided by the quality of the previous local approximation. Because of its computational efficiency, we use the method introduced by[?] to generate our quadtree model. The quadtree is stored in a tree where each node has four sons. Initially, the bounding box of data is defines the root of the quadtree. Then, we iteratively subdivide each cell into four quadrants according to the residual $E_{local}(\mathcal{O}_i)$ of the least square approximation on the points (see section 2). We subdivide the node containing $\mathcal{O}_i$ when $E_{local}(\mathcal{O}_i) \geqslant E_{threshold}$, where $E_{threshold}$ is a user defined threshold. The process stops when either all leaf nodes are found or when quadrants are smaller than the minimal size allowed ($Size_{min}$). The parameter $E_{threshold}$ drives the quality of the final surface result. Indeed a low value forces the division of the quadtree to get better local approximating patches. The parameter $Size_{min}$ is fixed by taking into account the resolution of the $2D$ grid used to filter local minima (see 2.2). At least, a minimum of six points per quadrant is needed to compute the local approximation g_i in equation 5.2, since the local approximating patch has six coefficients.

3.5 Correction of approximation errors

Once the quadtree division process is over, the algorithm iterates through leaf cells to correct its approximating surface patch if needed. For each cell $\mathcal{O}_i$, one of the three following scenario happens: (1) The cell contains only ground points, so the local approximation is maintained. (2) The cell contains both ground and non-ground points. In that case, the non-ground points are filtered out and the local patch is approximated again with remaining ground points only. (3) The cell contains only vegetation points. The whole set of points is then filtered out and the local approximation is computed using data from the neighboring cells.

In practice, $E_{local}(\mathcal{O}_i)$ is computed for each leaf node, and quantifies the local quality of the reconstruction. As mentioned earlier, the quadtree subdivision process may eventually produce cell of high $E_{local}(\mathcal{O}_i)$. This is the case when the maximum subdivision level is reached but the cell contains both ground and non-ground points. Therefore, based on this value, we define two different filters, which actually come to improve segmentation of ground, non-ground and noise points.

3.5.1 Histogram filtering

The first filter relies on the commonly accepted hypothesis that the ground is a continuous surface. Hence, the projection over the Oz axis of the surface restricted to a cell, should be a segment of $\mathbb{R}$. Thus, the vertical distribution of points falling within a quadtree cell is expected to be unimodal. In practice, let us consider a cell that cannot be divided anymore because its size is the minimum size allowed $Size_{min}$. If the vertical distribution of points is not unimodal, then $E_{local} > E_{threshold}$. Only the points of the lowest cluster (that is of the first mode) describe the ground topography. Therefore we filter out the highest cluster and keep only the information of the first mode (that is the lower cluster). To do so, we fill an histogram with the distribution of the elevation of all the points of the cell (see Fig. 4.4), smooth that histogram with a median-filter, and finally detect the last bin of the first peak, which provides a limit of elevation for the lowest cluster. Then we filter out the points above that limit, and compute a new local approximation. The size of the histogram bin $Size_{bin}$ and the width of the median filter $Width_{filter}$ are input parameters of the algorithm. They need to be tuned by the user depending on the structure of the vegetation and on the slope of the scanned area.

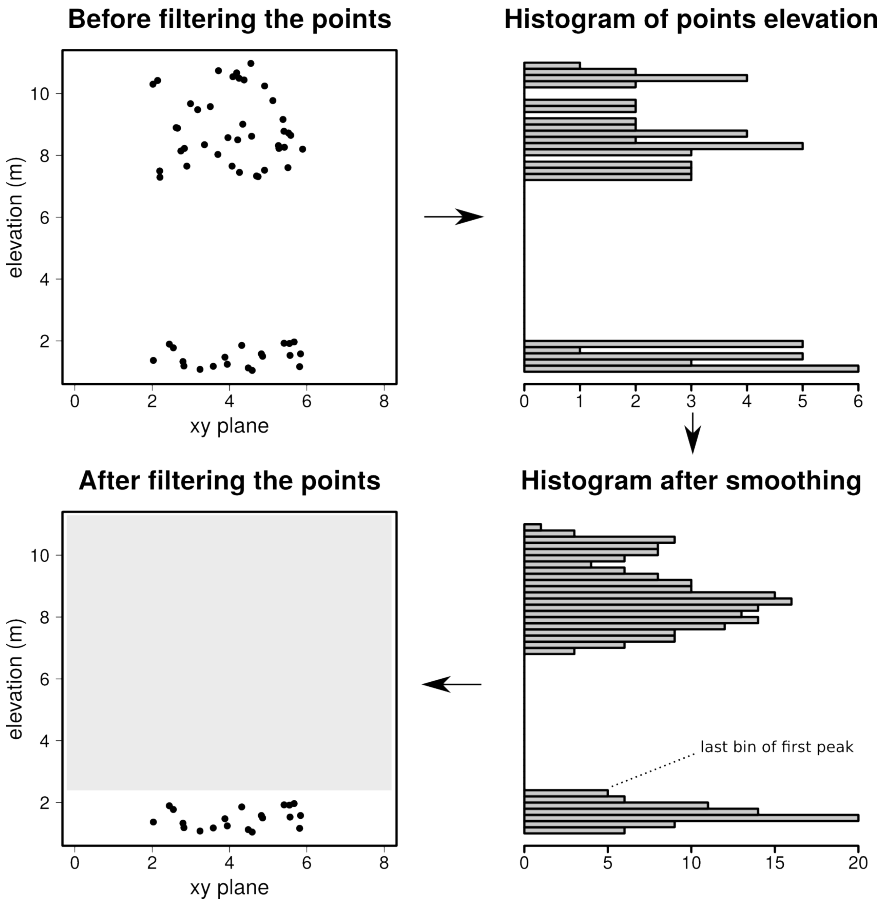


Figure 4.4: Histogram filtering method

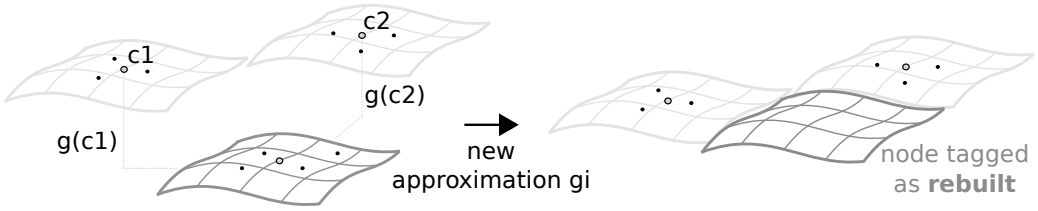


Figure 4.5: Neighborhood filtering method

3.5.2 Neighborhood filtering

Another filtering criterion is the disruptive change in local parametric patches with respect to the neighboring cells of the quadtree. For each leaf node, we estimate $E_{neighbors}(\mathcal{O}_i)$ as the averaged sum of squared values of the local quadric g_i at each centers of its neighbors. More precisely, let $C_{neighbors} = \{c_1, ..., c_M\}$ denote the centers of the neighbors of $\mathcal{O}_i$:

$$E_{neighbors}(\mathcal{O}_i) = \sqrt{\frac{1}{M} \sum_{j=1...M} g_i(c_j)^2} \tag{4.4}$$

For a given leaf node l_i, if $E_{neighbors}(\mathcal{O}_i)$ is higher than a threshold E_{max} defined by the user, we delete the set of points of that node and consider the set of all points of the neighboring nodes to compute a new local approximation g_i by least square minimization such as eq. (5.2). Such a node is then tagged as *rebuilt* (see Fig. 3).

Let us denote by $P_{filtered}$ the set of points that remains after the previous filtering steps (3.5.1) and (3.5.2). All further computation only rely of those points.

3.6 Computation of the global implicit model

Based on the previous steps, we can now compute a global implicit model by sewing together local patches by means of Wendland's CSRBF used as partitions of unity. The global implicit model $f : \mathbb{R}^3 \mapsto \mathbb{R}$ is computed as:

$$f(x) = \sum_{c_i \in C} g_i(x) \cdot \Phi \quad _i(\|x - c_i\|) \tag{4.5}$$

where Φ is the CSRBF defined in 2.3 and g_i is a local implicit approximation, similar to equation 5.1 in 2.3) and computed according to the leaf node l_i (see Algorithm **??**) and to σ^i (as defined below).

We first compute σ^i. Let us point out that f has support inside $\cup_{i=1\ldots N} \mathcal{B}(c_i, \sigma^i)$, Therefore, we define and compute σ^i such that this union of spheres is a covering of the bounding box of the input point cloud. More precisely, given a cell $\mathcal{O}_i$ of center c_i and $C^i_{neighbors} = \{c^i_1, ..., c^i_M\}$ the centers of the neighboring nodes, we compute the radius σ^i as:

$$\sigma^i = \max_{j=1\ldots M} \|c_i - c^i_j\| \tag{4.6}$$

The global implicit model (equation 5.3) is then obtained by computing, for each leaf node l_i of center c_i: either a refined local approximation (over the points included in $\mathcal{B}(c_i, \sigma^i)$) if l_i is not tagged *rebuilt*, or the approximation g_i computed in the neighborhood filtering when l_i has been tagged *rebuilt*.

Algorithm 1 Computation of the final implicit model

let N be the number of leaf nodes of the quadtree
$P_{filtered}$ is the set of points remaining after filtering
$L = \{(l_i, \sigma^i)\}, i \in \{1, N\}$
$f = 0$
for (each $l_i \in L$) **do**
  if (l_i is not tagged as *rebuilt*) **then**
    compute $\mathcal{P}_i = \{p \in P_{filtered}\ ;\ \|c_i - p\| \leqslant \sigma^i\}$
    compute local approximation g_i of the set $\mathcal{P}_i$
  else
    set g_i as computed in the neighborhood filter
  end if
  $f = f + g_i(x) \cdot \Phi_{\ i}(\|x - c_i\|)$
end for

3.7 Extraction of the DTM as a mesh

Following from our modeling choices (namely Wendland's RBF and quadrics), the implicit function f is C^2. Hence it is possible to use a polygonization algorithm to represent the zero-level set of the implicit function f. In the space defined by the union of the spheres of radius σ^i, f is discretized into a scalar field. From this scalar field, the polygonizer computes a set of triangles approximating the implicit surface f, by using a marching cube like algorithm.

Table 4.1: Processing time scaling with the number of local minima

2D grid	Local minima	Leafs in quadtree	Time
20 cm	10k points	295	5s
10 cm	38k points	1057	28s
5 cm	149k points	3226	4m53s

3.8 Complexity

The study of the complexity shows that the algorithm is of order $\mathcal{O}(N)$ (where N is the initial number of minimal points, see section 2.2). Indeed, the computation of the quadtree is of order $\mathcal{O}(N \times N_{cells})$, where the constant $N_{cells} = 4^{Level_{max}}$ is the number of cells in the worst case. Histogram filtering as well as the computation of g_i are linear, that is of order $\mathcal{O}(N)$; the neighborhood filtering and the computation of the set of parameters σ^i are constant, of order $\mathcal{O}(N_{cells})$. The computations were done on an Intel Xeon E3-1200 (3.1 GHz) with 8Gb RAM. Table 4.1 shows examples of processing times according to the grid size and corresponding point density.

4 Experimental

The algorithm has been developed as a plugin for Computree platform (http://computree.onf.fr), dedicated to the processing of TLS data in forest environment. It uses the point cloud library (PCL) (pointclouds.org) and the GNU scientific library (GSL) (gnu.org/software/gsl).

Our method has been evaluated using two distinct approaches:
- Simulated data were used to assess the performance of the method on simple forest scene of known structure using the Hausdorff distance criteria.
- Real TLS scan acquired over 4 forest plots were used to test the robustness of the method over complex terrain and forest structures. Due to the absence of field truth, scans of different point of view were compared using the same criteria.

4.1 Simulated point cloud

Four TLS like point clouds were generated from meshed landscapes: we intended to assess the response of our approach to various terrain configurations (ridge, bumps and slope) and occlusion rates. To do so, we designed four terrain configurations (see Fig. 4.6): a ridge (plot s1), a flat surface with bumps (plot s2), a 10° slope scene (plot s3) and a 30° slope scene with similar bumps (plot s4). Each terrain was planted with five virtual trees to simulate occlusion by

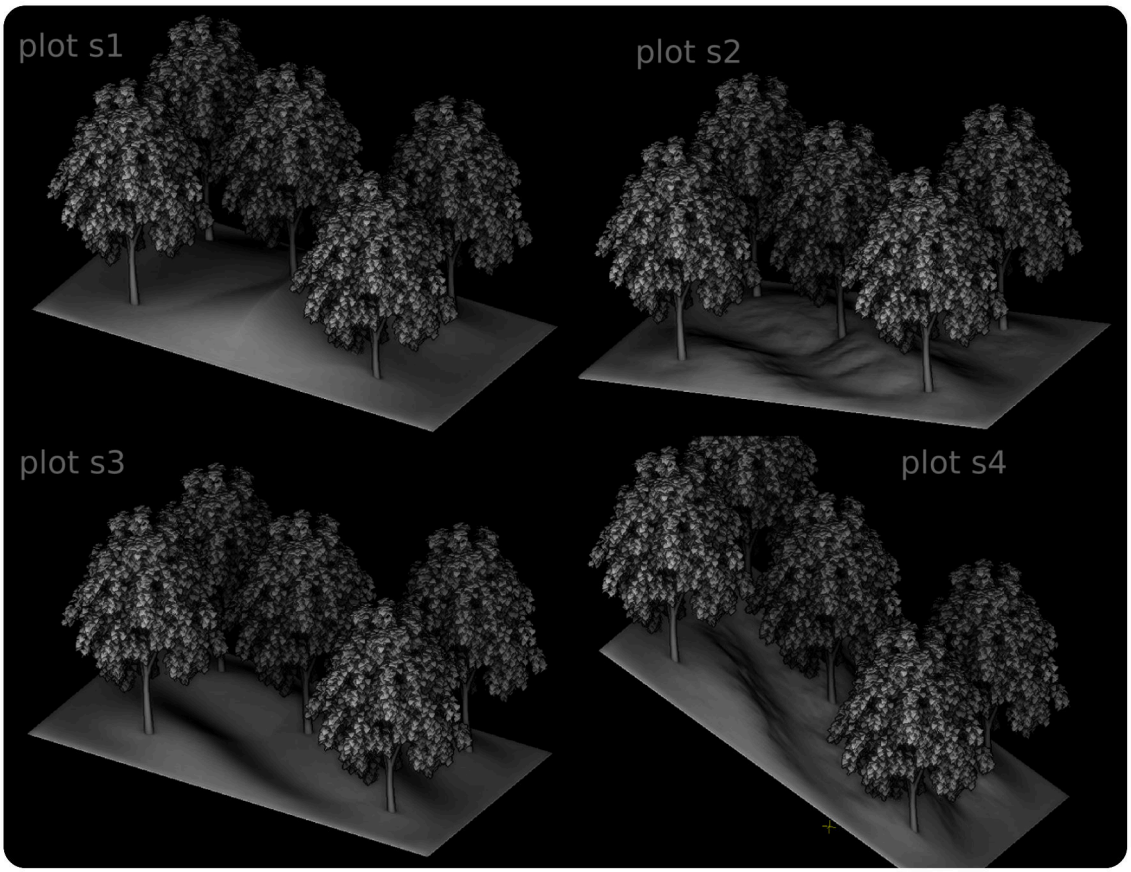


Figure 4.6: Landscape models designed on Blender for simulation purpose : a) Plot s1 - Ridge b) Plot s2 - Bumps on flat ground c) Plot s3 - Bumps on 10° ground slope d) Plot s4 - Bumps on 30° ground slope

the vegetation. A TLS simulator was used to generate the points clouds. It is based on PBRT (http://www.pbrt.org/) and emulates the interactions using a ray tracing method. Starting from a 10 meters side length mesh model we generated using Blender (http://www.blender.org/) and a fixed position of the scanner facing the slope, we produce TLS like point clouds of the scenes for feeding the algorithm. The vegetation structure, composed of five trees, was kept identical within all the generated landscape to reduce its effect in the statistical analysis.

Quality assessment was done by comparing the mesh models of the terrain surfaces with the outputs of the algorithm using the Hausdorff distance implementation available in Meshlab (http://meshlab.sourceforge.net/). The Hausdorff distance evaluates the distance between two surfaces represented as triangular meshes by adopting a surface sampling approach. For each scene, we generated a map of occluded areas in order to analyze the quality of the reconstruction both inside and outside occlusions.

Table 4.2: real data plots features

	vegetation	topography	slope
plot r1	high trees	simple	small
plot r2	small trees	simple	small
plot r3	small trees	complex	medium
plot r4	high trees	complex	strong

Hausdorff distance were also evaluated independently for visible and occluded areas to assess the quality of the reconstruction. For each plot, the occlusion mask is generated by projecting the set of points on the xy plane and further discretizing the plane to obtain a 2D binary grid with a value of 1 if the pixel contain data and a value of 0 otherwise. The overall occlusion rate is computed from the maks as the number of pixel containing data to the total number of pixels.

4.2 Real data

The method was further tested using real data collected within a hardwood-dominated forest (Phalsbourg, France). The ground topography is complex, with important slope variability and the presence of abrupt elevation changes locally. The ground supports a large diversity of forest structures, from low density stands with dense understory vegetation principally made of ferns, to very dense and multi-layered stands of different age structure. The dataset consisted in four, 20x20 m squared plots, scanned using an Optech ILRIS 3D from 3 different point of views. The plots (Table 4.2) were selected in order to maximize the variability of both the ground topography and the vegetation structure. The resulting point clouds show complex and dense occlusion near the ground, thus providing challenging data for ground reconstruction.

As no reference surfaces were available, the quality assessment was done by comparing the output of the algorithm for each couple of scans. To do that, we measure the likelihood of the results by computing the Hausdorff distances between the output surfaces. A map of intersection inter scan was also generated in order to analyze the quality of the reconstruction on the pooling area of both scans. The intersection map was computed as the intersection of the two corresponding occlusion masks. From this intersection map, we computed an overlap ratio as the surface of the intersection map divided by the surface of the scanned area.

In order to evaluate the quality of our method, our results were also compared with those obtained using a tin-iterative approach[13] has implemented within LAStools. As indicated above, the TIN-iterative approach is among the best methods for both ground point filtering and ter-

rain reconstruction and is well suited for assessing the performance of the proposed method. For the sake of this study, the tin-iterative algorithm was run using a 10 cm resolution for the selection of the local minima with the finer parameters in order to recover the smaller features on the ground. Fig. 4.12 presents the Hausdorff distance between Axelsson's DTM and ours and Fig. 4.11 illustrates recurring artifacts observed with Axelsson's approach.

4.3 Sensitivity analysis

The algorithm parameters were submitted to a sensibility algorithm to assess their influence on the quality of the reconstructed surface. Simulated data were well suited to optimize the quadtree structure, including their minimum size $Size_{min}$ and the error associated to the quadratic patches $E_{threshold}$ (see 3.4). Because the simulation environment was not complex enough to assess to sensibility of the filtering parameters (ie. $Size_{bin}$, $Width_{median}$ and E_{max}), the sensibility analysis of those parameters has been conducted on the real data only, and assessed by measuring the mean Hausdorff distance between individual scans has base line of comparison.

5 Results and discussion

5.1 Simulated point cloud

Fig. 4.7 highlights some of the key points of the algorithm using plot s1 as a reference: Fig. 4.7a and Fig. 4.7b show respectively the raw point cloud and the resulting quadtree structure. It can be seen that the complexity of the quadtree correspond well to the complexity of the scene, with respect to both the topographical variations and occlusion patterns. Fig. 4.7c illustrates the persistence of non ground points in the initial set of local minima, and the importance of filtering to identify the ground points. Finally, Fig. 4.7d shows the mesh resulting from the polygonization of the implicit function level-set, and its capabilities in managing large occlusions.

The simulations were run using the parameters values presented Table 4.3. These values were selected using a trial and error experiment and further refined through the sensitivity analysis. The resolution of the 2D grid used to select the local minima was fixed to 5 cm in order to provide a detailed topographical reconstruction. The sensitivity analysis was limited to the following parameters: the minimum size of the leaf of the quadtree ($Size_{min}$), the threshold on the error of approximation within a leaf of a quadtree ($E_{threshold}$) (see 3.4). The filtering parameters were not considered here, because the simulated scenes were not complex enough in term of vegetation

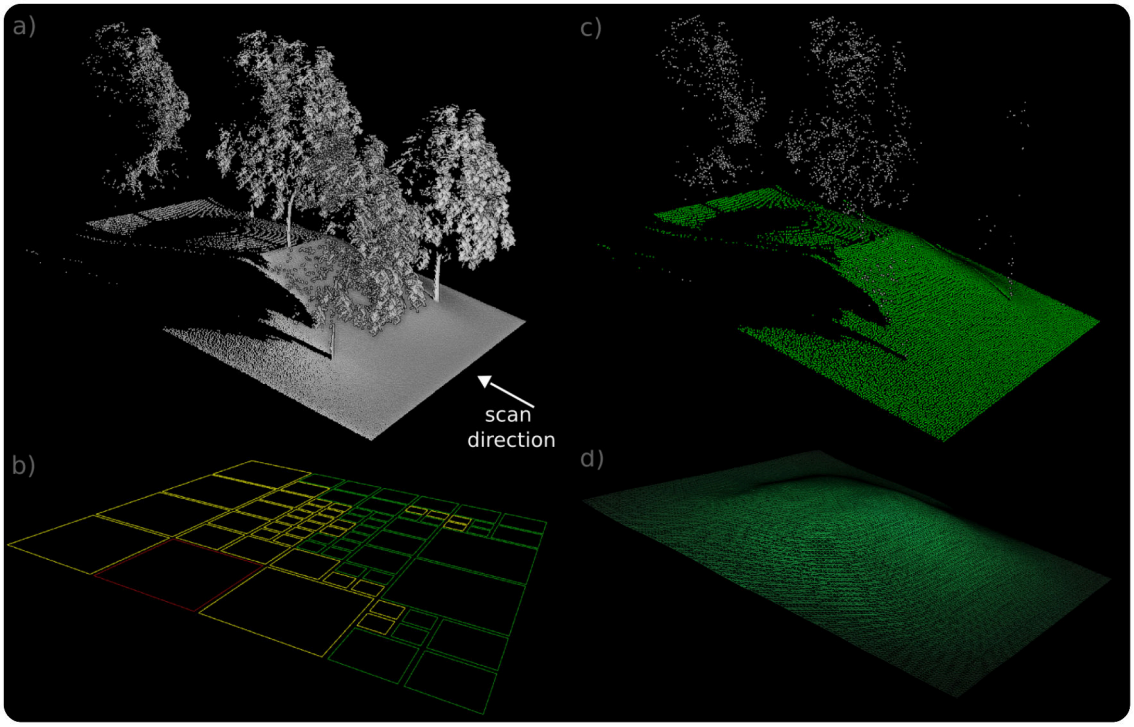


Figure 4.7: Simulation process: a) raw point cloud produced by the TLS simulator (with scan direction), b) the optimized quadtree structure, green cells are not filtered, yellow cells are filtered by the histogram method and red cells are filtered with the neighborhood method, c) the filtering of non remaining non-ground points, d) the resulting terrain model defined as the level-set of the continuous function.

Table 4.3: Statistical indices on Hausdorff distance (cm) for the simulated scenes, inside/outside the occlusions and on the full area (total)

scene	plot s1 (occlusion: 40%)			plot s2 (occlusion: 19%)		
($E_{threshold}$, $Size_{min}$)	0.1cm , 20cm			0.1cm , 20cm		
	inside	outside	total	inside	outside	total
mean	3.25	2.06	2.40	3.42	1.56	1.76
percentile 50	2.52	1.24	1.61	3.14	1.07	1.16
percentile 70	3.82	2.20	2.71	4.79	1.86	2.02
percentile 90	6.92	4.44	5.39	6.61	3.52	4.31
scene	plot s3 (occlusion: 27%)			plot s4 (occlusion: 42%)		
($E_{threshold}$, $Size_{min}$)	0.1cm , 20cm			0.1cm , 50cm		
	inside	outside	total	inside	outside	total
mean	4.67	1.48	2.21	13.01	2.94	7.18
percentile 50	3.33	1.08	1.30	9.34	2.07	3.51
percentile 70	5.68	1.77	2.22	16.93	3.58	6.74
percentile 90	10.77	3.19	5.12	31.38	6.39	20.23

to run such an analysis. Indeed the filter parameters were fixed using a simple trial and error experiment. Fig. 4.8a) and 4.8b) present the evolution of the mean Hausdorff distance between the reference mesh and the reconstructed one when those parameters vary. Overall, the mean distance increases when the quadtree gets coarser. On the other hand, setting a minimum size too low leads to incorrect reconstruction, especially when the landscape is complex and the occlusion rate high, for instance for the scene with a slope of 30°. The threshold on the weighted sum of the squared residuals drives directly the quality of the reconstruction, the lowest values allowing the sharpest reconstructions.

Table 4.3 shows the Hausdorff distance computed for each plot and the corresponding set of parameters. Overall, mean distances varied from 1.76 cm (plot s2, flat topography) to 7.18 cm (plot s4, 30° slope). The more the slope, the more the mean distance and extreme values. Strong topographical variations, like ridges (plot s1) also showed slightly higher distances compared to smoother terrain undulations (plot s2). This was expected, because fast elevation shift generate large data gaps (40% of occlusions) and are difficult to reconstruct. For all the plots, the Hausdorff distances increase in occluded areas. Mean values ranges from 1.48 cm (plot s2) to 2.94 cm (plot s4) in sampled areas, and from 3.25 cm (plot s1) to 13.01 cm (plot s4) in occluded ones. This corresponds to increase in errors ranging from a factor 1.57 (plot s1, 40% of occlusions) to a factor 4.4 (plot s4, 42% of occlusions), which is quite good considering the proportion of reconstructed surface. Interestingly, the extreme values are following the same trends, highlighting the robust-

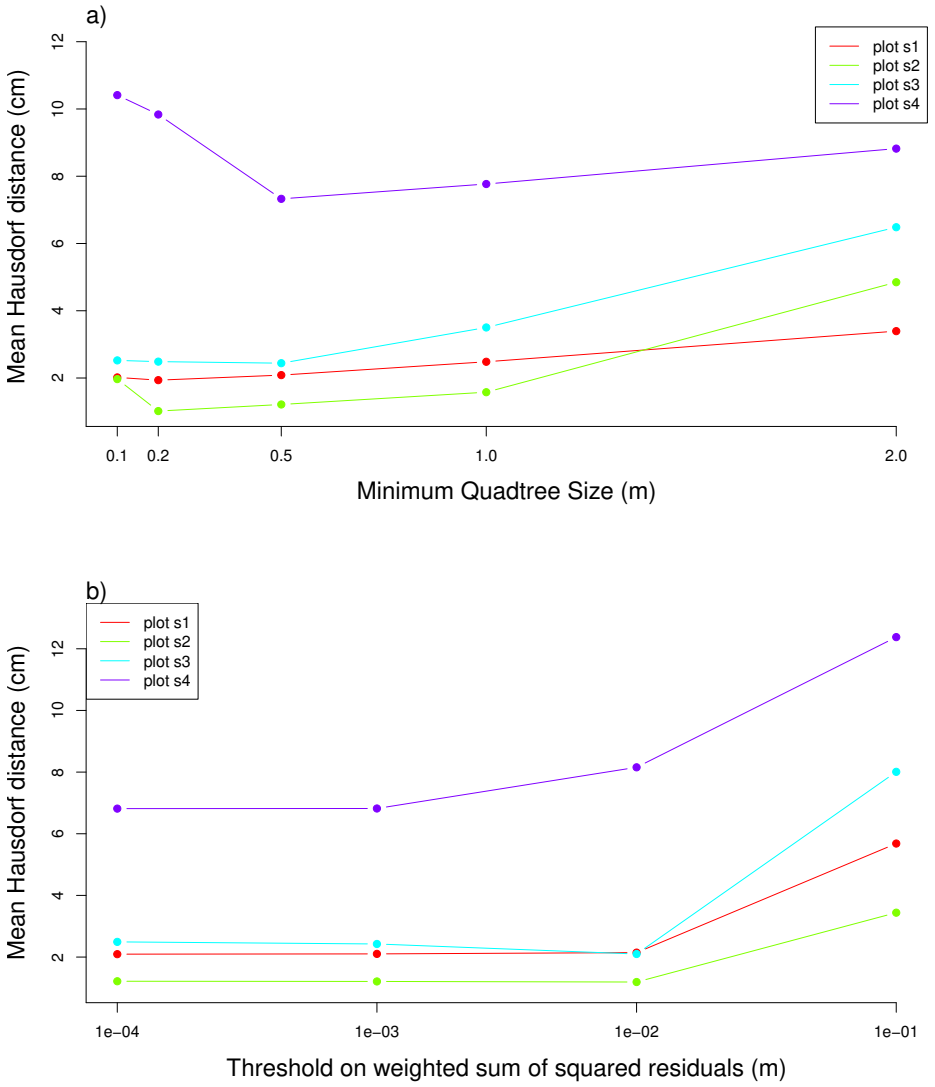


Figure 4.8: Sensitivity of the minimum size of a) the leaf of the quadtree $Size_{min}$ and b) the threshold on the the weighted sum of squared residuals $E_{threshold}$

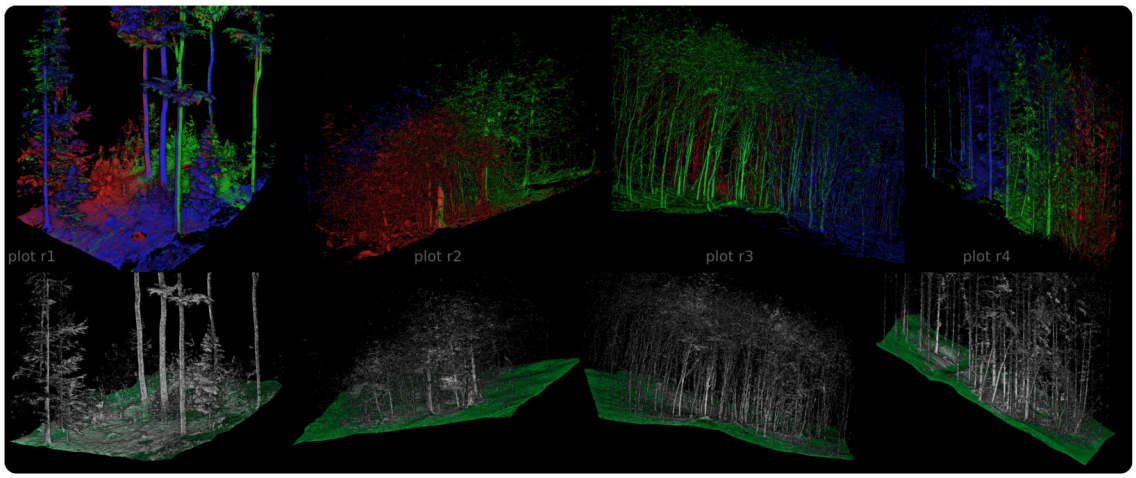


Figure 4.9: On the top row: Three georeferenced scans acquired from different points of view (red, green and blue point clouds) for plots r1, r2, r3 and r4. On the bottom row: Merged point clouds of plots r1, r2, r3 and r4 and the resulting reconstructed DTM(the point density has been reduced and an eye-dome lightning shader has been used for display purpose)

ness of the approach. Indeed, the distance of the 90th percentile varied from a factor 1.55 (4.44 to 6.92 cm, plot s1) to a factor 4.91 (6.39 to 31.38 cm, plot s4).

5.2 Real data

Table 4.5 presents the parameters values used for processing the point cloud obtained for the four forested plots. The quadtree-related parameters were fixed according to the sensibility analysis performed on the simulated data. The filter parameters were set following a sensitivity analysis. The performance of the histogram filter is based on the bin size ($Size_{bin}$, Fig. 4.10a)) and the number of bins of the median filter ($Width_{filter}$, Fig. 4.10b)). While the two parameters are dependent, Fig. 4.10a) and 4.10b) presente the response curve of the mean Hausdorff distance for each parameter separately, the value of the other parameter being set to its optimal value. The response of the Hausdorff distance to the bin size presents minimum values in the range 0.1-0.15 m. Smaller values added noise to the histogram, making it difficult to have clear peaks in the histogram. On the opposite, large $Size_{bin}$ values might mix ground and non-ground points, thus smoothing the histogram and lowering the precision of the filter. The size of the median filter ($Width_{filter}$) was found to be sensitive to low value (*ie.* from 1 to 5), especially in high slopes (Fig. 4.10b)). Above this threshold, the sensitivity remains low, despite an optimum could be find in the range 5 to 7. The parameter of the neighborhood filter (E_{max}) behaves like the size

Table 4.4: Statistical indices on Hausdorff distance (cm) for plot 1,2,3 and 4, on the overlapping area (overlap) and on the full area (total)

	plot r1		plot r2	
	overlap	total	overlap	total
	scan 1 on scan 2 (overlap 18%)		scan 1 on scan 2 (overlap 18%)	
mean	3.69	8.21	4.56	16.68
percentile 70	4.32	8.92	5.06	17.26
percentile 90	7.96	17.72	9.22	50.46
	scan 2 on scan 3 (overlap 33%)		scan 2 on scan 3 (overlap 22%)	
mean	2.87	8.74	3.60	10.33
percentile 70	2.99	10.28	3.95	9.45
percentile 90	7.28	22.04	8.03	31.75
	scan 3 on scan 1 (overlap 20%)		scan 3 on scan 1 (overlap 13%)	
mean	4.53	9.24	5.84	19.99
percentile 70	5.33	9.98	6.59	16.17
percentile 90	9.34	21.40	10.96	59.99
	plot r3		plot r4	
	overlap	total	overlap	total
	scan 1 on scan 2 (overlap 21%)		scan 1 on scan 2 (overlap 12%)	
mean	3.20	16.28	5.60	12.48
percentile 70	3.36	9.57	6.72	11.65
percentile 90	6.46	45.95	11.60	30.01
	scan 2 on scan 3 (overlap 20%)		scan 2 on scan 3 (overlap 12%)	
mean	3.80	8.79	6.30	18.11
percentile 70	3.87	10.08	6.81	19.79
percentile 90	7.93	23.02	15.34	43.12
	scan 3 on scan 1 (overlap 23%)		scan 3 on scan 1 (overlap 7%)	
mean	2.75	16.04	8.80	18.39
percentile 70	3.06	10.75	10.48	17.31
percentile 90	5.59	44.77	15.86	45.70

Table 4.5: Parameters set optimized for the DTM reconstruction of the real data

parameters	plot r1	plot r2	plot r3	plot r4
$E_{threshold}$(cm)	0.1	0.1	0.1	0.1
$Size_{min}$(cm)	20	20	30	40
$Size_{bin}$(cm)	0.15	0.05	0.05	0.1
$Width_{filter}$	3	5	10	5
E_{max}(m)	6	6	2	8

of the median filter, with a higher sensitivity to small values and an optimal value around 6.0 m (Fig. 4.10c)). This was expected because the neighborhood filter is related to both the quadtree size and the terrain slope. By linking neighbor patches and putting a threshold on the difference between them, E_{max} quantify the stiffness of the resulting surface.

The plots were processed using the parameters values presented in Table 4.5. Because the optimal parameters are expected to vary according to the plot structure, an optimization procedure was developed for each plot. The approach consists in providing a range of values for each parameter and to select the set of parameter values minimizing the Hausdorff distance between scans of the same plot.

Table 4.4 presents the results obtained over the four forested plots. Fig. 4.9 presents on its top row the point clouds colored by scan, to provide an idea of the scene complexity and on its bottom row the DTM computed from the merged point clouds. A notable difference with the simulation study is the increased complexity of the vegetation structure and of the terrain roughness. This resulted in increased occlusions and in more complex patterns of point densities and spatial distributions. This effect of vegetation density could be seem through the very small amount of overlapping areas between scans (Table 4.4), with values ranging from 7 (plot r4) up to 33% (plot r1). In this context of intense reconstruction, the Hausdorff distance obtained here are higher than those achieved from the simulated data (Table 4.3). The mean and 99th percentile distance values range from 8.21 cm (plot r1) to 19.99 cm (plot r2), and from 17.72 to 59.99 cm respectively. But most of these differences originated from reconstructed areas. Indeed, the increase is much more limited in the overlapping areas, with mean distance ranging from 2.75 cm (plot 3) to 8.80 cm (plot r4), and values of the 90th percentile limited to the range 7.96 (plot r1) -15.96 cm (plot r4).

The degradation of the Hausdorff distance with respect to the simulation might have different origins. Overall, and as indicated above, the driving factor is the higher density of the vegetation as well as the presence of low vegetation. While the former generated complex occlusion patterns,

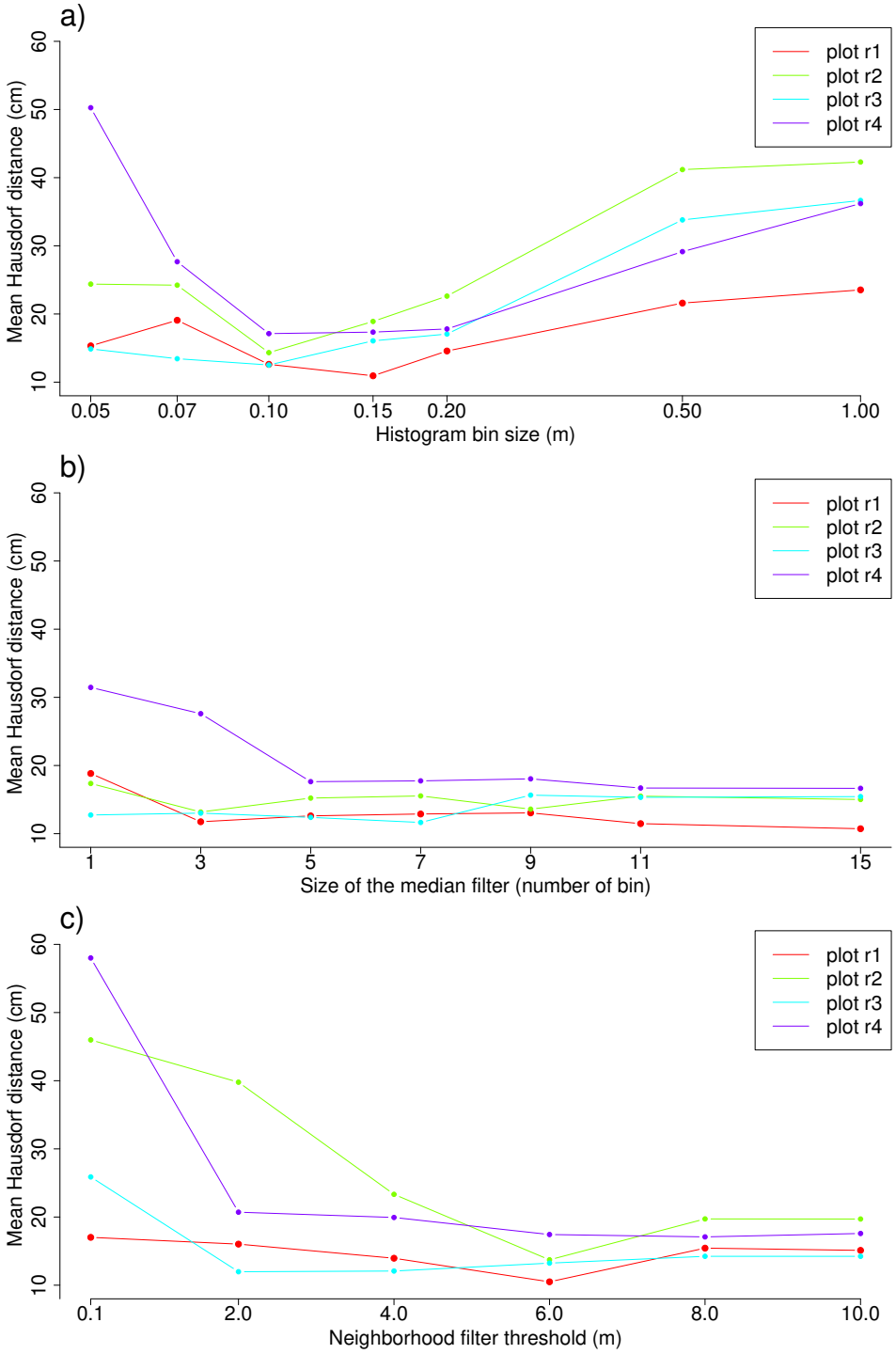


Figure 4.10: Sensitivity of a) the size of the bin of the histogram filter $Size_{bin}$, b) the width of the median filter of the histogram filter $Width_{filter}$ and c) the threshold of the neighborhood filter E_{max}

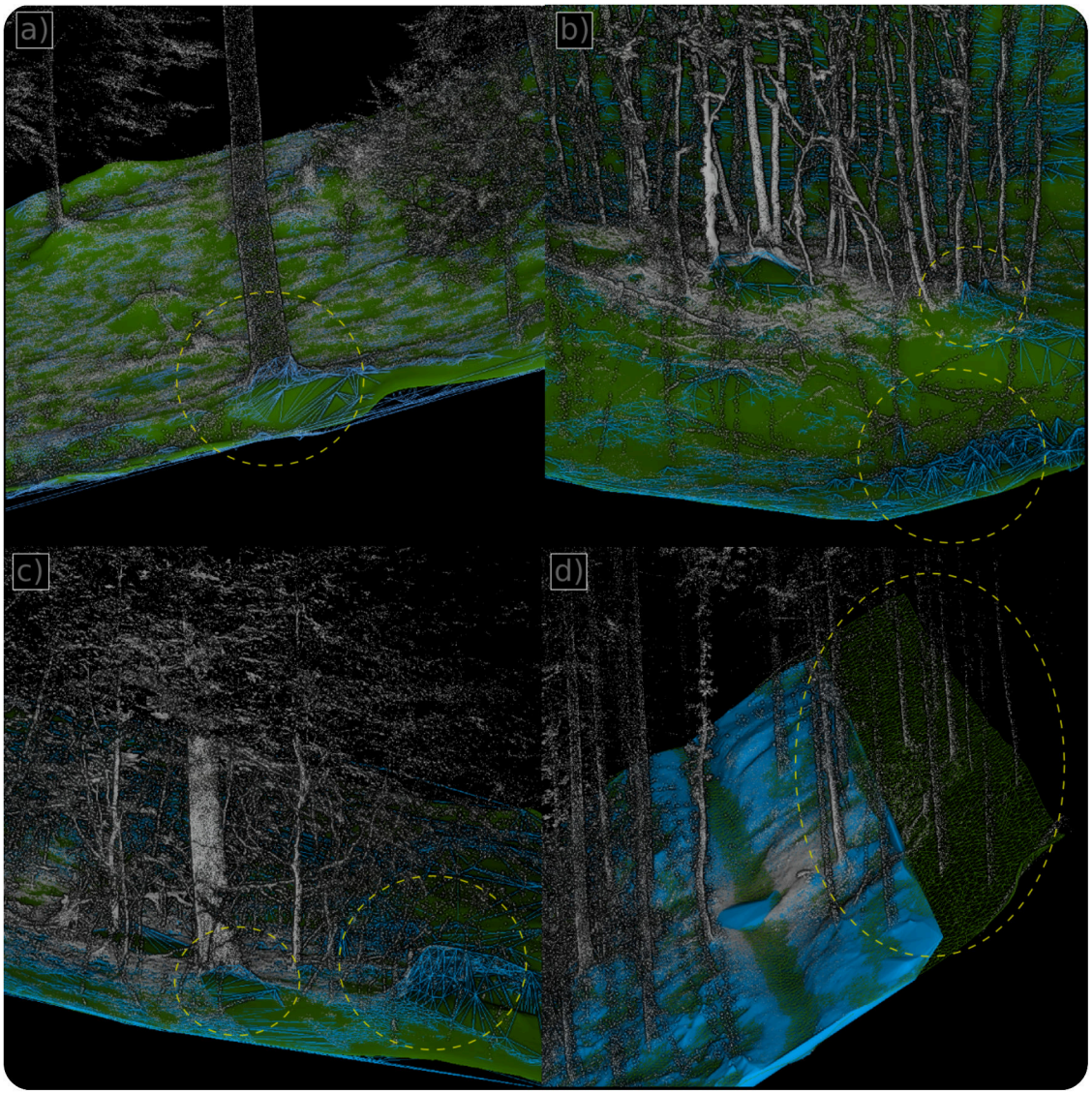


Figure 4.11: Comparison of the MNT produced with Axelsson's algorithm (in light blue) and our method (in green). We identify four types of recurrent errors: a) problem of reconstruction at the base of the trunk, b) instability of MNT at the edge of the plots, c) issue on occlusion management, d) error in points classification (the point density of the point clouds have been reduced, an eye-dome lightning shader has been used and one of the two surfaces is displayed as a wireframe for display purpose)

the later added noise at the ground level, making it more difficult to identify it precisely. Such an issue with low vegetation have also been reported with ALS data[139],[106]. Because occluded areas are reconstructued from the local approximations, any degradation of the local patches will translated into a degradation of the reconstructed part. The major constribution of the increased hausdorff distance within the occluded areas rely to the amount of occlusions, wich is dramatically higher compared to the simulations.[2] used infrared information from a digital camera to filter out vegetation from TLS data and further generate a DTM. Such an approach is however difficult to apply within forest plots due to both the thickness and number of vegetation layers.

The histogram filtering methods proves to perform well on simulated and on real data. Nevertheless it uses two sensitive parameters which needs to be fixed with *a priori* assumptions about the terrain and vegetation complexity, and so about the number or shape of the modes of the histogram. In order to improve the vegetation filtering, we are currently implementing a more efficient mode detector algorithm. We think that such approach would compromise more on the limitation between ground and vegetation points to filter out more precisely the vegetation.

5.3 Comparison with ALS algorithm

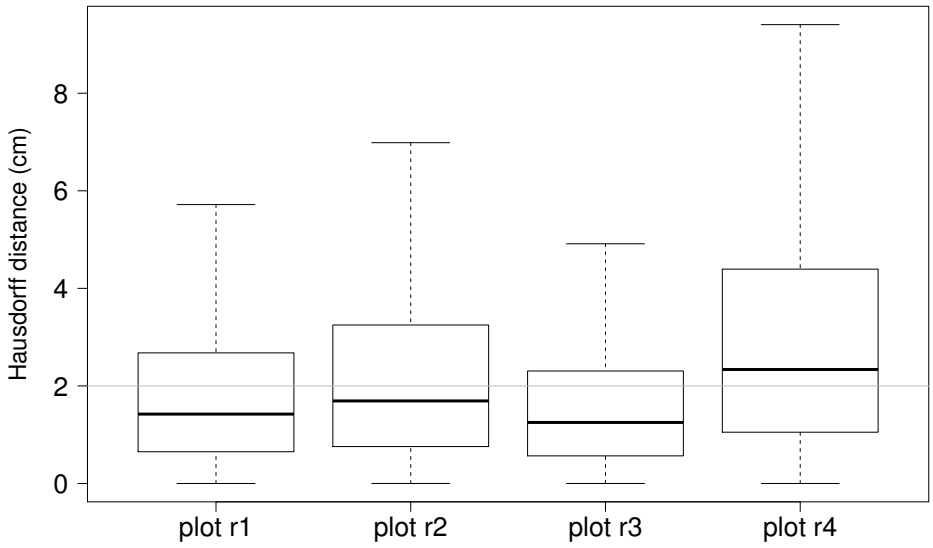


Figure 4.12: Statistical indices on the Hausdorff distance between the DTM surface produced by LAStools and our method for the four plots

In Fig. 4.12 the mean Hausdorff distances between Axelsson's DTM and ours remain around 2 cm for the four plots, which shows a global similarity between Axelsson's surfaces and the ones

produced by our method. Nevertheless, we noticed that LAStools exhibits four kinds of recurring errors on specific point cloud patterns, as shown in Fig. 4.11. Fig. 4.11 a) shows a common error of classification at the base of a trunk. Fig. 4.11 b) shows an example of surface instability at the edge of a plot. Fig. 4.11 c) shows some artifacts produced by occlusions. Fig. 4.11 d) shows an error of classification. Because of the strong slope and the low density of point, the upper part of the plot is classified as non ground. Overall, the surfaces produced by our method are more stable, smoother, with less artifacts and cover the full extend of the plots. As DTM extraction is a key point to asses points altitude, such stability is clearly an advantage of our approach.

6 Conclusion

This paper introduces a ground surface approximation algorithm dedicated to DTM production from TLS data. The proposed algorithm is designed to approximate the ground as the zero level set of an implicit function, rather than directly segmenting ground points (which is the case in most "virtual deforestation" algorithms found in the literature). The presentation of the result as an implicit function enables further developments to improve the quality of the reconstructed surface. Indeed, energy-minimizing deformable models would improve the DTM by pushing the surface towards the data points.

This algorithm has first been tested on simulated data, in order to explicitly measure the quality of ground approximation as a distance between the input surface and the output surface. Overall, the approach proved accurate in terms of ground point segmentation, DTM reconstruction, both on simulated and on real data. While errors tend to increase with the occlusion rate, the reconstructed surface remains consistent with the reference one, highlighting the quality of our hole filling process. One limitation is the difficulty of correctly modeling the ground surface along the border of the sampling areas when large occlusion occurs. However, such a problem is independent from the method itself and can be solved by optimizing the TLS sampling area.

The method has been developed for forest application, but can be used in various environments. Regarding DTM computation, future developments will focus on improving the results computed on complex forest environments by applying a deformable model in order to correct the digital terrain model and further improve data approximation.

5

Geometric model of trees

From allometric models based on field measurement to architectural and quantitative structure models based on terrestrial LiDAR scanning data, every current model found in the literature approximates the trees shape using cylinders. Such a cylindrical representation could be a good approximation for tree having either a simple structure, such as conifer trees, or broad-leaved trees showing regular trunk and branches. However, cylinders are expected to be too rigid to describe more complex shapes (for example, buttresses) resulting from either genetical traits or stresses related to the mechanical environment of the tree[109]. Indeed, the cross sections of tropical stems often present alternating convex and concave portions, as illustrated on Figure 2.4. Improved description of the tree shape might be of prime interest to compute accurate volume or biomass, but also to support our understanding of the effect of the tree environment on its morphological traits, and to assess its economic value. Based on these assumptions, we proposed a more flexible and continuous model expressing the geometry of tree parts as an implicit function. This chapter introduces our new model based on quadratic local approximations blended together with compactly supported radial basis functions in order to transform a discontinuous tree model based on a stack of independent building blocks to a continuous implicit function.

1 Introduction

In many parts of the world, forests are subjected to intense human pressures, and the ongoing global warming further constraint their dynamics and functioning in directions and magnitudes that can strongly varied in space and time as demonstrated by[36] with an emphasize on tree growth. Monitoring forest ecosystems require more and more detailed information about forests attributes, as also needed for example for international reporting regarding carbon cycle and the development of wood energy[104]. While field data acquisitions are time consuming and costly, thus limiting the development of precise field databases of forest parameters, terrestrial laser scanning (TLS) emerged as a practical tool to acquire accurate data on forests at either plot or stand level[47], and to further estimate forest parameters.

TLS systems sensed the local environment using laser signal to produce very detailed 3D point cloud of the scene surrounding the sensor. Newham[114] acknowledged that TLS point clouds catch forest structure information with much more details than traditional field measurements can do. Also extracting information from such 3D point clouds is not easy. The point cloud structure is not homogeneous. The point spacing decreases with distance to the sensors. And the point density can be heavily affected by occlusions generated by the objects spread within the scene.

Also in the last decade, many methods have been used to extract forest information from TLS data at either plot or tree level[94,114,47]. Plot level approaches mostly focused on the estimation or traditional forest parameters such as dbh, height and/or stem profile[99,76,98]. But others approaches directly dealt with the entire point cloud to retrieve plots properties like vegetation profile, vegetation density, or the leaf area index[31,17,159,53].

From very high density point clouds, advanced tree reconstruction approaches have been proposed. Most of them have been developed for the processing of individual trees. [45] proposed an architectural model combining a skeletal curve approach to retrieve the tree structure and allometric relationship to build the branching structure and further assess the amount of foliage. Dassot[47] introduced a semi-automatic approach to model tree architecture using cylinders and further estimate tree parameters such as tree volume. [128] introduced a method involving a clusterisation and segmentation of the point cloud followed by the reconstruction of the tree architecture using cylinders. They further summarized these geometric and hierachical information into the concept of quantitative structure modeling (QSM). A similar cylinder-based tree reconstruction method has been proposed by[73], using a sphere-following approach to progressively reconstruct the tree structure from the ground to the apex. Jumping from the tree level recon-

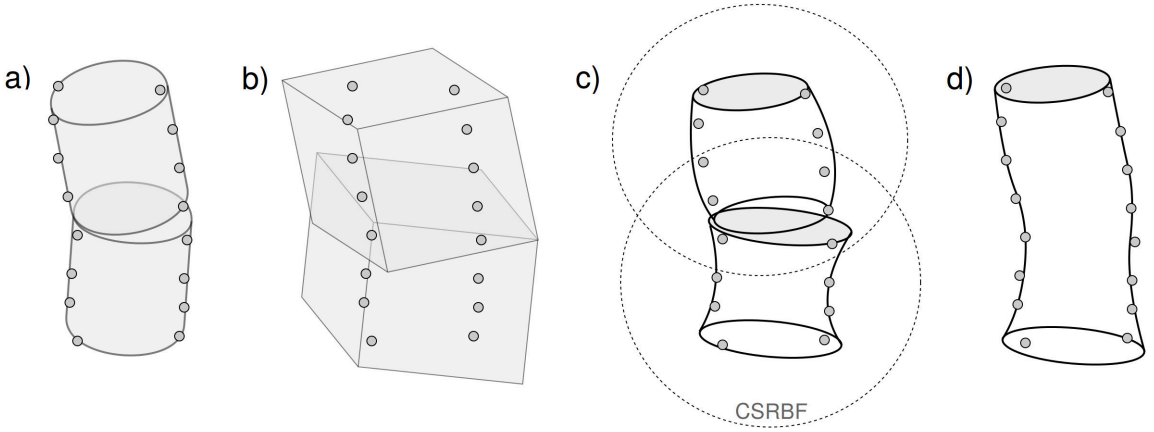


Figure 5.1: Overview of the method for two cylinders adjusted on the point cloud describing a part of the trunk: (a) cylinders of the VHT adjusted to point clouds; (b) bounding boxes of the cylinders defining the space partition; (c) local approximations with cylindrical quadrics and their CSRBF, (d) Extraction of geometric model as the the zero levelset of the implicit function resulting of the merge of local approximations.

struction to the plot level reconstruction is not an easy task. Intermingling crowns and occlusions by branches and leaves in the signal path makes is difficult to accurately segment trees.[127] used a clusterisation approach to first detect tree bases followed by a distance-based expansion procedure to allocate the remaining clusters to the detected trees.[141] used both a clustering and a shortest-path algorithm to detect trees and segment associated crowns. Combining Hough transform and active contours,[129] under a multi-scale framework introduced a method to automatically detect and reconstruct trees. While it has the potential the reconstruct the full tree architecture, the method in its present form has only been validated on tree stems.

Despite the efficiency of state of the art tree reconstruction methods and the ongoing development of segmentation approaches, it would be of interest to rely more on the point cloud structure to produce more realistic 3D tree models. Taking take of the local point cloud structure might allow an improved description of the tree shape as well as the computation of additional criteria describing the wood quality. Because TLS point clouds are prone to occlusions, and also because those data-sets are in-homogeneous, heavy and noisy, they are not reliable for the direct construction of the geometric shape as a mesh model. Instead, the modeling as an implicit surface appears as an lightweight and efficient alternative, to smooth noise, in order to propose a continuous model of the woody part of the tree. Expressed as an implicit function, a shape can be then improved by additional computational step.

Here, we provided a methodological framework to compute an implicit surface modeling of tree woody structure. From a set of geometrical shapes (ie. QSM) we proposed to improve the

tree reconstruction where point data is available and to preserve the supporting geometrical shape in the absence of data. The proposed method is based on local quadratic patches merged together by radial basis functions, following the partition of unity.

The paper is organized as follows. Section 2 introduces definitions and concepts used throughout this paper, and then describes the different steps of our algorithm. Section 3 presents our validation method based on simulated point clouds and section 4 presents the results. Finally, further developments are discussed in Section 6.

2 Algorithm

2.1 Overview of the approach

The method presented here is part of a complete processing line dedicated to the reconstruction of forests plot attributes model. Using TLS point cloud acquired in forets, this processing line automatically detects the trees and models their the woody structure with a continuous surface. The approach presented here consists in the continuous modeling of the tree structure as an implicit surface based on a QSM (*i.e.* a discontinuous sequence of cylinders). While our method is adapted to any kind of QSM, it uses a variant of the Hough transform in the scope of that paper. The method is made of three main steps illustrated in Figure 5.1.

The first one detects the stem and produces the QSM: it divides the dataset along the stem into a sequence of components to transform the global problem into a set of local problems (Section 2.2.2). The second step computes by least square fitting an approximating local quadratic patch in each local component (Section 2.3). Finally the implicit function is computed by merging the local approximations and extracts the final model as the zero-levelset of that implicit function (Section 2.4).

2.2 Input informations

We consider a point cloud acquired with TLS that describes the trunk and branches of an individual tree. That set of un-oriented data points, provided by the sensor, and denoted by $\mathcal{P} = \{p_i\}_{i=1\ldots N} \subset \mathbb{R}^3$ is the input of our algorithm. As mentioned previously, to divide the global surface approximation problem, we build upon a discrete tree model (QSM) to define a division of the space: the set of independent building block of the QSM defines a partition of the space used by our method to pass from global to local. In practice we run a variant of the Hough transform

(VHT), introduced by[129] to obtain a set of cylinders distributed along the bark of the tree (Section 2.2.2). The bounding boxes of those cylinders constitutes the structure of space. Because that method uses not only the position of the points but also their normal, we first detail briefly the normal estimation in Section 2.2.1.

2.2.1 Estimation of the normals

A normal n_i is computed for each point p_i. To do so, a tangent plane at each point p_i is estimated by a principal component analysis on the k-neighborhood of p_i (where k is an user defined parameter). The direction of n_i is chosen orthogonal to the tangent plane. However, such a choice does not guarantee that the resulting set of normal is coherent (*ie.* either all of them are inwards or outward). Therefore, the vector field defined by $\{n_i\}_{i=1...N}$ is corrected by modeling the problem as a graph optimization problem to adjust each n_i with its nearest neighbors[78].

2.2.2 Variant of the Hough Transform

The classical Hough transform was first used in image analysis to identify imperfact instances of image lines. This technique was later extended to identify positions of arbitrary shapes[15]. The variant of the Hough transform approach (VHT) introduced by[129] adjusts a sequence of cylinders along the stem axis.

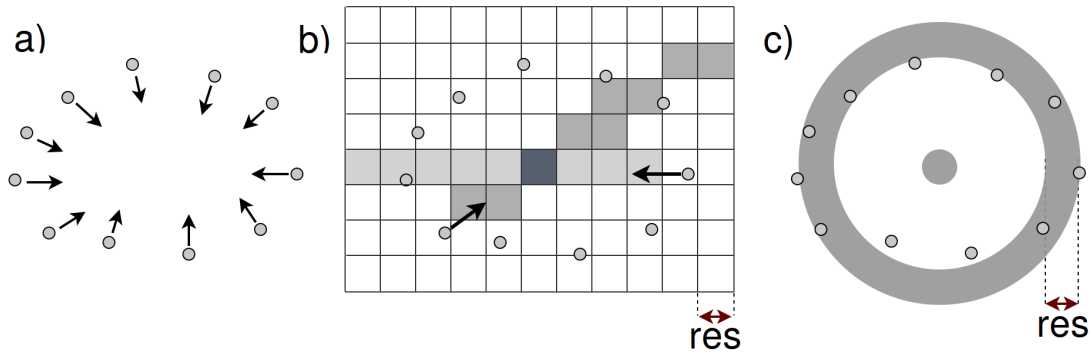


Figure 5.2: Illustration of the Hough transform in 2D: (a) computation of the point normals; (b) definition of a 4D Hough space (x,z,y,r) of "res" resolution. Vote of two points along their normals, the darker cell cumulates the vote of both points; and (c) after the vote of the whole set of points, extraction of the most probable circle approximating the points with a given uncertainty linked to the resolution of the Hough space. The uncertainty passes on the center coordinates and on the radius.

The VHT (Figure 5.2) is based on the idea on the following idea: a point p_i of normal n_i, votes for a small set of cylinders, actually a line directed by n_i in the accumulator space discretized along

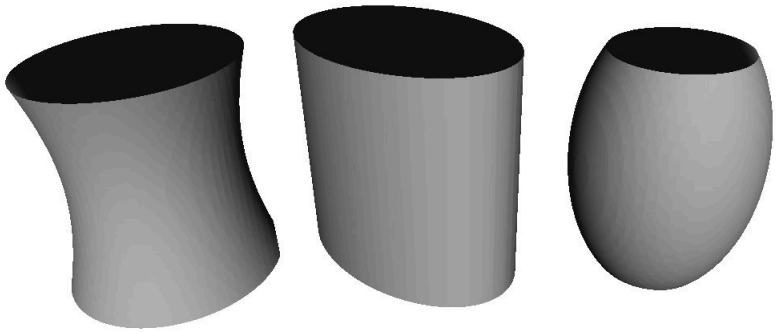


Figure 5.3: Class of shapes modeled by Equation 5.1 : (left) hyperboloids , (center) cylinders with an ellipse cross section and (right) ellipsoids

(x, y, z, r) where x, y, z is the center and r the radius of the cylinder. This line in the accumulator space is computed by a fast voxel traversal algorithm based on[5]. The most probable cylinder in the slice is estimated as the local maximum in that discrete slice. This cylinder (x_i, y_i, z_i, r_i) provides a bounding box $\mathcal{B}_i$ and its center c_i. The set of bounding box $\mathcal{B}_i$ constitutes the spatial structure we build on in the following step (Figure 5.1b)).

2.3 Local approximation of data contained in a cell

For each local frame, we first compute a Frenet coordinate system. Actually, let w denote the eigenvector associated with the smallest eigenvalue in the principal component analysis of the normals. We complete this vector so that (u, v, w) denotes an orthonormal coordinate system. The points of $\mathcal{B}_i$ are approximated by a local implicit cylindrical quadric function of the following form:

$$g_i(u, v, w) = A_i \cdot u^2 + B_i \cdot v^2 + C_i \cdot u \times v + D_i \cdot w^2 - r_i \tag{5.1}$$

r_i is the radius given by the Hough transform and coefficients A_i, B_i, C_i, D_i are determined by the least square minimization of:

$$\mathcal{E}_i = \sum_{\substack{j \in \{1 \ldots N\} \\ p_j = (u_j, v_j, w_j) \in \mathcal{O}_i}} g_i(u_j, v_j, w_j)^2 \tag{5.2}$$

Under such a minimization, only few shapes are allowed : cylinders with an ellipse cross section, ellipsoids and hyperboloids (the signs of coefficient A and B determine whether the hyperboloid is one sheet or two sheets).

Once g_i is defined (by least square minimization of Equation 5.2, the total residual $\mathcal{E}_i$, which

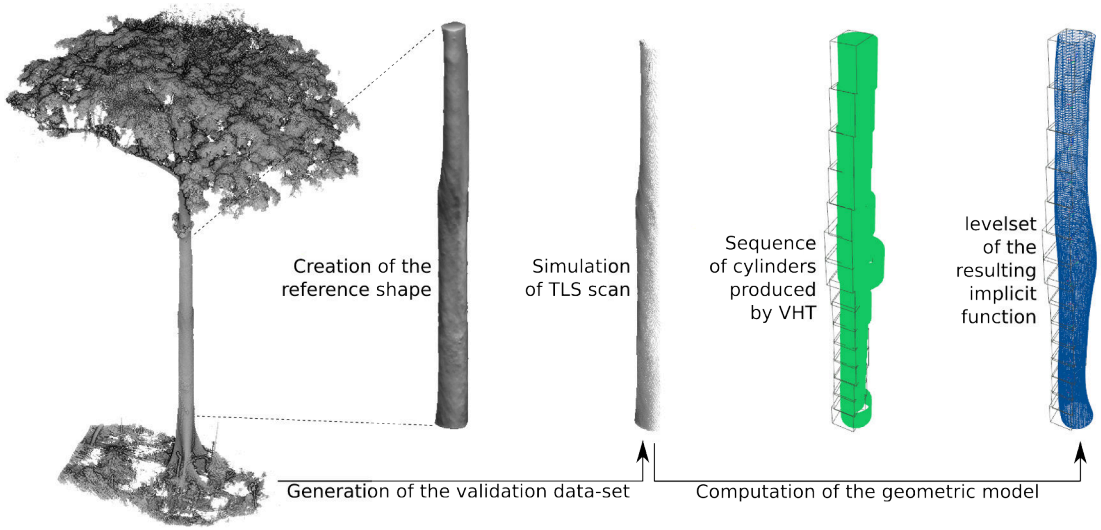


Figure 5.4: Illustration of the validation process of our approach : (a) multi-view scanned tree, (b) the raw data is clipped and cleaned to keep only 15 meters of the trunk. , (c) the Poisson surface reconstruction provides the reference mesh and its volume, (d) the LiDAR simulator computes a simulated single scan LiDAR point cloud using a ray tracing algorithm, (e) Hough cylinders distributed along the trunk, (f) levelset of the implicit function, (g) close-up on the implicit on the levelset.

is the weighted sum of squared residuals, quantifies the quality of the fit. If $\mathcal{E}_i$ is superior to a threshold defined by the user or if the coefficient A and B have different signs (which is given by a two sheets hyperboloid), we consider that the approximation failed and proceed to a simpler one, constraining $D_i = 0$. This allows to keep control over the local approximation by forcing it to be cylindrical. The flexibility of the quadric is thus constrained to produce a coherent approximation and prevent from data gaps in the final volume computation

2.4 Computation of the global implicit model

After estimating a local approximation in each cell $\mathcal{B}_i$, the global geometry of the tree is computed by smoothing all the quadratic patches into a surface model. In practice, compactly supported Wendland's RBF functions[153] were used as a partition of unity to merge these local approximations and compute a global implicit model. More precisely, we use $\phi(r) = (1-r)_+^4(1+4r)$, where $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise, which is $\mathcal{C}^2$ and radial on $\mathbb{R}^3$. Hence in each cell $\mathcal{B}_i$, we use, as a partition of unity, the function $\Phi_{\sigma^i}(\|x - c_i\|) = \phi(\frac{\|x-c_i\|}{\sigma^i})$, where the parameter σ^i controls the size of the zone of influence centered around c_i.

The global implicit model $f : \mathbb{R}^3 \mapsto \mathbb{R}$ is computed as:

$$f(x) = \sum_{c_i} g_i(x) \cdot \Phi_{\sigma^i}(\|x - c_i\|) \tag{5.3}$$

where Φ_{σ^i} is the CSRBF defined previously, σ^i is its radius and is defined hereafter and g_i is the local implicit approximation, computed in Section 2.3.

As σ^i is the radius of the smoothing function Φ_σ, this parameter control the influence of each local approximations: a smaller value tends to keep the influence of the quadric of a given cell $\mathcal{B}_i$ really local, whereas a larger value tends to spread its influence along the stem axis of the tree. Moreover, the support of the function f(Equation 5.3) is the union of the supports of the CSRBF Φ_{σ^i} it is composed of. To guarantee that this union of supports covers the input point cloud, we compute σ^i as:

$$\sigma^i = \max(h, r_i) * (1 + \delta) \tag{5.4}$$

where h is the height of the bounding box $\mathcal{B}_i$ and r_i is the radius of the cylinder provided by the VHT. The coefficient δ is optimized by the user and allows to control the influence of the approximations.

2.5 Extraction of the wrapping surface as a mesh

Following from our modeling choices (namely Wendland's RBF and quadrics), the implicit function f is C^2. Hence it is possible to use a polygonization algorithm to represent the zero-level set of the implicit function f. In the space defined by the union of the spheres of radius σ^i, f is discretized into a scalar field, whose resolution is set by the user. From this scalar field, the polygonizer computes a set of triangles approximating the implicit surface f, by using a marching cube like algorithm. We use a Bloomenthal-like polygonizer[22] which is well suited for our functional RBF.

2.6 Complexity

The complexity study shows that the algorithm is of order $\mathcal{O}(N)$, where N is the initial number of data points (section 2.2). Indeed, the local approximation is of order $\mathcal{O}(N \times N_{cells})$, where the constant N_{cells} is the number of slices provided by the VHT.

3 Experimental

The experiments were conducted under a simulation framework. In order to model realistic shapes with known reference volume, we set up a three step process (Figure 5.4). From real trees ($n = 4$) scanned with a high density, portions of trunks ($\sim 15m$ length) with a limited amount of occlusions were extracted. Then a Poisson surface reconstruction (PSR) algorithm [86] (Section 3.1) was applied to the point cloud to generate a realistic shape and further compute its volume. From these references shapes and volumes, a LiDAR simulator based on a ray tracing algorithm (Section 3.2) was then used to generate point clouds from various point of views and train the modeling. Indeed, each point cloud only describes the visible side of the trunk, thus providing large occlusions. To evaluate the performance of the proposed method for tree volume estimation from occluded shapes, three different reconstruction approaches were compared (see Figure 5.5): 1) the sum of the volume of the cylinders adjusted by VHT, 2) the sum of the volume of the cylinders obtained by least square fitting and 3) our method based on cylindrical quadrics and RBF.

3.1 The reference volume

To realistic tree surface such model we considered detailed TLS scans of tree individuals and used the Poisson surface reconstruction (PSR) method to produce a closed surface of part of the trunks. These TLS scans were acquired during a special field campaign dedicated to the study of few individuals : the low vegetation surrounding the trees have been cut and the trees themselves have been scanned form multiple point of view. This special modus operandi produced homogeneous point clouds depicting accurately the tree shape details. Thus, from these multi-scan point clouds, we produced reliable tree surface model with PSR. Represented as a triangle mesh, we computed their volumes by adding the signed volume of each of the tetrahedra formed by the origin and each triangle of the mesh [158]. Figure 5.6 presents the four individuals chosen for the study and the mesh resulting of the PSR. The tree 1 and 2 are *Terminalia superba* and the trees 3 and 4 are *Pycnanthus angolensis*.

3.2 Generation of the testing data

From the triangle meshes produced by the Poisson reconstruction surface implemented in Meshlab (http://meshlab.sourceforge.net/), we ran a TLS simulator to generate several single-view scans from varying point of view. This simulator is based on PBRT (http://www.pbrt.org/)

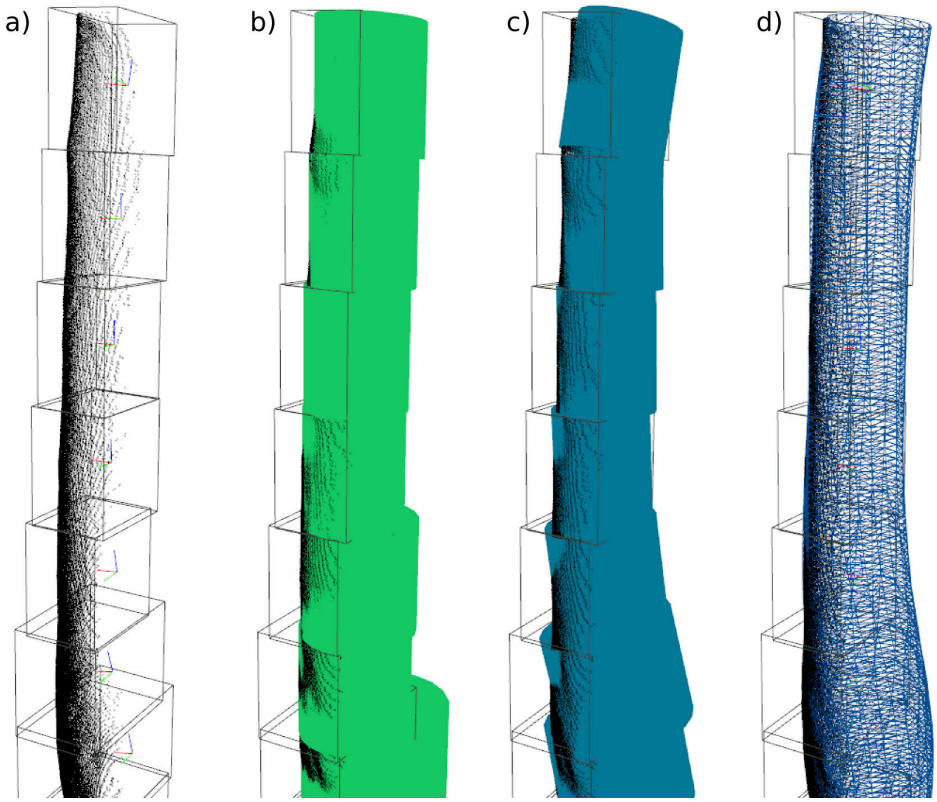


Figure 5.5: Close up on the modeling of the tree from single view point simulated scan with : (a) the cylinders derived from the VHT, (b) the cylinders from least square fitting, (c) the quadrics surface before the merge with RBF, (d) the levelset of the implicit function resulting of the merge of the quadrics with the RBF.

and emulates the interactions using a ray tracing method. To artificially increase the size of our testing data sample, we generated a set of single point of view scans by setting up virtual scan positions evenly distributed around the reference mesh (*i.e.* 16 for each trunk model, locate 22.5° apart from each other).

4 Results and discussion

Volume statistics are shown as box-and-whisker plots in Figure 5.8 for the global set of volume estimation, and Figure 5.7 details the results for individual trees. For each data set of each methods, a box is drawn to indicate the position of the lower and upper quartile of the data. The box has a center line indicating the median of the data. The whiskers above and below the box indicates the maximum and minimum values in the data set (except for outliers).

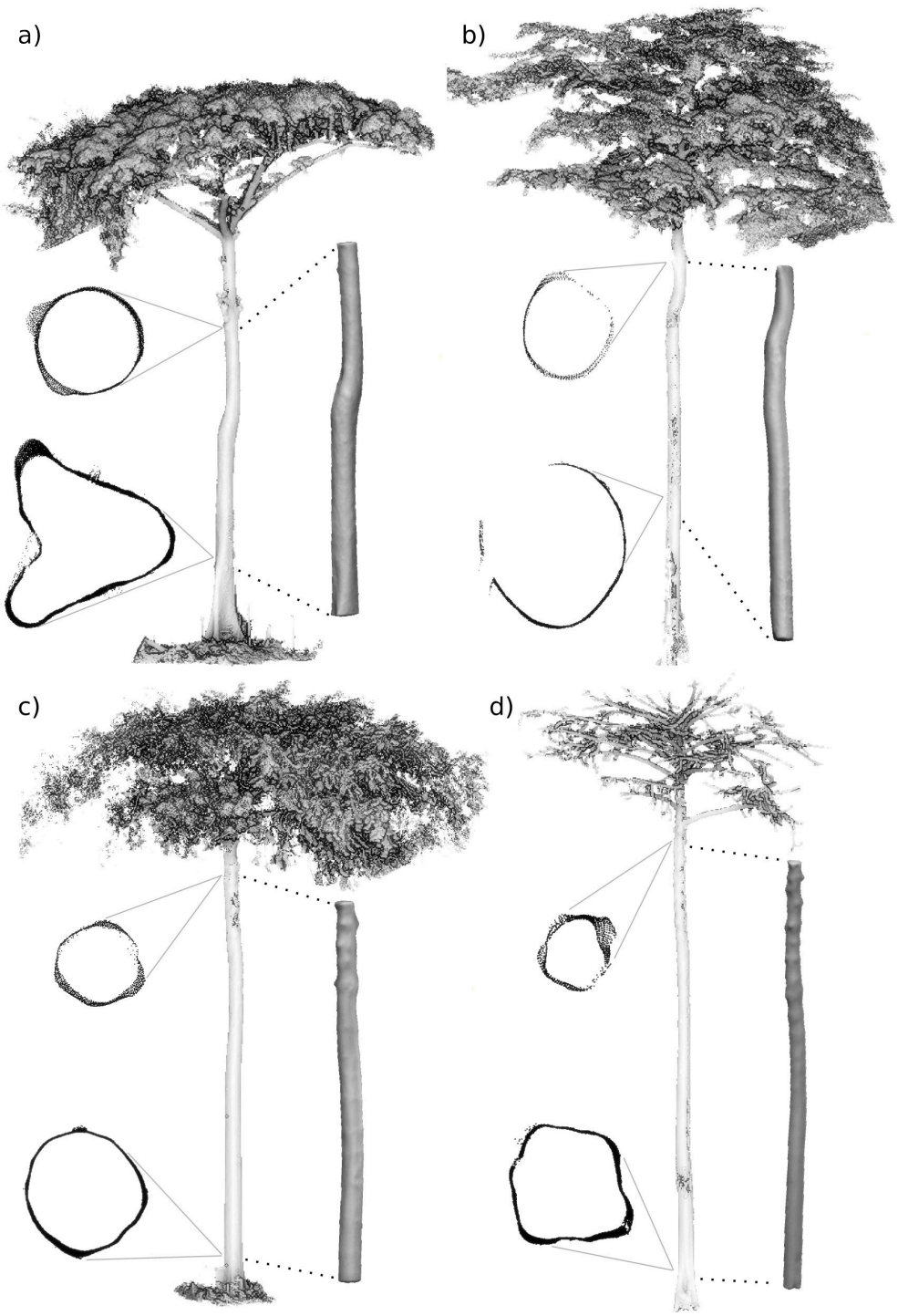


Figure 5.6: The multi scan LiDAR point clouds of four trees along with their trunk PSR. a) and b) are *Terminalia superba* (common name : Fraké). c) and d) are *Pycnanthus angolensis* (common name : Ilomba). For each tree, the mesh resulting from PSR is display on their right side, with the cross sections of the point cloud on the left side.

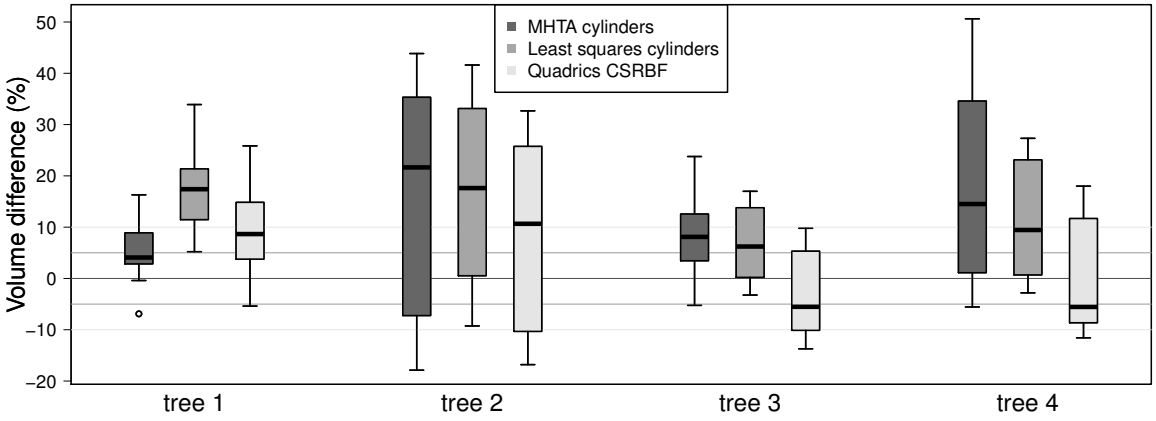


Figure 5.7: Statistical indices on the normalized difference of volume between the estimation and the reference for the three methods detailed for each tree.

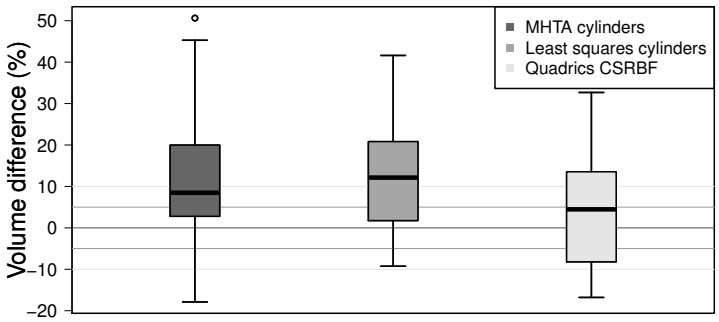


Figure 5.8: Statistical indices on the normalized difference of volume between the estimation and the reference for the three methods tested on the whole set of trees.

Overall, the volume estimated by our method is closer to the reference than the fitting of cylinders method what shows a clear improvement in the use of cylindrical quadrics and CSRBF over the use of cylinders. Except for the tree 1, the proposed method show a light overestimation of the volume but is unbiased. This results highlight the importance of developing methods more adapted to the real shape of the trees. Indeed, cylinder fitting or other forms of geometric fitting are not well suited to model complex tree structures.

The figure 5.8 shows a mean difference of estimation around 10% of the reference volume for the fitting of cylinders with VHT and least square method. That mean difference is reduced to less than 5% with our method. This highlights a improvement in the estimation of the volume, but also an improvement in the modeling of the tree itself. Indeed, computing the volume of a tree has a strong smoothing effect. A small improvement in the volume estimation means a noticeable improvement in the tree modeling.

The effect of tree complexity on modeled volume can be seems on box-plots describing the four trunk samples independently (Figure 5.7). Overall, the simpler the tree structure, the better the results. Indeed, the simpler tree shapes tend to be enough described by a single view-scan, but more complex shape, like the shape of the tree 1, would need homogeneously distributed information to transcribe all the details of the shape. That explains the performance of the VHT cylinders for the tree 1 : the discretization on the radius and the smoothing effect of the computation of the volume as the sum of the cylinders make VHT to perform well in that case. On the other hand, this effects is very sensitive to the parameters and the modeling of the tree remains improvable.

Our algorithm requires a geometric support, provided as a QSM. The methods that reconstruct a coherent set of cylinders along the trunk and the main branches are currently in development. Once finalized, they would extend the structure of the space to support our reconstruction towards the branches.

5 Results based on SimpleTree

While the previous paper described geometric model based on QSM coming from the Hough transform introduced by Ravaglia[129], we propose here a continuous modeling based on SimpleTree[73] software results. Simpletree is a C++ based implementation which builds tree models by browsing the point cloud with a 3D sphere. It describes the trunk and every branches individually by a sequence of cylinders. Unlike the processing of the single trunk section presented in the

previous paper, turning SimpleTree models into a continuous model induce the correct management of the branch junctions. To do so, we lean on the segment breakdown of the tree structure provided by Simpletree. We model each segment following the method described in Section 2.3. Considering a tree composed of N segments, we compute for each segment $j \in \{0, \ldots, N-1\}$ an implicit surface f_j of the form:

$$f_j(x) = \sum_{c_i} g_i(x) \cdot \Phi_i(\|x - c_i\|) \tag{5.5}$$

where Φ_i is the CSRBF centered in c_i and g_i is the local implicit approximation as defined in Section **??**.

In order to limit interference of shapes at the segment junctions, we define g the global implicit function as:

$$g(x) = \mathbf{max}[f_0(x), \ldots, f_{N-1}(x)] \tag{5.6}$$

The global implicit function g is finally polygonized to retrieve the surface mesh model of the whole tree. Figure 5.9 illustrates a watertight mesh model of a an evergreen sub-tropical tree (*Erytrophleum fordii*) produced by such approach.

6 Conclusion

Using this method to compute the volume of trees has proven to reduce the error by 50% on average, which demonstrates an improvement in the global geometrical modeling. We proved that the use of cylindrical quadrics and RBF improve the computation of the volume of trees over cylindrical approximations. Our method is about more than just an estimation of the volume, it models the tree itself as an implicit function, which can be with further work better shaped. Indeed, the implicit function resulting of the merge of the local approximations can initiate a deformable model in order to push the surface towards the data point, thus improving the estimation of the volume.

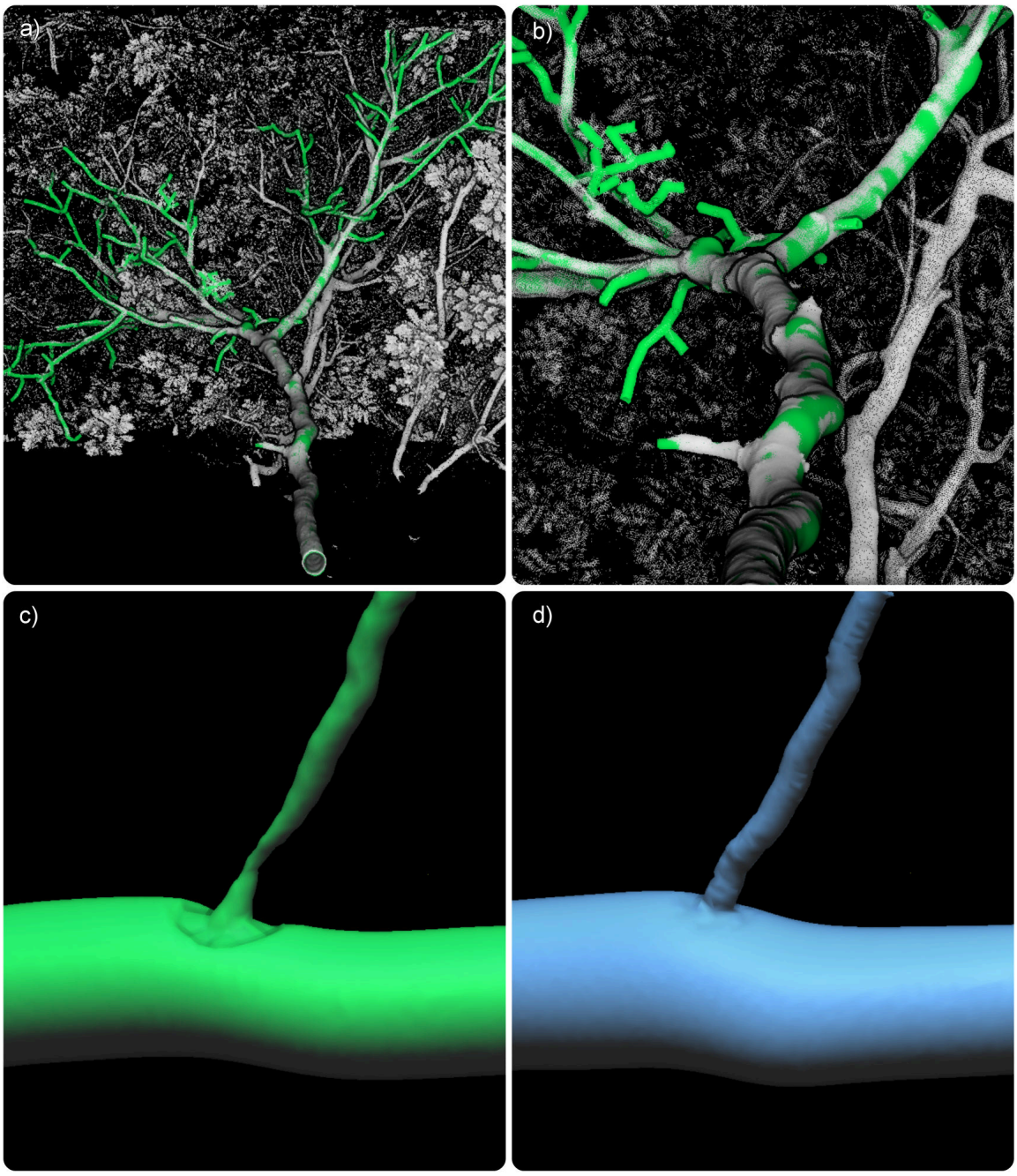


Figure 5.9: Watertight tree mesh model of a an evergreen sub-tropical tree (*Erytrophleum fordii*) based on SimpleTree: (a) and (b) raw point cloud and the continuous surface model from two points of view, (c) branch junction without the treatment detailed in Equation 5.6 and (d) the same junction after correction.

6

Reconstruction of partially occluded objects

In the previous chapter we introduced a method turning a discontinuous geometric tree model made of a linked stack of cylinders into a continuous implicit function. Based on the locations and directions of the building blocks of a QSM, our continuous tubular model fills the occluded areas and improves the approximation of more complex shapes. This chapter focuses on the improvement of the continuous tubular model: as it is defined as an implicit function, the surface model is further re-shaped to approximate the 3D sample points. From this insight emerges two manners of model enhancement: (i) a traditional surface reconstruction approach using the tubular continuous model as a safeguard to handle the occlusions and (ii) a deformable approach initiated by the tubular continuous model. We explored these two ways of improvement. On one hand, we developed an original Poisson surface reconstruction specifically designed to manage occlusions (Section 1 to 5). On the other hand, we address the problem following a deformable approach which has not yet been published and whose theoretical background is described in Section 6.

1 Introduction

This paper presents an original Poisson surface reconstruction method using compactly supported radial basis function (CSRBF), specially designed for the reconstruction from 3D point clouds of partially described objects.

With the emergence of new scanning technologies, reconstruction of 3D surfaces from scattered data receives an increasing attention. Pioneering works in the late nineties explored different options. First, triangular mesh reconstruction (through Delaunay triangulations, alpha shape reconstruction or Voronoi diagrams, for instance [20,23,56,9]). However scanning devices such as LiDAR scanners produce dense, noisy and potentially occluded point clouds which can hardly be modeled by meshes. In order to cope with these data features, smooth surfaces are an efficient solution and many works have been carried out both on parametric and implicit surfaces. The unstructured, non planar nature of point clouds makes implicit surfaces a key modeling tool. Global reconstruction methods based on radial basis functions (initiated by Carr and al. [32]) take advantage of the approximating properties of RBF. However, polyharmonic RBF have a global support, and hence reconstruction entails dense ill-conditioned matrices. In order to mitigate this problem, further works focused on compactly supported radial basis function (either used directly [78] or as blending functions between local reconstructions [115]). Another approach takes advantage of both global and local fitting schemes by approximating the field of estimated normals through a Poisson reconstruction scheme [86].

Our work has been developed in the context of surface tree reconstruction from Terrestrial Laser Scanning (TLS) data-set and more precisely for data-sets acquired in forest environments. TLS technology is nowadays broadly used in the domain of forests study. It allows to acquire the 3D forest structure as point clouds in record time [47]. However, the features of such data are challenging when it comes to reconstruct models and extract informations (actually more challenging than classical applicative data such as urban environments or isolated objects): clouds are extremely dense, inhomogeneous, noisy and present large occlusions. Specific algorithms are required to extract the useful information. One of the Forestry applications is to retrieve automatically the tree structure, branching organization and the branch size distribution. The modeling through quantitative structure models (QSM), that summarize those information by describing the whole tree components in an hierarchical order, has been explored by [72,128,129]. While those methods are effective to extract the structural parameters, they propose a discontinuous model and rely on local cylindrical approximations which has proven to be the best local approximation but can lead to substantial errors. Indeed, to assume that a tree is a cylinder can be far from the

reality, specially when we consider tropical trees.[39] underlined an error of 50% on the computation of the volume from field measurement at the tree level by such approximation in tropical forests. sharper 3D modeling are required to better describe the shape and improve the geometric model, in order to assess precisely tree properties.

To overcome this approximation, and to use the millimeter accuracy of the scanner, classical surface reconstruction methods[86] can be used to produce tree models. While excellent results can be obtain on isolated trees specially well scanned, the complexity of forested environments creates occlusion that are hard to handle for those approaches. Even with dense scanning designs, occlusions may still occur in the upper part of the trees. Indeed, the surface of the trunk and branches sampled by the scanner is often locally occluded by the surroundings. Actually, only the surfaces visible from the sensor point of view are described, the remaining part is occluded. Usually, one combines scans acquired from several point of view around the targeted tree in order to limit the occlusion ratio. However, the multiplication of scans cannot get ride off the entire occlusion because of the spatial configuration between the sensor and the objects scanned. Therefore, none of the existing approaches can be used as such. Our works provide a new Poisson reconstruction algorithm based on CSRBF functions; this underlying model lets us therefore combine the resulting reconstruction with a priori models computed for occluded areas and expressed with CSRBF functions.

2 Overview of the approach

Our method focuses on approximating accurately the data points and managing the occlusions to produce a closed surface. To do so, it identifies and processes separately visible and occluded areas. In visible portions on the object, a Poisson surface algorithm is used to reconstruct the surface. In occluded areas, the tree-like object surface is approximated using a tubular profile. The smoothing characteristic of the compactly supported radial basis functions (CSRBF) guarantees the continuity of the shape between the two models.

Illustrated on Figure6.1a), the two following sections describe both pre-processing steps on which our method builds upon: the computation of the points normal that are required by the Poisson surface solver, and on the other hand the tubular shape that guides the reconstruction in occluded areas.

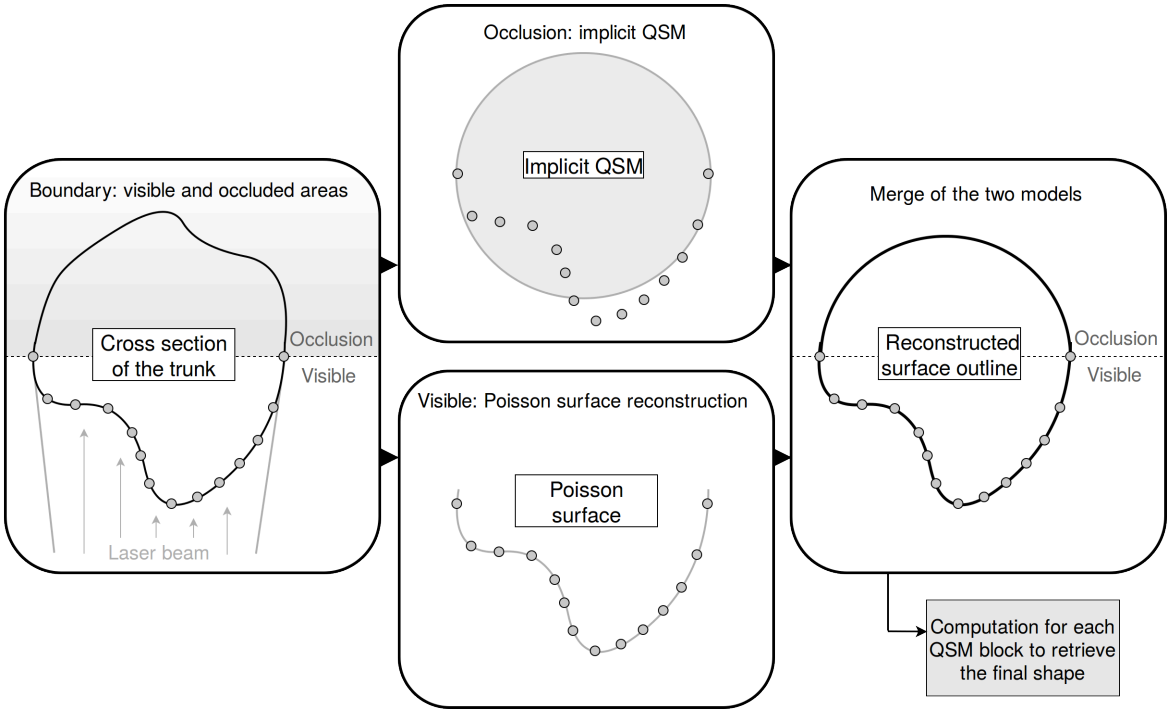


Figure 6.1: Overview of the method: a) as a pre-processing, we compute a QSM and the normal of the points. Then we consider each building block of the QSM independently. b) data points are distributed over the part of the trunk exposed to the LiDAR beam, dividing the space in occlusion and visible areas, c) based on a QSM, we estimate the tree shape as a tubular model expressed as an implicit surface d) in the visible area, we compute a Poisson surface reconstruction based on CSRBF e) the algorithm approximates the surface of the tree by a Poisson surface in the visible area and closes the surface by using the tubular model in the occlusion. The tree surface is retrieved by blending the set of models computed for each QSM building block.

2.1 Points normal computation

Let $\mathcal{P} = \{\mathbf{p}_i\}_{i=1\ldots N} \subset \mathbb{R}^3$ be the input point cloud. The pre-processing step towards Poisson surface reconstruction is to compute a consistent normal fields for each sample. We use the method introduced by Hoppe[78] to compute a normal $\mathbf{n}_i$ at each point $\mathbf{p}_i$. First, we compute for each point $\mathbf{p}_i$ a co-variance matrix $\mathcal{C}$ created from N_k, the k-nearest neighbors of $\mathbf{p}_i$.

$$\mathcal{C} = \frac{1}{k} \sum_{\mathbf{p}_j \in N_k} (\mathbf{p}_j - \overline{\mathbf{p}})(\mathbf{p}_j - \overline{\mathbf{p}})^T \tag{6.1}$$

where k is the number of point in the neighborhood of $\mathbf{p}_i$ and $\overline{\mathbf{p}}$ is the center of the nearest neighbors. We attribute to $\mathbf{n}_i$ the value of the Eigen vector of the matrix $\mathcal{C}$ corresponding to its lowest eigenvalue.

The previous step computes a normal field for $\mathcal{P}$ but cannot guarantee its consistency. Indeed, when dealing with surface reconstruction the normal field should be coherent and verify the following characteristic: for two points $\mathbf{p}_i$ and $\mathbf{p}_j$ that are geometrically close, $\mathbf{n}_i \cdot \mathbf{n}_j \approx 1$. Otherwise either $\mathbf{n}_i$ or $\mathbf{n}_j$ should be flipped. In order to obtain a consistent global orientation of the normal, we fill a graph $\mathcal{G}$ with the data set $\mathcal{P}$. We attribute a weight w_i to the edges connecting each points $\mathbf{p}_i$ to its k-neighbors as:

$$\mathbf{p}_j \in \mathcal{N}_k, w_i = 1 - |\langle \mathbf{n}_i, \mathbf{n}_j \rangle| \tag{6.2}$$

Then, we compute $\mathcal{K}$ a sub graph of $\mathcal{G}$ with a minimum-spanning-tree algorithm [91]. We visit $\mathcal{K}$ with a depth-first search and propagate the normal orientation through its nodes: For a point $\mathbf{p}_i$ of the data set $\mathcal{P}$, we call $\mathbf{p}_j, j = 1 \dots m$ its neighbors in the graph $\mathcal{K}$. We flip the normal of $\mathbf{n}_j$ if $\langle \mathbf{n}_i, \mathbf{n}_j \rangle$ is negative. At the end of the process, the input point cloud $\mathcal{P}$ is enriched by a globally consistent normal $\mathbf{n}_i$ for each of its data point $\mathbf{p}_i$

2.2 Tubular guide estimation

In a pre-processing step, we compute a QSM of the tree. From this discontinuous stack of cylinders, the global outline of the tree is then modeled by a continuous tubular shape as in [110]. Following this approach, the tubular model is expressed as an implicit function built as a stack of quadratic approximation g_i merged together with compactly supported radial basis function (CSRBF), namely Wenland's CSRBF [154]. Our method takes as input the function f defined as:

$$f(\mathbf{q}) = \sum_{\mathcal{Z}_i} g_i(\mathbf{q}) \cdot \Psi_{\sigma^i}(\|\mathbf{q} - \mathbf{c}_i\|) \tag{6.3}$$

where $\mathcal{Z}_i$ is the given slice along the Oz axis, g_i is a quadratic function and Ψ_{σ^i} is a CSRBF centered in $\mathbf{c}_i$.

3 CSRBF Poisson surface reconstruction guided by a model known a priori

As illustrated on Figure 6.1, our method is based on several steps: 1) we divide the space into two parts, the occluded and the visible areas, by analyzing the angular distribution of the points. 2) We define a space of functions based on CSRBF with high resolution near the points and near the surface of the tubular model, and coarser resolution away from them. 3) In the visible area,

we set up and solve the Poisson equation. 4) In the occluded area, we express the tubular shape in our space of functions and inject in the solution. 5) Finally, we extract an iso-surface of the resulting implicit function.

3.1 Mark off the occluded areas

Illustrated on Figure 6.1b), the pre-processing QSM computation described in Section 2.2 provides a division of space into several slices $\mathcal{Z}_i$ along the stem axis. In each subset, an analyze of the angular distribution of the points let us detect holes: among the points contained in $\mathcal{Z}_i$, if two points are separated by an angle larger than a given user-defined threshold, then the portion of space in between is considered as occluded. Such process on the whole set of slice $\mathcal{Z}_i$ defines a boundary between visible and occluded areas.

3.2 Problem Discretization

We use an octree division of space as the resolution of our model needs to be refined close to the points and the surface of the tubular model. To do so, we define the octree to be the minimal octree so that every point falls into a leaf of a resolution fixed by the user. Then we force the octree division close to the tubular shape by analyzing at the eight corners of each leaf falling in the occluded area the sign of the implicit function given in Section 2.2, and illustrated in Figure 6.1b). A different sign for at least one of the corners of a leaf results in its subdivision. Finally, we refine this octree by allowing a maximum difference of levels equal to one between a leaf and its neighbors to settle a smooth decrease of resolution. $\mathcal{O}_{visible}$ and $\mathcal{O}_{occluded}$ are respectively the leafs falling in the visible area and the ones falling in the occluded area as defined in Section 3.1. The set of leafs $\mathcal{O}$ form a partition of the space.

3.3 Definition of the function space

Given $\{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$ a partition of $\mathcal{P}$ (in our case, the octree subdivision of space previously defined), for each subset $\mathcal{O}_i$ (or cell), we denote by $\mathbf{o}_{c_i}$ the center of $\mathcal{O}_i$ (we will call it *center* of $\mathcal{O}_i$). In order to build our implicit model, we define a space of functions as the span of translates of Wendland's CSRBF: for every node $\mathcal{O}$, we set Φ_{σ_o} to be the CSRBF centered in $\mathbf{o}_{c_i}$ with a radius σ_o:

$$\Phi_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|) = \phi(\frac{\|\mathbf{q} - \mathbf{o}_c\|}{\sigma_o}) \tag{6.4}$$

More precisely, we use $\phi(r) = (1-r)_+^4(1+4r)$, where $+$ means $(x)_+ = x$ if $x > 0$ and $(x)_+ = 0$ otherwise. The radii σ_o have been adjusted to recover details entailed to smaller Φ_{σ_o}, while limiting the overlapping of greater functions. With this in mind, we computed σ_o as:

$$\sigma_o = a \times \text{Size}_{\text{min}} + \sqrt{3}/2 \times \text{Size}_o \tag{6.5}$$

Size_{min} is the size of the smallest octree leaf, Size_o is the size of the leaf $\mathcal{O}$ and a is a parameter tuned by the user. $Span\{\Phi_{\sigma_o}\}$, the set of independent basis function Φ_{σ_o} is used as a basis to express the implicit functions computed in the following chapters.

3.4 Setup of the Poisson problem

Poisson surface reconstruction initially consists in computing the characteristic function χ of a closed surface. In[?], the autors highlight an integral relation between the gradient $\nabla\chi$ and the field of oriented points normal. Hence χ can be computed as an implicit function χ whose gradient approximate a vector field $\mathcal{V}$ induced by the normal of the sample points: $\nabla\chi \approx \mathcal{V}$. Because $\mathcal{V}$ is rarely integrable, this variational problem is turned into a standard Poisson problem by applying the divergence operator:

$$\Delta\chi = \nabla.\mathcal{V} \tag{6.6}$$

The definition of a discrete approximation of $\mathcal{V}$ is a critical step. The influence of each sample normal needs to be distributed in its surrounding octree cells to produce a smooth points normal vector field. Considering a smoothing filter $\mathcal{F}$ and that a point $\mathbf{p}_i$ of normal $\mathbf{n}_i$ describes an elementary patch of surface $\mathcal{A}_{\mathbf{p}_i}$,[?] proposes the approximating formula:

$$\mathcal{V}(\mathbf{q}) = \sum_{\mathbf{p}_i \in \mathcal{P}} \mathcal{A}_{\mathbf{p}_i} \cdot \mathcal{F}(\mathbf{q} - \mathbf{p}_i) \cdot \mathbf{n}_i \tag{6.7}$$

With this in mind, in the setting of CSRBF we consider $\mathcal{N}(\mathbf{p}_i)$ the neighborhood of a sample point $\mathbf{p}_i$ as: $\mathcal{N}(\mathbf{p}_i) = \{o \in \mathcal{O}_{visible}, \|\mathbf{p}_i - \mathbf{o}_c\| < \sigma_o\}$. For $\mathbf{q} \subset \mathbb{R}^3$, we define $\mathcal{V}(\mathbf{q})$ as:

$$\mathcal{V}(\mathbf{q}) = \sum_{\mathbf{p}_i \in \mathcal{P}} \frac{1}{\mathcal{W}(\mathbf{p}_i)} \sum_{o \in \mathcal{N}(\mathbf{p}_i)} \alpha_{o,\mathbf{p}_i} \Phi_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|) \cdot \mathbf{n}_i \tag{6.8}$$

where $\alpha_{o,\mathbf{p}_i}$ is defined as:

$$\alpha_{o,\mathbf{p}_i} = \frac{\Phi_{\sigma_o}(\|\mathbf{p}_i - \mathbf{o}_c\|)}{\sum_{o' \in \mathcal{N}(\mathbf{p}_i)} \Phi_{\sigma'_o}(\|\mathbf{p}_i - \mathbf{o}'_c\|)} \tag{6.9}$$

and where the density $\mathcal{W}$ is defined as:

$$\mathcal{W}(\mathbf{q}) = \sum_{\mathbf{p}_i \in \mathcal{P}} \sum_{o \in \mathcal{N}(\mathbf{p}_i)} \alpha_{o,\mathbf{p}_i} . \Phi^1_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|) \tag{6.10}$$

with $\Phi^1(r) = 1 - r$ being a Wendland's CSRBF of first order.

3.5 Resolution of the Poisson equation

We express χ in the space $Span\{\Phi_{\sigma_o}\}$ as a linear combination:

$$\chi(\mathbf{q}) = \sum_{o \in \mathcal{O}_{visible}} x_o . \Phi_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|) \tag{6.11}$$

By projecting the Poisson equation onto this space of functions, we obtain:

$$\forall o' \in \mathcal{O}_{visible}, \langle \Delta\chi, \Phi_{\sigma o'} \rangle = \langle \nabla \cdot \mathcal{V}, \Phi_{\sigma o'} \rangle \tag{6.12}$$

Where $\Delta\chi(\mathbf{q}) = \sum_{o \in \mathcal{O}_{visible}} x_o \cdot \Delta\Phi_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|)$

Let us note $\mathcal{L}$ the matrix of elements $\langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma o'} \rangle$, v the vector of $\langle \nabla \cdot \mathcal{V}, \Phi_{\sigma_{o'}} \rangle$ elements and x the unknown vector, composed of the linear coefficient of χ. Because our function basis is not orthonormal, the Poisson problem then resumes to a least square minimization of $\|\mathcal{L} \cdot x - v\|^2$. In our case, the minimum is computed by resolving $\mathcal{L} \cdot x - v = 0$. Indeed, as the matrix $\mathcal{L}$ is self-adjoint and positive definite thanks to the properties of functions $\Phi_{\sigma o}$ and $\Delta\Phi_{\sigma o}$. Moreover it is sparse because of the compact support of the basis functions, thus solving this problem directly is not expensive. The system is then inverted by using the LDL^T Cholesky decomposition implemented in the Eigen Library. As the efficient resolution of the Poisson problem with CSRBF is technical and complex, see Section 3.6 for further details on the system of equations and optimization implemented for its resolution.

3.6 Optimization

The Poisson Equation 6.6 can be express in its matrix form as:

$$\begin{bmatrix} \ddots & & \\ & \langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ x_o \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \langle \nabla \cdot \mathcal{V}, \Phi_{\sigma_{o'}} \rangle \\ \vdots \end{bmatrix} \tag{6.13}$$

The dot product is that of $L^2(\mathbb{R})$. Given $f : \mathbb{R}^3 \to \mathbb{R}$ and $g : \mathbb{R}^3 \to \mathbb{R}$ two compactly supported functions and Ω the union of their compact supports, the dot product can be expressed as follows:

$$\langle f, g \rangle = \int_{\mathbb{R}^3} f(\mathbf{q}) \cdot g(\mathbf{q}) \cdot d\mathbf{q} = \int_{\Omega} f(\mathbf{q}) \cdot g(\mathbf{q}) \, \mathrm{d}\mathbf{q} \tag{6.14}$$

Keeping up with the definitions introduced earlier in this document, we denote by $\mathcal{L}$ the laplacian matrix and by $\mathcal{V}$ the vector $(\langle \nabla \cdot \mathcal{V}, \Phi_{\sigma_{o'}} \rangle)_{o' \in \mathcal{O}}$.

3.6.1 Evaluation of the integral terms

Let us first compute the coefficients of the matrix $\mathcal{L}$, that is $\langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle$. Integrating by parts compactly supported functions, we obtain $\int_\Omega f'' \cdot g = -\int_\Omega f' \cdot g'$. So we can express the terms of the matrix as:

$$\langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle = -\langle \frac{\partial \Phi_{\sigma_o}}{\partial x}, \frac{\partial \Phi_{\sigma_{o'}}}{\partial x} \rangle - \langle \frac{\partial \Phi_{\sigma_o}}{\partial y}, \frac{\partial \Phi_{\sigma_{o'}}}{\partial y} \rangle - \langle \frac{\partial \Phi_{\sigma_o}}{\partial z}, \frac{\partial \Phi_{\sigma_{o'}}}{\partial z} \rangle \tag{6.15}$$

As pointed in Section 3.3, we use Wendland's CSRBF to define our basis, that is for $\mathbf{q}(x, y, z) \in \mathbb{R}^3$, $\Phi_{\sigma_o}(\mathbf{q}) = (1 - \frac{\|\mathbf{q} - \mathbf{o}_c\|}{\sigma_o})_+^4 (1 + 4\frac{\|\mathbf{q} - \mathbf{o}_c\|}{\sigma_o})$. The derivative along the first coordinate thus gives:

$$\frac{\partial \Phi_{\sigma_o}}{\partial x} = -\frac{20}{\sigma_o^2}(1 - \frac{\|\mathbf{q} - \mathbf{o}_c\|}{\sigma_o})^3 (\mathbf{q} - \mathbf{o}_c)_x \tag{6.16}$$

where $(\mathbf{v})_w$ denotes the coordinate of a vector $\mathbf{v}$ along the w axis (with w respectively x, y and z).

Let us set $\mathbf{q}_o = \mathbf{q} - \mathbf{o}_c$ and $\mathbf{q}_{o'} = \mathbf{q} - \mathbf{o}'_c$. The combination of Equations 6.15 and 6.16 gives:

$$\langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle = -(\frac{400}{\sigma_o^2 . \sigma_{o'}^2}) \int_\Omega (1 - \frac{\|\mathbf{q}_o\|}{\sigma_o})_+^3 (1 - \frac{\|\mathbf{q}_{o'}\|}{\sigma'_o})_+^3 \langle \mathbf{q}_o, \mathbf{q}_{o'} \rangle \mathrm{d}\mathbf{q} \tag{6.17}$$

We then simplify Equation 6.17 as:

$$\langle \Delta\Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle = -(\frac{400}{\sigma_o^2 . \sigma_{o'}^2}) \mathcal{L}_{o,o'} \tag{6.18}$$

where:

$$\mathcal{L}_{o,o'} = \int_\Omega (1 - \frac{\|\mathbf{q}_o\|}{\sigma_o})_+^3 (1 - \frac{\|\mathbf{q}_{o'}\|}{\sigma'_o})_+^3 \langle \mathbf{q}_o, \mathbf{q}_{o'} \rangle \, .\mathrm{d}\mathbf{q} \tag{6.19}$$

Following the same process, we express the projection of the divergence of $\mathcal{V}$ on our base of functions by:

$$\langle \nabla \cdot \mathcal{V}(q), \Phi_{\sigma_{o'}} \rangle = \sum_{\mathbf{p}_i \in \mathcal{P}} \sum_{o \in \mathcal{O}} \alpha_{o,\mathbf{p}_i} \int_\Omega \nabla \cdot [\Phi_{\sigma_o}(\|\mathbf{q}_o\|) \cdot \mathbf{n}_i] \Phi_{\sigma_{o'}}(\|\mathbf{q}_{o'}\|) \, .\mathrm{d}\mathbf{q} \tag{6.20}$$

where:

$$\nabla.[\Phi_{\sigma_o}(\|\mathbf{q}_o\|) \cdot \mathbf{n}_i] = \langle \nabla \Phi_{\sigma_o}(\|\mathbf{q}_o\|), \mathbf{n}_i \rangle = -(\frac{20}{\sigma_o^2})(1 - \frac{\|\mathbf{q}_o\|}{\sigma_o})_+^3 \langle \mathbf{q}_o, \mathbf{n}_i \rangle \tag{6.21}$$

Finally, we rewrite Equation 6.20 as:

$$\langle \nabla \cdot \mathcal{V}(q), \Phi_{\sigma_{o'}} \rangle = \sum_{\mathbf{p}_i \in \mathcal{P}} \sum_{o \in \mathcal{O}} \alpha_{o,\mathbf{p}_i} (\frac{-20}{\sigma_o^2}) \sum_{t \in \{x,y,z\}} \mathcal{D}_{t,o,o'} \, (\mathbf{n}_i)_t \, \mathrm{d}\Omega \tag{6.22}$$

where:

$$\forall t \in \{x, y, z\}, \;\; \mathcal{D}_{t,o,o'} = \int_\Omega (1 - \frac{\|\mathbf{q}_o\|}{\sigma_o})_+^3 \, \Phi_{\sigma_{o'}}(\|\mathbf{q}_{o'}\|) \, (\mathbf{q}_o)_t \, \mathrm{d}\mathbf{q} \tag{6.23}$$

3.6.2 Study of symmetries

In order to fasten and simplify the computation of the matrix and vectors terms of Equation 6.13, we study symmetries of $\mathcal{L}_{o,o'}$ and $\mathcal{D}_{t,o,o'}$ and show that actually only few of these terms need to be computed.

Symmetries for the computation of $\mathcal{L}_{o,o'}$.

Let us define the function f by $f(\mathbf{x}) = (1 - \mathbf{x})_+^3$. Equation 6.19 becomes:

$$\mathcal{L}_{o,o'} = \int_\Omega f(\frac{\|\mathbf{q} - \mathbf{o}_c\|}{\sigma_o}) f(\frac{\|\mathbf{q} - \mathbf{o}'_c\|}{\sigma_{o'}}) \langle \mathbf{q} - \mathbf{o}_c, \mathbf{q} - \mathbf{o}'_c \rangle \, \mathrm{d}\mathbf{q} \tag{6.24}$$

Let us first show that this expression is invariant by a translation $t : \mathbf{q} \mapsto \mathbf{q} - \mathbf{o}_c$.

$$\mathcal{L}_{o,o'} = \int_\Omega f(\frac{\|\mathbf{q}\|}{\sigma_o}) f(\frac{\|\mathbf{q} + \mathbf{o}_c - \mathbf{o}'_c\|}{\sigma_{o'}}) \langle \mathbf{q}, \mathbf{q} + \mathbf{o}_c - \mathbf{o}'_c \rangle \, |\det(\mathcal{J}_t)| \, \mathrm{d}\mathbf{q} \tag{6.25}$$

$\det(\mathcal{J}_t)$ is the determinant of the Jacobian of the translation t whose value is 1.

Then let us consider a rotation $r : \mathbf{q} \mapsto \mathcal{R}\mathbf{q}$ whose matrix in the canonic basis is $\mathcal{R}$ and sending the vector $\mathbf{o}_c - \mathbf{o}'_c$ to $\|\mathbf{o}_c - \mathbf{o}'_c\|e_1$ along the x axis, with $e_1 = (1, 0, 0)$. Equation 6.25 becomes:

$$\mathcal{L}_{o,o'} = \int_\Omega f(\frac{\|\mathcal{R}\mathbf{q}\|}{\sigma_o}) f(\frac{\|\mathcal{R}\mathbf{q} + \|\mathbf{o}_c - \mathbf{o}'_c\|e_1\|}{\sigma_{o'}}) \langle \mathcal{R}\mathbf{q}, \mathcal{R}\mathbf{q} + \|\mathbf{o}_c - \mathbf{o}'_c\|e_1\rangle \, |\det(\mathcal{J}_r)| \, \mathrm{d}\mathbf{q} \tag{6.26}$$

As the rotation $\mathcal{R}$ is an isometry, it maintains norms and dot products.

So, by using $|\det(\mathcal{J}_r)| = 1$, Equation 6.26 becomes:

$$\mathcal{L}_{o,o'} = \int_\Omega f(\frac{\|\mathbf{q}\|}{\sigma_o}) f(\frac{\|\mathbf{q} + \|\mathbf{o}_c - \mathbf{o}'_c\|e_1\|}{\sigma_{o'}}) \langle \mathbf{q}, \mathbf{q} + \|\mathbf{o}_c - \mathbf{o}'_c\|e_1\rangle \, \mathrm{d}\mathbf{q} \tag{6.27}$$

Therefore, Equation 6.27 proves that $\mathcal{L}_{o,o'}$ depends only on σ_o, $\sigma_{o'}$ and $\|\mathbf{o}_c - \mathbf{o}'_c\|$.

Symmetries for the computation of $\mathcal{D}_{t,o,o'}$

Following the same ideas, we study the symmetries of Equation 6.23:

$$\forall t \in \{x, y, z\}, \quad \mathcal{D}_{t,o,o'} = \int_\Omega f(\frac{\|\mathbf{q}\|}{\sigma_o}) \, \Phi_{\sigma_{o'}}(\|\mathbf{q} + \mathbf{o}_c - \mathbf{o}'_c\|) \, (\mathbf{q})_t \, \mathrm{d}\mathbf{p}$$

We can prove in a similar way that $\mathcal{D}_{t,o,o'}$ depends only on σ_o, $\sigma_{o'}$, $\|\mathbf{o}_c - \mathbf{o}'_c\|$ and the distance $\|\mathbf{o}_c - \mathbf{o}'_c\|$ projected on t.

3.6.3 Resolution

The integrals $\mathcal{L}_{o,o'}$ and $\mathcal{D}_{t,o,o'}$ are computed with MISER algorithm of Press and Farrar implemented in GSL. This Monte Carlo technique reduces the overall integration error by concentrating integration points in the regions of highest variance.

Moreover, the use of invariance properties we just proved reduces drastically the number of integrals needing to be computed. Indeed, we showed that the integrals in $\mathcal{L}_{o,o'}$ depend only on σ_o, $\sigma_{o'}$ and $\|o - o'\|$, and that the integrals in $\mathcal{D}_{t,o,o'}$ depend only on σ_o, $\sigma_{o'}$, $\|o - o'\|$ and $(o - o')_t$.

In practice, we stored the values of the integrals in hash-tables whose hashes were computed from the variables σ_o, $\sigma_{o'}$, $\|o - o'\|$ and the extra $(o - o')_t$ for the integrals in the $\mathcal{D}_{t,o,o'}$. Actually, those hash-keys are converted into integer values to avoid floating point precision problems. For each couple o, o', we thus compute the integrals $\mathcal{L}_{o,o'}$ and $\mathcal{D}_{t,o,o'}$ only if the value is absent in the hash-table. Table **??** illustrates the drastic reduction of the number of computation in the case of the reconstruction of the Stanford bunny. For each value of radii explored, among the hundreds

a	Non-zero entries in matrix $\mathcal{L}$	Computations of $\mathcal{L}_{o,o'}$	Computations of $\mathcal{D}_{t,o,o'}$
1.5	4773131	14	31
1.6	4781877	16	38
1.7	4785120	18	44
1.8	4785766	19	47
1.9	4786511	21	52
2.0	4786627	22	54

Table 6.1: Evolution of the *number of integrals computations* $\mathcal{L}_{o,o'}$ and $\mathcal{D}_{t,o,o'}$ for the reconstruction of the Stanford bunny model according to the radii σ_o of the basis functions (parameter a in Equation 6.5) for an octree resolution of 1 mm.

of thousands matrix entries that need to be estimated, only few dozen are actually computed thanks to the symmetries, the remaining being retrieved by a search in the hash-table.

3.7 Injection of the tubular shape

In order to include the tubular shape defined in Section 2.2, f, the implicit function describing the tubular shape, needs to be stated in the base $Span\{\Phi_{\sigma_o}\}$. To do so, we consider the system:

$$\begin{bmatrix} \ddots & & \\ & \mathcal{A}_{o,o'} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ y_o \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ f(\mathbf{o}'_c) \\ \vdots \end{bmatrix} \tag{6.28}$$

where $\mathcal{A}_{o,o'} = \langle \Phi_{\sigma_o}, \Phi_{\sigma_{o'}} \rangle$

Thanks to the properties of Wendland's CSRBF Φ_{σ_o}, this matrix is self-adjoint and positive definite. Thus, the system is inverted by using the LDL^T Cholesky decomposition implemented in Eigen library.

At last, we obtain the linear decomposition of f in $Span\{\Phi_{\sigma_o}\}$ as:

$$f(\mathbf{q}) = \sum_{o' \in \mathcal{O}_{occluded}} y_{o'} \cdot \Phi_{\sigma_{o'}}(\|\mathbf{q} - \mathbf{o}'_c\|) \tag{6.29}$$

3.8 Smoothing between visible and occluded areas

In most cases, the transition between χ the implicit function defined in the visible area and f the one defined in the occluded areas is handled by the smoothing characteristic of the Wendland's

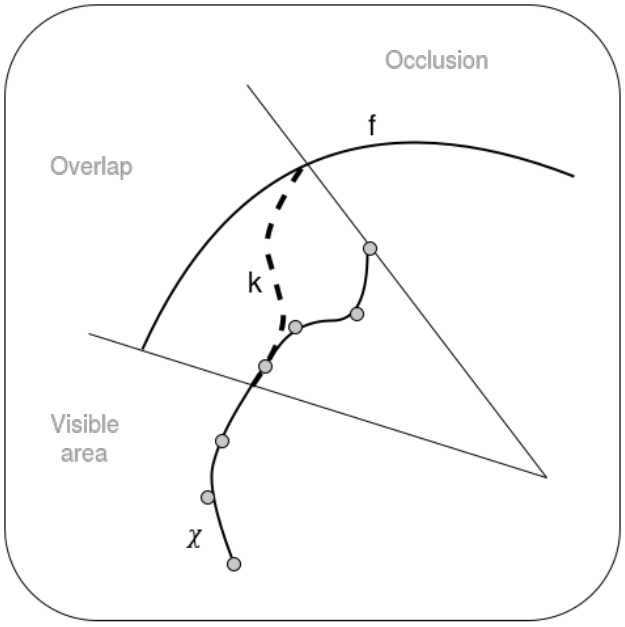


Figure 6.2: Smoothing of the final surface in the overlap area by combining linearly the implicit function solution of the Poisson equation χ and the implicit function of the tubular model f.

CSRBF. Nevertheless, while dealing with complex tree shapes, the transition might be too rough. To overcome this issue, we define an overlap portion of space, whose size is user defined, in which we mix both implicit function linearly. In this sub-part of $\mathcal{O}_{visible}$ called $\mathcal{O}_{overlap}$, we consider the implicit function k defined as:

$$k(\mathbf{q}) = \sum_{o \in \mathcal{O}_{overlap}} [\alpha_o \cdot x_o + (1 - \alpha_o) \cdot y_o] \cdot \Phi_{\sigma_o}(\|\mathbf{q} - \mathbf{o}_c\|) \tag{6.30}$$

where α_o is the coefficient of the linear interpolation between x_o and y_o according to the angular position of the centre of o in $\mathcal{O}_{overlap}$.

Finally, our resulting implicit function h is expressed as: $h = \chi + k + f$, as illustrated on Figure 6.2.

3.9 Surface extraction

Following from our modeling choices (namely Wendland's RBF and quadrics), the implicit function f is $\mathcal{C}^2$. Hence it is possible to use a polygonization algorithm to represent the zero-level set of the implicit function h. In the space defined by the union of the spheres of radius σ_o, h is

discretized into a scalar field, whose resolution is set by the user. From this scalar field, the polygonizer computes a set of triangles approximating the implicit surface h, by using a marching cube like algorithm [22].

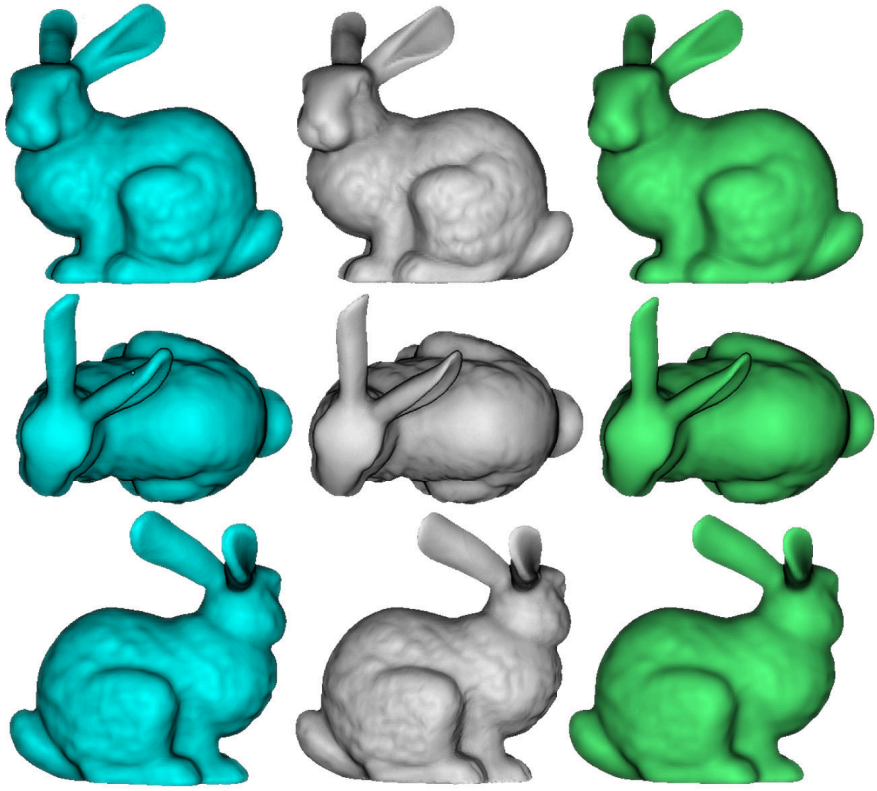


Figure 6.3: Reconstruction of the Stanford bunny surface with different methods: (Center, white) The reference surface reconstructed by the "zipper" program and provided by the Stanford University. (Right, green) The Poisson surface reconstructed by Kazdan's algorithm implemented in Meshlab for an octree depth of 7. (Left, blue) The Poisson surface reconstructed by our method for an octree depth of 7.

4 Results and validation

To assess the quality of our algorithm, we validate it by part: in Section 4.1, we compare the surface generated by our approach to the Poisson surface reconstruction found in the literature, namely the un-screened version of the Poisson reconstruction developed in[?]. Then in Section 4.2, we compare how those two methods handle occlusions. Finally, in Section 4.3, we analyze the improvement brought to the tree modeling.

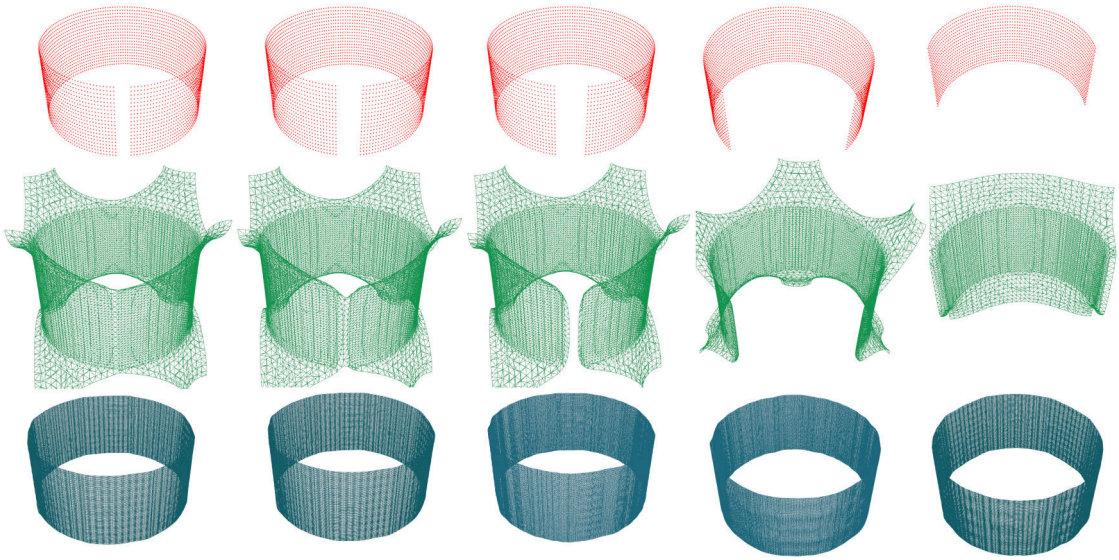


Figure 6.4: Result of classic Poisson surface reconstruction (green mesh) and of our approach (blue mesh) on simulated point cloud (red points) that features an angular occlusion of 10°, 15°, 20°, 90 ° and 180°.

Table 6.2: Hausdorff distance statistics (mm) for the reconstruction of the Stanford bunny

	min	max	mean	RMS
Un-screened PSR	0.00	2.17	0.21	0.32
CSRBF PSR	0.00	1.74	0.22	0.30

4.1 Comparison without occlusion

In order to test the robustness of our version of Poisson surface reconstruction, we compare the surface we generate to the one created by Kazhdan version which stands as a reference in this field of research. To do so, we solved the Poisson problem on a non-occluded well known standard model (the Stanford bunny), and produced directly a surface model without injecting any shape in the solution, *ie.* we ran the part of the algorithm described in Section 3.4, 3.5 and 3.9 only. Figure 6.3 presents the reconstructed surfaces of the bunny.

Then, we estimated the distances to the reference for the un-screened Poisson surface reconstruction and for our method. This quality assessment was done by comparing the reference mesh model with the outputs of the algorithms using the Hausdorff distance implementation available in Meshlab (http://meshlab.sourceforge.net/)

4.2 Occlusion management

The development of our method ensues from the need of holes management in the 3D point samples collected in forests. Indeed, traditional surface reconstruction techniques found in the literature struggle to handle big holes in point distribution. In particular, dealing with trees described by single TLS scans, only the bark facing the scanner is described by the point samples, the back side remains occluded. To compare our approach to the traditional PSR in such situation, we generated point clouds distributed on a cylinder of 1 meter radius with a decreasing angular distribution to simulate a widening occlusion. A visual assessment of the surface reconstruction on occluded cylinders (Figure 6.4) demonstrated how our approach can handle data gaps over 50% of the surface. In such case, point samples are distributed only on half of the cylinder, like a single scan in forests. On the opposite side, traditional PSR shows shape inconsistencies starting at 15° and failed to close the shape when the occluded areas exceeded 20°.

4.3 Modeling of trees

The following experiment focus on testing the efficiency of our reconstruction approach with respect to cylindroid fitting approaches. To that end, portions of real trees were used to generate a reference model and associated volume. The efficiency of the method has been evaluated from a volume estimations and from distance computation from the input points to the computed surface mesh. We chose four portion of real trees modeled as a triangulated mesh whose volume is previously estimated to serve as reference. On this four references, we ran a TLS simulator to generate several single-view scans from varying point of view. To artificially increase the size of our testing data sample, we generated a set of single point of view scans by setting up virtual scan positions evenly distributed around the reference mesh. The simulated point clouds were used to compute tree volume from two approaches: 1) the method based on cylindrical quadrics and RBF and 2) our Poisson surface reconstruction method based on CSRBF. Sixteen simulated point clouds were produced for each tree, which gives a validation set of a total of 64 computations of volume with the two methods.

5 Discussion

First, we compared our method to the un-screened Poisson surface reconstruction well recognized in the computer graphics field of surface reconstruction. Despite a longer computation

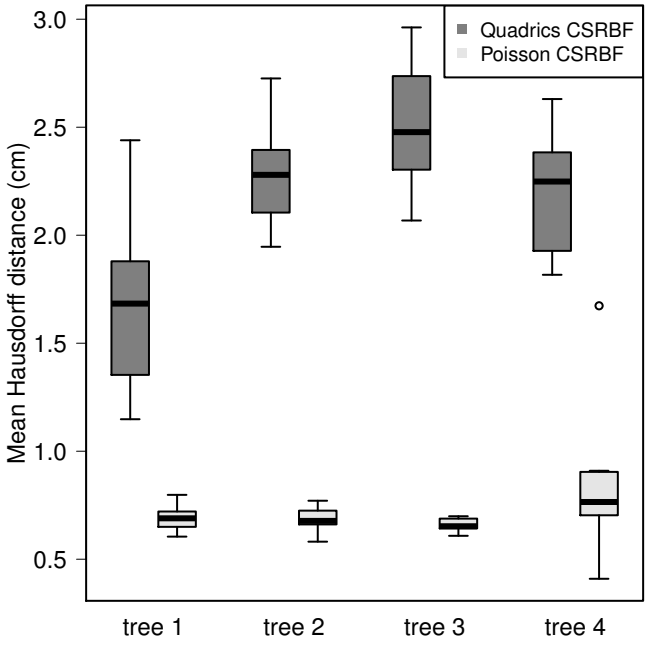


Figure 6.5: Mean Hausdorff distance between the mesh computed and the input point cloud for the two methods on the four trees

time due to the Monte Carlo integration needed to compute the matrices terms, our method have proven to be as efficient to rebuild the surface mesh of the Standford bunny, which is well described, with less noise and no hole. Indeed, the distance between the input point cloud and the reconstructed surface presented in Table 6.2 have the same magnitude for the two methods. Moreover, our method produces a slightly more detailed surface thanks to a smaller radius of the kernels. Indeed, the un-screened Poisson reconstruction surface produces a smoother and less detailed surface for the same octree refinement.

The second step of the validation was to evaluate the effect on occlusion on both methods. The results of the experiment presented in Figure 6.4 shows clearly the limit of the un-screened Poisson surface: for an angular occlusion higher than 20°, which correspond to hole of 35 cm on a cylinder of points of 1 meter radius, the method doesn't manage to close the hole. On the other hand, our method, guided by the shape of the tubular, handle occlusions, even for higher angular distribution upto 180°.

The last part of the validation was to measure the added value of our method in the modeling of tree and in the computation of their volumes. The model that our method produces was compared to the tubular shape based on quadrics and CSRBF described in Section 2.2. Overall, the proposed poisson-CSRBF-based approaches provided an improved approximation of the object

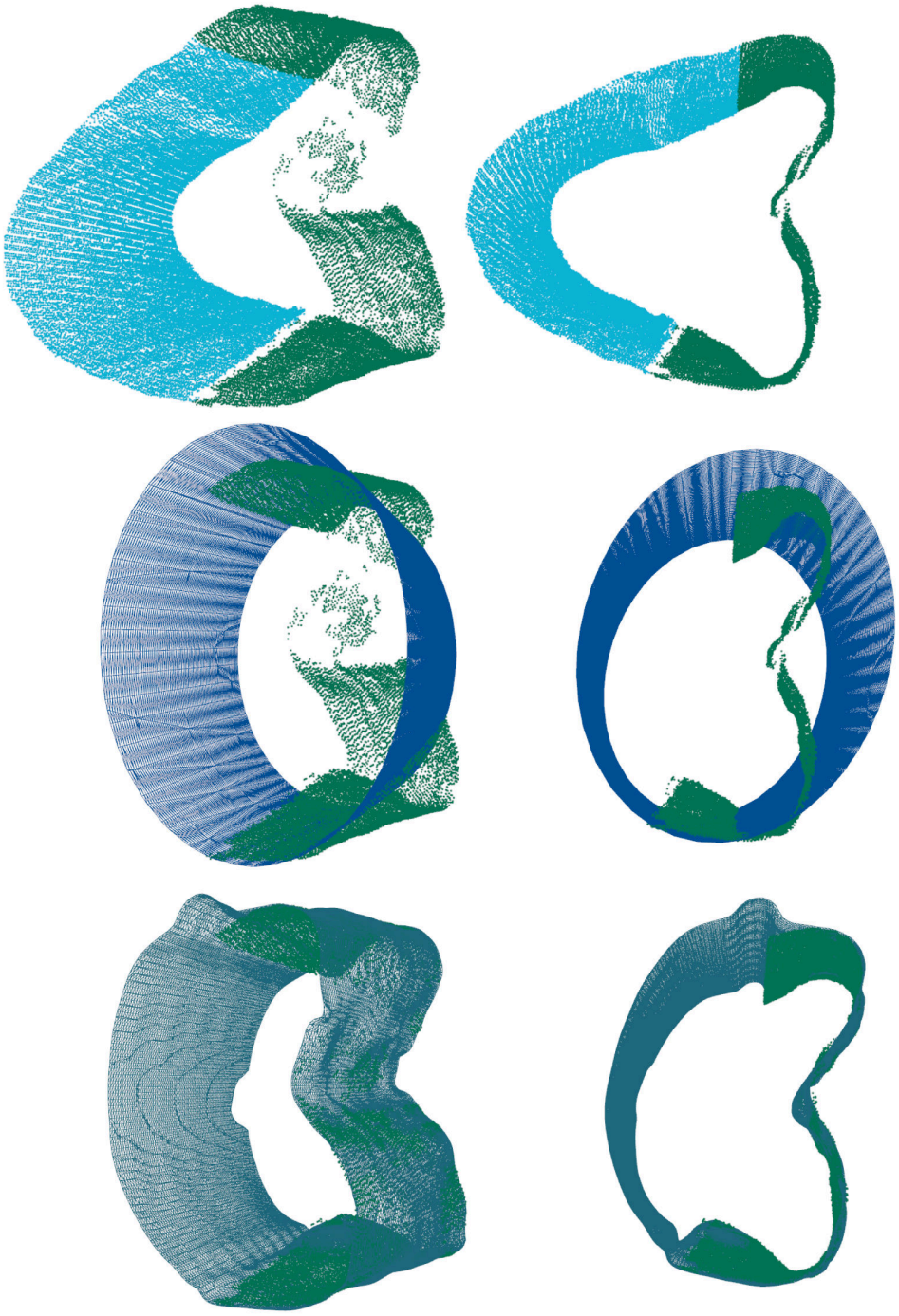


Figure 6.6: Close-up from two point of view of the reconstructed surfaces of a slice of point distributed on the lower part of a tree presenting buttresses: in the upper part, the input point cloud with deleted points in blue that artificially creates occlusion and the remaining points in green; in the middle part the surface reconstructed with the Quadrics and CSRBF method; In the lower part, the surface reconstructed with our Poisson reconstruction method based on CSRBF

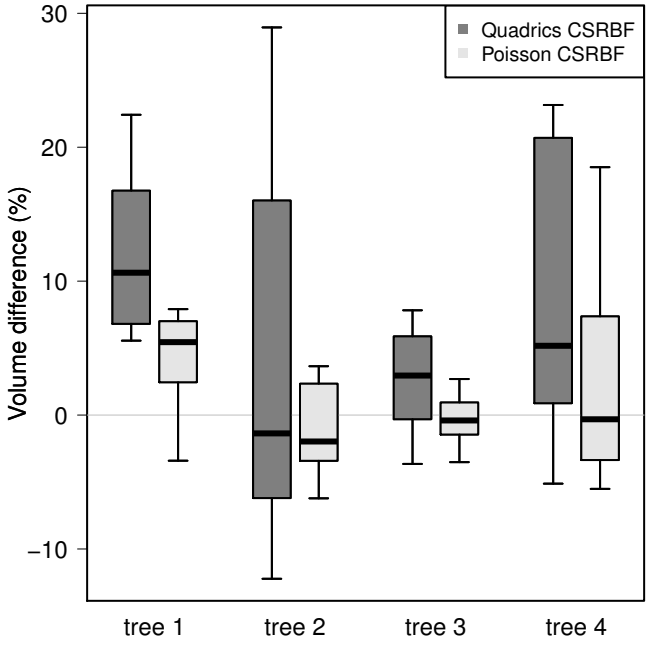


Figure 6.7: Normalized difference of volume between the estimation and the reference for the two methods tested on the four trees.

shape as well as the object volume. The method is particularly well suited to reconstruct irregular shapes, thus being more flexible to handle the diversity of shapes found in natural environments. The method would be particularly useful to reconstruct crown branches whose shape could be distorted due to gravity effect, as well as tree buttresses that remain difficult to model and could contain a significant portion of the tree volume and biomass. Figure 6.6 presents a example of surface reconstruction on an unfavorable case of point cloud distributed on the buttress at the base of a tree. While the tubular shape struggle to fit to the points, our method approximates correctly the shape of the buttress and keep the shape of the tubular in the occlusion. Figure 6.5 highlights the improvement on the tree modeling from the tubular shape to our Poisson surface reconstruction based on CSRBF. For the whole set of trees used in the validation, the tubular shape model stay over 1.5 cm on average. Our method reduce this distance to an average of 6 mm. That remaining distance between the model and the points is due to the smoothing of the implicit function described in Section 3.8, where we set aside input points and surface to handle complex tree shapes. Moreover, because of the due to point density shift, our method produces slightly less good results on the thinner trees, for instance the tree 4. This issues defines the two axis to improve the method: First, we will improve the occlusion detection to merge more wisely the surface coming from the resolution of the Poisson equation to the shape known a priori. Sec-

ond, we will implement a multi-scale approach to the Poisson surface reconstruction in order to better handle the variation of points density. By following the framework defined in[?] where the matrices are partially recomputed at each refinement iteration, we will build an adaptive solver to solve the Poisson equation that exploits the refinability of the space of functions. This will also lead to a better memory management and a faster computation.

6 Another approach: improvement of tree models with a convection model

Inspired by the level-sets formulations[140,146,119] in the field of image processing, this second method recovers the shape surface by propagating an interface represented by the zero level set of a smooth function, as introduced by Gelas[66]. In practice, the evolution of the interface is driven by a time-dependent partial differential equation (PDE) where the so-called velocity term reflects the 3D point cloud features. The level-set PDE is then solved through a collocation method using CSRBF. The next Sections present the computation of the velocity terms and introduce the convection evolution equation. Because of time constraints, the resolution of the evolution equation has not been achieved yet. Once completed and validated, the method will be submitted.

6.1 Defining the function space

Based on the space segmentation provided by the N building blocks of the QSM, we define a subdivision of the space $\{\mathcal{O}_1, ..., \mathcal{O}_N\}$, where each cell $\mathcal{O}_i$ contains the i^{th} building block. In each cell, we define a function space $\mathcal{F}_{\mathcal{O}_i}$ on which the level-set PDE is discretized. In practice, the function space construction follows the same process as in Section **??**. In each cell $\mathcal{O}_i$, we use an octree division of the space. This octree is the minimal octree so that every point falls into a leaf of a resolution fixed by the user. Then we force octree divisions close to the continuous tubular model shape by analyzing the sign of its implicit function at the eight corners of each leaf : a different sign for at least one of the corner of a leaf results in its subdivision. Finally, we refine this octree by allowing a maximum difference of level equal to one between a leaf and its neighbors to settle a smooth decrease of resolution. The function space is then defined as the span of translates of Wendland's CSRBF Φ_{r_o} centered in each leaf center $\mathbf{o}_c$ of the octree:

$$\Phi_{r_o}(\|\mathbf{q} - \mathbf{o}_c\|) = \phi(\frac{\|\mathbf{q} - \mathbf{o}_c\|}{r_o}) \tag{6.31}$$

where r_o is the support and depends on the depth of the leaf.

6.2 From continuous tubular model to CSRBF model

The first step of the method consists in stating the continuous tubular model in the base $Span\{\Phi_{r_o}\}$ as the linear combination $g(\mathbf{x}) = \sum_{\mathbf{o}_c} \alpha_{\mathbf{o}_c} \cdot \Phi_{r_{o_c}}(\|\mathbf{x} - \mathbf{o}_c\|)$. To do so, we consider the system :

$$\begin{bmatrix} \ddots & & \\ & \mathcal{A}_{\mathbf{o}_c,\mathbf{o}'_c} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \alpha_{\mathbf{o}_c} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ f(\mathbf{o}'_c) \\ \vdots \end{bmatrix} \tag{6.32}$$

where o_c,o'_c are the center of two octree leafs, $\mathcal{A}_{\mathbf{o}_c,\mathbf{o}'_c} =< \Phi_{r_{o_c}}(\|\mathbf{x} - \mathbf{o}_c\|), \Phi_{r_{o'_c}}(\|\mathbf{x} - \mathbf{o}'_c\|) >, \forall \mathbf{x} \in \mathbb{R}^3$ and the implicit function f, describing the tubular model is expressed in the base $Span\{\phi_{r_i}\}$ as:

$$f(\mathbf{x}) = \sum_{\mathcal{O}_i} g_i(\mathbf{x}) \cdot \phi_{r_i}(\|\mathbf{x} - \mathbf{c}_i\|) \tag{6.33}$$

where c_i is the center of the cell $\mathcal{O}_i$. It is important to note that $Span\{\phi_{r_i}\}$ differs from $Span\{\Phi_{r_o}\}$: the functions of $Span\{\phi_{r_i}\}$ are used to merge the cylindrical quadrics of the continuous tubular model by partition of unity whereas the base $Span\{\Phi_{r_o}\}$ is used to express our model as a linear combination of its functions.

Thanks to the properties of Wendland's CSRBF Φ_{r_o}, the matrix of the $\mathcal{A}_{\mathbf{o}_c,\mathbf{o}'_c}$ elements is self-adjoint and positive definite. Thus, the system is inverted by using the LDL^T Cholesky decomposition.

6.3 Solving the convection evolution equation using CSRBF collocation

We define now $\mathcal{V}(\mathbf{p}, t), \mathbf{p} \in \mathbb{R}^3, t \in \mathbb{R}$, a velocity function reflecting the geometrical properties of the interface according to the data, and quantifying the local deformations over time. Our collocation approach implies the evaluation of the velocity $\mathcal{V}(\mathbf{p}, t)$ at each CSRBF center o_c. Then, for each o_c, the velocity is computed by minization of an energy function inspired by the snake approaches introduced by Kass[85]. Indeed, we define the energy function $\xi(\mathbf{q})$, for each $\mathbf{q} \in \mathbb{R}^3$ to be the sum of two terms: $\xi_{data}(\mathbf{q})$ which gives rise to the sample points force and $\xi_{surface}(\mathbf{q})$ which

gives rise to the external constraint forces. A weighting scalar parameter γ, defined by the user, adjusts the influence of both terms.

$$\xi(\mathbf{q}) = \gamma \cdot \xi_{data}(\mathbf{q}) + (1 - \gamma) \cdot \xi_{surface}(\mathbf{q}) \tag{6.34}$$

Considering a set of points $\mathbf{p}_i, i \in [1, \ldots, N]$, a local fitting plane of normal $\mathbf{n}$, for each $\mathbf{q} \in \mathbb{R}^3$, the distance t between $\mathbf{q}$ and the plane, ξ_{data} and $\xi_{surface}$ are expressed as function of t and $\mathbf{n}$, at the point $\mathbf{p} = \mathbf{q} - t \times \mathbf{n}$:

$$\begin{aligned} \xi_{data}(t, \mathbf{n}) &= \sum_{\mathbf{p}_i} \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \frac{\mathbf{n}}{\|\mathbf{n}\|} \rangle^2 . \Im \left(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\| \right) \\ \xi_{surface}(t, \mathbf{n}) &= \langle \mathbf{x}' - \mathbf{q} + t \times \mathbf{n}, \mathbf{n}' \rangle^2 \end{aligned} \tag{6.35}$$

where $\mathbf{n}'$ is the gradient of the implicit function defining the continuous tubular model, and $\mathbf{x}'$ is the projection of $\mathbf{p}$ on the zero level-set of this function. For clarity sake, $\xi_{surface}$ quantifies the distance between the continuous tubular model and the *MLS* model[11] approximating locally the data. The gradient of ξ is then expressed as $\nabla\xi = \gamma \cdot \nabla\xi_{data} + (1 - \gamma) \cdot \nabla\xi_{surface}$.

Following equations 6.36 and 6.40 detail the computation of both data and surface gradient terms, respectively ξ_{data} and $\xi_{surface}$.

$$\begin{aligned} \nabla\xi_{data}(t, \mathbf{n}) = \quad & \sum_{p_i} 2 \cdot \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \frac{\mathbf{n}}{\|\mathbf{n}\|} \rangle \cdot \nabla \langle \ldots \rangle \cdot \Im \left(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\| \right) \\ & + \langle \mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}, \tfrac{\mathbf{n}}{\|\mathbf{n}\|} \rangle^2 \mathcal{M}^T_{(t,\mathbf{n})} \cdot \nabla \Im \left(\|\mathbf{p}_i - \mathbf{q} + t \times \mathbf{n}\| \right) \end{aligned} \tag{6.36}$$

where $\nabla\langle \ldots \rangle$ is computed as:

$$\nabla \langle \ldots \rangle = \begin{bmatrix} \frac{1}{\|\mathbf{n}\|} \cdot (I_3 - \frac{\mathbf{n} \cdot \mathbf{n}^T}{\|\mathbf{n}\|^2}) \cdot (\mathbf{p}_i - \mathbf{q}) + t \cdot \frac{\mathbf{n}}{\|\mathbf{n}\|} \\ \|\mathbf{n}\| \end{bmatrix} \tag{6.37}$$

$\mathcal{M}_{(t,\mathbf{n})}$ is a $3x4$ matrix expressed as:

$$\mathcal{M}_{(t,\mathbf{n})} = \left[\begin{array}{c|c} & n_x \\ I_3 \cdot t & n_y \\ & n_z \end{array} \right] \tag{6.38}$$

and considering ϑ as a Wendland's CSRBF $\vartheta(r) = (1-r)^4 \cdot (1+4\cdot r)$, $\nabla\vartheta$ is expressed as:

$$\nabla\vartheta(r) = (-20\cdot r)\cdot(1-\|r\|)^3 \cdot \frac{r}{\|r\|} \tag{6.39}$$

The second gradient term is computed as:

$$\nabla\xi_{surface}(t,\mathbf{n}) = 2\cdot\mathcal{M}^T_{(t,\mathbf{n})}\cdot\langle\mathbf{x}' - \mathbf{q} + t\times\mathbf{n}, \mathbf{n}'\rangle\cdot\mathbf{n}' \tag{6.40}$$

Then, we proceed to a multidimensional minimization that requires finding a point $(t,\mathbf{n})$ such that the scalar function $\xi(t,\mathbf{n})$, defined in Equation 6.34, takes a value which is lower than at any neighboring point. A one-dimensional line minimization is performed along the direction $\nabla\xi(t,\mathbf{n})$ until the lowest point is found to a suitable tolerance. The search direction is then updated with local information from the function and its derivatives, and the whole process repeated until the true minimum is found. In practice, we retrieve $(t_{min},\mathbf{n}_{min})$, minimizing $\xi(t,\mathbf{n})$ with the Fletcher-Reeves[65] conjugate gradient algorithm available in the GSL library and initiated to $(t_0,\mathbf{n}_0) = (0,\mathbf{n}')$.

From this minimization process computed at each collocation center arises a deformation vector defined as $\mathbf{v}_{o_c} = -t_{min}\times\mathbf{n}_{min}$. Figure 6.8 illustrates an example of the deformation vector field induced by a slice of 3D points distributed on a tree trunk.

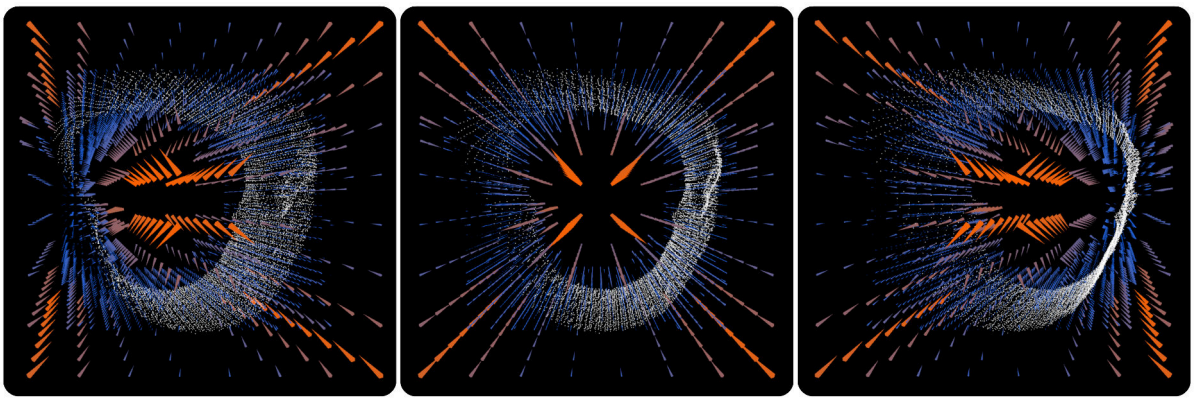


Figure 6.8: Deformation vector field, colored according to the intensity, computed on a slice of 3D points (in white) distributed on a trunk.

To push the surface to the data points according to the deformation vector field, we propose to follow a convection model as proposed by Osher[119]:

$$\frac{df(\mathbf{p},t)}{dt} = \mathbf{v}_\mathbf{p}\cdot\nabla f(\mathbf{p},t) \tag{6.41}$$

where $\mathbf{v_p}$ is the deformation vector expressed at $\mathbf{p}$.

To solve Equation 6.41, we assume that space and time are separable, we decompose $f(\mathbf{p}, t)$ as the product of a line vector $\Phi(\mathbf{p})$ and a column vector $\mu(t)$:

$$f(\mathbf{p}, t) = \Phi(\mathbf{p}) \cdot \mu(t) \tag{6.42}$$

Injecting 6.42 into 6.41 gives the ordinary differential equation of evolution:

$$\Phi(\mathbf{p}) \cdot \frac{d\mu(t)}{dt} = \mathbf{v_p} \cdot [\mu(t) \times \nabla\Phi(\mathbf{p})] \tag{6.43}$$

After the application of Euler's method, Equation 6.41 becomes:

$$\mu(t + \tau) = \mu(t) - \tau \cdot \Phi(\mathbf{p})^{-1} \cdot (\mathbf{v_p} \cdot [\mu(t) \times \nabla\Phi(\mathbf{p}))] \tag{6.44}$$

where τ is the time step.

Thanks to the properties of Wendland's CSRBF, the matrix $\Phi(\mathbf{p})$ is sparse, selfadjoint and positive definite. Thus, $\Phi(\mathbf{p})$ is inverted by using the LDL^T Cholesky decomposition implemented in Eigen library. The evolution of the CSRBF coefficients is finally given by Equation 6.44.

In practice, a convection evolution problem is defined in each subdivision $\mathcal{O}_i$. Once each convection problem has been solved, we combine the set of CSRBF corresponding to each subdivision $\mathcal{O}_i$ to build a global model describing the whole tree. From this global set of CSRBF we compute an implicit function and finally extract its zero levelset to produce the tree surface model.

6.4 Complexity

The complexity of the various steps of our approach is detailed as:

- Using an octree data structure for the collocations layout, as advised by Wendland[155], the matrix $\Phi(\mathbf{p})$ computation is $\mathcal{O}(NlogN)$ and the implicit function evaluation f is $\mathcal{O}(logN)$.

- According to Botsch[25], the inversion of $\Phi(\mathbf{p})$ through the Cholesky factorization is $\mathcal{O}(nzf)$, where nzf is the number of nonzero factors, which depends on the CSRBF center position and on the CSRBF support size.

6.5 Comparison of the tree models

While we showed previously promising results of our Poisson reconstruction surface guided by a model known a priori, it obviously needs improvements to be used efficiently in the computation of tree model. Indeed, it relies on several «small»Poisson surface reconstructions, one for each building block of the QSM it uses as a guide, and then becomes un-affordable at the tree scale. Nevertheless, if the input point cloud presents minor occlusions, and describes faithfully the structure of the tree, the usefulness of the QSM guide decreases. In that case, leaving the QSM guide out and running the Poisson surface reconstruction on the point cloud describing the woody part produces a surface model of the tree in a timely manner.

In that context, we compare our modeling and the volume it generates to three different approaches: (i) the field measurement where the tree is chunk into logs and then weighted, (ii) the allometric model which establishes quantitative relations between the height, diameter and the volume of the tree, and (iii) the quantitative structure modeling previously defined. In practice, using each of these methods, we compute the volume of four tropical trees of different species from different architectural groups: *Terminalia superba* (common name: Fraké), *Triplochiton scleroxylon* (common name: Ayous), *Pycnanthus angolensis* (common name: Ilomba) and *Erythrophleum suaveolens* (common name: Tali).

For each method, the volume were estimated as follow:

- Concerning the volume estimation derived from the destructive measurement of the field campaign, the volume itself is not directly available. Actually, once the trees were felled at ground level, their biomass was measured by weighting small branches (diameter below 20 cm) and by approximating the bigger parts from geometrical considerations[77,63]. Using the average wood density sampled on the different part of the tree, we can then express the volume of the tree as:

$$V_{Field} = AGB_{field}/\varrho_{field} \tag{6.45}$$

  where AGB_{field} (kg) is the measured biomass and ϱ_{field} is the average measured density.

- Regarding the allometric approach, we estimated the volume following Chave[39] equation that gives the above ground biomass as a function of height, diameter and density:

$$AGB_{est} = 0.0559 \cdot \varrho \cdot D^2 \cdot H \tag{6.46}$$

where AGB_{est} (kg) is the biomass, ϱ $(g \cdot cm^{-3})$ is the density, D (cm) is the diameter, and H is the height of the tree. After adjusting the dimensions, the volume can be expressed as:

$$V_{Allometric} = 0.559 \cdot D^2 \cdot H \tag{6.47}$$

For each tree, we consider a slice of 10 cm of points above the buttresses and project it on the xy plane. From $P_{textconvex-hull}$, the perimeter of the convex-hull of the resulting 2D points, we estimate the diameter as $D = P_{textconvex-hull}/\pi$. We choose to deduce D from the perimeter rather than by circle fitting to stick to the field reality, where diameters are estimated 50 cm above the last buttress using a tape meter surrounding the trunk.

- With regard to the estimation through QSM, the volume is computed as the sum of the volume of each cylinders of the model. QSM models were produced using SimpleTree, a C++ based implementation of a method building tree models developed by Hackenberg[73]. This software proposes a way to enhance the model by correcting the vertical scrolling of diameter evolution with the use of an allometric equation. Considering the «growth volume» for each cylinder, computed as the volume of the given cylinder plus the «growth volume» of its child, a model of the form $y = a \cdot x^b$ is fitted against the radius of the cylinders. The radii of the sequence of cylinders is then corrected according to this fit.

- To compute the volume enclosed in the continuous surface arising from our Poisson surface reconstruction, we followed the method described by Zhang[157]: we summed the signed volume of the tetrahedron formed by the center of gravity of the mesh and each of its triangle.

Figure 6.9 and 6.10 present the point clouds describing the trees selected for this comparison, their section at breast height (1.3 m) and also the section picked for their diameter estimation. They also display the 3D QSM and the corresponding surface reconstructed with our Poisson method. Figure 6.11 compares the 3D model precision on both *Terminalia superba* and *Pycnanthus angolensis*.

Because of the limited size of the testing sample, the volume estimated in Figure 6.12 are given for information purposes only. By comparing them to field volume methods, we give an idea of the performance of state of the art methods which compute tree volume using LiDAR. Even though no statistical conclusion on the methods efficiency can be deduced from such comparison, it is interesting to note the strong difference on the volume assessment computed by each

methods. The reconstruction of complex geometry can be hardly validated with conventional field measurements, that do not focus on estimating a shape but rather a volume by doing cylindrical approximation because of practical reasons and financial matters. Setting out from this premise, simulation stands as the most relevant way to validate the method. This observation highlights the need of a controlled simulation environment producing complex tree geometrical model with required details, adapted buttresses and surface defaults. Based on AmapSim[16] of UMR AMAP laboratory or on the models produced by L-Architect[44] for instance, the simulator would produce complicated tree model with a known volume, evolving from a defined disturbance level.

7 Conclusion

Our Poisson surface reconstruction approach with CSRBF has proven to be effective to handle point clouds imperfections in order to produce accurate geometric model of the woody part of trees. Several improvements, that are clearly identified, need to be implemented to complete the method and make it usable in production conditions. The future development will enhance the method in order to handle point density shift by following a multi scale approach which will also speed up the resolve of the Poisson equation. We plan to manage holes in the data more ingeniously in order to better control the transition of the model between visible and occluded are. On the other hand, encouraging results arise from the second method dedicated to the enhancement of tree model: the deformation vector fields produced are consistent. Nevertheless, further work is required to finalize the convection evolution equation solving. Once completed, both methods will be compared in terms of accuracy and computation time.

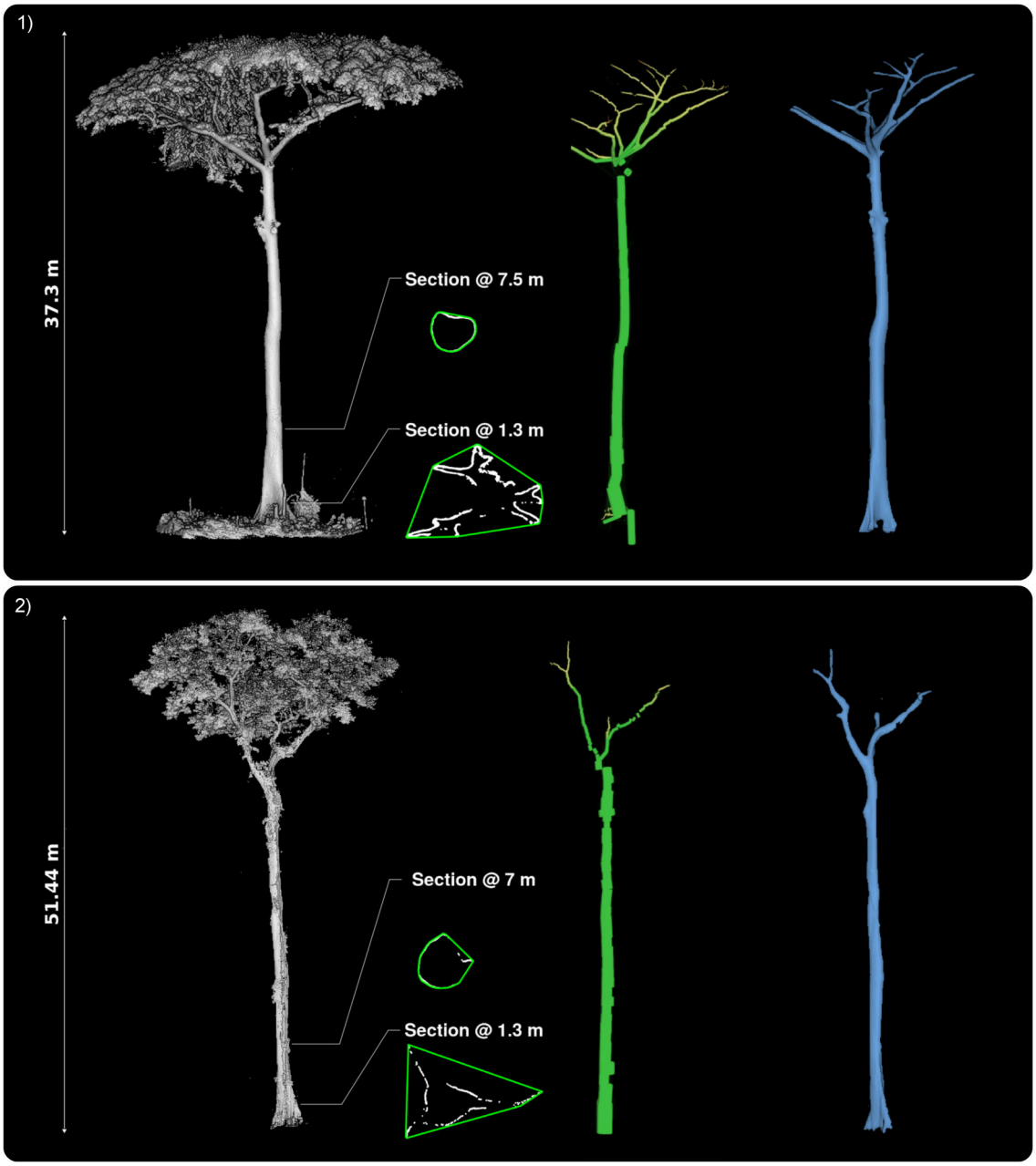


Figure 6.9: Illustration of the different models for the *Terminalia superba* (top) and the *Triplochiton scleroxylon* (bottom). For each tree is plotted, (left) the point cloud along with the section of points at 1.3 m and above the buttresses, (center) the QSM in green and (right) the Poisson surface in blue.

Figure 6.10: Illustration of the different models for the *Pycnanthus angolensis* (top) and *Erythrophleum suaveolens* (bottom). For each tree is plotted, (left) the point cloud along with the section of points at 1.3 m and above the buttresses (The Tali have no buttress), (center) the QSM in green and (right) the Poisson surface in blue.

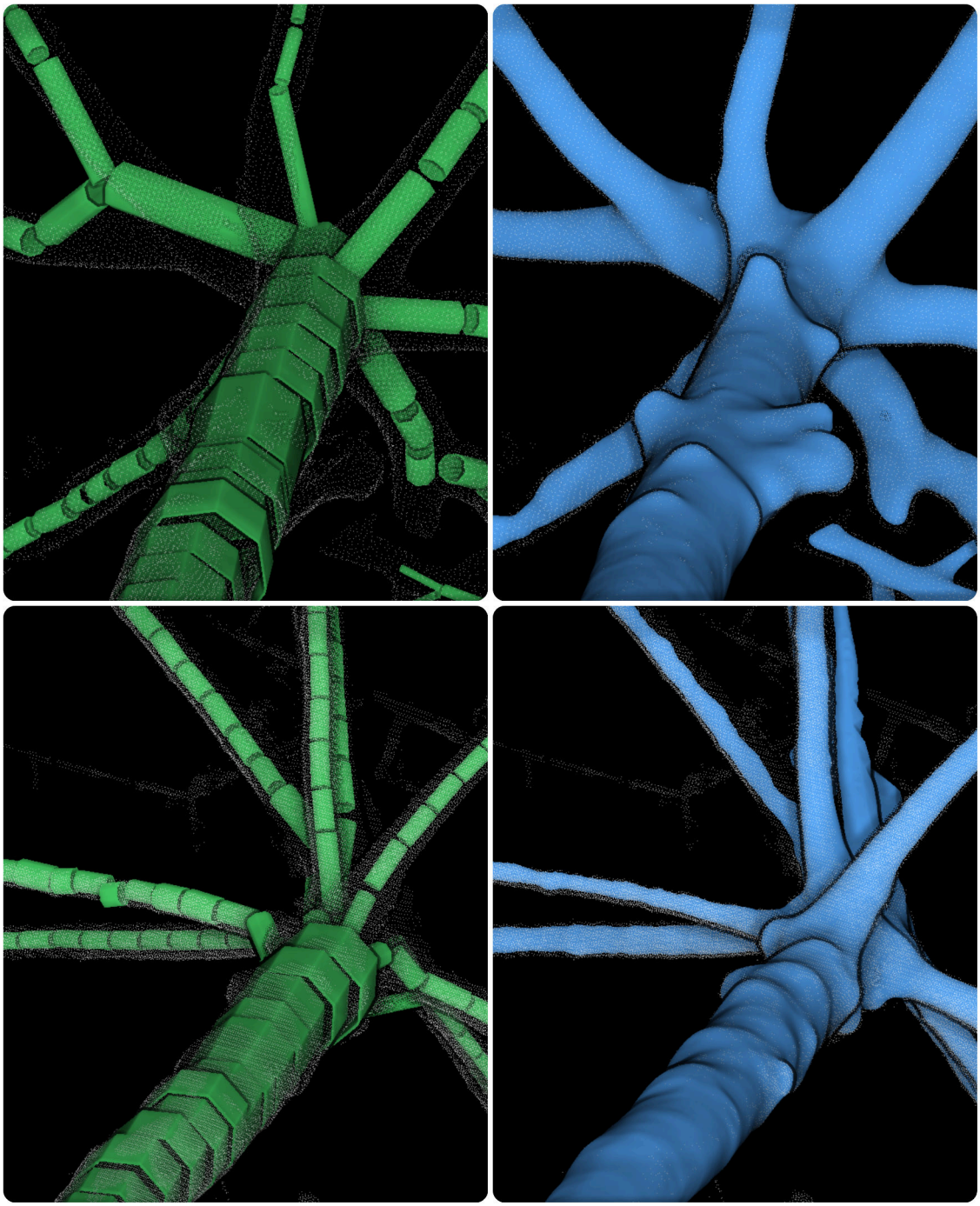


Figure 6.11: Zoom on QSM and Poisson models for the *Terminalia superba* (top row) and the *Pycnanthus angolensis* (bottom row). Each plot represents the point cloud describing the woody part of the trunk, the QSM model (green) and the Poisson surface (blue).

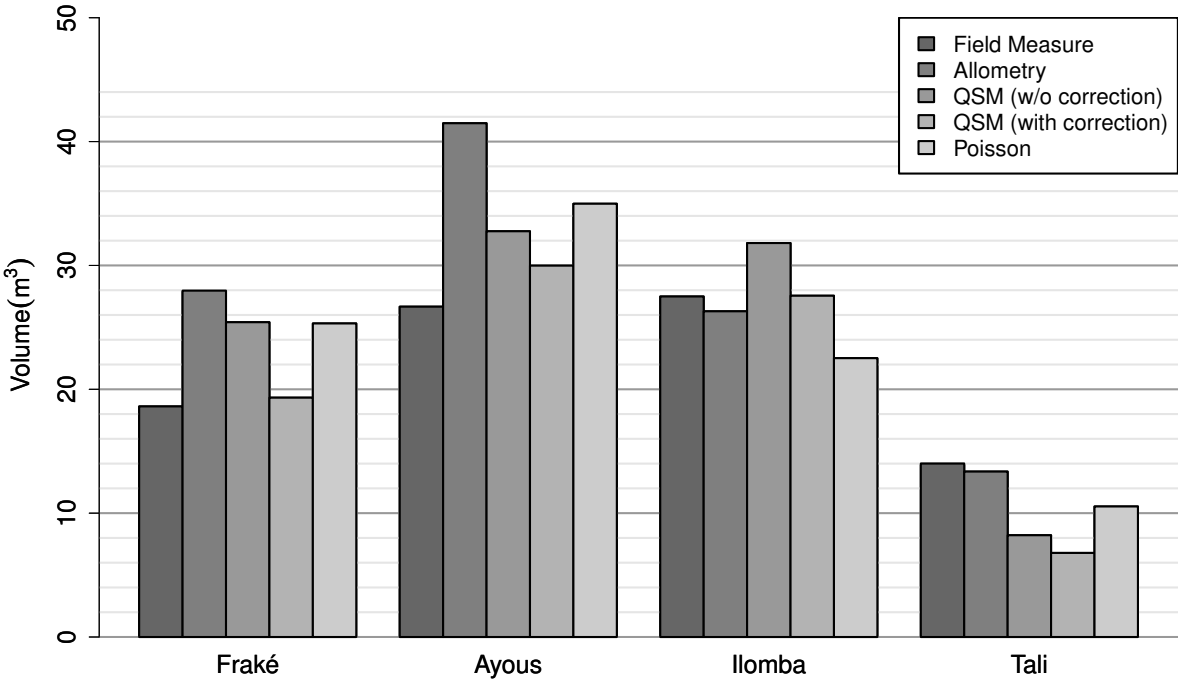


Figure 6.12: Comparison of tree volume computation on four different individuals.

7

Conclusion and perspectives

Extracting knowledge from TLS point clouds acquired in forest environments involves dealing with objects of different natures and geometrical properties, such as the ground, the tree trunks and the leaves. In the past, ground and trunk modeling have been investigated through independent and unrelated approaches. In this thesis, we investigated the modeling of dense 3D point clouds describing forest environments through implicit functions. The method framework was designed to be flexible and the various tools developed were integrated under a fully automated processing line addressing both terrain and tree structure modeling.

The flexibility of the method takes advantage of the huge diversity of modeling framework allowed by RBF. For ground modeling, the implicit model was guided by a dynamic, multi-resolution quadtree division of space, allowing for dealing with open surfaces. For closed surfaces, such as tree trunks, the quadtree structure was replaced by quantitative structure models of trees, which can be derived from multiple approaches such as Hough transform or cylinder fitting among others.

Overall, our results demonstrated the usefulness of such a family of approaches for modeling forest scenes.

The modeling of terrain surfaces was found to be robust for reconstructing large occlusion gaps and was not affected locally by the presence of objects at the ground level. Interestingly, the quadtree division is also a very convenient tool for generalizing the ground surface and computing terrain properties according to either the local ground complexity or the required level of detail. While the diversity of plots used to validate the method was quite high, the number of plots remained limited. Also, it would be interesting to validate the method on a higher number of plots to run a complete sensitivity analysis and to ensure its effectiveness. Also, if the approach has been developed for TLS point clouds, we considered that the underlying modeling principles are generic enough to be applied to low density point clouds such as those generated from airborne data. To do so, we planned to test the algorithm on the bench-marking data-base provided by the ISPRS (http://www.itc.nl/isprswgiii-3/filtertest/, last consulted October 31, 2016). In its current version, the quality of the modeling depends on a filter whose performance relies on the point density and distributional properties. The effectiveness of the method and its generic nature might thus be improved by a deeper analysis of the histogram of the point distribution and by a refinement of the neighborhood filtering. Indeed, an in-depth study of the histogram modes would lead to a better screening of the vegetation and a detection of local continuity would also have significant positive repercussions.

While the application framework of this thesis was the processing of forest TLS data, this algorithm can be easily generalized and adapted to recover surfaces from 3D point clouds in other scopes. For instance in the field of archaeology or architecture, the method would lead to interesting results in the processing of point clouds produced by photogrammetry, structure from motion or RGB-D camera describing bas-relief of cultural heritage monument. Indeed, similarly to TLS point clouds acquired in forests, such point clouds present noise and outliers that would be smoothed by our method filters. Moreover, our specific modeling relying upon the stitching of surface patches would help to fill the holes.

At the tree level, we investigated three different approaches to tree reconstruction. While current state of the art methods summarized trees as a list of geometrical primitives, such as circles (Hough) or cylinders (QSM), our method used these outputs to guide a reconstruction approach locally approximating the point structure and providing more realistic of the tree shape. Chapter 5 introduced a model based on quadratic local approximations blended together with compactly supported radial basis functions using the partition of unity concept, in order to retrieve a global continuous surface model approximating the tree shape. The late developments on this

method focus on describing accurately the branches junctions by encoding the insertions. The forestry application of this modeling are relevant and needs to be developed. We invoked the weaknesses and limits of the current modeling through discontinuous stacks of cylinders; our modeling stands as a more efficient alternative without being more complex.

Chapter 6 provides an improved modeling approach. While the model introduced in Chapter 5 built upon QSM to overcome occlusions, it is defined as an implicit function and so can be further improved. Consequently, we presented two new methods that enhance the modeling. The first one injects the tubular model in a Poisson surface reconstruction method to retrieve a closed surface even in the case of occlusions. The second one uses the tubular model as the starting point of a deformable model to push the surface model towards the data points.

Our results demonstrated that Poisson surface reconstruction is a very convenient approach to model the variability of tree shapes. Compared with the QSM approach, our modeling allows for providing unbiased approximations of tree volume and it provided the amount of detail necessary to give insights on the wood quality for industry. Also, the Poisson approach can be used without any QSM support in the absence of large occlusions in the point cloud making it interesting for modeling different kinds of object shape. The formalism introduced to constrain the Poisson surface reconstructions by an implicit function describing a shape known *a priori* could be further improved. From our experience, we think that it would be interesting to investigate new locations for the centers data structure, for example using a multiscale approach which will bring higher precision and efficiency to set up and solve the Poisson equation. This would lead to a better discretization of the numerical scheme and a faster solving of the problem.

The opposite approach, consisting in the deformation of the model known a priori through a data driven convection flow, has been barely explored. However, it is also promising and its completion is part of our future research. The comparison of the two methods in terms of accuracy and computation time would lead to interesting results.

During the development of our methods, and especially during their validation, we realized the need for a controlled simulation environment. Being able to generate complex tree shapes presenting buttresses and surface defaults as a closed surface mesh would help to assess the performance of the different reconstructing approaches. On the basis of existing plant simulators and growth model of the literature, which produce accurate architectural skeletons of trees, our modeling based on implicit surface could be used to deform and complicate the geometrical tree

shapes in order to reflect imperfect, and so more realistic tree models.

Finally, building on its success and the positive feedback of the community, we intend to pursue and further the research and development of the Android application «LiDAR VR Viewer». Initially designed to display forest point clouds and meshes, this application appeared to be used in different fields. Indeed, it is actually generic and allows to load any point clouds and surfaces: for instance it is used by professionals in the construction industry for the immersive visualization of 3D buildings, or also by VR film makers to get an insight of the VR world before the final rendering. The future developments will improve the VR showcase of our work on forests and will benefit at the same time to users from various scopes.

A

Android application «LiDAR VR Viewer»

During the finalization and testing of our algorithms, the quality of our reconstructed models happened sometimes hard to asses visually: flooded in middle of the data samples, the surface can be partially covered and also can itself hide points. The trend on virtual reality (VR) raised in the past few years in the scientific and technological worlds brought the concept on visualizing our data in an immersive manner. The introduction in June 2014 of the Google cardboard, a low-cost device that uses a smart-phone screen and sensors to turn into a virtual reality (VR) headset, ended up convinced us to produce a VR viewer to display our input/ouput in a precise and entertaining way. Using the Cardboard software development kit and Android studio, we developed «LiDAR VR Viewer»and released it on the Google Play Store. It counts today over 3500 downloads for a rating of 4.6/5.

1 An Android application to visualize in VR point clouds and meshes

Given the success of the "LiDAR VR Viewer" application, we prepared a short article to introduce its features. It is presented here and will be submitted to WSCG conference in the near future.

An Android application to visualize point clouds and meshes in VR

Jules Morel
IFP, LSIS
11 saint louis street
605001 Pondicherry,
India
jules.morel@ifpindia.org

Alexandra Bac
LSIS
163 AV. de Luminy
13288 Marseille, France
alexandra.bac@univ-amu.fr

Cédric Véga
IGN
14 rue Girardet
54000 Nancy, France
cedric.vega@ign.fr

ABSTRACT

This paper presents an Android application dedicated to the visualization of point clouds and surfaces as an immersive experience through a virtual reality head mounted display. Using the headtracking sensors of the smartphone, a generic Bluetooth controller and a virtual reality headset, this application has proven to be a powerful tool to investigate and explore 3D point clouds and meshed surfaces as a virtual reality environment.

Keywords

Point clouds, surface meshes, Visualization, Virtual reality, Android application

1 INTRODUCTION

The increasing ease of use and popularity of 3D acquisition entails a wider and wider use of digital represantations. Computer graphics play a major role in this development, providing modeling, recognition and analysis of 3D data. Acquisition techniques range from active techniques such as LiDAR scanner, motion sensing input devices like Microsoft Kinect, as well as passive ones such as multi-view stereo and surface from motion. They somehow capture 3D point cloud sampling the real world. Surface reconstruction algorithms build upon such data to produce watertight surface usually represented as mesh models. In order to visualize such point clouds and asserted reconstructed surface, computer scientists and professionals rely on dedicated visualization tools using Oculus Rift or other similar virtual reality (VR) headsets. However, those devices are expensive and need to be powered by high end desktop PC. The present paper presents an effective alternative to such costly virtual reality solutions : "LiDAR VR Viewer", an Android application, freely available on the Google Play Store (over two thousand downloads and rated 4.5/5), immersing the user into a virtual reality made of 3D point clouds and possibly reconstructed surfaces as mesh models. It stands as an affordable VR experience with the use of simple head-mounted displays such as Google cardboard while offering a possible navigation into the data by the mean of a bluetooth controller.

The paper is organized as follows. Section 2 presents the design of the application, its user interface, and summarizes its screens lifecycle, Section 3 highlights advanced features implemented to improve the rendering and the overall VR experience. Section 4 presents the performance of the application and few examples of in-app screenshots. Finally, further developments are discussed in Section 5.

2 DESIGN

The application has been developped following Android developper best practices proposed by Google. It is made of three different screens (called Activity among Android developpers) as shown in Figure 1. The first one, appears when the application starts and presents the settings that allow users to modify the application features. Then, loading is done asynchroneously and a format check on the file is performed. When the data file is not conform, an error message is displayed and the first activity is reseted. Finally, once the data are loaded, the last activity shows up as an OpenGL canvas to render the 3D points and meshs.

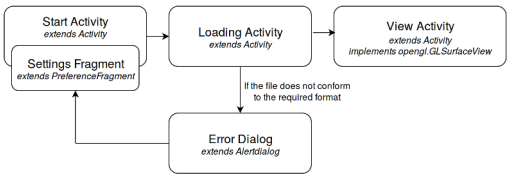


Figure 1: Application activity lifecycle

The user can access several options that define the application behavior and the rendering of data, as shown in Figure 2. Under the "Display" section, the user can define the rendering: an immersive mode, dedicated to smartphones used with a reality head mounted device, or a classic mode, to use the device as a window on the virtual world of its data. Under the "Point cloud" section, one can setup the file containing the points and

the parameters for their rendering such as their colors or sizes. The path to the file containing the mesh is defined under the section "Mesh". The inputs for the camera movement are defined under the "Camera motion mode" section.

3 FEATURES

The following sections give details on the main features we chose to implement in OpenGL shading language [3, 6] to improve the visualization while keeping a flowing experience. Actually, this was the main challenge: to provide an immersive experience through devices with low computing power.

3.1 Immersive mode

As shown on the Figure 3, the application proposes two rendering modes : 3a) a simple non-immersive mode designed to be used without VR headset as a window on the world of the data, where the navigation is done by using the touchscreen, and 3b) an immersive mode dividing the screen in two parts, and designed to be used with a VR headset to reproduce the 3D effect with double-view stereo. In the last mode, chromatic aberration and distortion induced by the lenses of the VR device are corrected.

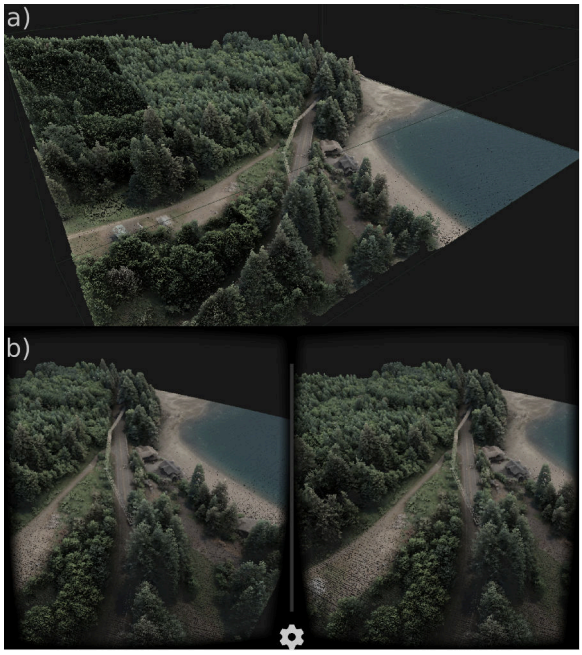


Figure 3: Rendering in classic mode (a) and immersive mode (b) of an aerial LiDAR point cloud (The point cloud contains 2 million points, and has been colored using aerial imagery and illuminance computation pre processing)

3.2 Optimization of buffers

The complexity of our scenes with millions of points and triangles on lightweight devices requires the use of

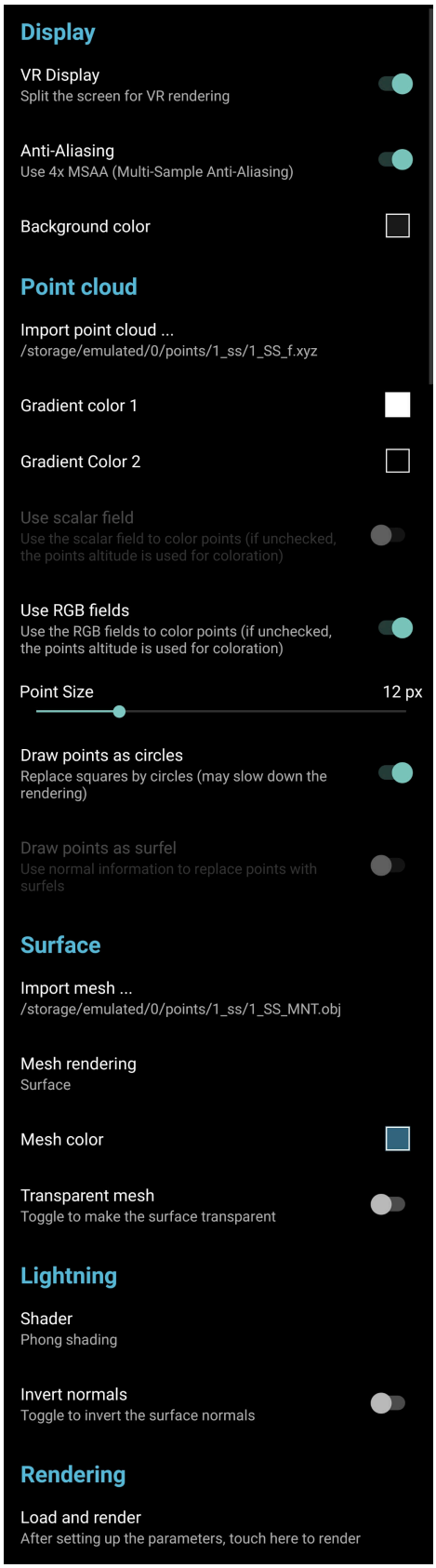


Figure 2: Graphical user interface of the scrollable parameters activity. It allows the user to setup the rendering of the 3D data.

vertex buffer objects. Instead of transferring vertex information from client memory at every frame, the information is transferred once and rendering is then performed directly from the graphics memory cache. Plus, instead of using a separate buffer for positions, normals and colors, we implemented packed buffers storing all of this data. Packed buffers have proven more efficient for GPU rendering as all the informations needed to render an object are located in the same block of memory. One should note that because of Froyo's broken OpenGL ES 2.0 bindings, the targeted plateform are devices running Gingerbread (2.3) and higher. This seems reasonable as the application requires a GPU recent enough to handle the graphic load.

3.3 Meshes and points rendering

To improve the rendering of surfaces, we shade them according to a partial Phong reflection model [5]. As specular reflection needs several computation in the fragment shader, only ambient and diffuse reflection were implemented to keep a flowing experience on highly detailed models.

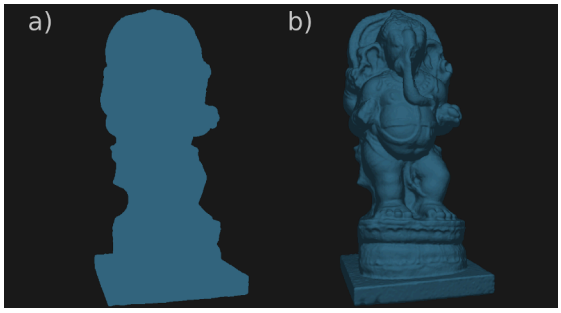


Figure 4: Rendering of a mesh (a) with ambient lightning only (no diffuse lightning) and (b) with partial Phong lightning.

Moreover, the application allows to chose between two rendering modes for the mesh, either as a smooth surface composed of a set of triangles or as a wireframe to improve visualization when needed. Our procedure for wireframe drawing is based on [1]. By providing the barycentric coordinates of the vertex to the fragment shader, we compute for each fragment the shortest distance to the edges of the polygon. Then, by setting a threshold on that distance and using alpha testing, triangles edges are drawn directly as a part of triangle rasterization, without resorting to line primitive, and so without any performance hit.

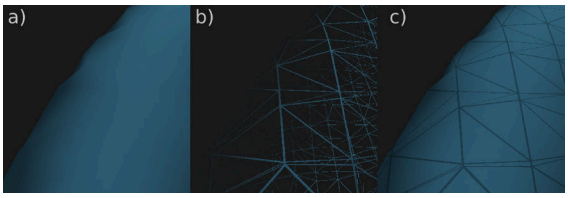


Figure 5: Close up on the rendering of mesh as smooth surface (a), as a wireframe (b) and as a surface with superimposed wireframe(c).

Similar lightning methods have been explored for the point clouds, especially the eye-dome lightning shadding technique proposed in [2], but such shaders involve too costly computation for smartphones or tablets GPU resulting in big drops of FPS. Indeed, provided the large size of handled point clouds and the double computation rendering for immersive mode (one for each eye), GPU cannot ran more than really simple shaders. Nevertheless, preprocessing illuminance computation drives to consistant FPS ratio together with a crisp and clear rendering (on Figure 3, RGB values on each point are scaled with the illuminance factor computed with a generalization of [7]).

3.4 Control management

As the application rendering is based on OpenGL, the concept of camera motion is not directly available. Nevertheless we simulate such a motion by moving all objects of the scene in the reverse direction, giving the illusion that user is moving. To define a camera we need its position in world space and a local coordinate system made of three vectors: the direction in which the camera is pointing, a vector pointing to the right and a vector pointing upwards from the camera.

The coordinate system formed by the camera direction, the right and up vectors are given by the Android SDK as a view transformation matrix linked to the accelerometers and gyroscopes output of the smartphone and updated at every frame. By applying this matrix to the world coordinates, we obtain vertex coordinates as seen from the camera's perpective at every frame, thus allowing the application to have a flowing headtracking of the camera direction.

To define the camera position, the user has three options. In classic mode, the camera can be moved forward and backward by using the touchscreen : a swap from bottom to top of the screen move the camera forward and a swap from top to bottom bring it backward. In immersive mode, the user can trigger the "motion mode" by using the magnetic trigger. In that mode, the camera position is updated at every frame according to the camera direction. The user can move in the world by looking at where he wants to go. Finally, the most precise option to control the camera is by using a

bluetooth generic controller. Moreover, the controller allows to modify several rendering options on the go. The mapping of the buttons is given in Figure 6.

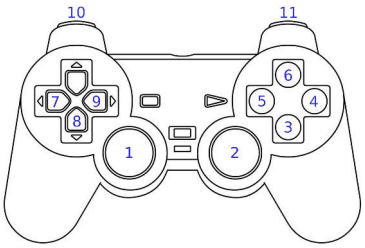


Figure 6: Button mapping on bluetooth controller : 1) move forward/backward and strafe leftright, 2) (if available) : go up/down, 3) go down, 4) go up, 5) reduce velocity, 6) increase velocity, 7) toggle points rendering, 8) switch mesh rendering between surface and wireframe, 9) toggle mesh rendering, 10) reduce points size and 11) increase points size

4 EXAMPLE OF RENDERING AND PERFORMANCE

Figure 7 shows examples of rendering through the non-immersive mode of the application. It highlights, amongst others, the different coloration of points implemented : 7a) a color ramp scaling with a scalar field, 7b) a color ramp scaling with the altitude of the points or 7c) RGB values. Figure 7d) displays an airborne LiDAR dataset colored with the altitude of the point and Figure 7e) displays a point cloud constructed with the structure from motion method; it describes a temple from south India.

Figure 8 shows an analysis of the performance of our application ran on an Adreno 330 GPU with Qualcomm Snapdragon 801 2.5 GHz CPU on a 720p screen. It presents the average framerate with respect to the number of sample points rendered. According to this result, for the given hardware, users can load point clouds with upto 2 millions points while keeping a flowing experience.

5 CONCLUSION

The optimization of the OpenGL rendering allows our application to maintain a descent 30 fps framerate for well detailed 3D data on lightweight devices. It is not enough to completly avoid motion sickness, but it is descent enough to propose a clear, flowing and affordable VR experience. The new trends in development and the evolution of the technology will lead to much better performance in the near future. Indeed, the upcoming generations of smartphones with more powerful GPU promises enhancement of this virtual reality experience. The current developments focus on implementing a level of details (LOD) approach for the data managment. By following an octree division of

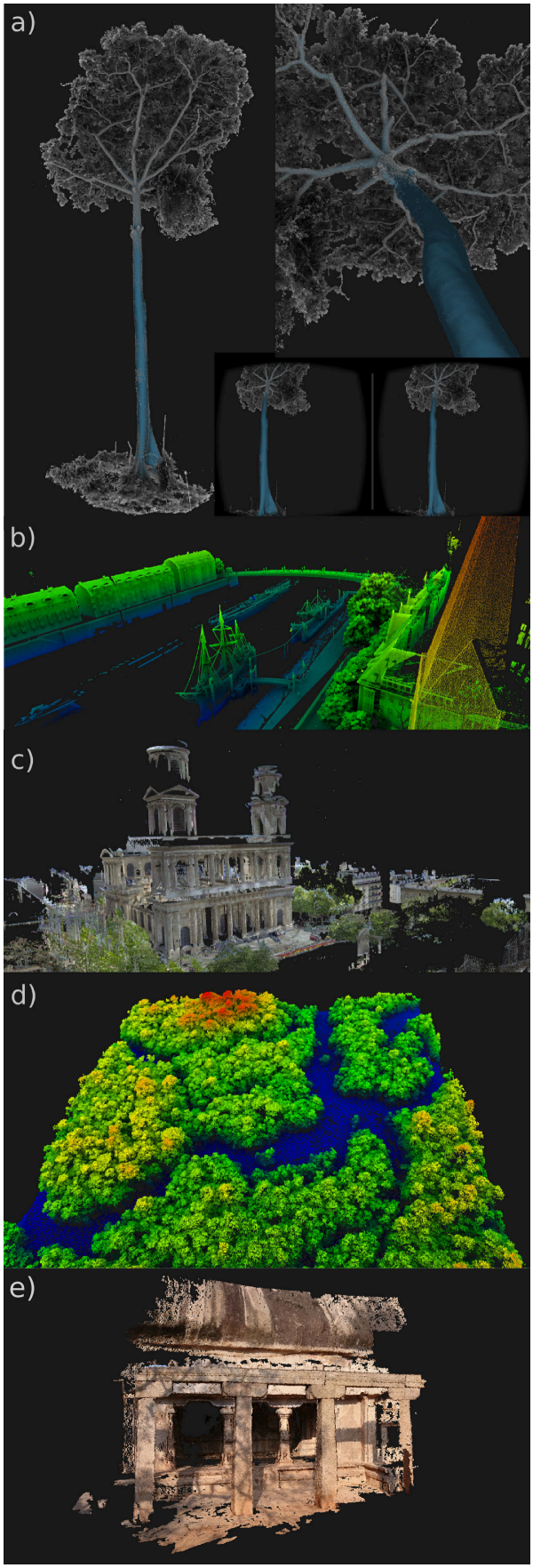


Figure 7: Examples of rendering through in-app screenshots

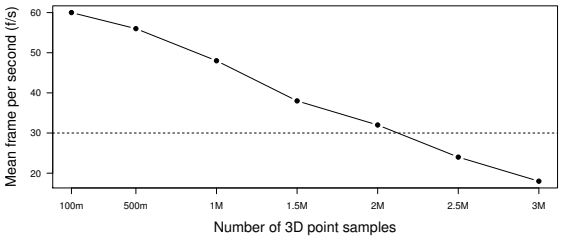


Figure 8: Evolution of the average frame per second ratio with the number of points displayed by the application in immersive mode.

the point clouds as introduced by [4] and loading on GPU only the relevant part, we expect a huge boost in performance. Moreover, the introduction of the Vulkan API, meant to replace OpenGL, will also boost the performance and allow the broadcast of the application on other operating system. The application is currently being ported on the Oculus Mobile systems: the implementation of parts of the app using native-code languages such as C and C++ thanks to the Android NDK, and the extra inertial measurement units of the Samsung Gear VR will enhance this VR experience.

ACKNOWLEDGEMENTS

LiDAR data used in Figure 7c) were collected in the Nouragues Research station under the ANR grants CEBA: ANR-10-LABX-25-01.

6 REFERENCES

[1] A. Baerentzen, S. L. Nielsen, B. D. Larsen M. Gjol, and N. J. Christensen. Single-pass wireframe rendering. *Siggraph '06 ACM*, 2006.

[2] C. Boucheny. Interactive scientific visualization of large datasets: towards a perceptive-based approach. *PhD thesis*, 2009.

[3] J. Kessenich, D. Baldwin, and R. Rost. The opengl shading language. 2004.

[4] Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society, 2002.

[5] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 1975.

[6] Randi J Rost, Bill Licea-Kane, Dan Ginsburg, John M Kessenich, Barthold Lichtenbelt, Hugh Malan, and Mike Weiblen. *OpenGL shading language*. Pearson Education, 2009.

[7] Marco Tarini, Paolo Cignoni, and Roberto Scopigno. Abstract visibility based methods and assessment for detail-recovery. *Visualization, IEEE*, 2003.

2 Recent developments

As presented in the previous paper, several specific techniques were implemented to overcome the relatively low processing power of smart-phones CPU and GPU. To keep a high frame rate while displaying millions of points and complex mesh on the high definition screens of those devices, we kept simple OpenGL Shading Language (GLSL)[88,130] shaders to limit the computation per pixel. However, simple techniques found in the literature and lately implemented as GLSL shaders to be hardware supported, allowed to improve the rendering of surfaces. Among other, the focus was on surface elements rendering and lightning models.

2.1 Surfels

At the interface between point cloud and surface resides the concept of *surface elements* (surfels)[107]: able to render complex geometric objects, surfels are point primitive without explicit connectivity, represented as small disks oriented with the local normals. Pfister[124] defines a surfel as "a zero-dimensional n-tuple with shape and shade attributes that locally approximate an object surface". While using surfel to render an object geometry, we transfer two costly tasks as pre-processing: (i) the re-sampling of the point cloud to limit the redundancy of geometric information at a given resolution and (ii) the computation of the normals. Then surfels become tailored primitive for fast rendering.

Inspired by Gross[71], we parametrize the screen-space square over $[-r, r]^2$, where r is the surfel radius. For each of the pixel $(x, y) \in [-r, r]^2$, a depth offset δz from the surfel center $\mathbf{p}$ is computed as a linear function depending on the surfel normal vector $\mathbf{n} = [n_x, n_y, n_z]^T$ expressed in the camera space:

$$\delta z = -\frac{n_x}{n_z} \cdot x - \frac{n_y}{n_z} \cdot y \tag{A.1}$$

This depth offset is then used to compute the 3D distance from the surfel center: the pixel (x, y) corresponds to a point inside the surfel if $\|(x, y, \delta z)\| \leqslant r$. Otherwise, it is discarded.

The quality of the model rendered rely on the expertise of the user and in the calibration of the pre-processing step: the point density and the surfels radius must match the expected output resolution of the image.

Our implementation of the surfels lightning is detailed in Listing A.1.

Listing A.1: GLSL code of the surfels lightning vertex and fragment shaders

```
// Vertex shader
uniform mat4 u_MVP;          // Map from world space to camera space
uniform mat4 u_MVMatrix;     // Map from camera space to screen space
attribute vec4 a_Position;   // Position of the vertex
attribute vec3 a_Normal;     // Normal of the vertex
varying vec3 v_Position;     // Position given to fragment shader
varying vec3 v_Normal;       // Normal given to fragment shader

void main() {
    // Transform the vertex into eye space.
    v_Position = vec3(u_MVMatrix * a_Position);

    // Transform the normal's orientation into eye space.
    v_Normal = vec3(u_MVMatrix * vec4(a_Normal, 0.0));

    // Final point position in screen coordinates.
    gl_Position = u_MVP * a_Position;

    //Scale the surfel size with the distance to the camera
    gl_PointSize = u_Size/gl_Position.w;
}

// Fragment shader
precision mediump float;        // Set the default precision to medium
uniform vec3 u_LightPos;        // The position of the light in eye space
varying vec3 v_Position;        // Interpolated position for this fragment
varying vec3 v_Normal;          // Interpolated normal for this fragment

void main()
{
  // light vector
  vec3 lightVector = normalize(u_LightPos - v_Position);

  // Calculate the dot product of the light vector and vertex normal.
  float diffuse = max(dot(normalize(v_Normal), lightVector), 0.4);

  //compute the coordinate of the given pixel center
  vec2 ptc = gl_PointCoord- vec2(0.5);

  //The depth for the given pixel
  float depth = -v_Normal.x/v_Normal.z*ptc.x+v_Normal.y/v_Normal.z*ptc.y;

  //Check if the pixel "pointCenter" corresponds to a point inside the surfel
  vec3 pointProj = vec3(ptc.x,ptc.y,depth);
  float r = length(pointProj)/0.5;

  //if not, it is discarded
  if(r >= 1.) discard;

  //Setting the fragment color
    gl_FragColor = v_Color * diffuse;
}
```

The upcoming developments on the surfel rendering will focus on the computation of the correct surfel radius. Currently, the splat-size is constant and set by the user, as shown in the vertex shader of Listing A.1. To estimate a correct radius dynamically, we propose to follow Pajarola[120] approach in order to force the elliptical disks to cover the surface at all levels of details. Moreover, this approach introduces a blending process to smoothly interpolate the texture color of irregularly distributed point samples. Both of these techniques are related to a crucial improvement

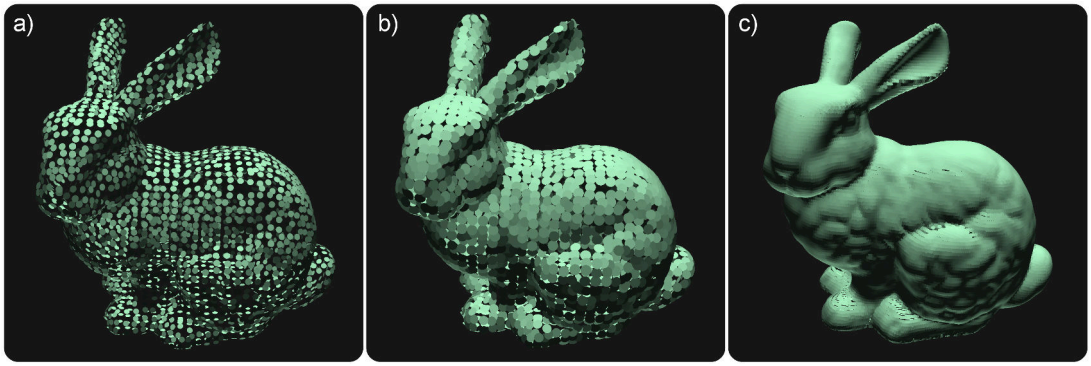


Figure A.1: Stanford bunny rendered with surfel: (a) and (b) 3k points with surfel of different radius and (c) 35k points.

needing to be implemented: the optimization of the data to be loaded into the GPU according to the viewpoint, as presented by Pauli[122].

2.2 Cook-Torrance model

In the recent developments, a special effort has been made to the rendering of surface, in particular to the lightning model. Indeed, inspired by Cook and Torrance[43], we implemented a realistic lightning model. This model, known to be closer to the physical reality, simulates the specular reflectance of different materials. The model treats each surface as consisting of many micro-facets: Very small facets that reflect the incoming light. On rough surfaces, the slopes of these microfacets vary greatly, and on smooth surfaces the micro-facets are oriented in a similar direction[144]. The model focuses on the specular reflection r_s computed as:

$$r_s = \frac{\mathcal{F} \cdot \mathcal{D} \cdot \mathcal{G}}{\Pi(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})} \tag{A.2}$$

where $\mathbf{n}$ is the local normal vector, $\mathbf{l}$ is the light vector and $\mathbf{v}$ is the view vector. $\mathcal{F}$, the Fresnel factor, defines what fraction of the incoming light is reflected and what fraction is transmitted. Because the original Fresnel formula is too computationally expensive, we used the Schlick approximation[135] to get similar results with faster computation. $\mathcal{D}$ is the directional distribution of the micro-facets. On smooth surfaces, all micro-facets have a similar orientation and therefore all the reflected light is close to the reflection vector. On rough surfaces, the light is more widely distributed. Some of the micro-facets block the incoming light before reaching the surface or after being reflected. $\mathcal{G}$, the geometrical attenuation factor, is a value from 0 to 1 that represents

the proportionate amount of light that remains after this shadowing or masking has taken place. This calculation assumes that all micro-facets have the form of V-shaped grooves and is described in details by Blinn [21].

Our implementation of the Cook-Torrance lightning is detailed in Listing A.2 and its effect is illustrated in Figure A.2.

Listing A.2: GLSL code of the Cook-Torrance lightning vertex and fragment shaders

```
// Vertex shader
uniform mat4 u_MVP;          // Map from world space to camera space
uniform mat4 u_MVMatrix;     // Map from camera space to screen space
attribute vec4 a_Position;   // Position of the vertex
attribute vec3 a_Normal;     // Normal of the vertex
attribute vec4 a_Color;      // Color of the vertex
varying vec3 v_Position;     // Position given to fragment shader
varying vec4 v_Color;        // Color given to fragment shader
varying vec3 v_Normal;       // Normal given to fragment shader

void main() {
    // Transform the vertex into eye space.
    v_Position = vec3(u_MVMatrix * a_Position);

    // Pass through the color.
    v_Color = a_Color;

    // Transform the normal's orientation into eye space.
    v_Normal = vec3(u_MVMatrix * vec4(a_Normal, 0.0));

    // Final point position in screen coordinates.
    gl_Position = u_MVP * a_Position;
}

// Fragment shader
precision mediump float;       // Set the default precision to medium
uniform vec3 u_LightPos;       // The position of the light in eye space
varying vec3 v_Position;       // Interpolated position for this fragment
varying vec4 v_Color;          // Interpolated color for this fragment
varying vec3 v_Normal;         // Interpolated normal for this fragment

void main()
{
  float C = 0.8; // roughness of the surface (0 : smooth, 1: rough)
  float ni = 0.2; // fresnel reflectance at normal incidence

  // Normalize interpolated normal
  vec3 N = normalize(v_Normal);

  // Light vector
  vec3 L = normalize(u_LightPos - v_Position);

  // Vertex-to-eye space (view vector)
  vec3 V = normalize(-v_Position);

  // Half-vector
  vec3 H = normalize(vec3(lightVector-v_Position));

  float NdotH = max(0.0, dot(N, H));
  float VdotH = dot(V, H);
  float NdotV = dot(N, V);
  float NdotL = dot(N, L);
```

```
    // Fresnel Refraction (F)
  float k = pow(1.0 - NdotV, 5.0);
  float F = 1.0 - k + ni * k;

  // Gaussian term (D)
  float alpha = acos(NdotH);
  float D = C * exp(- alpha * alpha * C);

  // Geometric factor (G)
  float G1 = 2.0 * NdotH * NdotV / VdotH;
  float G2 = 2.0 * NdotH * NdotL / VdotH;
  float G = min(1.0, min(G1, G2));

  // Calculate the dot product of the light vector and vertex normal.
  float diffuse = max(dot(normalize(v_Normal), L), 0.4);

  // Setting the fragment color
    gl_FragColor = v_Color * diffuse * (F * D * G) / NdotV;
}
```

Figure A.2: Surface model enlightened by (a) a Lambertian diffuse reflection model and (b) a Cook-Torrance model.

References

[1] Ackoff, R. L. (1989). From data to wisdom. *Journal of applied systems analysis*, 16(1), 3–9.

[2] Alba, M., Barazzetti, L., Roncoroni, F., & Scaioni, M. (2011). Filtering vegetation from terrestrial point clouds with low-cost near infrared cameras. *Italian Journal of Remote Sensing*, 43, 55–75.

[3] Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., & Desbrun, M. (2003). Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)*, volume 22 (pp. 485–493).: ACM.

[4] Alliez, P., Ucelli, G., Gotsman, C., & Attene, M. (2008). Recent advances in remeshing of surfaces. In *Shape analysis and structuring* (pp. 53–82). Springer.

[5] Amanatides, J., Woo, A., et al. (1987). A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87 (pp. 3–10).

[6] Amenta, N. & Bern, M. (1999). Surface reconstruction by voronoi filtering. *Discrete computing geometry*, (pp. 481–504).

[7] Amenta, N., Bern, M., & Eppstein, D. (1998). The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2), 125–135.

[8] Amenta, N., Choi, S., Dey, T. K., & Leekha, N. (2000). A simple algorithm for homeomorphic surface reconstruction. *Proceedings of the sixteenth annual symposium on Computational geometry*, (pp. 213–222).

[9] Amenta, N., Choi, S., & Kolluri, R. K. . (2001). The power crust. *Proceedings of the sixth ACM symposium on Solid modeling and applications.*, (pp. 249–266).

[10] Amenta, N., Choi, S., & Kolluri, R. K. (2003). Tight cocone: a water-tight surface reconstructor. *Proceedings of the sixth ACM symposium on Solid modeling and applications.*, (pp. 127–134).

[11] Amenta, N. & Kil, Y. J. (2004). Defining point-set surfaces. *ACM SIGGRAPH 2004 Papers*, (pp. 264–270).

[12] Aurenhammer, F. (1991). Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), 345–405.

[13] Axelsson, P. (2000). DEM Generation from laser scanner data using adaptive tin models. *International Archives of Photogrammetry and Remote Sensing*, (pp. 110–117).

[14] Bac, A., Ravaglia, J., Morel, J., & Véga, C. (2013). Detection of single trees in t-lidar scans of dense forest areas. *LSIS research report*.

[15] Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2), 111–122.

[16] Barczi, J.-F., Rey, H., Caraglio, Y., De Reffye, P., Barthélémy, D., Fourcaud, T., & Dong, Q. X. (2008). Amapsim: an integrative whole-plant architecture simulator based on botanical knowledge. *Annals of Botany*, 101(8), 1125–1138.

[17] Béland, M., Widlowski, J.-L., & Fournier, R. A. (2014). A model for deriving voxel-level tree leaf area density estimates from ground-based lidar. *Environmental Modelling & Software*, 51, 184–189.

[18] Berger, M., Tagliasacchi, A., Seversky, L., Alliez, P., Levine, J., Sharf, A., & Silva, C. (2014). State of the art in surface reconstruction from point clouds. 1(1), 161–185.

[19] Berlinet, A. & Thomas-Agnan, C. (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media.

[20] Bernardini, F., Mittleman, H. R. J., & Rushmeier, H. (1999). The ball-pivoting algorithm for surface reconstruction. *Transaction on visualization and computer graphics*, (pp. 349–359).

[21] Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH Computer Graphics*, volume 11 (pp. 192–198).: ACM.

[22] Bloomenthal, J. (1994). An implicit surface polygonizer. *Graphics gems IV*, 4, 324.

[23] Boissonnat, J.-D. (1984). Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics (TOG)*, 3(4), 266–286.

[24] Borrelli, V., Cazals, F., & Morvan, J.-M. (2003). On the angular defect of triangulations and the pointwise approximation of curvatures. *Computer Aided Geometric Design*, 20(6), 319–341.

[25] Botsch, M., Bommes, D., & Kobbelt, L. (2005). Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI* (pp. 62–83). Springer.

[26] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Lévy, B. (2010). *Polygon mesh processing*. CRC press.

[27] Brown, K. Q. (1979). Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5), 223–228.

[28] Brown, S. (1997). *Estimating biomass and biomass change of tropical forests: a primer*, volume 134. Food & Agriculture Org.

[29] Buhmann, M. D. (2003). Radial basis functions: theory and implementations. *Cambridge Monographs on Applied and Computational Mathematics*, (pp. 307–318).

[30] Calakli, F. & Taubin, G. (2011). Ssd: Smooth signed distance surface reconstruction. *Comput. Graph. Forum*, 30(7), 1993–2002.

[31] Calders, K., Armston, J., Newnham, G., Herold, M., & Goodwin, N. (2014). Implications of sensor configuration and topography on vertical plant profiles derived from terrestrial lidar. *Agricultural and Forest Meteorology*, 194, 104–117.

[32] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., & Evans, T. R. (2001). Reconstruction and representation of 3d objects with radial basis functions. In L. Pocock (Ed.), *SIGGRAPH* (pp. 67–76).: ACM.

[33] Carr, J. C., Fright, W. R., & Beatson, R. K. (1997). Surface interpolation with radial basis functions for medical imaging. *IEEE transactions on medical imaging*, 16(1), 96–107.

[34] Cazals, F. & Pouget, M. (2005). Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2), 121–146.

[35] Chang, W. (2008). *Surface reconstruction from points*. Department of Computer Science and Engineering, University of California, San Diego.

[36] Charru, M., Seynave, I., Hervé, J.-C., & Bontemps, J.-D. (2014). Spatial patterns of historical growth changes in norway spruce across western european mountains and the key effect of climate warming. *Trees*, 28(1), 205–221.

[37] Chasmer, L., Hopkinson, C., & Treitz, P. (2004). Assessing the three-dimensional frequency distribution of airborne and ground-based lidar data for red pine and mixed deciduous forest plots. *Proceedings of the Laser Scanners for Forest and Landscape Assessment–Instruments, Processing Methods and Applications*, (pp. 02–06).

[38] Chave, J., Andalo, C., Brown, S., Cairns, M., Chambers, J., Eamus, D., Fölster, H., Fromard, F., Higuchi, N., Kira, T., et al. (2005). Tree allometry and improved estimation of carbon stocks and balance in tropical forests. *Oecologia*, 145(1), 87–99.

[39] Chave, J., Réjou-Méchain, M., Búrquez, A., Chidumayo, E., Colgan, M. S., Delitti, W. B., Duque, A., Eid, T., Fearnside, P. M., Goodman, R. C., et al. (2014). Improved allometric models to estimate the aboveground biomass of tropical trees. *Global Change Biology vol 20.*, (pp. 3177–3190).

[40] Cheng, S.-W., Dey, T. K., & Shewchuk, J. (2012). *Delaunay mesh generation*. CRC Press.

[41] Cohen-Steiner, D. & F., D. (2004). A greedy delaunay based surface reconstruction algorithm. *The Visual Computer*.

[42] Cohen-Steiner, D. & Morvan, J.-M. (2003). Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry* (pp. 312–321).: ACM.

[43] Cook, R. L. & Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1), 7–24.

[44] Côté, J.-F., Fournier, R. A., & Egli, R. (2011). An architectural model of trees to estimate forest structural attributes using terrestrial lidar. *Environmental Modelling & Software*, 26(6), 761–777.

[45] Côté, J.-F., Widlowski, J.-L., Fournier, R. A., & Verstraete, M. M. (2009). The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote sensing and environment*, (pp. 1067–1081).

[46] Cox, M. G. (1972). The numerical evaluation of b-splines. *IMA Journal of Applied Mathematics*, 10(2), 134–149.

[47] Dassot, M., Constant, T., & Fournier, M. (2011). The use of terrestrial lidar technology in forest science: Application fields, benefits and challenges. *Annals of Forest Science*, (pp. 959–974).

[48] De Boor, C. (1972). On calculating with b-splines. *Journal of Approximation theory*, 6(1), 50–62.

[49] Desbrun, M., Meyer, M., Schröder, P., & Barr, A. H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (pp. 317–324).: ACM Press/Addison-Wesley Publishing Co.

[50] Dokken, T., Lyche, T., & Pettersen, K. . (2013). Polynomial splines over locally refined box-partitions. *Computer Aided Geometric Design*, 30(3), 331–356.

[51] Du, Q., Faber, V., & Gunzburger, M. (1999). Centroidal voronoi tessellations: applications and algorithms. *SIAM review*, 41(4), 637–676.

[52] Duchon, J. (1977). Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables* (pp. 85–100). Springer.

[53] Durrieu, S., Allouis, T., Fournier, R., Véga, C., & Albrech, L. (2008). Spatial quantification of vegetation density from terrestrial laser scanner data for characterization of 3d forest structure at plot level. In *SilviLaser 2008* (pp. p–325).

[54] Eck, M. & Hoppe, H. (1996). Automatic reconstruction of b-spline surfaces of arbitrary topological type. In J. Fujii (Ed.), *SIGGRAPH* (pp. 325–334).: ACM.

[55] Edelsbrunner, H., Kirkpatrick, D., & Seidel, R. (1983). On the shape of a set of points in the plane. *Transactions on Information Theory*, (pp. 551–559).

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

[57] Elmqvist, M., Jungert, E., Lantz, F., Persson, A., & Soderman, U. (2001). Terrain modeling and analysis using laser scanned data. *International archives of photogrammetry and remote sensing*, 34.

[58] Elseberg, J., B. D. . N. A. (2013). One billion points in the cloud – an octree for efficient processing of 3d laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, (pp. 76–88).

[59] Enquist, B. J. (2002). Universal scaling in tree and vascular plant allometry: toward a general quantitative theory linking plant form and function from cells to ecosystems. *Tree physiology*, 22(15-16), 1045–1064.

[60] Fasshauer, G. (2006). Meshfree methods. *Handbook of Theoretical and Computational Nanotechnology, American Scientific Publishers*, 27, 33–97.

[61] Fasshauer, G. E., Chui, C. K., Schumaker, L. L., & Stöckler, J. (2002). Matrix-free multilevel moving least-squares methods. *Approximation Theory X: Abstract and Classical Analysis*, (pp. 271–281).

[62] Fasshauer, G. E. & Zhang, J. G. (2006). Scattered data approximation of noisy data via iterated moving least squares. *Proceedings of. Curves and Surfaces Avignon*.

[63] Fayolle, A., Doucet, J.-L., Gillet, J.-F., Bourland, N., & Lejeune, P. (2013). Tree allometry in central africa: Testing the validity of pantropical multi-species allometric equations for estimating biomass and carbon stocks. *Forest Ecology and Management*, 305, 29–37.

[64] Fearnside, P. M. (1997). Wood density for estimating forest biomass in brazilian amazonia. *Forest ecology and management*, 90(1), 59–87.

[65] Fletcher, R. & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The computer journal*, 7(2), 149–154.

[66] Gelas, A., Bernard, O., Friboulet, D., & Prost, R. (2007). Compactly supported radial basis functions based collocation method for level-set evolution in image segmentation. *IEEE Transactions on Image Processing*, 16(7), 1873–1887.

[67] George, P. & Borouchaki, H. (1998). *Delaunay triangulation and meshing*.

[68] Gopi, M., Krishnan, S., & Silva, C. (2000). Surface recnstruction based on lower dimensional localized delaunay triangulation. *Computer graphics forum*.

[69] Gough, B. (2009). *GNU scientific library reference manual*. Network Theory Ltd.

[70] Gregorski, B. F., Hamann, B., & Joy, K. I. (2000). Reconstruction of b-spline surfaces from scattered data points. In *Computer Graphics International* (pp. 163–170).: IEEE Computer Society.

[71] Gross, M. & Pfister, H. (2011). *Point-based graphics*. Morgan Kaufmann.

[72] Hackenberg, J., Morhart, C., Sheppard, J., Spiecker, H., & Disney, M. (2014). Highly accurate tree models derived from terrestrial laser scan data: A method description. *Forests vol. 5*, (pp. 1069–1105).

[73] Hackenberg, J., Spiecker, H., Calders, K., Disney, M., & Raumonen, P. (2015). Simpletree—an efficient open source tool to build tree models from tls clouds. *Forests*, 6(11), 4245–4294.

[74] Hardy, L. R. (1971). Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8), 1905–1915.

[75] Heckbert, P. S. & Garland, M. (1997). *Survey of polygonal surface simplification algorithms*. Technical report, DTIC Document.

[76] Henning, J. G. & Radtke, P. J. (2006). Ground-based laser imaging for assessing three-dimensional forest canopy structure. *Photogrammetric Engineering & Remote Sensing*, 72(12), 1349–1358.

[77] Henry, M., Besnard, A., Asante, W., Eshun, J., Adu-Bredu, S., Valentini, R., Bernoux, M., & Saint-André, L. (2010). Wood density, phytomass variations within and among trees, and allometric equations in a tropical rainforest of africa. *Forest Ecology and Management*, 260(8), 1375–1388.

[78] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1992). Surface reconstruction from unorganized points. *ACM SIGGRAPH 1992 Proceedings*, (pp. 71–78).

[79] Horn, B. K. (1989). Obtaining shape from shading information. In *Shape from shading* (pp. 123–171).: MIT press.

[80] Howard, I. P. & Rogers, B. J. (1995). *Binocular vision and stereopsis*. Oxford University Press, USA.

[81] Huxley, J., Huxley, J. S., et al. (1993). *Problems of relative growth*.

[82] Interrante, V., Fuchs, H., & Pizer, S. (1995). Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization'95* (pp.52).: IEEE Computer Society.

[83] Kankare, V., Holopainen, M., Vastaranta, M., Puttonen, E., Yu, X., Hyyppä, J., Vaaja, M., Hyyppä, H., & Alho, P. (2013). Individual tree biomass estimation using terrestrial laser scanning. *J. Photogramm. Remote Sens*, (pp. 64–75).

[84] Karlsson, B. (2005). *Beyond the C++ standard library: an introduction to boost*. Pearson Education.

[85] Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4), 321–331.

[86] Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson surface reconstruction. *Symposium on Geometry Processing*.

[87] Kazhdan, M. & Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 29.

[88] Kessenich, J., Baldwin, D., & Rost, R. (2004). The opengl shading language. *Language version*, 1.

[89] Kimme, C., Ballard, D., & Sklansky, J. (1975). Finding circles by an array of accumulators. *Communications of the ACM*, 18(2), 120–122.

[90] Kolluri, R., Shewchuk, J. R., & F., O. J. (2004). Spectral surface reconstruction from noisy point clouds. *Symposium on geometry processing.*, (pp. 11–21).

[91] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1), 48–50.

[92] Lancaster, P. & Salkauskas, K. (1981). Surfaces generated by moving least squares methods,. *Mathematics of Computation*, (pp. 141–158).

[93] Levin, D., Brunnett, G., Hamann, B., Mueller, K., & Linsen, L. (2003). Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*.

[94] Liang, X., Kankare, V., Hyyppä, J., Wang, Y., Kukko, A., Haggrén, H., Yu, X., Kaartinen, H., Jaakkola, A., Guan, F., et al. (2016). Terrestrial laser scanning in forest inventories. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115, 63–77.

[95] Lipman, Y., Cohen-Or, D., & Levin, D. (2006). Error bounds and optimal neighborhoods for mls approximation. *Proceedings of the fourth Eurographics symposium on Geometry.*, (pp. 71–80).

[96] Liu, S. & Wang, C. C. (2010). Orienting unorganized points for surface reconstruction. *Computers & Graphics.*, (pp. 209–218).

[97] Lorensen, W. E. & Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21 (pp. 163–169).: ACM.

[98] Lovell, J., Jupp, D., Newnham, G., & Culvenor, D. (2011). Measuring tree stem diameters using intensity profiles from ground-based scanning lidar from a fixed viewpoint. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(1), 46–55.

[99] Maas, H., Bienert, A., Scheller, S., & Keane, E. (2008). Automatic forest inventory parameter determination from terrestrial laser scanner data. *International Journal of Remote Sensing,*, (pp. 1579–1593).

[100] Mairhuber, J. (1954). On haar's theorem concerning chebyshev approximation problems having unique solutions. *Proc. Amer. Math. Soc.*, (pp. 609–615).

[101] Markku, Å., Raumonen, P., Kaasalainen, M., & Casella, E. (2015). Analysis of geometric primitives in quantitative structure models of tree stems. *Remote Sensing*, (pp. 4581–4603).

[102] Max, N. (1999). Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2), 1–6.

[103] McDaniel, M. e. a. (2012). Terrain classification and identification of tree stems using ground based lidar. *Journal of Field Robotics*, (pp. 1–20).

[104] McKechnie, J., Colombo, S., Chen, J., Mabee, W., & MacLean, H. L. (2010). Forest bioenergy or forest carbon? assessing trade-offs in greenhouse gas mitigation with wood-based fuels. *Environmental science & technology*, 45(2), 789–795.

[105] McQuail, D. (1987). Mass communication theory: An introduction (2nd ed.). *Computer Aided Geometric Design*.

[106] Meng, X., Currit, N., & Zhao, K. (2010). Ground filtering algorithms for airborne LiDAR data: a review of critical issues. *Remote Sensing*, 2, 833–860.

[107] Merl, B. (1994). A survey and classification of real time rendering methods.

[108] Meyer, M., Desbrun, M., Schröder, P., & Barr, A. H. (2003). Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III* (pp. 35–57). Springer.

[109] Minamino, R. & Tateno, M. (2014). Variation in susceptibility to wind along the trunk of an isolated larix kaempferi (pinaceae) tree. *American journal of botany*, 101(7), 1085–1091.

[110] Morel, J., Bac, A., & Véga, C. (2016). Reconstruction of trees with cylindrical quadrics and radial basis functions (in revision). *Photogrammetric Engineering and Remote Sensing*.

[111] Morse, B., Yoo, T. S., Rheigans, P., Chen, D. T., & Subramanian, K. R. (2001). Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. *International Conference on Shape Modeling and Applications*, (pp. 89–98).

[112] Nagai, Y., Ohtake, Y., & Suzuki, H. (2009). Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. *Comput. Graph. Forum*, 28(5), 1339–1348.

[113] Newman, T. S. & Yi, H. (2006). A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5), 854–879.

[114] Newnham, G. J., Armston, J. D., Calders, K., Disney, M. I., Lovell, J. L., Schaaf, C. B., Strahler, A. H., & Danson, F. M. (2015). Terrestrial laser scanning for plot-scale forest measurement. *Current Forestry Reports*, 1(4), 239–251.

[115] Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., & Seidel, H.-P. (2003). Multi-level partition of unity implicits. *ACM Trans. Graph.*, (pp. 463–470).

[116] Ohtake, Y., Belyaev, A., & Seidel, H.-P. (2006). Sparse surface reconstruction with adaptive partition of unity and radial basis functions. *Graphical Models*, 68(1), 15–24.

[117] Olagoke, A., Proisy, C., Féret, J.-B., Blanchard, E., Fromard, F., Mehlig, U., de Menezes, M. M., Dos Santos, V. F., & Berger, U. (2016). Extended biomass allometric equations for large mangrove trees from terrestrial lidar data. *Trees*, 30(3), 935–947.

[118] Olson, J. S., Watts, J. A., & Allison, L. J. (1983). *Carbon in live vegetation of major world ecosystems*. Technical report, Oak Ridge National Lab., TN (USA).

[119] Osher, S. & Fedkiw, R. (2006). *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media.

[120] Pajarola, R., Sainz, M., & Guidotti, P. (2003). *Object-space blending and splatting of points*. Information and Computer Science, University of California, Irvine.

[121] Patanè, G. (2013). Multi-resolutive sparse approximations of d-dimensional data. *Computer Vision and Image Understanding*, (pp. 418–428).

[122] Pauly, M., Gross, M., & Kobbelt, L. P. (2002). Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02* (pp. 163–170).: IEEE Computer Society.

[123] Pfeifer, N., Gorte, B., Winterhalder, D., et al. (2004). Automatic reconstruction of single trees from terrestrial laser scanner data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf.*, (pp. 114–119).

[124] Pfister, H., Zwicker, M., Van Baar, J., & Gross, M. (2000). Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (pp. 335–342).: ACM Press/Addison-Wesley Publishing Co.

[125] Piegl, L. A. & Tiller, W. (1995). *The NURBS book*. Monographs in visual communication. Springer.

[126] Pueschel, P. (2013). The influence of scanner parameters on the extraction of tree metrics from faro photon 120 terrestrial laser scans. *J. Photogramm. Remote Sens*, (pp. 58–68).

[127] Raumonen, P., Casella, E., Calders, K., Murphy, S., Åkerblom, M., & Kaasalainen, M. (2015). Massive-scale tree modelling from tls data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3), 189.

[128] Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., & Lewis, P. (2013). Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sensing vol. 5*, (pp. 491–520).

[129] Ravaglia, J., Bac, A., & R., F. (2015). Tree stem reconstruction from terrestrial laser scanner point cloud using hough transform and open active contours. *Proceedings of Silvilaser*, (pp. 1067–1081).

[130] Rost, R. J., Licea-Kane, B., Ginsburg, D., Kessenich, J. M., Lichtenbelt, B., Malan, H., & Weiblen, M. (2009). *OpenGL shading language*. Pearson Education.

[131] Rusu, R. B. & Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 1–4).: IEEE.

[132] Savchenko, V. V., Pasko, A. A., Okunev, O. G., & Kunii, T. L. (1995). Function representation of solids reconstructed from scattered surface points and contours. In *Computer Graphics Forum*, volume 14 (pp. 181–188).: Wiley Online Library.

[133] Schaback, R. (2007). A practical guide to radial basis functions. *Electronic Resource*.

[134] Scharstein, D. & Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1 (pp. I–195).: IEEE.

[135] Schlick, C. (1994). An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13 (pp. 233–246).: Wiley Online Library.

[136] Seidel, D. e. a. (2011). Review of ground-based methods to measure the distribution of biomass in forest canopies. *Annals of Forest Science*, (pp. 225–244).

[137] Seidel, R. (1981). *A convex hull algorithm optimal for point sets in even dimensions*. PhD thesis, University of British Columbia.

[138] Shamos, M. I. & Hoey, D. (1975). Closest-point problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on* (pp. 151–162).: IEEE.

[139] Sithole, G. & Vosselman, G. (2004). Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds. *Journal of Photogrammetry and Remote Sensing*, (pp. 85–101).

[140] Suri, J. S., Liu, K., Singh, S., Laxminarayan, S. N., Zeng, X., & Reden, L. (2002). Shape recovery algorithms using level sets in 2-d/3-d medical imagery: a state-of-the-art review. *IEEE Transactions on information technology in biomedicine*, 6(1), 8–28.

[141] Tao, S., Wu, F., Guo, Q., Wang, Y., Li, W., Xue, B., Hu, X., Li, P., Tian, D., Li, C., et al. (2015). Segmenting tree crowns from terrestrial and mobile lidar data by exploring ecological theories. *ISPRS Journal of Photogrammetry and Remote Sensing*, 110, 66–76.

[142] Taubin, G. (1995). Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Computer Vision, 1995. Proceedings., Fifth International Conference on* (pp. 902–907).: IEEE.

[143] Thürrner, G. & Wüthrich, C. A. (1998). Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, 3(1), 43–46.

[144] Torrance, K. E. & Sparrow, E. M. (1967). Theory for off-specular reflection from roughened surfaces. *JOSA*, 57(9), 1105–1112.

[145] Trucco, E. & Fisher, R. B. (1995). Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), 177–182.

[146] Tsai, R., Osher, S., et al. (2003). Review article: Level set methods and their applications in image science. *Communications in Mathematical Sciences*, 1(4), 1–20.

[147] Turk, G. & O'brien, J. F. (1999). Variational implicit surfaces.

[148] Turk, G. & O'brien, J. F. (2005). Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses* (pp.13).: ACM.

[149] Van Leeuwen, M. & Nieuwenhuis, M. (2010). Retrieval of forest structural parameters using lidar remote sensing. *European Journal of Forest Research*, 129(4), 749–770.

[150] Véga, C., Durrieu, S., Morel, J., & Allouis, T. (2012). A sequential iterative dual-filter for Lidar terrain modeling optimized for complex forested environments. *Computers and Geosciences*, 44, 31–41.

[151] Versprille, K. J. (1975). Computer-aided design applications of the rational b-spline approximation form. *Electrical Engineering and Computer Science - Dissertations*, (pp. 1993–2002).

[152] Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *AICM*, (pp. 389–396).

[153] Wendland, H. (2002). Fast evaluation of radial basis functions: methods based on partition of unity. *Approximation Theory X: Abstract and Classical Analysis*, (pp. 473–483).

[154] Wendland, H. (2005a). Computational aspects of radial basis function approximation. *Topics in Multivariate Approximation and Interpolation*, (pp. 231–256).

[155] Wendland, H. (2005b). *Scattered Data Approximation*. Cambridge University Press.

[156] Xu, H., Gossett, N., & Chen, B. (2007). Knowledge and heuristic based modeling of laser-scanned trees. *ACM Trans. Graph.*

[157] Zhang, C. & Chen, T. (2001). Efficient feature extraction for 2d/3d objects in mesh representation. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 3 (pp. 935–938).: IEEE.

[158] Zhang, K. & Whitman, D. (2005). Comparison of three algorithms for filtering airborne lidar data. *Photogrammetric Engineering and Remote Sensing*, 71, 313–324.

[159] Zhao, K., García, M., Liu, S., Guo, Q., Chen, G., Zhang, X., Zhou, Y., & Meng, X. (2015). Terrestrial lidar remote sensing of forests: Maximum likelihood estimates of canopy profile, leaf area index, and leaf angle distribution. *Agricultural and Forest Meteorology*, 209, 100–113.