THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Thibault LEJEMBLE

Le 9 décembre 2020

Analyse multi-échelle de nuage de points

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Loïc BARTHE

Jury

Mme Julie DIGNE, Rapporteure
M. Marc ALEXA, Rapporteur
M. Mathias PAULIN, Examinateur
M. Mathieu DESBRUN, Examinateur
M. Loïc BARTHE, Directeur de thèse
M. Nicolas MELLADO, Co-directeur de thèse

Multi-scale Point Cloud Analysis

Thibault Lejemble

Remerciements

Je tiens à remercier tout particulièrement Nicolas et Loïc pour leur encadrement durant ces trois années. Ils ont formé un duo idéal d'encadrants et travailler avec eux fut un réel plaisir pour moi. Leur présence, leur curiosité et leur motivation m'ont permis d'apprendre beaucoup et m'ont aidé à avancer sereinement dans mes travaux.

Je remercie Julie Digne et Marc Alexa pour avoir relu mon manuscrit de thèse, et pour avoir écrit des rapports très encourageants. J'ai été ravi que Marc Alexa m'accepte au sein de son équipe à Berlin (bien que ma visite ait été stoppée au bout de 10 jours par la COVID-19...). Merci aussi à Mathieu Desbrun d'avoir accepter de participer à mon jury de soutenance en tant qu'examinateur, et à Mathias Paulin en tant que président du jury.

Mon doctorat s'est déroulé dans les meilleures conditions grâce aux chercheurs permanents de l'équipe STORM. Après avoir été mes professeurs à l'UPS, Mathias, Loïc, David et Nicolas m'ont accueilli dans une équipe dynamique et sympathique.

J'ai passé d'excellents moments avec tous les collègues qui m'ont accueilli dans `voxar` lorsque j'ai commencé mon stage – Nadine, Céline, Valentin, Anahid et Florian – et puis avec ceux qui sont arrivés ensuite pendant mon doctorat – Hugo, Olivier, François, Chems, Pierre et Amélie. Et je suis très reconnaissant envers Claudio, Chems et Jie avec qui j'ai pris beaucoup de plaisir à collaborer.

J'aimerais saluer mes amis du lycée et de l'ENSIMAG pour tous ces bons moments que nous partageons ensemble. Finalement, je remercie grandement ma famille, ma belle-famille et Estèle pour tout leur soutien.

Abstract

3D acquisition techniques like photogrammetry and laser scanning are commonly used in numerous fields such as reverse engineering, archeology, robotics and urban planning. The main objective is to get virtual versions of real objects in order to visualize, analyze and process them easily. Acquisition techniques become more and more powerful and affordable which creates important needs to process efficiently the resulting various and massive 3D data.

Data are usually obtained in the form of unstructured 3D point cloud sampling the scanned surface. Traditional signal processing methods cannot be directly applied due to the lack of spatial parametrization. Points are only represented by their 3D coordinates without any particular order.

This thesis focuses on the notion of scale of analysis defined by the size of the neighborhood used to locally characterize the point-sampled surface. The analysis at different scales enables to consider various shapes which increases the analysis pertinence and the robustness to acquired data imperfections.

We first present some theoretical and practical results on curvature estimation adapted to a multi-scale and multi-resolution representation of point clouds. They are used to develop multi-scale algorithms for the recognition of planar and anisotropic shapes such as cylinders and feature curves. Finally, we propose to compute a global 2D parametrization of the underlying surface directly from the 3D unstructured point cloud.

Résumé

Les techniques d'acquisition numérique 3D comme la photogrammétrie ou les scanners laser sont couramment utilisées dans de nombreux domaines d'applications tels que l'ingénierie inverse, l'archéologie, la robotique, ou l'urbanisme. Le principal objectif est d'obtenir des versions virtuels d'objets réels afin de les visualiser, analyser et traiter plus facilement. Ces techniques d'acquisition deviennent de plus en plus performantes et accessibles, créant un besoin important de traitement efficace des données 3D variées et massives qui en résultent.

Les données sont souvent obtenues sont sous la forme de nuage de points 3D non-structurés qui échantillonnent la surface scannée. Les méthodes traditionnelles de traitement du signal ne peuvent alors s'appliquer directement par manque de paramétrisation spatiale, les points étant explicités par leur coordonnées 3D, sans ordre particulier.

Dans cette thèse nous nous focalisons sur la notion d'échelle d'analyse qui est définie par la taille du voisinage utilisé pour caractériser localement la surface échantillonnée. L'analyse à différentes échelles permet de considérer des formes variées et ainsi rendre l'analyse plus pertinente et plus robuste aux imperfections des données acquises.

Nous présentons d'abord des résultats théoriques et pratiques sur l'estimation de courbure adaptée à une représentation multi-échelle et multi-résolution de nuage de points. Nous les utilisons pour développer des algorithmes multi-échelle de reconnaissance de formes planaires et anisotropes comme les cylindres et les lignes caractéristiques. Enfin, nous proposons de calculer une paramétrisation 2D globale de la surface sous-jacente directement à partir de son nuage de points 3D non-structurés.

Contents

List of Figures

List of Tables

Introduction

The development of 3D acquisition techniques have progressed rapidly during past decades. Mature technologies and devices now exist: photogrametry, Lidar systems, Kinect devices, etc... Many technical fields such as reverse engineering, archeology and urban planning use them more and more often. The objective is usually to obtain virtual copies of physical objects. These "digital twins" are easier to control and manipulate than their physical counterparts. Their visualization, analysis and processing are non-intrusive and can be automated. Acquisition techniques are now able to digitize elements with a wide range of sizes from small molecular structures to whole cities. Their popularity is also empowered by recent advances of autonomous cars, drones and smart-phones that can embed 3D captors. In addition, digital fabrication, 3D printing and the entertainment industry need more virtual 3D contents than ever. The increasing power and accessibility of acquisition technologies thus create a high demand for efficient algorithms to process 3D acquired data that become various and massive.

The raw 3D data is often represented in a regular grid structure like depth images, but after the registration of multiple acquisitions, the data become a point cloud without any spatial organization. The point cloud corresponds to a discrete set of 3D coordinates sampling the scanned objects surface. Its unstructured nature forbids the direct use of standard signal processing tools such as the Fourier transform and wavelets. The discrete convolution and other fundamental operations are difficult to perform on points only. Contrary to audio signals or images, there is no underlying temporal nor spatial parametrization represented by time series or regular grids. The set of points has no particular order, so a point cannot be referenced by its index since it can be set arbitrarily. There is not even triangles nor edges linking the sampled points as we can find on polygonal meshes. In geometry processing, this surface representation is extreme in a sense that parametric models contain few high orders polynomials as B-Splines, and polygonal meshes contain many smaller linear pieces as triangles. But point clouds have a high number of infinitesimal points that explicitly tell where the surface is located without any topological structure. Acquisition noise and outliers, partially missing data

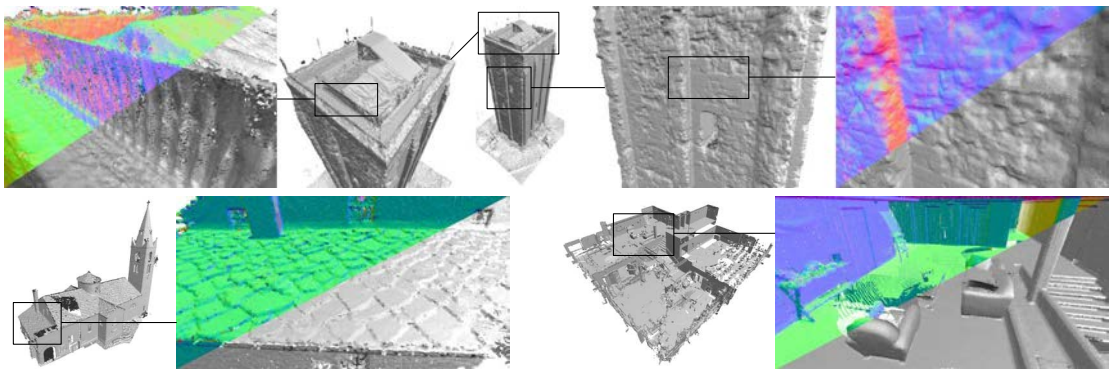


Figure 1: Point clouds examples. Loudun tower (top), Lans church (bottom left) and Euler building (bottom right) contain respectively 35M, 1M and 4M points. Different shapes are observed depending on the scale of observation.

due to occlusion, and sampling irregularities are common artefacts found in point clouds that also make their processing difficult.

Despite the lack of structures, another challenge concerns the notion of scale that intuitively corresponds to the lens size used to look at the point cloud. As shown by Figure 1, 3D models often exhibit shapes with a wide range of sizes. There are small details as tiles and stairs steps, medium details such as furnitures, as well as global elements like roofs and walls. Large models (see Figure 1-top) can even contain several nested levels of scale at the same time. In the context of pattern recognition for instance, the scale is a critical parameter since it determines directly the type of detected shapes. An algorithm performing at a single scale would be able to extract features with an equivalent spatial extent only. This is true even for relatively simple shapes as the twisted cable shown in Figure 2 that is made of two levels of scale.

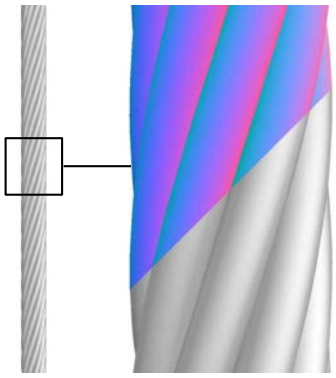


Figure 2: Twisted cable at two scales. This shape can be seen either as one straight vertical cylinder at large scale, or as 10 individual cylinders twisted together at low scale.

The concept of scale is very general as it takes its origin from biological perception. In visual computational models, the scale is defined by the size of a Gaussian kernel that mimics the retinal receptive fields size [Lindeberg 2013a]. This model is the fundamental principle of the so called scale-space framework [Iijima 1963, Witkin 1987] widely used in computer vision [Lindeberg 2013b]. Varying the scale generates more or less blurry images and enables the consideration of various features with different sizes in an image.

In computer graphics, several methods leverage the scale-space for meshing [Digne 2011] and registration [Gelfand 2005] for instance. Many local 3D shape signatures correspond to multi-scale features that are extracted from the point cloud at several scales [Pauly 2003, Pottmann 2007, Mellado 2012]. Multi-scale algorithms have additional advantages other than handling details as well as global features in the same framework. Real data coming from 3D scans frequently contain acquisition artefacts such as noise and outliers. If at small scale a point cloud is particularly noisy, then we can still rely on clean detected features at a higher scale. In addition, multi-scale representations of discrete 3D surfaces are useful in a wide range of computer graphics applications as real-time rendering and interactive editing. However, the question of scale is not always tackled by pattern recognition methods in 3D point clouds. Few approaches in the literature are able to associate a point to different patterns depending on the scale of observation. When an intuitive parameter exists, its setting is often left to the user. If the scale information is not known a priori, setting this parameter usually results in a tedious trial and error process.

Contributions of this thesis

In this thesis we focus on the notion of scale applied to unstructured 3D point cloud analysis. We provide in Chapter 1 a method to characterize the geometry of the input sampled-surface at multiple scales. This method is then used for geometric pattern detection in order to abstract the shape with planes, cylinders and curves (Chapters 2 and 3). To address the lack of structure, we also present in Chapter 4 a preliminary method for finding a global parametrization of the point cloud, i.e. a mapping from a 2D domain to the input 3D points.

In Chapter 1, we first perform a theoretical analysis of the algebraic sphere regression [Guennebaud 2007] used to locally characterize the shape. We demonstrate that the fitted sphere directly gives access to important properties of the point-sampled surface such as its mean curvature, a measure of its anisotropy and a higher order differential quantity. We then introduce an algorithm to estimate principal curvatures and show its numerical accuracy and robustness to noise in practice. This estimator is then integrated in an efficient multi-scale and multi-resolution representation of point clouds that is used in all our work.

The multi-scale differential quantities estimation provides effective descriptors well adapted to abstract the shape with geometric primitives. We propose in Chapter 2 a multi-scale algorithm to detect planar regions in point clouds. A persistence analysis of regions sharing the same differential properties at multiple scales enables the extraction of meaningful planes. A point can thus belong to different planar primitives depending on the scale of analysis.

In Chapter 3, we investigate the use of principal curvature lines drawn at several scales to extract anisotropic features. Instead of looking only at point-wise differential properties, the curvature lines bring some spatial coherence and decrease the sensitivity to data imperfections. Feature curves are located at curved locations where a large number of lines pass through. Moreover, cylindrical regions are segmented at any scale where curvature lines are mostly aligned with each other.

In order to process smoother shapes that potentially lack of prominent planar or anisotropic features, we propose in Chapter 4 to determine a global 2D parametrization of the input 3D point cloud. In this ongoing research project, we let the scale grow until it reaches the size of the whole shape while keeping points on their local surface approximation. In the end, all the points are flattened onto one plane or sphere, producing a planar or spherical parametrization. This mapping between a 2D domain and the set of 3D points makes a powerful structure that could be useful for many shape analysis tasks.

In general, we show the importance of the notion of scale in point cloud analysis. It provides robustness to pattern recognition algorithms and enables to detect geometric features of highly varying sizes (see Chapters 2 and 3). In our scale-space point cloud parametrization algorithm (Chapter 4), the scale is the key element to unfold the unstructured 3D points onto a 2D domain. Considering the scale is thus one step toward more powerful algorithms processing 3D acquired data, which contributes to improve the overall acquisition pipeline.

Chapter 1

Multi-scale differential analysis of point clouds

Contents

1.1 Introduction

The reconstruction of a surface and the estimation of its differentiable properties from unstructured point clouds is a fundamental problem in shape analysis. The algorithms we propose in the following chapters rely on this step to locally characterize the surface. Detecting geometric features (Chapters 2 and 3) or mapping the 3D point cloud to a 2D domain (Chapter 4) both need an accurate approximation of the underlying surface. As discussed in the introduction, the notion of scale is essential and needs to be part of the analysis. Details must be reconstructed at low scales, but do not necessarily appear in the surface approximation when the scale grows. Robustness to noise and computational efficiency are two other mandatory constraints to be able to handle large point clouds obtained from acquired data. This means

that differential estimators should imply as few numerical operations as possible to be fast enough. They also must remain efficient at any scale. In the same time, their estimations have to stay accurate even if some noise perturbs the point cloud.

The state-of-the-art presented in Section 1.2 identifies the Algebraic Point Set Surfaces (APSS) [Guennebaud 2007] as a relevant method for multi-scale approximation of point sampled surfaces. As a particular Point Set Surfaces (PSS) [Alexa 2001], the APSS approximate a smooth 2D manifold from discrete points and include an intuitive scale parameter in the form of a neighborhood radius. Thanks to its algebraic sphere regression, this technique shows in practice an advantageous robustness to data imperfections. However, no theoretical results yet exist in the literature that relates the best fitting algebraic sphere to the differential properties of the surface. As the APSS method involves the regression of an isotropic primitive, another question comes up: how to accurately estimate the principal curvatures of the APSS? Finally, a practical issue occurs at high scale. Although the APSS work locally, conventional space partitioning data structures quickly get overworked when the neighborhood radius become too large.

Contributions of this chapter

- Section 1.3 describes an asymptotic analysis of the algebraic sphere fitting and projection used in the APSS. Using this integral invariant viewpoint [Pottmann 2007], we prove that the fitted algebraic sphere properly captures the mean curvature as well as higher derivatives of the surface. We also show how this isotropic primitive still contains information about the anisotropy of the shape. We also define the associated geometric flow to obtain a robust analog to the mean curvature flow of plane fitting [Digne 2011].

- We propose in Section 1.4 a new method to calculate principal curvatures from the APSS. Prior work [Guennebaud 2007, Mellado 2020] only use partial shape operators (defined by Equation 1.28) while ours performs the complete differentiation of the algebraic sphere fit. A comparative study demonstrates the accuracy and the robustness to noise of our approach.

- Inspired from previous work on multi-scale shape analysis [Pauly 2003], we introduce in Section 1.5 an efficient multi-resolution representation combined to the multi-scale APSS. Our algorithm successfully balances between smoothing and decimation and drastically increases the performance compared to traditional APSS.

1.2 State-of-the-art

Surface reconstruction from a 3D point cloud is a vast and heavily studied scientific area. Several families of approaches are dedicated to specialized domains of application, e.g. urban [Musialski 2013] and indoor [Pintore 2020] scenes. They often focus on specific type of data such as shapes composed of simple primitives [Kaiser 2019], structured objects [Pauly 2008] or general defect-laden point clouds [Berger 2017]. Another substantial body of work tackles surface interpolation problems with a computational geometry point of

view [Cazals 2006]. In Section 1.2.1, we only focus on the fundamental problem of local surface approximation from unstructured point clouds. These methods are general enough to be implemented for a wide range of applications and on various types of shapes. They allow us to compute the low-level geometric features used as basic building block of all the methods described in the following chapters. In addition, local surface approximation methods are often naturally compatible with an intuitive notion of scale.

The notion of scale is a commonly known concept in the digital signal processing field. The idea of separating the small details from the global elements, in other words the high from the low frequencies, is well used for image compression, denoising and pattern recognition. Section 1.2.2 reviews the different approaches of multi-scale analysis of discrete 3D shapes and especially of point clouds.

1.2.1 Local surface approximation

Local surface approximation aims at finding a mathematical model of a smooth and regular surface embedded in $\mathbb{R}^3$ from a discrete set of points. They are local in the sense that they are used around a point and its neighborhood, and they do not require to process the whole point cloud at once. In contrast to interpolation, approximation is well suited to point clouds coming from an acquisition process due to the inherent presence of noise. In our context, the ultimate goal of these techniques is to project a point on the surface estimated around it, and to calculate differential invariants to pertinently describe the geometry nearby that point.

Taylor approximations In the vicinity of a point, an infinitely differentiable function can be expanded as Taylor series involving its successive derivatives. When the series are truncated to a finite order, the resulting Taylor polynomial only approximates the function up to some known errors. The Osculating Jets [Cazals 2005a] find the coefficients of the truncated Taylor expansion, also called jets, that best match the neighborhood of a point. They locally express the surface as a bivariate function over a plane that does not include the surface normal. Computing the K-order Osculating Jets from N points boils down to solve a Vandermonde system of size $N \times (K+1)(K+2)/2$. The result corresponds to the normal vector of the surface, its principal curvatures, as well as other higher order differential quantities. The dependence of the size of the system to the number N of neighboring points is the main issue for using this method in our context. Two different options are possible to query the neighbors. A k-nearest neighbors graph gathers a fixed number of neighbors for each point, but it is inappropriate for point clouds with a highly irregular sampling, strong noise or a large amount of outliers. On the other hand, radius-based neighborhoods that are more appropriate to defect-laden data lead to varying size systems across the point cloud. This second option is less appropriate for GPU implementations and potentially introduces large systems that become slower to solve.

Recently, Béarzi et al. (2018) decompose the bivariate Taylor approximation into a radial polynomial and angular oscillations, creating a new set of basis function called Wavejets. A local frequency analysis of the surface is possible, and differential invariants are also available. Furthermore, Wavejets come with stability properties so that the initial and potentially wrong tangent plane can be corrected afterward. However, they also suffer from the same issue as

the Osculating Jets [Cazals 2005a]. A system of the size of the neighborhood has to be solved which can significantly decrease the performance.

Covariance analysis Principal Component Analysis (PCA) is a popular method to locally characterize discrete data. When performed on a set of 3D points $\{\mathbf{p}_i\}_{i=1...N}$, the PCA consists in the eigendecomposition of the covariance matrix Σ of the points coordinates

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i\mathbf{p}_i^T - \bar{\mathbf{p}}\bar{\mathbf{p}}^T, \tag{1.1}$$

where $\bar{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i$ is the average position. The eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq 0$ correspond to the variance of the coordinates distribution along their associated orthonormal eigenvectors $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$. The vector $\mathbf{v}_2$ associated to the eigenvalue of least magnitude is often taken as the normal direction of the surface while $\mathbf{v}_0$ and $\mathbf{v}_1$ span the tangent plane.

Besides normal estimation [Liang 1990, Mitra 2003, Sanchez 2020], many other methods use this approach for surface reconstruction [Hoppe 1992], shape approximation [Cohen-Steiner 2004], dimensionality analysis [Demantké 2011, Brodu 2012], smoothing [Digne 2011], denoising [Narváez 2006] and so on. Note that a PCA is also an essential step for estimating an initial tangent plane for the Jets-based approaches [Cazals 2005a, Béarzi 2018] introduced in the previous paragraph. In addition, various combinations of eigenvalues give rise to several features encoding the local shape. Such covariance-based features appear in point cloud processing as the Surface Variation [Pauly 2002, Equation 5] and in machine-learning methods for semantic classification [Kalogerakis 2010, Kim 2013, Thomas 2018]. Apart from unstructured point clouds, local covariance analysis is successfully applied to 3D voxels [Coeurjolly 2013] and 2D images when using the Structure Tensor [Harris 1988] for instance. The PCA of normal vectors around a point is also a common way to estimate the principal curvatures and their directions [Berkmann 1994].

One great advantage of the PCA is its efficiency. The size of the covariance matrix Σ to be diagonalized corresponds to the dimension of the ambient space and does not depend on the number of neighboring points. Note that the right-hand side of Equation 1.1 highlights the advantageous fact that visiting the points $\mathbf{p}_i$ is required only once to build Σ. However, as shown by the integral invariant analysis [Pottmann 2007, Theorem 6], the eigenvalues of the PCA calculated on an infinitesimal surface patch asymptotically contain only a mix of principal curvatures. Two of the three eigenvalues do not even have surface curvatures appearing in their preponderant term. As such, these eigenvalues do not tend toward any curvature-related quantity when the size of the surface patch tends to zero. Therefore, many covariance-based features of 3D point clouds do not pertinently characterize the geometry since two different shapes in terms of curvature can have similar PCA eigenvalues. Furthermore, the PCA and other integral invariants in general are particularly stable for structured data such as 2D pixels and 3D voxels [Coeurjolly 2014], but they easily fail on not evenly spaced data like point clouds. Indeed, a varying density in the neighborhood of a point effects the coordinates variance giving a false sense of curvature in that direction.

Point Set Surfaces The Moving Least Squares (MLS) [Levin 1998] is a popular technique to approximate a function from scattered data. Contrary to other approximations that use Radial-Basis-Functions [Carr 2001] or spline fitting [Böhm 1984], the MLS approximation takes inspiration from differential geometry. Each spatial position has its own local coordinate system containing its own mapping. It avoids the difficult task of finding global reference domains, so partitioning the point cloud into multiple patches to get a piecewise parametrization is not required. The MLS is actually akin to local weighted least squares regressions where the weighting function depends on where the regression takes place and has a limited influence in space. The result of the MLS approximation of a set of points in $\mathbb{R}^d$ is a smooth surface [Levin 1998] corresponding to a $(d-1)$-manifold [Levin 2004].

The MLS approximation is the main ingredient of the Point Set Surfaces (PSS) [Alexa 2001] where the surface S is defined as the stationary points of a projection operator $\phi : \mathbb{R}^3 \to \mathbb{R}^3$

$$S = \left\{ \mathbf{x} \in \mathbb{R}^3,\ \phi(\mathbf{x}) = \mathbf{x} \right\}. \tag{1.2}$$

In its original form, the projection operator is a two-step procedure. First, a reference plane is found via a non-linear optimization. Then, a bivariate polynomial is fitted to the local neighborhood expressed over the reference plane. The operator $\phi(\mathbf{x})$ is in this case the projection of $\mathbf{x}$ onto this polynomial approximation.

A simpler and more efficient PSS is provided with an implicit formulation [Adamson 2003a] using weighted average position and covariance analysis (Equation 1.1). Another similar but slightly more general PSS is defined by the critical points of an energy function along lines determined by a vector field [Amenta 2004]. Many other variants exist [Cheng 2008] including Progressive PSS [Fleishman 2003] that builds a multi-resolution surface, Anisotropic PSS [Adamson 2006], which is more robust to irregular sampling and Parabolic-cylindrical PSS [Ridel 2015] enforcing developability. More global approaches based on PSS are also common [Avron 2010, Guillemot 2012, Huang 2019], but their global optimization process does not scale well for very large point clouds. From a PSS perspective, the Osculating Jets [Cazals 2005a] could be viewed as a first unweighted iteration of a MLS approximation without any further fitting steps to reach convergence. The absence of local weighting and iterations loose respectively the smoothness and manifoldness properties obtained with the MLS [Levin 2004].

Sharp features are known to be challenging for PSS because of their smooth aspect. They can be preserved with a forward-search algorithm [Fleishman 2005] where neighboring points are iteratively added to the regression process until their residual is too high. Although this method is able to preserve sharp edges and corners, it has several disadvantages such as the manually selected threshold that locally classifies outliers and that adapts poorly to noise. Performance is another issue since a priority list must be used locally, making parallelization difficult. In another way, iteratively re-weighted least squares simply adjusts the weighting kernel of the MLS approximation [Oztireli 2009]. Neighbors far from the current fitted model contribute less to the next fit by decreasing their weights. This method is robust and is well adapted to the MLS framework. We thus use it in the reaserches presented in this thesis.

Most of PSS methods use the PCA to estimates a first tangent plane. If this stage breaks down due to irregularities present in the data, then the rest of the approximation algorithm

certainly fails too. To avoid this, the Algebraic Point Set Surfaces (APSS) [Guennebaud 2007] directly fit an algebraic sphere [Pratt 1987]. This implicit surface is expressed as the following quadratic scalar field

$$f_{\mathbf{u}}(\mathbf{x}) = u_c + \mathbf{u}_\ell \,.\, \mathbf{x} + u_q \,\mathbf{x} \,.\, \mathbf{x} = \begin{bmatrix} 1 & \mathbf{x}^T & \mathbf{x} \,.\, \mathbf{x} \end{bmatrix} \mathbf{u}, \tag{1.3}$$

where $u_c, u_q \in \mathbb{R}$ and $\mathbf{u}_\ell \in \mathbb{R}^3$ are the parameters of the algebraic sphere compactly represented in the vector $\mathbf{u} = \begin{bmatrix} u_c & \mathbf{u}_\ell^T & u_q \end{bmatrix}^T$. The surface is defined by the 0-isosurface $S_0 = \left\{ \mathbf{x} \in \mathbb{R}^3,\ f_{\mathbf{u}}(\mathbf{x}) = 0 \right\}$. Contrary to PSS based on an implicit plane defined by the linear equation $u_c + \mathbf{u}_\ell \,.\, \mathbf{x} = 0$ [Shen 2004, Kolluri 2008], the algebraic sphere is more robust to irregular and noisy point clouds, especially near curved regions where the plane fitting is usually problematic [Guennebaud 2007, Figures 13-15]. The APSS thus directly fit a quadratic primitive without the need of the local linear basis obtained from the PCA. In case of a purely planar shape, the quadratic term u_q is null and the sphere smoothly changes into a plane. The least squares regression of an algebraic sphere to points equipped with oriented normals has a closed-form expression [Guennebaud 2008]. Two criteria are used in the form of two objective functions given in Equations 1.4 and 1.5. The first criterion ensures that the spatial gradients $\nabla f_{\mathbf{u}}(\mathbf{p}_i)$ match the normals $\mathbf{n}_i$ of the point cloud. The second criterion brings the surface as close as possible to the points $\mathbf{p}_i$ by minimizing the squared scalar field magnitude at these points. Note that the scalar field magnitude at a given point corresponds to its algebraic distance to the sphere.

$$E_1(\mathbf{u}, \mathbf{x}) = \sum_i w_t(\mathbf{p}_i - \mathbf{x}) \, \|\nabla f_{\mathbf{u}}(\mathbf{p}_i) - \mathbf{n}_i\|^2 \tag{1.4}$$

$$E_2(\mathbf{u}, \mathbf{x}) = \sum_i w_t(\mathbf{p}_i - \mathbf{x}) \, f_{\mathbf{u}}(\mathbf{p}_i)^2 \tag{1.5}$$

In sens of the MLS fit, $\mathbf{x}$ is the "moving" evaluation point of the regression and w_t is the weighting function of compact support size $t \in \mathbb{R}$ given by

$$w_t(\mathbf{x}) = K\left(\frac{\|\mathbf{x}\|}{t}\right), \tag{1.6}$$

where $K : (0,1) \to (0,1)$ is a smooth decreasing kernel, typically defined by the polynomial $K(x) = (x^2 - 1)^2$. Minimizing E_1 and E_2 with respect to the parameters $\mathbf{u}$ leads to the following results [Guennebaud 2008, Equation 6]

$$u_q(\mathbf{x}) = \frac{1}{2} \frac{\sum_i w_i \sum_i w_i \, \mathbf{p}_i.\mathbf{n}_i - \sum_i w_i \, \mathbf{p}_i. \sum_i w_i \, \mathbf{n}_i}{\sum_i w_i \sum_i w_i \, \mathbf{p}_i.\mathbf{p}_i - \sum_i w_i \, \mathbf{p}_i. \sum_i w_i \, \mathbf{p}_i}, \tag{1.7}$$

$$\mathbf{u}_\ell(\mathbf{x}) = \frac{1}{\sum_i w_i} \left(\sum_i w_i \, \mathbf{n}_i - 2 \, u_q(\mathbf{x}) \sum_i w_i \, \mathbf{p}_i \right), \tag{1.8}$$

$$u_c(\mathbf{x}) = -\frac{1}{\sum_i w_i} \left(\mathbf{u}_\ell(\mathbf{x}). \sum_i w_i \, \mathbf{p}_i + u_q(\mathbf{x}) \sum_i w_i \, \mathbf{p}_i.\mathbf{p}_i \right), \tag{1.9}$$

where $w_i = w_t(\mathbf{p}_i - \mathbf{x})$. If u_q is null, then a plane is fitted, which corresponds to averaging neighboring positions and normals.

The projection $\varphi_{\mathbf{y}}(\mathbf{x}) \in \mathbb{R}^3$ of a point $\mathbf{x}$ onto the surface defined by $f_{\mathbf{u}(\mathbf{y})}$ fitted at point $\mathbf{y}$ is given by

$$\varphi_{\mathbf{y}}(\mathbf{x}) = \begin{cases} \mathbf{x} - \frac{f_{\mathbf{u}(\mathbf{y})}(\mathbf{x})}{\|\nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x})\|^2} \nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x}) & \text{if } u_q(\mathbf{y}) = 0, \\ \mathbf{x} - \frac{\|\nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x})\| - \sqrt{\Delta_{\mathbf{u}(\mathbf{y})}}}{2u_q(\mathbf{y})\|\nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x})\|} \nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x}) & \text{otherwise,} \end{cases} \tag{1.10}$$

where $\nabla f_{\mathbf{u}(\mathbf{y})}(\mathbf{x}) = \mathbf{u}_\ell(\mathbf{y}) + 2u_q(\mathbf{y})\mathbf{x}$ is the gradient of the scalar field evaluated at $\mathbf{x}$, and $\Delta_{\mathbf{u}}$ is the discriminant of the quadratic form $f_{\mathbf{u}}$

$$\Delta_{\mathbf{u}} = \|\mathbf{u}_\ell\|^2 - 4\, u_q u_c = \frac{1}{\sum_i w_i} \sum_i w_i\, \nabla f_{\mathbf{u}}(\mathbf{p}_i)\,.\,\mathbf{n}_i. \tag{1.11}$$

It is equal to the squared norm of the gradient of the points lying on the 0-isosurface, and also to the average dot products between the normals $\mathbf{n}_i$ and the scalar field gradients at $\mathbf{p}_i$ (right-hand side of Equation 1.11). In the literature, $\Delta_{\mathbf{u}}$ is referred to as a normalization [Pratt 1987], or as fitness [Mellado 2012]. Note that if $\Delta_{\mathbf{u}} \leq 0$ then the scalar field is degenerated and does not describe any surface, but this extreme situation is never met in practice. In the end, the complete orthogonal projection [Alexa 2004] of the APSS defining the surface of Equation 1.2 is the limit

$$\phi(\mathbf{x}) = \lim_{n \to \infty} \varphi^n_{\mathbf{x}}(\mathbf{x}), \tag{1.12}$$

where the exponent denotes the composition of function. In practice, the iterative projections stop when a maximal number of steps is reached, or when the difference between two successive projections is negligible.

The first benefit of the APSS is its performance since one MLS iteration only involves simple summations over the neighborhood without any complex system to store nor to solve. Secondly, the resulting algebraic sphere is analytically differentiable which gives access to principal curvature directions although the sphere is originally an isotropic shape. The Ponca library [Mellado 2020] provides an implementation of the shape operator following this idea. As we show in Section 1.4, this shape operator is only partial, so we propose a more accurate and more robust version. Derivatives with respect to the support size t of the weighting function of Equation 1.6 are also available. This opens the door to an efficient multi-scale framework [Mellado 2012] discussed in Section 1.2.2.

Primary usages of PSS include ray tracing [Adamson 2003a, Adamson 2003b], triangulation [Scheidegger 2005], interactive modeling [Zwicker 2002] and real-time rendering [Alexa 2003, Guennebaud 2008]. This thesis adopts another point of view. The PSS essentially aim at estimating differential quantities of the underlying surface such as normals [Alexa 2004] and curvatures [Yang 2007]. It is the fundamental basis of our higher level of analysis for the extraction of planes (Chapter 2), lines and cylinders (Chapter 3). We slowly leave the infinitesimal world of the MLS approach to reach a more local frame of analysis composed of planar patches (Section 2.3.1) and flow lines (Section 3.3)). An even more global viewpoint is also adopted in Chapter 4 where a global parametrization is computed for the

whole point cloud. Besides the various advantages of the APSS presented previously, one of the main reason for choosing them for local surface approximation is their ability to define an intuitive parameter of scale as explained in the following section.

1.2.2 Multi-scale analysis

The notion of scale is widely spread in the digital signal processing community. The goal is usually to separate noise, small details, and broader elements composing the signal. In our context of geometric feature detection, a multi-scale representation of the point cloud has several advantages. Noise and irregularities on the sampling are robustly handled. It also enables more variety in the detected features, from small details to wider shapes. We review in this section the main approaches to model a 3D surface at multiple scales and emphasize their use on 3D unstructured point clouds.

Spectral methods Discrete 1D signals and 2D images can be decomposed into a sum of basis functions of different frequencies using discrete Fourier transform or wavelets. Basically, noise would correspond to the highest frequencies, details to medium ones while global variations are related to the lowest ones. The core element of spectral analysis of 3D triangular meshes [Taubin 1995] is the Laplace-Beltrami operator (LBO), which extends the Laplace operator on surfaces. Eigenfunctions of the LBO that are usually called Manifold Harmonics constitute an orthonormal basis in which the vertices coordinates are decomposed, and its eigenvalues magnitude play the role of frequencies. Such extension of Fourier analysis to surfaces is a fundamental tool in geometry processing [Zhang 2010]. It is the basis of several multi-scale point-wise descriptors [Reuter 2006, Sun 2009, Aubry 2011] usually used for shape retrieval [Bronstein 2011]. Other applications include mesh compression [Karni 2000], shape matching [Ovsjanikov 2012], segmentation [Sharma 2009, Huang 2009, Reuter 2009] and quadrangulation [Dong 2006, Huang 2008, Ling 2015] among others. A multi-scale representation can also be obtained by reconstructing the shape using only specific ranges of frequencies [Vallet 2008].

On unstructured point clouds, computing the LBO is not as clear as on manifold triangular meshes [Meyer 2003]. The umbrella operator (or graph laplacian) [Taubin 1995, Desbrun 1999] is the simplest option, but it is often subject to large errors due to its coarse approximation. Better approximations are based on local [Belkin 2009, Liu 2012, Qin 2018] or global Delaunay triangulations [Sharp 2020]. They cast the LBO discretization problem from unstructured point clouds to the more standard mesh domain. A completely mesh-free LBO can be obtained using Smoothed Particle Hydrodynamics [Petronetto 2013] involving a computational intensive optimization procedure. Another possibility is to decompose the point cloud onto simpler patches and to perform spectral analysis on each of them [Pauly 2001]. The complexity is largely reduced, but the patch decomposition and blending are crucial and not really straightforward.

One of the main drawback of such spectral approaches is their global nature as the whole data is processed at once. Even if the LBO is represented as a sparse matrix, the eigendecomposition remains inefficient regarding both memory and time when processing large point clouds. Only a subset of eigenvectors associated to the smallest eigenvalues

magnitude are usually required which speeds up their computations thanks to shift-invert solvers [Vallet 2008]. Still, computing 100 eigenvalues and eigenvectors for 600K points could take several hours [Liu 2012, Table 1].

Recent work address the performance issue [Nasikun 2018] or the lack of locality [Melzi 2018] for Laplacian mesh analysis. However, the LBO is intrinsic in essence as it only considers geodesic distances along the surface. This feature explains the success of spectral methods for isometric problems such as shape matching, deformation and segmentation. On the other hand, it can be considered as a problem since isometric but very different surfaces embedded in 3D cannot be distinguished. For pattern recognition on acquired point clouds, discerning such shapes is important, which limits the use of spectral frameworks. Extrinsic operators [Liu 2017, Wang 2018b] are proposed to solve this issue, but they are only defined on triangular meshes and are costly to apply on data with more than a million of vertices.

Finally, the notion of scale used in spectral analysis is not truly intuitive. The magnitude and the rank of an eigenvalue are difficult to link to some meaningful geometric properties of the shape. Furthermore, the parameter controlling the size of the neighborhood used to compute the LBO on point cloud defines an additional scale parameter on top of the eigenvalues that is not studied in the previously mentioned references.

The scale-space theory The scale-space refers to a family of progressively smoothed version of a digital signal where a scale parameter $t \in \mathbb{R}^+$ controls the width of a Gaussian kernel convolving the signal [Witkin 1987]. Applied to an image, it produces a discrete stack of images going from the original detailed data to a highly blurred image. The Gaussian scale-space of an image F is actually the solution of the diffusion equation [Koenderink 1984]

$$\dot{F} = \Delta F, \tag{1.13}$$

subject to initial condition $F_0 = F$, where $\dot{F}$ is the temporal derivative and ΔF is the Laplacian of F. It has the nice property of non-enhancement of local maxima [Lindeberg 1990], meaning that the image is necessarily smoother as the scale grows. Differential properties of the scale-space offer a large variety of features that are invariant to translation, rotation and scaling [Lowe 1999]. Nevertheless, the scale-space is based on the regular grid structure of images, which makes it non-trivial to apply on 3D meshes and point clouds.

A first extension to 3D point clouds proposes to compute a local geometric feature called Surface Variation [Pauly 2002, Equation 5] from the k-nearest neighbors with several values of k [Pauly 2003]. Similar scale-space methods with varying neighborhood size are also used to detect interest regions in 3D point clouds [Unnikrishnan 2008], or for the analysis of volumetric data [Levallois 2015]. However, the Surface Variation is not truly discriminative. It mixes principal curvatures [Digne 2014, Theorem 4], so two different shapes can have a similar descriptor. Nevertheless, projecting each point on their local PCA plane (Equation 1.1) asymptotically amounts to a mean curvature flow [Digne 2011] that corresponds to a discretization of Equation 1.13. In this approach, the neighborhood size used to compute the PCA is fixed, so the diffusion process is very slow. Overall, this kind of techniques suffers from the instabilities of the covariance analysis.

The Point Set Surfaces [Alexa 2001] introduced in Section 1.2.1 are good candidates for a

scale-space representation of a point cloud. They involve a scale parameter in their weighting function [Pauly 2006] and can process scattered points without requiring any spatial parametrization. The Growing Least Squares (GLS) [Mellado 2012] develop this idea and take advantage of the analytical expression of the algebraic sphere fit (Equations 1.7-1.9) introduced in the APSS [Guennebaud 2008]. The support size $t \in \mathbb{R}^+$ of the weighting function given in Equation 1.6 plays the role of the scale parameter. The combination of the APSS and the scale-space theory results in a useful depiction of an unstructured 3D point cloud. The closed form expressions of Equations 1.7-1.9 can be differentiated according to both scale and space. This leads to pertinent GLS descriptors detecting geometric changes during scale variations, which is used for shape matching [Mellado 2012], registration [Mellado 2015a] and modeling [Nader 2014]. Although the GLS characterize the shape at multiple scales, they follow the MLS structure by considering one local surface approximation for each spatial position. For this reason, they are essentially pointwise and make difficult the extraction of more regional features such as planar and cylindrical parts. We thus propose in Chapters 2 and 3 a higher level of analysis starting from this GLS-based multi-scale representation, and Chapter 4 investigates the limit of the scale-space when t tends to infinity.

One potential drawback of using the APSS as a scale-space approach comes from neighborhood queries at high scale. The use of a kd-tree is very efficient for a small range query since few nodes are visited. When the scale parameter t grows, we may loose the performance advantage by visiting much more kd-tree nodes. To overcome this problem, we propose in Section 1.5 a new multi-resolution approach coupled to the multi-scale representation.

Finally, a theoretical question arises regarding the projection on a fitted algebraic sphere (Equation 1.10). While the projection of a point onto a PCA plane asymptotically corresponds to a mean curvature flow [Digne 2011, Theorem 2], we do not have such knowledge on the algebraic sphere. Thanks to the closed form expression of the fit, we perform in Section 1.3 an asymptotic analysis of the algebraic sphere regression and projection to get a better understanding of the APSS from the integral invariant viewpoint [Pottmann 2007].

1.3 Asymptotic analysis of algebraic sphere regression

This section presents an asymptotic analysis of the algebraic sphere fit and projection involved in the APSS. The regression has an analytical solution given in Equations 1.7-1.9 that only requires local summations over the neighborhood of basic quantities related to positions and normals. This simple observation leads to the following important remark: fitting an algebraic sphere is actually linked to integral invariants that are well studied in the geometry processing community [Manay 2004, Clarenz 2004, Pottmann 2007, Pottmann 2009, Digne 2014]. Many of these work focus on PCA, but no result exist regarding the algebraic sphere fit. This clearly raises the following question: what are the differential invariants linked to the coefficients u_c, $\mathbf{u}_\ell$ and u_q of the best fitting sphere? The same question applies for the projection of a point onto the sphere for which an analytical solution is also available (Equation 1.10).

Section 1.3.1 first introduces the usual asymptotic framework adopted for the investigation of integral invariants. We prove in Section 1.3.2-Theorem 1 that the fitted sphere gives access to the mean curvature, a measure of anisotropy, and higher order surface derivatives.

Section 1.3.3 and Theorem 2 gives similar results but for the projection of a point onto the algebraic sphere. We also determine the robust geometric flow defined by the iterative projection onto the algebraic sphere, which is the counterpart of the mean curvature flow for the PCA plane [Digne 2011].

1.3.1 Asymptotic settings

We study a smooth regular surface S embedded in $\mathbb{R}^3$ and focus the analysis to one of its point $\mathbf{p} \in \mathbb{R}^3$ and its close neighborhood within a fixed distance $t \in \mathbb{R}$. The frame of analysis is the so-called principal frame [Pottmann 2007] (also called local canonical frame [Do Carmo 1976, Section 1.6] or local intrinsic coordinate system [Digne 2011]), where $\mathbf{p} = \mathbf{0}$ is placed at the origin, and the surface S is locally expressed as a height field over its tangent plane by using the mapping

$$\mathbf{f}(x, y) = \begin{bmatrix} x & y & z(x, y) \end{bmatrix}^T . \tag{1.14}$$

The coordinates x and y on the plane are aligned with the directions of principal curvatures κ_1 and κ_2. The height z is given by the following Taylor expansion of order 4

$$z(x, y) = \frac{1}{2}\left(\kappa_1 x^2 + \kappa_2 y^2\right) + \sum_{k=3}^{4} \sum_{j=0}^{k} \binom{k}{j} \frac{x^j y^{k-j}}{k!} a_{j,k-j} + o\left(x^4 + y^4\right). \tag{1.15}$$

A lower order would not be sufficient to obtain the results presented in this section. The coefficients $a_{j,k-j} = \frac{\partial^k z}{\partial x^j \partial y^{k-j}}$ correspond to the successive derivatives of z evaluated at $\mathbf{p}$. For a better understanding, the principal curvatures are explicitly written as $\kappa_1 = a_{20}$ and $\kappa_2 = a_{02}$, and the mean and Gaussian curvature are respectively denoted by $H = (\kappa_1 + \kappa_2)/2$ and $K = \kappa_1 \kappa_2$. Note that this local principal frame is chosen so that $a_{00} = a_{10} = a_{01} = a_{11} = 0$. The Laplace operator applied to the mean curvature, which is also half of the bilaplacian of z, is also explicitly denoted by $\Delta H = \frac{1}{2}\Delta^2 z = \frac{1}{2}(a_{40} + 2a_{22} + a_{04})$. In this smooth setting, any discrete sum appearing in Equations 1.7-1.9 is replaced by an integral over the surface patch

$$\mathcal{P}_t = \mathcal{B}_t(\mathbf{p}) \cap S = \left\{(x, y) \in \mathbb{R}^2,\ \|\mathbf{f}(x, y)\| < t\right\}, \tag{1.16}$$

where $\mathcal{B}_t(\mathbf{p})$ is the ball of center $\mathbf{p}$ and radius t. Note that for simplicity, this study considers a constant weighting $w_i = 1$ instead of the smooth decreasing weighting kernel of Equation 1.6 that is used in practice. Apart from changing multiplicative constants, this modification does not impact the results of the two following sections.

1.3.2 Algebraic sphere fitting

Theorem 1 *The parameters of the algebraic sphere fitted to the surface patch $\mathcal{P}_t$ have the following asymptotic expansions*

$$u_q = -\frac{H}{2} + o(1), \tag{1.17}$$

$$\mathbf{u}_\ell = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{a_{30}+a_{12}}{8} \\ \frac{a_{03}+a_{21}}{8} \\ \frac{H^2-K}{4} \end{bmatrix} t^2 + o(t^2), \tag{1.18}$$

$$u_c = -\frac{1}{96}(9H^3 - 5KH - \Delta H)t^4 + o(t^4). \tag{1.19}$$

Moreover, the norm of $\mathbf{u}_\ell$ is given by

$$\|\mathbf{u}_\ell\| = 1 - \frac{H^2 - K}{4}t^2 + o(t^2). \tag{1.20}$$

The full proof is given in Appendix A.3.

Interpretation A first remark concerns the parameter u_q that contains the mean curvature H of the surface in its preponderant term. If we calculate the surface mean curvature $\tilde{H}$ from the scalar field $f_{\mathbf{u}}$ itself (Equation 1.3) as half the trace of its shape operator (see Equation 1.32 for more details), we obtain $\tilde{H} = 2u_q/\sqrt{\|\mathbf{u}_\ell\|^2 - 4u_q u_c} = -H + o(t)$, which tends to H (up to a sign convention). It means that the mean curvature estimated by the APSS not only is the curvature of some geometric model fitted to the data as a proxy but also asymptotically converges toward the actual mean curvature of the surface. Figure 1.1-right illustrates how well u_q is proportional to H.

The linear parameter $\mathbf{u}_\ell$ converges to the normal of the surface according to Equation 1.18. In addition, its norm deviates from 1 by a multiple of $H^2 - K = (\kappa_1 - \kappa_2)^2/4$. This positive quantity measures the anisotropy of the shape and is equal to zero only if $\kappa_1 = \kappa_2$, which is true for the sphere and the plane. It can be seen on Figure 1.1-middle that $1 - \|\mathbf{u}_\ell\|$ takes high values around very anisotropic regions such as the crease of the neck, the edge of the ear and the outline of the hair curls. Medium values appear along the nose and the eyebrow. Zero is reached on spherical regions like the tip of nose or the chin, and on mostly planar areas like the jowl. The discriminant Δ of the algebraic sphere (Equation 1.11) is very similar as its asymptotic expansion since $\Delta = 1 - \frac{H^2-K}{2}t^2 + o(t^3)$. It also quantifies the deviation of the surface from a sphere or a plane, which validates its role of fitness score used in prior work [Mellado 2012].

Finally, the parameter u_c, that is also the algebraic distance between the analyzed point $\mathbf{p}$ and the surface, involves combination of H, K and ΔH. It gives access to higher order derivatives of the surface through ΔH. It explains why u_c exhibits higher frequencies than $\mathbf{u}_\ell$ or u_q as shown by Figure 1.1-left. The extent of its variations are usually smaller than the scale t used to gather the neighborhood of each point for the sphere fitting.

Similar features called the GLS parameters τ, $\boldsymbol{\eta}$ and κ [Mellado 2012] are equal to u_c, $\mathbf{u}_\ell$ and u_q divided by $\sqrt{\Delta}$ so that the gradient of $f_{\mathbf{u}}$ is unitary on the 0-isosurface. They are

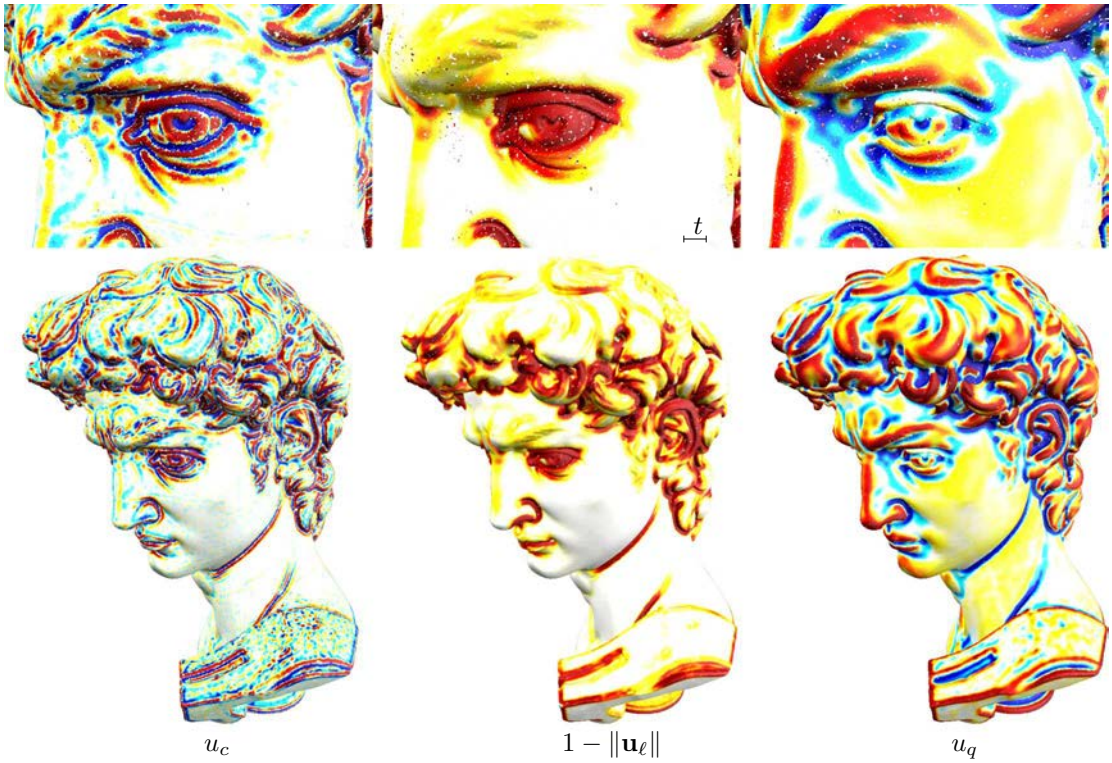


Figure 1.1: Algebraic sphere parameters. Left: u_c highlights high frequencies of the surface. Middle: The deviation of $\|\mathbf{u}_\ell\|$ from 1 measures the anisotropy of the shape. Right: u_q estimates the mean curvature H. Values are negative in blue, null in white and positive in red. These visualizations are consistent with the asymptotic results of Theorem 1. The scale t used to fit the sphere is shown in the middle top figure.

actually equivalent in this asymptotic setting except for $\boldsymbol{\eta}$ that loses its capacity to measure anisotropy as $\mathbf{u}_\ell$ does through its norm. This is not surprising because $\mathbf{u}_\ell$ and Δ describe equivalently the surface anisotropy so their quotient cannot naturally maintain this property.

1.3.3 Algebraic sphere projection

Theorem 2 *If $u_q = 0$, a plane is fitted to the surface patch $\mathcal{P}_t$ and the projection $\varphi(\mathbf{p})$ of the origin point $\mathbf{p}$ asymptotically yields*

$$\varphi(\mathbf{p}) = \begin{bmatrix} 0 & 0 & -\frac{\Delta H}{96} \end{bmatrix}^T t^4 + o(t^4), \tag{1.21}$$

otherwise the projection on the fitted sphere is

$$\varphi(\mathbf{p}) = \begin{bmatrix} 0 & 0 & \frac{1}{96}(9H^3 - 5KH - \Delta H) \end{bmatrix}^T t^4 + o(t^4). \tag{1.22}$$

The proof stems from Theorem 1 and Equation 1.10 and is detailed in Appendix A.4. Note that since $\mathbf{p}$ is supposed to be the origin of the principal frame, the right-hand sides of Equation 1.21 and 1.22 are also equal to the displacement vectors $\varphi(\mathbf{p}) - \mathbf{p}$ induced by the projection operator.

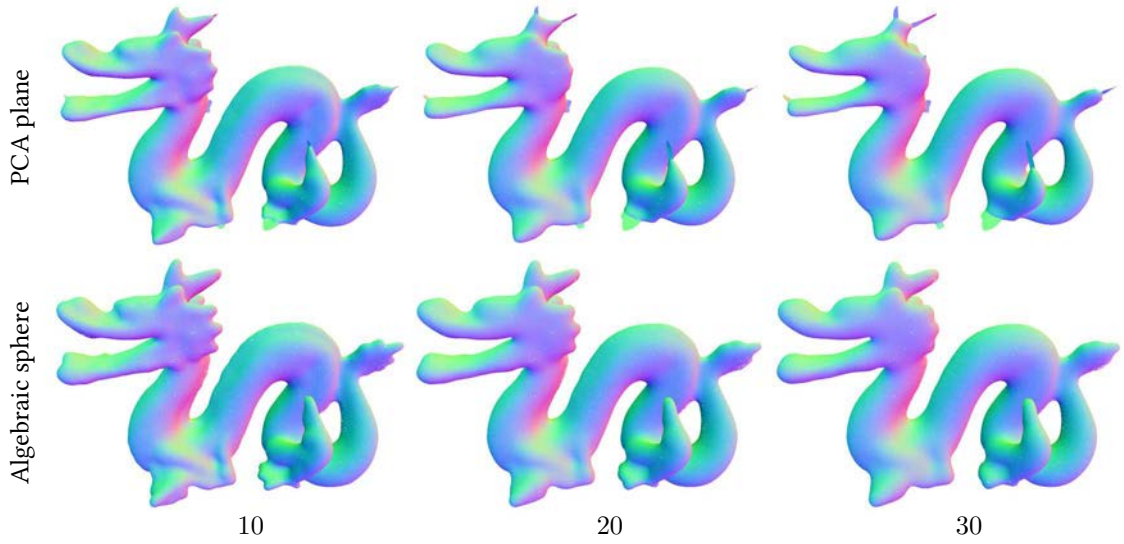


Figure 1.2: Geometric flows comparison. Three intermediate steps of iterative projections onto PCA planes (top) and algebraic spheres (bottom). The first exhibits singularities near elongated regions which is typical of the mean curvature flow. Our more stable flow avoids such extreme shrinkage.

Interpretation To better understand the meaning of Theorem 2, we can look at what happens when every points of a surface S are iteratively projected onto the algebraic sphere fitted to their neighborhoods. This iterative procedure is equivalent to a forward Euler scheme $\mathbf{p}^{n+1} = \mathbf{p}^n + \varphi(\mathbf{p}^n)$. Using Theorem 2 and the fact that if u_q is null then $H = 0$ (Equation 1.17), the iterative projections on the best fitting algebraic spheres are the solution of the following diffusion equation

$$\dot{S} = \lambda(9H^3 - 5KH - \Delta H)\mathbf{n} \tag{1.23}$$

where $\dot{S}$ is the derivative with respect to time, λ is a positive diffusion coefficient, and $\mathbf{n}$ is the normal vector of S. If the surface evolves under this flow, each point moves along the normal direction with a velocity proportional to $(9H^3 - 5KH - \Delta H)$. If $H = 0$, the flow amounts to a bilaplacian flow. This results in a robust geometric fairing process that is compared to the PCA-based mean curvature flow [Digne 2011] in Figure 1.2. The singularities of the mean curvature flows happening near the medial axis of the shape are not present in our algebraic sphere flow. Instead, the surface under our flow is more and more rounded and does not excessively shrink as shown by Figure 1.2 near the extremities of the dragon tail and horns.

1.4 Robust differential properties estimation

Differential properties such as the normal vector and the curvatures of a surface plays an important role in our studies. Their estimation needs to be as accurate as possible to avoid any loss of quality in further processing. When dealing with acquired data, their robustness to noise is one of the main critical aspect.

We develop in Section 1.4.2 an accurate method to estimate principal curvatures from the

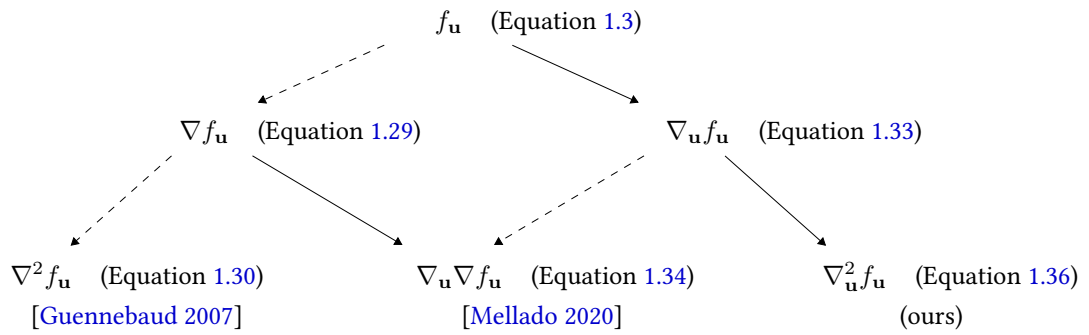


Figure 1.3: Scalar field differentiation. Dashed arrows represent *partial* differentiation denoted by $\nabla \cdot$ where the algebraic sphere parameters $\mathbf{u}$ are considered as constant. Solid arrows represent *complete* differentiation denoted by $\nabla_{\mathbf{u}} \cdot$ that considers space-varying $\mathbf{u}$ as a result of the fitting procedure. The three Hessian matrices below lead to different shape operators given in Equations 1.31, 1.35 and 1.38.

APSS. The way the APSS scalar field is differentiated is the key element of our approach as illustrated by Figure 1.3. The proposed estimator includes all the advantages of the APSS such as its efficiency and its approximation power. A numerical evaluation is performed in Section 1.4.3 on simple and dense points clouds to assess its robustness to different types of noise. Compared to baseline techniques such as PCA, Osculating Jets [Cazals 2005a], PSS [Alexa 2001] and prior APSS-based approaches [Guennebaud 2007, Mellado 2012] our method is empirically more accurate and robust. Section 1.4.1 first introduces general curvatures computations and then describes the existing curvature estimations on APSS.

1.4.1 Prior work on APSS curvatures

Since APSS represent the surface implicitly, we first give a brief background on curvature computation on a generic implicit surface. Two existing methods are then described. The first is the initial APSS [Guennebaud 2007] that estimates only the mean curvature using a simple version of the shape operator. The second is implemented in the Ponca library [Mellado 2020] where a better approximation of the shape operator is used.

Implicit surface curvatures The normal $\mathbf{n}$ of an implicit surface defined by an iso-surface of a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$ is defined by the normalized gradient

$$\mathbf{n} = \frac{\nabla f}{\|\nabla f\|}. \tag{1.24}$$

The normal curvature $\kappa_{\mathbf{n}}$ in a tangent direction $\mathbf{t} \in \mathbb{R}^3$ is equal to the variation of the normal in that direction. It is given by the following quadratic form

$$\kappa_{\mathbf{n}}(\mathbf{t}) = \mathbf{t}^T \nabla \mathbf{n} \mathbf{t}. \tag{1.25}$$

The spatial derivative of the normal $\nabla\mathbf{n}$ is obtained from Equation 1.24 as

$$\nabla\mathbf{n} = \left(I_3 - \mathbf{n}\mathbf{n}^T\right)\frac{\nabla^2 f}{\|\nabla f\|}, \tag{1.26}$$

where I_3 is the 3-by-3 identity matrix and $\nabla^2 f$ is the Hessian matrix of the scalar field. By computing an orthonormal basis $\{\mathbf{t}_1, \mathbf{t}_2, \mathbf{n}\}$ of the tangent plane, we express any 3D tangent vector $\mathbf{t}$ by a linear combination $P\mathbf{u}$, with $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^T$ its coordinates in the tangent space and $P = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 \end{bmatrix}$ the 3-by-2 transfer matrix from the 2D tangent plane to the 3D space. The normal curvature of Equation 1.25 expressed in the tangent plane corresponds to the second fundamental form of the surface

$$\kappa_{\mathbf{n}}(\mathbf{u}) = \mathbf{u}^T W \mathbf{u}, \tag{1.27}$$

where W is the matrix of the shape operator that is simplified to

$$W = P^T \nabla\mathbf{n} P = P^T \frac{\nabla^2 f}{\|\nabla f\|} P. \tag{1.28}$$

Finally, the principal curvatures κ_1 and κ_2 of the surface are the eigenvalues of W, which correspond to extrema of the normal curvature $\kappa_{\mathbf{n}}$. Note that every quantities introduced in this paragraph depend on the evaluation point $\mathbf{x}$ that is omitted for clarity. Computing the principal curvatures of the APSS boils down to calculate the gradient ∇f and the Hessian matrix $\nabla^2 f$ of the scalar field $f_{\mathbf{u}}$ of Equation 1.3. As illustrated by Figure 1.3, the different approaches presented below differ by how they compute these first and second order derivatives.

Simple shape operator Considering the parameters $\mathbf{u}$ as constant when $\mathbf{x}$ varies is the simplest way to differentiate the scalar field $f_{\mathbf{u}}(\mathbf{x}) = u_c + \mathbf{u}_\ell \,.\, \mathbf{x} + u_q \mathbf{x} \,.\, \mathbf{x}$ (Equation 1.3). This approach does not account for all the fitting procedure and considers the resulting scalar field as a pure algebraic sphere only. This *partial* differentiation leads to the following approximation of the gradient and Hessian

$$\nabla f_{\mathbf{u}} = \mathbf{u}_\ell + 2u_q\mathbf{x}, \tag{1.29}$$

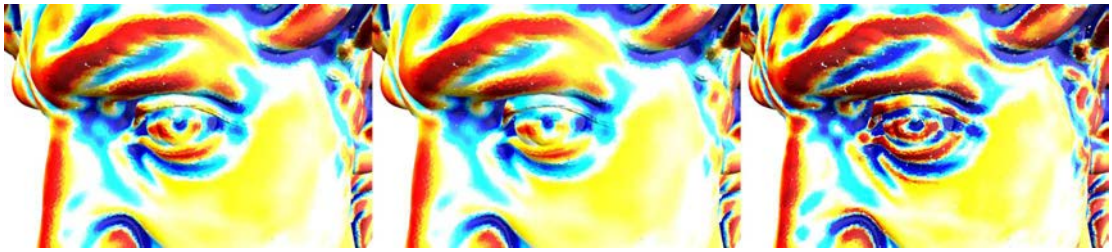
$$\nabla^2 f_{\mathbf{u}} = 2u_q I_3, \tag{1.30}$$

If u_q is null, then the gradient is the orthogonal vector $\mathbf{u}_\ell$ defining the plane, otherwise it is a vector in the direction between the sphere center and $\mathbf{x}$. Normalizing the gradient $\nabla f_{\mathbf{u}}$ defines the normal $\mathbf{n}$ of the surface and is also denoted by $\boldsymbol{\eta}$ in the GLS parameters [Mellado 2012].

The shape operator resulting from Equations 1.29 and 1.30 is

$$W_1 = \frac{2u_q I_2}{\|\mathbf{u}_\ell + 2u_q\mathbf{x}\|}. \tag{1.31}$$

This diagonal matrix makes the principal curvatures necessarily equal which is not surprising since we only consider the plain algebraic sphere. Considering only the resulting sphere definitely leads to wrong principal curvatures unless the surface is isotropic. On the other hand, it is still capable of estimating the mean curvature. Calculating W_1 at $\varphi(\mathbf{x})$ (the projection of



$\tilde{H}$ Equation 1.32 [Guennebaud 2007] W_2 Equation 1.35 [Mellado 2020] W_3 Equation 1.38 (ours)

Figure 1.4: Mean curvature comparison. Our shape operator (right) produces more visually accurate mean curvatures than previous methods. More details are visible in the eye hollow and on the upper eyelid. Values range from -0.2 in blue to $+0.2$ in red via 0 in white.

$\mathbf{x}$ onto the sphere (Equation 1.10)) gives the mean curvature

$$\tilde{H} = \frac{2u_q}{\sqrt{\|\mathbf{u}_\ell\|^2 - 4u_q u_c}}. \tag{1.32}$$

It is equal to the inverse radius of the algebraic sphere as initially proposed by the APSS method [Guennebaud 2007], and it also corresponds to the GLS parameter κ.

Approximate shape operator Thanks to the analytical formulas of the fitted algebraic sphere given in Equations 1.7-1.9, the scalar field parameter $\mathbf{u}$ is actually a function of $\mathbf{x}$. By taking into account this dependence we no longer see the resulting scalar field $f_\mathbf{u}$ as only a simple algebraic sphere. The Ponca library [Mellado 2012] performs this *complete* differentiation of the scalar field that gives a second variant of the gradient

$$\nabla_\mathbf{u} f_\mathbf{u} = \nabla u_c + \mathbf{u}_\ell + \nabla \mathbf{u}_\ell^T \mathbf{x} + 2u_q \mathbf{x} + \nabla u_q \mathbf{x} \,.\, \mathbf{x}, \tag{1.33}$$

where $\nabla \mathbf{u}_\ell$ is the Jacobian matrix of $\mathbf{u}_\ell$. The subscript $\mathbf{u}$ in $\nabla_\mathbf{u}$ means that a complete differentiation is done (see Figure 1.3). The partial differentiation of $\nabla_\mathbf{u} f_\mathbf{u}$ is performed giving a partial version of the Hessian matrix

$$\nabla_\mathbf{u} \nabla f_\mathbf{u} = \nabla \mathbf{u}_\ell^T + 2u_q I_3 + 2\mathbf{x} \nabla u_q^T. \tag{1.34}$$

Note that this is also equivalent to the complete differentiation of $\nabla f_\mathbf{u}$ (see Figure 1.3). It leads to the second approximate shape operator

$$W_2 = \frac{P^T (\nabla \mathbf{u}_\ell^T + 2u_q I_3 + 2\mathbf{x} \nabla u_q^T) P}{\|\nabla u_c + \mathbf{u}_\ell + \nabla \mathbf{u}_\ell^T \mathbf{x} + 2u_q \mathbf{x} + \nabla u_q \mathbf{x} \,.\, \mathbf{x}\|}, \tag{1.35}$$

where P is a tangent plane basis computed from $\nabla_\mathbf{u} f_\mathbf{u}$ (see Equation 1.28). Contrary to W_1, W_2 is no longer diagonal, so its eigenvalues can be different, which enables the calculation of principal curvatures.

1.4.2 Accurate APSS shape operator

As shown by Figure 1.3, none of the two existing Hessian matrices and their associated shape operators take completely into account the dependence of $\mathbf{u}$ to $\mathbf{x}$. We thus propose a more accurate form of Hessian matrix by performing the complete differentiation of the complete gradient $\nabla_{\mathbf{u}} f_{\mathbf{u}}$ (Equation 1.33 [Mellado 2020]). We obtain the following Hessian matrix

$$\nabla^2_{\mathbf{u}} f_{\mathbf{u}} = \nabla^2 u_c + \nabla \mathbf{u}_\ell + \nabla \mathbf{u}_\ell^T + \nabla^2 \mathbf{u}_\ell \mathbf{x} + 2\nabla u_q \mathbf{x}^T + \mathbf{x}\,.\,\mathbf{x} \nabla^2 u_q + 2u_q I_3 + 2\mathbf{x} \nabla u_q^T. \quad (1.36)$$

In 3D, the term $\nabla^2 \mathbf{u}_\ell \mathbf{x}$ is the product between the rank-3 tensor $\nabla^2 \mathbf{u}_\ell$ and the 3 dimensional vector $\mathbf{x}$ that is equal to

$$\nabla^2 \mathbf{u}_\ell \mathbf{x} = x \nabla^2 \mathbf{u}_{\ell x} + y \nabla^2 \mathbf{u}_{\ell y} + z \nabla^2 \mathbf{u}_{\ell z}, \quad (1.37)$$

where $\nabla^2 \mathbf{u}_{\ell x}$ is the Hessian matrix of the first coordinate of $\mathbf{u}_\ell$. Plugging $\nabla^2_{\mathbf{u}} f_{\mathbf{u}}$ in Equation 1.28 together with $\nabla_{\mathbf{u}} f_{\mathbf{u}}$ yields the complete shape operator

$$W_3 = P^T \frac{\nabla^2_{\mathbf{u}} f_{\mathbf{u}}}{\|\nabla_{\mathbf{u}} f_{\mathbf{u}}\|} P. \quad (1.38)$$

Its eigenevalues are not the curvatures of a simple algebraic sphere as those of W_1 but correspond to the actual curvatures of the reconstructed MLS surface. The difference between W_2 and W_3 is difficult to appreciate by looking only at their equations. Figure 1.4 shows the mean curvatures computed with the different methods, where we can already observe that W_3 produces more accurate mean curvatures on a clean shape. The numerical comparison conducted in the next section validates that our shape operator produces curvatures that are more accurate and more robust in the presence of noise than those obtained with previous methods.

1.4.3 Numerical comparison

We perform a comparison between our approach to compute curvatures with several baseline methods. Dense point clouds sampling smooth geometric primitives are used as input so that the theoretical values of curvatures are known. The goal is to show that differentiable properties calculated from the APSS, especially using our complete shape operator, are both accurate and robust to noise on positions and normals. The main result is two plots of several estimators as a function of noise presented in Figure 1.6 and 1.7.

Input data The comparison is performed on three primitives : the unit sphere, a cylinder of radius 1 and an hyperbolic paraboloid corresponding to a saddle shape. The theoretical principal curvatures (κ_1, κ_2) are $(1, 1)$, $(1, 0)$ and $(2, -1)$, and the theoretical mean curvature H is 1, 0.5 and 0.5 for the sphere, cylinder and saddle respectively. All the estimations take place on the same point on these surfaces and uses a radius equal to 1 to define the neighborhood. The primitives are randomly sampled as dense point clouds, and around 800K points are located within the neighborhood ball. Sparser data could be tested but the goal here is to measure the robustness to noise and not to sampling variation.

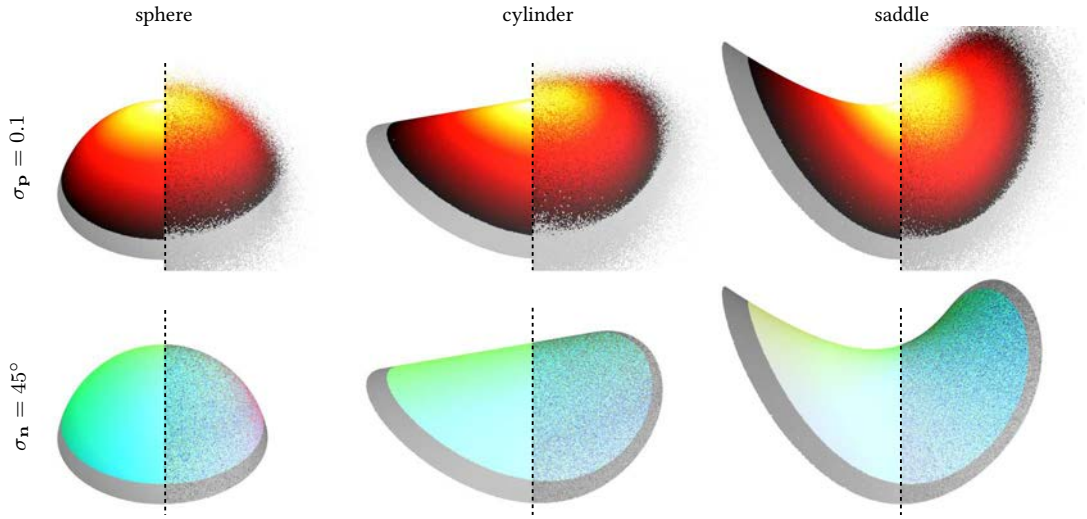


Figure 1.5: Input data. Each sub-figure shows the clean point cloud on the left and the noisy version on the right. Colors are determined by the distance to the evaluation point for the positions noise (top row) or by the normal orientation for the normals noise (bottom row).

Two types of noise are introduced in order to measure the robustness of the different estimators. In a first experiment (Figure 1.6), a Gaussian noise with zero mean and standard deviation $\sigma_{\mathbf{p}}$ is added to the coordinates of the positions. The standard deviation $\sigma_{\mathbf{p}}$ varies from 0 to 0.5. A second experiment (Figure 1.7) measures the robustness against noise on the normal vectors of the point cloud. However, a basic Gaussian noise is not enough to differentiate the accuracy between one method and another. So every normal is rotated around a globally fixed direction that is diagonal to the principal directions of curvature. The angle follows a Gaussian distribution of zero mean and standard deviation $\sigma_{\mathbf{n}}$ that is truncated to constrain the angles to be positive. The standard deviation $\sigma_{\mathbf{n}}$ varies from 0 to 45 degrees. A few examples of input data can be observed in Figure 1.5.

Selected methods The two existing approaches computing the principal curvatures from the APSS algebraic sphere presented in Section 1.4.1 are included in the evaluation under the name Sphere 1 for W_1 (Equations 1.31 [Guennebaud 2007]) and Sphere 2 for W_2 (Equations 1.35 [Mellado 2020]). Our method explained in Section 1.4.2 using the complete shape operator W_3 given by Equation 1.38 is called Sphere 3.

The PCA plane fitting gives another estimator of the mean curvature. Contrary to previous work [Pottmann 2007, Digne 2011], we use a weighted version of the PCA given in Equation 1.1 using the same weighting function of the APSS (Equation 1.6). With this minor change, the normal component of the projection displacement is asymptotically equal to $\frac{Ht^2}{8} + o(t^2)$, where t is the neighborhood radius that is equal to 1 in our experiments. Therefore, we can estimate H with $\frac{8d}{t^2}$ where d is the point-to-plane distance measured in practice. The same asymptotic expansion holds for the neighborhood barycenter. Averaging the positions and the normals is referred to as the Average method.

We also compare the pioneer PSS [Alexa 2001], the Osculating Jets [Cazals 2005a] and the Wavejets [Béarzi 2018] referred to as Quadric, Jets and Wavejets respectively. These methods

Method	δH	$\delta\kappa_1$	$\delta\kappa_2$	$\angle\mathbf{n}$
Mean	0.190	-	-	0.027
Mean (MLS)	0.304	-	-	4.844
Plane	0.190	-	-	0.082
Plane (MLS)	0.366	-	-	0.101
Sphere 1	0.148	-	-	**0.020**
Sphere 1 (MLS)	0.147	-	-	0.021
Sphere 2	0.142	0.346	0.082	0.050
Sphere 2 (MLS)	0.139	0.341	0.084	0.049
Sphere 3	0.070	0.153	**0.027**	-
Sphere 3 (MLS)	**0.067**	**0.149**	**0.027**	-
Quadric	0.266	0.453	0.238	0.054
Quadric (MLS)	0.235	0.426	0.217	0.065
Jets	0.445	0.796	0.449	29.898
WaveJets	0.383	0.690	0.359	13.754

Table 1.1: Differential properties estimation errors. Average of absolute errors on mean curvature δH, principal curvatures $(\delta\kappa_1, \delta\kappa_2)$ and normals $\angle\mathbf{n}$ (in degrees) for every shapes and every noise strength used in Figure 1.6 and 1.7. Our method (Sphere 3) gives the best results for curvatures estimation.

use the PCA plane as initial tangent plane and then fit a bivariate polynomials. The degree of the polynomials is set to 2 because it is sufficient to calculate curvatures informations. Jets and Quadric differ in the second step to perform the polynomials regression. The former solves a Vandermonde system with a size equal to the number of neighbors whereas the PSS solve a more compact system with a size equal to the dimension of the embedded space. Moreover, the PSS solve a weighted least squares problem since it is a MLS method, which is not the case for the Jet. Wavejets perform differently by considering polar coordinates decomposing the polynomials into radial magnitudes and angular oscillations.

The PCA, the Quadric and the Jets methods cannot infer the actual sign of curvatures as they do not take into account normals. Their resulting sign are modified a posteriori to match the reference normal orientation. We still include them in the second experiment involving noise on normals although they all give constant results.

Each method is tested after a single fitting step and several MLS quasi-orthogonal projections. The iterations stop when 20 steps are reached, or if the distance between two iterations points is less than 0.001 (0.1% of the neighborhood radius). We implemented our shape operator W_3 in the Ponca library [Mellado 2020] using Eigen [Guennebaud 2010]. All the other implementations also come from Ponca except for the Jets that are part of CGAL [The CGAL Project 2020], and the Wavejets that we coded in C++ from the authors Matlab function.

Results For comparison we report the mean curvature H, the principal curvatures (κ_1, κ_2) as well as the deviation $\angle\mathbf{n}$ in degree between the estimated and reference normal. These four features are plotted as a function of the positions noise standard deviation $\sigma_\mathbf{p}$ in Figure 1.6, and the normals noise standard deviation $\sigma_\mathbf{n}$ in Figure 1.7. Table 1.1 summarizes the average errors for each method during all these experiments. It includes the average of absolute errors on mean curvatures δH, principal curvatures $(\delta\kappa_1, \delta\kappa_2)$, and on the normals deviations $\angle\mathbf{n}$.

Overall, the Sphere 3 that uses our complete shape operator W_3 (Equation 1.38) gives better results. On average, we obtain the lowest errors of curvatures estimation as shown in bold font by Table 1.1. Our estimations (in red on Figures 1.6 and 1.7) are always the closest to the references and vary less than the others when both noise strengths increase. The response to noise of Sphere 1 and 2 are closer to Sphere 3 than the other methods, but their values are a bit farther from the references. Theses two experiments clearly show the accuracy and robustness of the APSS curvatures especially when the complete differentiations of its scalar field are considered as we proposed in Section 1.4.2. Although one exception is observed concerning the normal deviation $\angle\mathbf{n}$ on Figures 1.6 and 1.7 (bottom) and on the right column of Table 1.1. Sphere 1 gives slightly better results than Sphere 2 for the two types of noise. This counter intuitive observation suggests that curvatures are better estimated using the most complete shape operator W_3 while the gradient $\nabla f_{\mathbf{u}}$ (Equation 1.29) of the plain algebraic sphere is enough to estimate the normal vector in presence of noise.

Quadric, Jets and Wavejets methods show a comparable unstable behavior to deal with noise. The former is usually more accurate which confirm the benefit of considering a compact weighted least squares problem particularly when the data is noisy. The large systems that the Jets and Wavejets have to solve may also introduce more numerical errors. One common issue certainly comes from the PCA plane on which they are all based on. As shown in blue by the bottom of Figure 1.6, the noise strongly affects the normal obtained from the PCA.

The MLS version of each method does not improve much the results in these two experiments. This observation does not really reflect real cases where MLS approaches clearly improve the accuracy of the surface approximation. Possible reasons to explain this inconsistency are the simplicity of the shapes or the synthetic nature of the noise considered in these experiments. The only techniques showing a noticeable difference are Average and Plane. During the MLS iterations, the PCA plane and the barycenter gradually move away from the initial point. This well known shrinking effect (see Figure 1.2 and [Guennebaud 2007, Figures 13-15]) inevitably increases the mean curvature since it is proportional to the distance from the evaluation point to the plane or barycenter. With the Average method on the sphere, the moving point progressively shifts in an arbitrary tangent direction due to the random sampling. The neighborhood gathered around that point is no longer centered which explains why the estimated normal of Average is so biased.

Performance Figure 1.8 plots the average time spent by each method to perform the estimation at one point. It includes the three shapes and every noise type and strength used in Figure 1.6 and 1.7. We recall that all the estimations are done for one neighborhood size, so this parameter is not evaluated here although it can have an impact on the different methods. As expected, the execution time of the single fit methods is roughly comparable to the degree of the fitted shape (point, plane, sphere and quadric over plane).

Jets is the slowest due to the large system it has to solve. Although the single fitting of Average and Plane are very efficient, their MLS variants are quite slow compared to Sphere 1 for instance. The reason comes from their shrinking effect that decreases their convergence rate.

The performances for the different Sphere methods follow the complexity of their shape operators. Our shape operator W_3 (Equation 1.38) involves more terms than W_1 and W_2,

which make it approximately four and two time slower respectively. Its single fit version is slightly slower than the one of Quadric, but it is faster with the MLS version meaning that APSS converge more quickly than PSS on average in these experiments. This performance evaluation highlights one drawback of our method. Its accuracy and robustness are obtained at the price of a performance overhead. Sphere 1, 2 and 3 take around 20, 33 and 63 seconds to compute mean curvatures on 2M points in Figure 1.4. In case performance is critical, and only the mean curvature is necessary, then Sphere 1 remains a good choice. Choosing between Sphere 2 and 3 to get principal curvatures depends on the required level of accuracy and the importance given to the execution time.

1.5 Efficient multi-scale representation

The scale-space representation of a 2D image is obtained by successive Gaussian convolutions of varying support size [Witkin 1987], which corresponds to the solution of a diffusion equation [Koenderink 1984] (Equation 1.13). The diffusion-based approach can be implemented for 3D point clouds [Digne 2011] as illustrated by Figure 1.2. High frequency details of the shape are quickly removed, but it remains very slow to smooth out the low frequencies. Indeed, more than 2 minutes are required to obtain the barely smooth 30^{th} iteration of 1M points in Figure 1.2. Another standard scale-space technique consists in the variation of the neighborhood size t used for local features estimation [Pauly 2003, Yang 2006, Mellado 2012]. The APSS can be used as a local reconstruction operator that provides the necessary differential features such as normals and curvatures. When t tends toward the whole size of the shape, the approach looses its local nature and many neighbors are considered for the APSS calculations at each point. Although the APSS is fast to compute, the computational time at high scales is still too high, even if a space partitioning data structure is used.

For this reason, we introduce in Section 1.5.2 a new multi-resolution representation integrated within the scale-space framework. Taking inspiration from a prior work [Pauly 2003], the idea is to consider a low resolution version of the point cloud at high scale, so that neighbors queries maintain their efficiency. The main challenge is to appropriately link decimation and scale together. While Pauly et al. first decimate the point clouds at several resolution and then define the scale according to the local density, we take the apposite direction. We first determine the scales as explained in Section 1.5.1, and the decimation we propose in Section 1.5.2 is done accordingly. An evaluation is conducted in Section 1.5.3 to show that our approach produces an appropriate trade-off between the scale-space smoothing and the multi-resolution decimation thanks only one parameter.

1.5.1 Discrete scale-space sampling

A set of M values must be selected in $\mathbb{R}^+$ to define the scales. The method to determine the interval $(t_{\min}, t_{\max})$ defining the scale-space bounds is first explained. Then, we describe how the scales are sampled inside this interval.

Scale bounds Since the scale parameter t is simply a distance in the ambient space, these values can be linked to the size of the analyzed 3D point cloud. The highest scale $t_{\max}$ can be

reasonably bounded by the global size of the shape. We define it as the diagonal length of the axis-aligned box bounding the point cloud.

The determination of the minimal scale $t_{\min}$ is more critical. A high value implies that many smaller geometric details are smoothed and do not appear in the multi-scale representation. On the other hand, a too low value can be problematic for the APSS fitting that requires at least a few points to be stable.

We describe two ways for specifying the minimal scale $t_{\min}$. A first solution is to adapt $t_{\min}$ to the local density of the point cloud [Pauly 2002]. Similarly to balloon estimators in kernel density estimation [Terrell 1992], each point $\mathbf{p}_i$ admits its own scale-space by using the enclosing sphere radius $r_{i,K}$ of its K nearest neighbors as minimal scale. This *adaptive* scale-space ensures that each APSS projection step is stable enough and is mainly employed in Chapter 4. Linking the scale to the density is not optimal, especially when dealing with acquired data showing strongly varying sampling and noise, and the choice of K could be critical. But contrary to previous work [Pauly 2003], we perform this density-based scale selection only once to determine $t_{\min}$. However, varying sampling density still leads to very different minimal scales for each point. Analyzing the shape at one particular level of scale j leads to consider different scales at the same time. So we take the median of $r_{i,K}$ instead as a common minimal scale $t_{\min}$ for every points. This *homogeneous* scale-space is used in Chapters 2 and 3 as it is more appropriate in the context of analysis.

Scale sampling The simplest method to distribute M values in the scale interval $(t_{\min}, t_{\max})$ is the arithmetic series

$$t_j = t_{\min} + \left(\frac{j-1}{M-1}\right)(t_{max} - t_{\min}), \qquad j \in \{1, \dots, M\}, \tag{1.39}$$

where the difference between two successive scales is constantly equal to $\frac{t_{\max}-t_{\min}}{M-1}$. This *linear* scale produces a uniform scattering of scale samples with as many low values as high values. We use it in Chapter 4 to continuously map the point cloud to a common domain without too high gaps from one scale to another.

If the goal is to analyze the geometric features of the underlying surface (Chapter 2 and 3), geometric series are better used

$$t_j = \left(\frac{t_{\min}}{t_{\max}}\right)^{\frac{j-1}{M-1}} t_{\min}, \qquad j \in \{1, \dots, M\}, \tag{1.40}$$

where the ratio between two successive scales is constantly equal to $\left(\frac{t_{\min}}{t_{\max}}\right)^{\frac{1}{M-1}}$. This widely used *logarithmic* scale in scale-space methods has several benefits. In practice, only few high values are necessary because shapes usually vary less at a global scale. On the contrary, more samples are required near the lowest scale to better represent every fine details. This is perfectly accomplished by Equation 1.40. Furthermore, the logarithm transforms multiplication into addition, so scaling the geometry in the spatial domain is equivalent to a translation in the logarithm scale-space. This useful property is widely used to design scale-invariant descriptors [Kokkinos 2008, Bronstein 2010] and for data registration [Zokai 2005, Mellado 2015a].

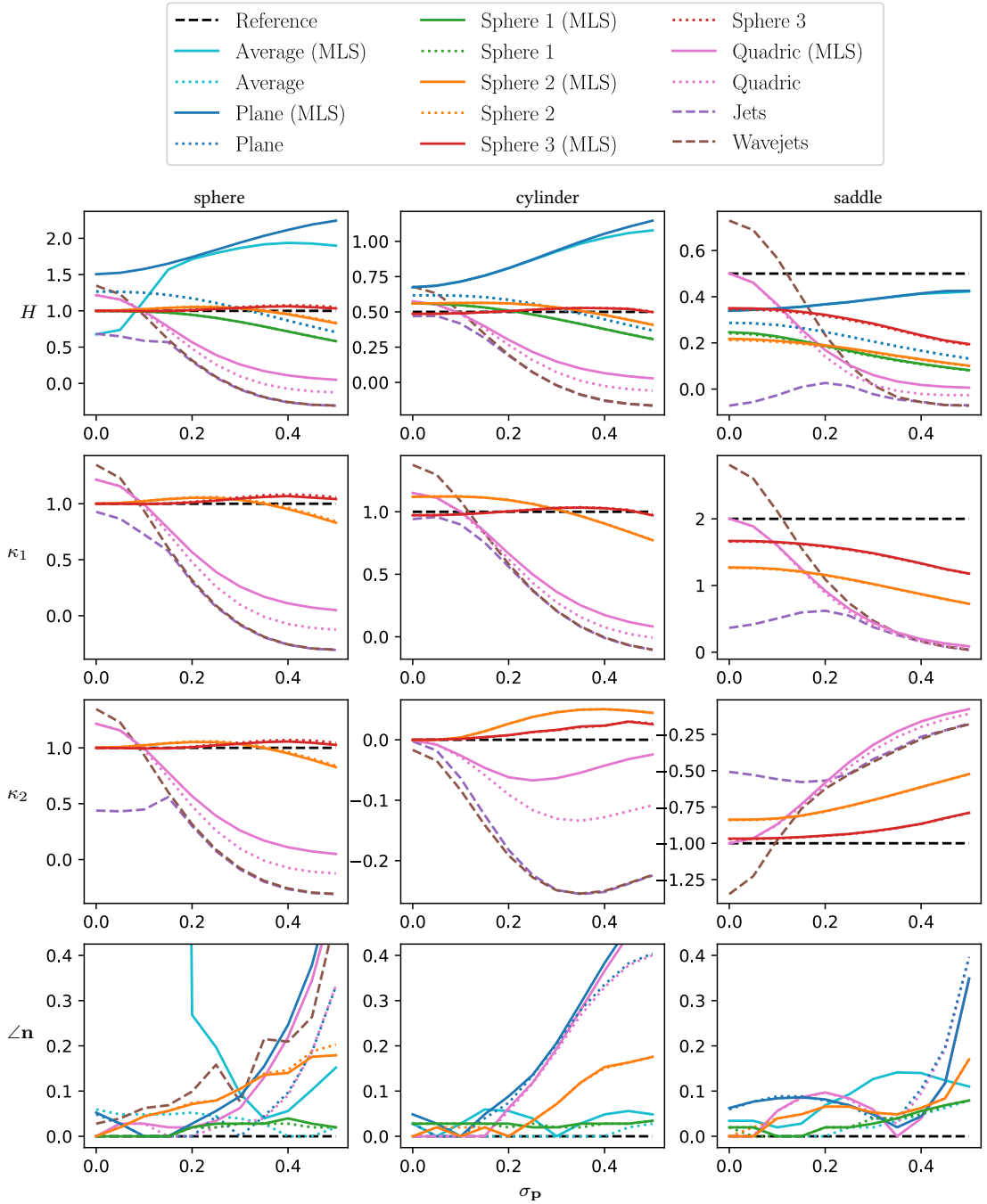


Figure 1.6: Differential properties with noise on positions. The mean curvature H, principal curvatures (κ_1, κ_2) and normal deviation $\angle\mathbf{n}$ (in degree) are plotted as a function of the position noise standard deviation $\sigma_\mathbf{p}$ with the neighborhood radius equal to 1. Our curvatures estimator (red) is the least prone to noise and is very close to the reference curvatures in general.

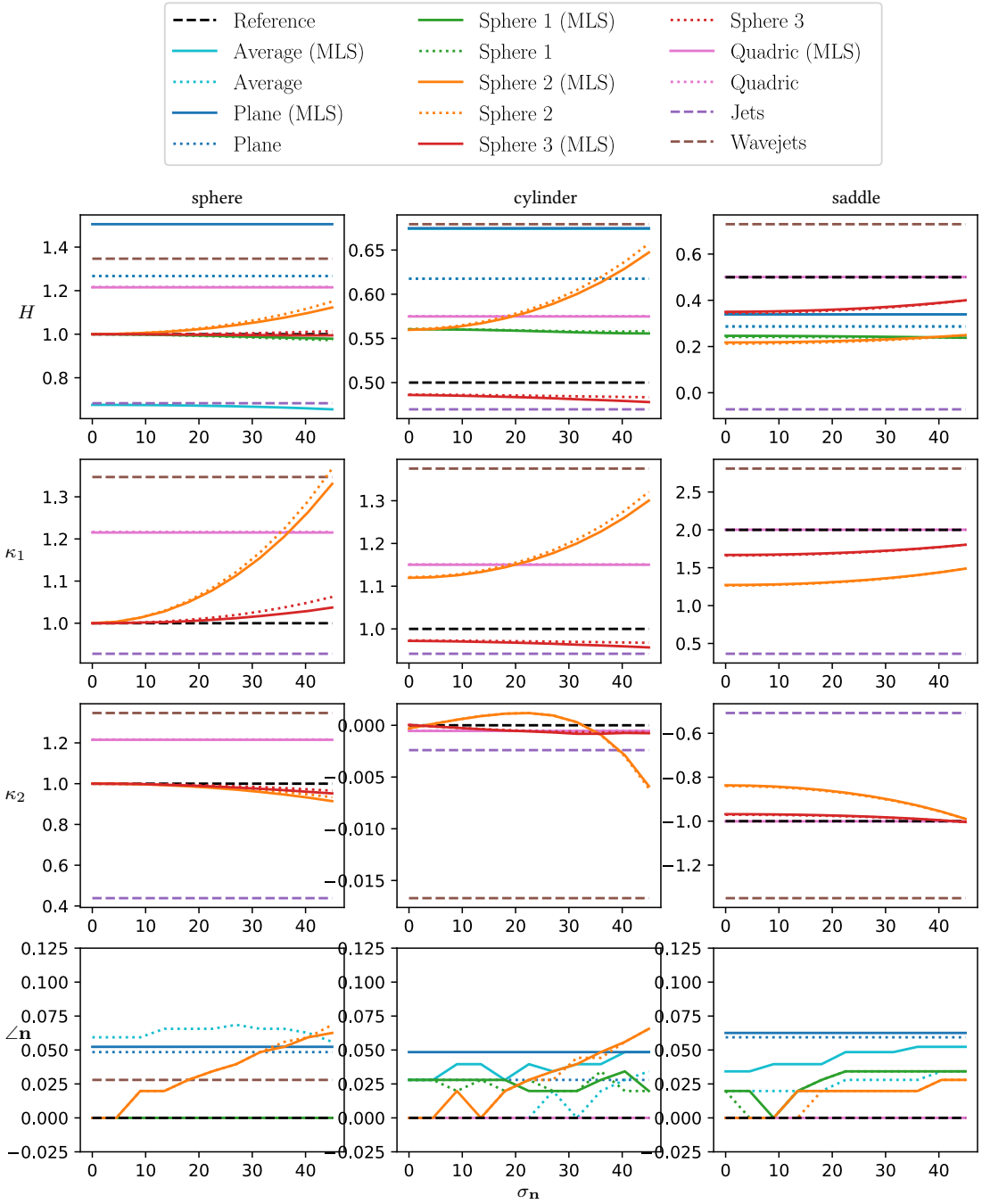


Figure 1.7: Differential properties with noise on normals. The mean curvature H, principal curvatures (κ_1, κ_2) and normal deviation $\angle\mathbf{n}$ (in degree) are plotted as a function of the normal noise standard deviation $\sigma_\mathbf{n}$ (in degree) with the neighborhood radius equal to 1. Our curvatures estimator (red) is barely subject to noise on normals.

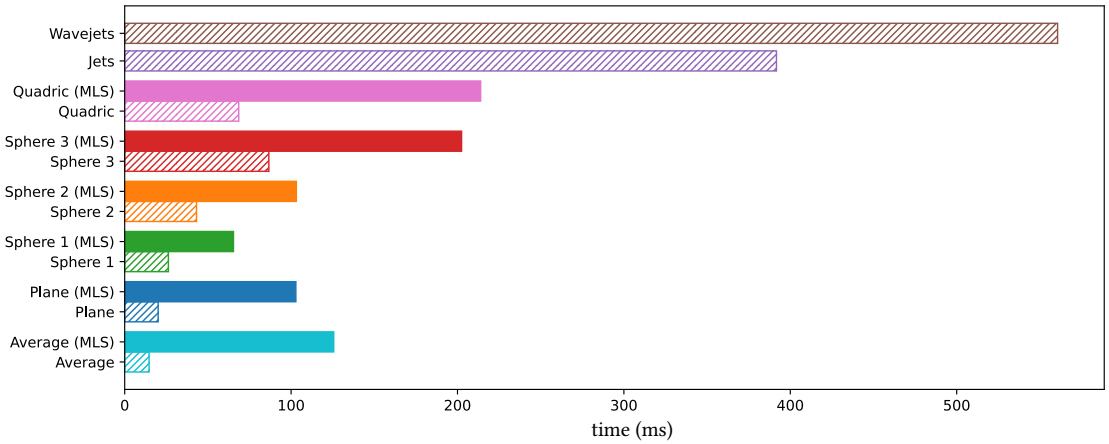


Figure 1.8: Differential properties estimations times. Average time spent of each method to compute curvatures and normals at one point for every shapes and every noise strength used in Figure 1.6 and 1.7.

1.5.2 Spatial sub-sampling

Although few high scales are sampled thanks to the logarithmic sampling (Equation 1.40), computing the full scale-space representation remains extremely slow when the amount of data become large. The reason comes from the costly neighborhood queries performed at large t even though a spatial partitioning data structure is used. To overcome this performance issue, we introduce a multi-resolution version of the point cloud inspired by a previous work on multi-scale point cloud analysis [Pauly 2003]. The idea is to compute the APSS projection of all initial points but using a different down-sampled version of the point cloud as illustrated by Figure 1.9. The sub-sampling is progressive such that at a very low scale almost all the points are considered whereas only a few points are treated at high scale.

The main challenge is to appropriately relate together the notion of scale and the down-sampling strength. Otherwise, an excessively decimated point cloud relatively to a given scale would result in a fast but inaccurate APSS, and a too conservative sub-sampling keeps too many points, slowing down the overall process. Many techniques exist to decimate a point cloud [Alexa 2001, Pauly 2002, Fleishman 2003] but the desired number of points or an error tolerance are used as criterion to drive the decimation. They usually cannot directly use our intuitive definition of scale as parameter.

Poisson disk sampling We propose to perform a Poisson disk sub-sampling [Yan 2015] of the point cloud used to compute the APSS at each level of scale. The disk radius is defined as a fraction of the current scale t_j as

$$r_j = \alpha\, t_j, \tag{1.41}$$

where $\alpha \in (0, 1)$ is the only parameter involved that controls the final resolution. In case an adaptive scale-space is used (Section 1.5.1), each point $\mathbf{p}_i$ defines its own scale so Equation 1.41 is distinct for each i while α remains a single global parameter.

For the homogeneous scale-space, we use a simple algorithm to perform the Poisson disk sampling using only a kd-tree to accelerate neighbors queries. Each point has the same prob-

Figure 1.9: Multi-scale and multi-resolution representation. Top right: three results of the projections of 1M initial points (left) at scale levels 10, 20 and 30 among 50. Bottom right: the corresponding sub-sampled point clouds with $\alpha = 0.1$ (Equation 1.41). The heat map highlights the neighborhood used to compute the APSS for the same blue point at the three scales.

ability to be selected as candidate and is included in the resulting samples set if it is marked as available. Each neighbor within a radius equal to $2r_j$ of a selected candidate is then marked as unavailable. The process is repeated until all the input points are treated. For the adaptive scale-space, a slightly more complex algorithm is required since r_j depends on each point $\mathbf{p}_i$. For each candidate, a first neighborhood query of radius r_j checks if all neighbors are available. The candidate is selected if the Poisson disk property is not violated, and another neighborhood query of the same radius marks all the neighbors as unavailable.

Although this naive algorithm is greedy and the resulting Poisson disk sampling is never maximal, it still produces an evenly spaced distribution of points appropriate in practice for the APSS. The intrinsic link between the scale and the sub-sampling strength (Equation 1.41) makes the overall algorithm well balanced between decimation and smoothing as shown in Figure 1.9. The sequential nature of the sampling can also be a source of performance issue, but since at each level of scale the previously sub-sampled point cloud is taken as input, the process is faster when the scale grows and the number of samples decreases.

This progressive approach is conceptually close to restriction phases of multi-grid methods often used on regular grids [Briggs 2000] and on meshes [Botsch 2005] to quickly solve partial differential equations. From this point of view, the Poisson disk sampling can be considered as part of the V-cycle solving our diffusion equation defined in Equation 1.23. The APSS plays the role of the smoothing operator that rapidly attenuates high frequencies such as noise and small details of the surface. Its slow speed to smooth low frequencies corresponding to larger shapes is then bypassed by a loss of resolution and an increase of scale.

Our multi-resolution representation can give a rough idea on the upper bound of the number of neighbors that can be visited to compute the APSS at each point and at a given scale t. For example, it is conjectured that at most 91 disks of radius approximately equal to 0.095 can be packed in a circle of radius 1 [Lubachevsky 1997]. If we imagine an infinitely dense point cloud on a plane, then the Poisson disk sampling would select at most 91 points within a distance t from any point if we chose $\alpha \approx 0.095$. The next section provides a practical evaluation of α and shows why we select $\alpha = 0.1$ in all our following experiments.

1.5.3 Evaluation

This section gives some insights on the impact of the sampling factor α on both accuracy and execution time. We also study the performance of the overall algorithm as the scale increases and show how our multi-scale approach benefits from the use of massively parallel implementations.

Impact of the sampling factor The main parameter that must be set is the sampling factor α of Equation 1.41. If $\alpha = 0$, the point cloud is kept unchanged so the APSS is computed as usual but the execution time is very slow at high scale. In the other extreme case where $\alpha = 1$, each point would have no neighbor, which is a situation to avoid. An appropriate value for α should give approximately the same APSS as if no sub-sampling is performed and should bring a noticeable improvement in execution time. We evaluate these criteria in Figure 1.10 on the 1M points shown in Figure 1.9. Errors on positions and normals are measured between each point on the resulting APSS obtained with a certain α and the same point but on the APSS of

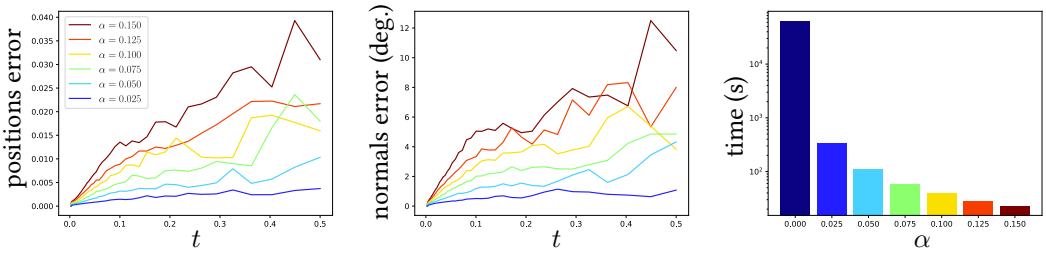


Figure 1.10: Impact of the sub-sampling. Left and middle: median errors on positions (relatively to the scale) and on normals (in degrees) for each scale between the APSS obtained with different values of sampling factor α and the APSS of reference ($\alpha = 0$). The scale t is given as a factor of the point cloud bounding box diagonal length. Right: total time in log-scale to compute the whole multi-scale representation as a function of α. We always use $\alpha = 0.100$ (yellow) for its sufficient trade-off between accuracy and speed.

reference computed without any sampling ($\alpha = 0$). The overall timings are recorded using an implementation of the APSS projections parallelized using Cuda.

In the rest of this thesis we empirically chose the setting $\alpha = 0.100$ (in yellow in Figure 1.10). It introduces no more than 2% of median error on positions with respect to the scale, and the median error on the normals does not exceed 8 degrees. Moreover, the largest errors tend to occur at high scales, which is not too limiting since we usually want more accuracy near the low scale to better catch tiny details. The execution time is largely reduced from 18 hours for $\alpha = 0$ to 40 seconds for $\alpha = 0.100$, which is totally acceptable for processing 1M points with 50 scales.

Performance study We study the detailed performance of our algorithm when the scale grows. Figure 1.11-left shows the time spent in the Poisson disk sampling and in the APSS projection at each level of scale. The sampling step is always done using one CPU thread and only takes around 10% of the total time. Two parallel implementations of the APSS projections are tested using either 8 threads parallelized using OpenMP with an Intel Xeon CPU 3.70GHz or using Cuda with an NVIDIA GeForce GTX 1080. As expected the GPU implementation outperforms the CPU one by a factor 2 in this experiment.

The main reason for using the Poisson disk sampling described in Section 1.5.2 is to reduce the number of points visited locally when computing the APSS. This is confirmed by the timings discussed previously in Figure 1.10-right, but also by the Figure 1.11-right that plots the number of neighbors visited at each level of scale. The average number of neighbors stays relatively constant between 14 and 38 for any scale. The total number of samples decreases logarithmically although we only control it indirectly via the logarithmic scale sampling of Equation 1.40 and the Poisson disk radius of Equation 1.41.

1.6 Conclusion

This chapter introduces theoretical as well as technical contributions for the use of APSS in multi-scale point cloud analysis. In asymptotic settings, Theorem 1 links the fitted algebraic sphere to the surface derivatives and especially to its mean curvature. Higher order derivatives

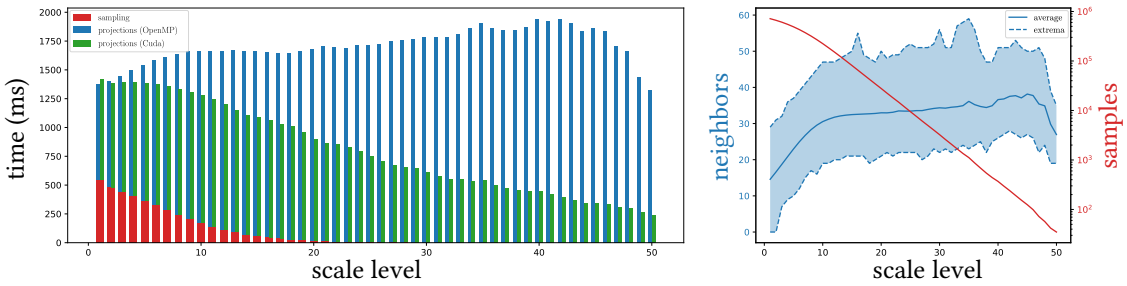


Figure 1.11: Timings and amounts of samples processed. Left: timings per step for 1M points and 50 levels of scale (Figure 1.9). The sub-sampling (red) is always done using 1 CPU thread while the APSS projections are parallelized either with 8 CPU threads using OpenMP (blue), or with a GPU using Cuda (green). In the end, the GPU improves the performance by a factor 2. Right : the neighbors count used for each APSS projection (blue) stays relatively constant while the total number of samples (red) decreases logarithmically.

and a measure of anisotropy also appear in the asymptotic expansions of the algebraic sphere parameters. We also propose a new shape operator (Equation 1.38) that completely considers the fitting procedure. A numerical comparison shows that it is more accurate and more robust to different types of noise than other existing methods. Finally, Section 1.5 explains how we compute efficiently the APSS at different level of scale using a multi-resolution representation of the input point cloud. The proposed Poisson disk sampling gives an appropriate trade-off between speed and accuracy with a good balance between scale-space smoothing and multi-resolution decimation.

In the rest of this thesis, we adopt the APSS, the new shape operator and the new multi-resolution algorithm as a multi-scale representation of the point cloud. Chapter 2 that focuses on plane detection uses normals and curvatures computed from our shape operator to group similar points together. In Chapter 3, the directions of principal curvatures are used to generate flow lines in order to extract anisotropic geometric features. The planar segmentations and the flow lines both take advantage of our efficient multi-resolution algorithm especially at high scale. Finally, Chapter 4 proposes a geometric flow inspired from the one described in Equation 1.23 to smoothly project the point cloud onto a plane or a sphere, resulting in a global 2D parametrization.

Future work The asymptotic analysis of the algebraic sphere fit performed in Section 1.3 could integrate noise on positions or normals to obtain stability proof similar to the integral invariant [Pottmann 2007]. Asymptotic expansions of many more quantities would also be interesting to derive such as the scale derivatives used in the GLS method, the spatial derivatives, our shape operator, etc... The algebraic sphere fit without normals [Guennebaud 2007] or with only non-oriented normals [Chen 2013] are also other methods to investigate using such asymptotic settings.

The numerical comparison of Section 1.4.3 includes only 3 densely sampled shapes, 2 types of noise, and 1 single scale. The few methods that are compared are either close to the APSS, such as the PSS [Alexa 2001] and the Osculating Jets [Cazals 2005a], or their asymptotic expansions exists as for the PCA plane and the barycenter method [Pottmann 2007]. It could be in-

teresting to perform an extensive comparison of differential properties estimators on unstructured point clouds using more complex shapes and noise. Different settings could be tested with multiple scales and various sampling densities and patterns. The comparison should include more existing methods including those based on statistical fitting [Kalogerakis 2007], Voronoi diagram [Mérigot 2010] and machine-learning [Guerrero 2018].

Another possible research direction is to consider algebraic quadrics. Its scalar field defined as $u_c + \mathbf{u}_\ell^T \mathbf{x} + \mathbf{x}^T U_q \mathbf{x}$ is similar to the algebraic sphere scalar field (Equation 1.3), but the quadratic parameter U_q is now a 3-by-3 matrix. In some sens, this trivariate Taylor polynomials of order 2 corresponds to the implicit version of the Osculating Jets (or the PSS) of degree 2. The direct regression to oriented points using the same least squares problems of Equations 1.4 and 1.5 [Guennebaud 2008] is also possible. It leads to a small simplified Sylvester equation [Bartels 1972] to obtain U_q. The parameters $\mathbf{u}_\ell$ and u_c are obtained using similar formulas as Equations 1.8 and 1.7. Principal curvatures are directly accessible from the scalar fields itself contrary to the simple shape operator of Equation 1.31. Its asymptotic analysis and its numerical comparison to the other methods are left as future work.

Chapter 2

Plane detection using persistence analysis of graph

Related publications

- T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Persistence Analysis of Multi-scale Planar Structure Graph in Point Clouds.* Computer Graphics Forum, 2020.
- T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Multi-Scale Point Cloud Analysis.* Eurographics posters, 2019.
- T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Multi-scale Planar Segments Extraction from Point Clouds.* Journées Françaises d'Informatique Graphique, 2018.

Contents

Figure 2.1: Planar shapes. Planes are ubiquitous especially in man-made objects.

2.1 Introduction

Many existing 3D shapes can be abstracted by arrangements of planes as shown by Figure 2.1, e.g. cities and buildings, indoor rooms, furniture and stairs, as well as most of computer-aided designed objects. Therefore, point clouds obtained from these scanned elements can be represented by a set of planar primitives. Such compact description also offers the possibility to link the geometry of the sampled surface to its semantic, e.g. floors, roofs, tables and desks. For these reasons, detecting planar primitives from point clouds is an important problem for many technical fields as reverse engineering, robot locomotion and urban planning to name a few.

Although planes are one of the simplest shape to characterize, their recognition in 3D point clouds remains challenging. The unstructured nature of the data, the large amount of points and the acquisition artifacts are common issues. Moreover, the notion of scale has a critical impact on the detected primitives. As shown in Figure 1, one point can be considered as part of either a large roof or a small tile depending on the scale of analysis. This ambiguity shows that a recognition algorithm may be exhaustive if a multi-scale approach is used.

Key observation Our approach is based on the key observation that smoothing a surface at increasing scale generates *stable* areas that are characterized by similar differential properties at different locations and scales. In Figure 2.2, we illustrate this concept on a 2D parametric curve (Figure 2.2(a)). The curvature scale-space [Witkin 1987] of this curve is plotted in Figure 2.2(b), where the color represents the curvature value (blue positive, orange negative and white null), the abscissa is the curve parameter, and the ordinate is the scale of analysis.

Areas of similar curvature values are revealed by this plot. Figure 2.2(c) illustrates these *stable* curvature areas in space and scale using a few colors for the representative curvature values. Figure 2.2(d) finally shows the shape components of corresponding curvature value for each of the *stable* areas. As we can see, these shape components are particularly meaningful for interpreting the curve shape, depending on the scale of observation (large scale in Figure 2.2(d-top), medium scale in Figure 2.2(d-middle) and low scale in Figure 2.2(d-bottom)). While multi-scale methods usually focus on peaks of curvatures along the scale-space to find meaningful salient features [Witkin 1987, Pauly 2003], we look instead for stable areas that persist over scale to robustly extract planes that potentially exist at different scales.

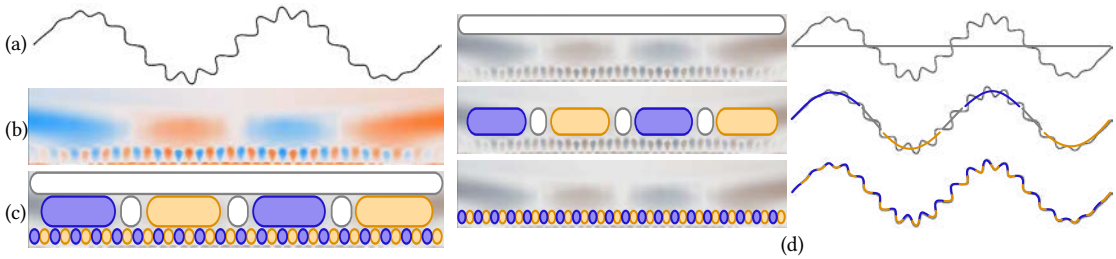
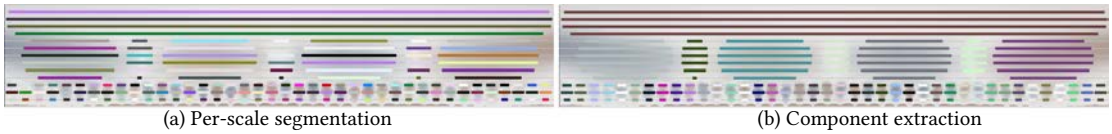


Figure 2.2: Curvatures at multiple scales. Illustration of the curvature scale-space of a curve and its application to meaningful components detection. (a) Plot of a parametrized 2D curve. (b) Curvature scale-space of the curve, where the color represents the curvature value (blue-white-orange for positive-null-negative). The abscissa is the curve parametrization, and the ordinate is the scale of analysis. (c) We define *components* as *stable* areas in scale-space. (d) Components represent the geometrical structures at different scales, and any part of the curve can belong to several components at different scales.



(a) Per-scale segmentation (b) Component extraction

Figure 2.3: Segmentations at multiple scales. (a) Visualization of the segmentation result per scale. The abscissa represents the points of the point cloud, the ordinate is the scale, and colors denote the different segmented regions. (b) Following the key observation presented in Section 2.1 and Figure 2.2, the colors show the resulting components defined by stable regions over scales.

Contributions of this chapter

- A new algorithm is developed in Section 2.3 to automatically extract planes based on our multi-scale representation and our differential property estimations introduced in Chapter 1. Meaningful planar regions are extracted even if they exist at different scales.
- In Section 2.4, we propose several interactive tools exploiting our proposals of planes. They provide an intuitive user-guided exploration of 3D point clouds improving interactive selection, segmentation, and reconstruction.

2.2 State-of-the-art

We review two classes of method for detecting geometric primitives and especially planes in 3D point clouds. The former class only detects simple primitives whereas the latter considers how they relate to each other in order to extract their underlying structure.

2.2.1 Primitive detection

Decomposing a raw 3D point cloud into patches corresponding to simple geometric primitives is a basic first step in many 3D processing pipelines [Chauve 2010, Mattausch 2014, Monszpart 2015]. Since real-world entities (man-made objects in particular) can be approximated in a piece-wise planar way, planes are among the most common primitives considered. Several methods [Rabbani 2006, Poppinga 2008] extract planar patches using region growing.

They expand a patch from a starting seed point by aggregating neighbors that have low offset and low normal deviation. To increase efficiency, small planar patches can be represented as voxels on which a similar region growing is performed [Vo 2015]. Other requirements can be used such as a minimum fitting quality or bounds on the number of points inside a region [Lafarge 2012]. Note that these thresholds are normally specified as fixed input parameters, making these approaches effective only in the presence of low or known levels of noise.

Stochastic method effectively handle noise and other defects of point clouds. They are often based on randomly generating a large set of primitive hypotheses (not restricted to planes) from the input data. In some pipelines inspired by the Ransac algorithm [Fischler 1981], the primitives that best explain the input data in terms of number of inliners are selected [Schnabel 2007]. Other approaches based on the well-known Hough Transform [Hough 1962], let each candidate primitive cast a vote in a discretization of the parameter space and select the primitives corresponding to the most voted parameter sets [Borrmann 2011]. Though robust and not restricted to planar primitives, such randomized approaches require testing a high number of primitives to ensure that all relevant features are captured.

Some approaches use the output of both region growing and randomized algorithms as starting point for a more global formulation [Pham 2016, Dong 2018, Guinard 2019]. These approaches are mostly based on minimizing an energy function designed to penalize the fitting error to the underlying data while favoring the use of a reduced number of models to explain the data [Yu 2011, Isack 2012]. Similar techniques are used when the input is an RGB-D image [Silberman 2012] or a sparse point cloud [Sinha 2009]. However, this strategy significantly increases the technical and computational complexity of the processing while only partially improving the initial segmentation.

Although effective in many specific use-cases, the success of the primitive detection techniques presented so far is highly dependent on the correct setting of their parameters, which is often unintuitive. In addition, fixing these parameters implicitly defines one single scale at which the detection is performed. In our pipeline, we rather apply a simple region growing approach with fixed parameters and vary the scale at which the underlying features are computed. This leads to results that reflect different scales of observation by simply varying an intuitive parameter like the scale of observation.

For a more in-depth review of primitive extraction approaches we refer the reader to the recent survey by Kaiser et al. (2019).

2.2.2 Structure detection

The output of the previous methods can be used to build more structured abstractions of the input data. The detected primitives can be arranged in a topological graph based on spatial proximity [Schnabel 2008]. The connectivity of this graph combines adjacent primitives in a hierarchical way, either starting from unrefined planar regions and merging them based on planarity [Feng 2014], or by considering individual points as initial primitives and aggregating them into more general shapes [Attene 2010]. The hierarchical nature of this graph makes it amenable for scale-aware reasoning, though this direction is not explored in these work. In

contrast, we build a hierarchical graph that represents the planar primitives at several scales of analysis and use their *persistence* inside this graph to discover relevant structures across different scales.

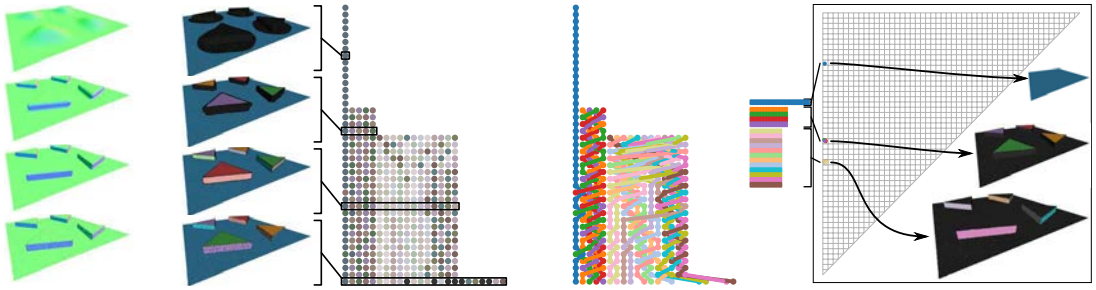
Other techniques detect global relationships between simple fitted primitives and use them to guide an optimization-based fitting. Some approaches detect non-planar primitives (e.g. cylinders, spheres, cones) as well as planar parts [Li 2011b], while others only focus on planes and extract both the primitive parameters and their inter-relationships in a joint manner, maximizing robustness [Monszpart 2015, Oesau 2016]. These approaches capture global regularities, but do not convey any explicit information on the scale at which the analysis is performed. Rather than focusing on regularity relations, our goal is to discover the relevance of the structures at different scales of observation.

In fact, the extraction of scale-aware representations of raw 3D models has only recently emerged as a meaningful research problem. Using a data-driven approach, Hu et al. (2018) learn a patch-based label assignment, extracting the patches at a single scale and using the available labels to constrain their boundaries. During testing, the segmentation into patches is performed at several geometric scales and the scale that best matches the learned patch-based labeling is selected. This approach yields a scale-aware segmentation, but it heavily relies on the availability of labeled input data and it does not consider the problem of how to convert the output segmentation to a suitable representation.

Fang et al. (2018) observe that the different scales of abstraction of a model can be obtained by exploring the 2D space defined by the shapes size σ and their fitting tolerance ε. Since the pairs (ε, σ) corresponding to meaningful scales are located on the diagonal of this space, they generate a redundant set of abstractions corresponding to samples on this diagonal, arguing that this amounts to repeatedly applying local geometric contractions to the input model. Finally, they select a fixed number of abstractions, optimally matching them to the learned preferences of a group of users. While we move from similar motivations, we do not rely on greedy simplifications to generate the meaningful abstractions and extract a large set of candidate meaningful parts at different scales of analysis. In addition, instead of relying on a learnt definition for the meaningful scale, we automatically propose meaningful parts by analyzing their persistence across different scales of analysis and allow the user to explore and refine our proposals interactively using a variety of intuitive tools.

2.3 Automatic extraction of multi-scale planar structures

We present a new algorithm to automatically extract meaningful planar regions at multiple scales from 3D unstructured point clouds following a procedure inspired by the multi-scale analysis presented in Section 2.1. The first step of our pipeline illustrated in Figure 2.4 builds the multi-scale representation of the input point cloud as we explain in Section 1.5. Recall that the scale is defined as the size of the neighborhood used to calculate the Algebraic Point Set Surfaces (APSS) [Guennebaud 2007]. We sample $M = 50$ scale values logarithmically (Equation 1.40) and uniformly over the whole point cloud (see Section 1.5.1). Since we focus on planes, a more robust version of the APSS is used as illustrated by Figure 2.5. We enhance the APSS algebraic sphere regression by an iteratively re-weighted least squares fit [Oztireli 2009]



(a) Smoothed input (b) Graph nodes (c) Component extraction (d) Persistence diagram

Figure 2.4: Pipeline. Our approach starts by (a) reconstructing the point cloud at different scales using robust APSS. (b) The reconstructed surfaces provide parameters for segmenting the point cloud in planar regions at multiple scales. Regions are stored as nodes in a hierarchical graph where each level corresponds to a scale. (c) Edges in the graph link the stable regions at successive scales and the colors show the different corresponding *components*. (d) The persistence analysis of the extracted components enables the characterization of planar structures at multiple scales.

improving the surface approximation near sharp features.

The differential informations obtained from the APSS (Section 1.4) parametrize a region growing algorithm that groups points in planar *regions* at each scale (Section 2.3.1). Points in abscissa of Figure 2.3(a) are grouped in regions (different colors) by scale in ordinate. Then we store regions as *nodes* of a graph and create edges between regions extracted at successive scales if they share enough points (Section 2.3.2). In Section 2.3.3, we define *components* as stable regions linked in the graph from which a persistance analysis is done. Figure 2.3(b) shows with different colors the components obtained from the per-scale region segmentations illustrated in Figure 2.3(a).

2.3.1 Planar segmentations

Our method defines a planar region as a set of points sharing similar normal vectors and presenting a low curvature. Normal vectors are estimated at any scale from the APSS as presented in Section 1.4. They are the main features to steer the segmentation and provide the per-scale planar regions. Figure 2.5 illustrates the reconstructed surface at four different scales using a robust version of the APSS [Oztireli 2009]. Our expectation is that the segmentation at two successive scales generates similar regions where the underlying scale-space is stable. We thus seek a simple, yet stable region segmentation algorithm.

Region growing By construction, seedless region growing offers strong stability as the segmentation result is uniquely defined for a given local threshold criterion. We could grow regions from neighbors to neighbors if their normals are approximately the same [Rabbani 2006]. However, this approach cannot control any global property of a region, and thus the planarity of a region cannot be guaranteed. For instance, a sphere can be detected as a single region, as long as the normals vary under the threshold over it. Curvature may also drive the segmentation, but it would add an additional threshold difficult to parametrize in the local criterion.

For these reasons, we rather use a seed-based region growing. The principal curvatures (κ_1, κ_2) of a point $\mathbf{p}_i$ at scale t_j are estimated from our shape operator introduced in Equa-

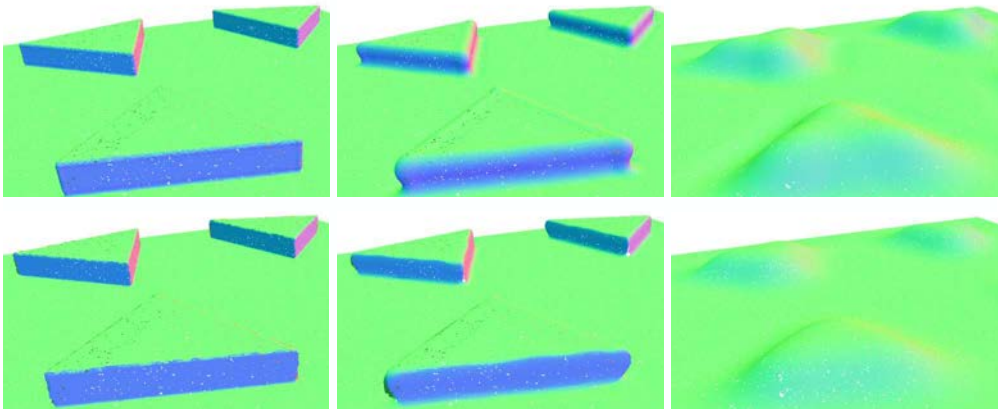


Figure 2.5: Robust APSS. Comparison between standard APSS (top) and robust APSS (bottom). Reconstructions are done at 3 increasing scales from left to right. Colors are determined by normals orientations.

tion 1.38. They define a planarity measure for this point as $(\kappa_1^2 + \kappa_2^2)^{-1}$ that is used to select seeds over the point cloud. For each scale, we rank the points based on their planarity and start the region extraction from the most planar points. Spatially neighboring points are inserted in the currently growing region until their normal vector deviate too much.

In practice, considering the seed point $\mathbf{p}^j_{\text{seed}}$ with the highest planarity at the j^{th} scale, and with normal vector $\mathbf{n}^j_{\text{seed}}$, we define an initial region $\mathcal{R}^j_{\text{seed}}$. The region is then expanded by visiting the spatially close points using the k-nearest neighbors graph. A point $\mathbf{p}_i$ is inserted in $\mathcal{R}^j_{\text{seed}}$ if it is not already assigned to another region, and if the angle between its normal $\mathbf{n}^j_i$ and the seed normal $\mathbf{n}^j_{\text{seed}}$ is lower than an angle θ. This process is repeated on the non assigned points until all points of the cloud belong to a region. In our experiments, we set $\theta = 5°$ and $k = 10$ for all scales. Note that the same value of k is used to select the minimum scale t_{min} (see Section 1.5.1). This value is also a good compromise as using a smaller value might separate points that are in practice similar to the reconstructed surface. In contrast, a higher value might lead to unwanted jumps during the region growing, where distant points are assigned to the same region even though they are separated by another thin region. Note that range-based neighborhoods could be considered instead but their performance are worse for equivalent results.

Filtering Each region obtained with this process represents a spatial aggregation of points considered as a planar structure at a specific scale t_j. As we use neighborhoods of size t_j, regions can only be representative of structures whose spatial extent is at least comparable to t_j. For this reason, we discard all regions $\mathcal{R}^j_i$ that have a surface area $a^j_i < 2 \cdot t_j$, with a^j_i being the area of the α-shape of $\mathcal{R}^j_i$ [Edelsbrunner 1983], computed using $\alpha = 2 \cdot t_j$.

For a given scale t_j, the planar segmentation yields a set of regions $\mathcal{S}_j = \{\mathcal{R}^j_1, \ldots, \mathcal{R}^j_{N_j}\}$. These regions form by construction a partition of the input point cloud $\mathcal{P}$ at scale t_j. By repeating in an independent manner the region growing at each t_j, we obtain a set of output segmentations $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, which corresponds to collection of regions sampling the scale-space of the input point cloud. Figure 2.4(b)-left shows several segmentations at four scales.

2.3.2 Multi-scale region graph

Our goal is now to extract the planar components from the evolution of regions of $\mathcal{S}$ along scales. Each of them corresponds to a set of similar regions persisting at several consecutive levels of scale. We describe in this section the hierarchical graph structuring the regions contained in $\mathcal{S}$ in order to relate them to each other. We then propose a similarity measure between two regions to compare and collect them into distinct clusters.

Hierarchical graph We structure the segmentation set $\mathcal{S}$ in a multi-scale region graph $G = (V, E)$ in which each region $\mathcal{R}_i^j$ is associated to exactly one node in V, denoted v_i^j. As each node v_i^j holds at a specific scale value, we treat G as a hierarchical structure in which all nodes corresponding to the same scale t_j form the j^{th} level of the graph and levels are ordered by increasing scale. Figure 2.4(b) shows the nodes of a multi-scale region graph sorted by level. Once regions are organized by levels in the graph, we connect by an edge all pairs of regions computed at successive scales.

Similarity measure As we designed our segmentation to give similar results at consecutive scales in stable areas of the scale-space, we expect that regions in stable areas share a sufficient number of points. To measure the overlap between two regions v_i^j and v_k^{j+1}, we use the Jaccard index $J(v_i^j, v_k^{j+1})$. It is defined as the sum of points in the intersection over the sum in the union of the regions

$$J(v_i^j, v_k^{j+1}) = \frac{|\mathcal{R}_i^j \cap \mathcal{R}_k^{j+1}|}{|\mathcal{R}_i^j \cup \mathcal{R}_k^{j+1}|}. \tag{2.1}$$

The Jaccard index $J(v_i^j, v_k^{j+1})$ is symmetric, is equal to 1 when two regions share exactly the same set of points, and drops to 0 for non-overlapping regions. It has the particular advantage to remain only combinatorial, which makes it fast to evaluate and trivial to implement. It is also generic as it does not depend on the type of considered primitives.

Components In our settings, two regions at the same level t_{j+1} cannot overlap, so a node at level t_j has at most one node at level t_{j+1} for which $J(v_i^j, v_k^{j+1}) > 0.5$. According to this observation, we only connect in the graph G all pairs of nodes laying at consecutive scales and having a Jaccard index strictly greater than 0.5 (Figure 2.4(c)). As illustrated in Figure 2.6, this simple rule connects nodes corresponding to regions with similar coverage, while preventing the connection of nodes having an ambiguous relation. One could consider using a stricter threshold in the range $(0.5, 1)$, but from our experiments this does not improve the method. Each set of connected nodes in the graph finally define a *component* in the topological sense. Each component $\mathcal{C}$ is characterized by a birth level l_b and a death level l_d (respectively, the lowest and highest levels of its regions), as well as all the regions $\left\{\mathcal{R}^{l_b}, \ldots, \mathcal{R}^{l_d}\right\}$ it contains. As demonstrated by Figure 2.4(c), considering these components clearly helps to interpret the raw graph shown in Figure 2.4(b).

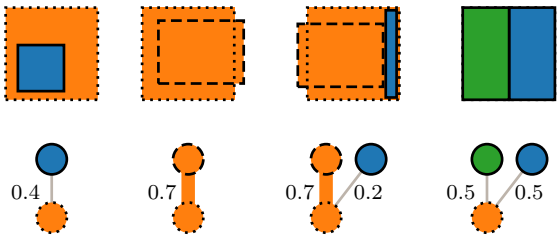


Figure 2.6: Nodes similarity. Examples of relations between nodes. We connect nodes when their Jaccard index is strictly superior to 0.5. Connected nodes are colored with the same color and belong to the same component.

2.3.3 Persistence analysis

With this new graph representation, we propose a new analysis tool for multi-scale plane detection in point clouds. As our graph is organized as a collection of components with birth and death scales, we directly benefit from the toolbox developed in the domain of Topological Data Analysis (TDA) [Chazal 2017]. More specifically, we consider the concept of *persistence* (inspired by the more formalized notion of *topological persistence* [Edelsbrunner 2000]) and define it for a component $\mathcal{C}$ as the difference between its death and birth scales: $\text{pers}(\mathcal{C}) = l_d - l_b$. The persistence of components can be easily described using a persistence diagram. This 2D diagram displays each component $\mathcal{C}$ as a point with coordinates (l_b, l_d). In such diagrams, as illustrated in Figure 2.7(a), components of equal persistence $\text{pers}(\mathcal{C})$ are on the same diagonal line. Those having the lowest scale of birth appear on the left side whereas those with the highest scale of birth appear on the right side. The diagonal $y = x$ in the middle of the plot contains components of null persistence. They only have one region at one scale that is not similar enough to any other regions above nor below in the scale-space. These mono-region components generally correspond to noisy part of the point cloud. The bottom-right part of the persistent diagram is empty since $l_d > l_b$ by definition. The components of lower persistence ($\text{pers}(\mathcal{C}) = 11$ in the figure) are those on the first diagonal line above the middle one, and those with the highest possible persistence are on the top-left corner.

Following these observations, we can see in Figure 2.4(d) that we have three sets of components grouped by equal persistence. All of them have a birth scale at the lowest scale and those having the lowest death scale are those with the lowest persistence (the closest to diagonal in the diagram). These components have a large enough persistence, meaning that they are stable over scales and they define meaningful planes explaining the data for the scales they cover. The slightly higher ones have a larger scale of death. They are also representative and define planes explaining the data up to higher scales. Finally, a single component exhibits a high scale of death with a very large persistence. These components explain data from small to large scales: at a very high scale, the single component of the overall plane is prominent, while at lower scales, the components of the parts of this plane that have no detail are more representative.

Figure 2.7(b) shows two components of identical persistence. The blue one is representative up to the lowest scales because the point cloud is very clean in this part. The orange one has a higher birth scale. This is due to the noise slightly degrading this part of the point cloud.

Similarly to TDA, components with larger persistence are more likely to represent promi-

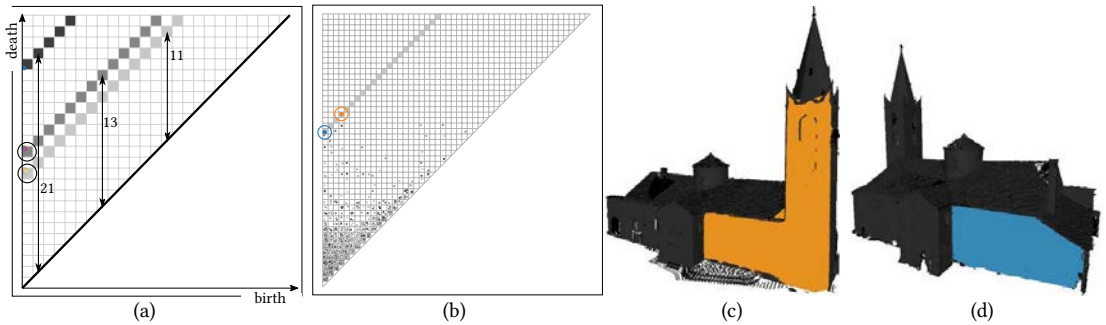


Figure 2.7: Persistence diagram. (a) Persistence diagram breakdown for the Tri scene (see Figure 2.4). (b)-(d) Comparison between two components with equal persistence on the Lans scene. The surface covered by the orange component (c) is more noisy at small scales than the blue one (d), which delays its level of birth and shifts it to the right in the persistence diagram (b).

nent structures in the graph. In particular, as illustrated in Figure 2.4(d), the persistence diagram of the components of a graph G allows to easily discriminate between the relevant geometric structures associated to persistent components. In Figure 2.14, we show how persistence diagrams are affected by noise with increasing variance.

2.4 Interactive tools

We propose new interactive tools for user-guided exploration of point clouds. They are based on the automatic extraction of multi-scale planar structures described in the previous section. To analyze a point cloud, the planar segmentations (Section 2.3.1) and the graph (Section 2.3.2) are pre-computed. The tools we describe next leverage our persistence analysis proposed in Section 2.3.3.

2.4.1 Persistence-based thresholding

In TDA, persistent structures are considered as meaningful, and non-persistent structures as noise. We propose a simple tool where the user selects a minimum persistence value and the system visualizes on the input point cloud the components $\mathcal{C}$ for which the persistence $\mathrm{pers}(\mathcal{C})$ is greater than this minimal value. The unique threshold is an integer between 0 and M (the number of scales). As shown in Figure 2.8, this approach is effective for point clouds whose components are distinctly clustered in the persistence diagram.

2.4.2 Scale-based point cloud segmentation

This tool performs a segmentation of the whole input point cloud with respect to a scale value t given by the user as illustrated in Figures 2.15 and 2.16. For every point $\mathbf{p}$, this tool selects the most persistent component among all the components including a region at the given scale t and holding the point $\mathbf{p}$. Points that are not associated to any component are kept unlabeled. The scale threshold gives control on the level of detail of the segmentation.

2.4.3 Interactive brush-based component selection

In order to let the user focus on a subset of the point cloud, we propose a brush-based interface to select specific planar components in real-time. With this tool, the user defines a set of query points $\mathcal{P}_Q$ by directly painting on the point cloud. Interactively, our system returns a set of components that best match the selected points in terms of overlapping and persistence (see Figure 2.9). More formally, we list all the components that hold any of the selected points, and rank them using the following score

$$\mathrm{s}(\mathcal{C}) = \alpha_{\mathrm{points}}(\mathcal{C}) \cdot \alpha_{\mathrm{pers}}(\mathcal{C}), \tag{2.2}$$

with $\alpha_{\mathrm{points}}(\mathcal{C}) = |\mathcal{P}_Q \cap \mathcal{P}_\mathcal{C}|/|\mathcal{P}_Q|$ and $\alpha_{\mathrm{pers}}(\mathcal{C}) = \mathrm{pers}(\mathcal{C})/M$. The score function $\mathrm{s}(\mathcal{C}) \in (0, 1)$ accounts both for the fraction of selected points that are held by the component $\mathcal{C}$ and for the persistence of $\mathcal{C}$. As the persistence score $\alpha_{\mathrm{pers}}(\mathcal{C})$ is normalized by the total number of scale M, it acts as a penalty factor modulating the overlap ratio between the query and the component. Once a component is selected, the user can either colorize the point cloud accordingly or keep its α-shape for further use.

2.4.4 Interactive similarity search

Another interesting aspect of the components $\mathcal{C}$ is that they offer a high level descriptor that can be used for similarity queries. We propose an interactive search where components are matched with respect to a query and presented to the user as shown in Figure 2.17. The tool performs as follows. The user selects a component using the brush-based selection tool. Then, a second brush is used to select a set of points defining the search area. As for the selection tool, all the components that hold a point belonging to the search area are considered as similarity candidate. In order to verify if a candidate component $\mathcal{C}_c$ matches the query component $\mathcal{C}_q$, we propose to combine multiple criteria depending on the usage case (any criterion can be adjusted interactively):

1. the birth and death levels of $\mathcal{C}_c$ coincide with those of $\mathcal{C}_q$, plus or minus a given threshold,
2. the planes approximating the components $\mathcal{C}_c$ and $\mathcal{C}_q$ are parallel at an angular threshold,
3. the ratio between the surface area of the α-shapes reconstructing the components $\mathcal{C}_c$ and $\mathcal{C}_q$ is lower than a given threshold, expressed as a percentage of the area of the query α-shape.

2.5 Experiments

This section presents the experimental results we obtain with our approach and compares it to prior work. We have developed the prototype of our pipeline in C++, using Eigen [Guennebaud 2010] for linear algebra operations and CGAL [The CGAL Project 2020] for α-shapes computation.

Datasets Our test scenes, presented in Table 2.1, consist of 10 point cloud datasets of varying complexity. In these models, 4 were obtained by Multi-View Stereo (MVS), 3

Model	#Points	Data source	Surf. Recon.	Segm.	Filtering	Graph	Comp.	Total
Tri	0.5M	Synthetic	77.78	1.59	27.19	3.42	0.00	109.98
Stairs	1M	Synthetic	194.62	4.22	22.51	5.41	0.00	226.76
Cubes	10M	Synthetic	1884.62	47.87	922.09	67.85	0.00	2922.43
Lans	1.2M	LiDAR	310.85	6.31	15.87	4.37	0.01	337.41
Pisa	2.5M	MVS	719.91	6.84	33.96	6.25	0.02	766.98
Church	4.3M	MVS	1490.85	15.29	92.02	14.65	0.03	1612.84
Loudun	35.5M	MVS	12299.40	147.15	606.82	126.43	0.84	13180.64
Room	1.1M	MVS	372.93	2.31	20.90	5.77	0.00	401.91
Euler	3.9M	LiDAR	1052.24	10.14	62.48	33.74	0.09	1158.69
Munich	6.5M	LiDAR	1664.66	33.00	260.93	50.63	0.07	2009.29
Empire	1M	Synthetic	383.72	2.21	15.87	4.24	0.00	406.04

Table 2.1: Timings. Description of our test datasets and of the computing time in second required in preprocess for each subroutine: APSS surface reconstruction, planar segmentations, regions filtering, graph construction, and components extraction.

by LiDAR scans of indoor and outdoor scenes, and 4 are synthetic point clouds generated by sampling hand-modeled meshes. These synthetic models represent relatively simple arrangements of geometric shapes and can be easily corrupted with controlled levels of noise. The real-world point clouds represent large-scale building structures (Loudun, Lans [Falcidieno 2004], Pisa [Mellado 2015b]), groups of buildings and their surroundings (Church [Sketchfab 2020], Munich [Hackel 2016]), and indoor environments (Euler [Monszpart 2015], Room [Armeni 2016]). We applied our different tools to all of our datasets, and we present a subset of our experiments in the following sections.

2.5.1 Results

We give a visual overview of the results obtained with our various interactive tools described in Section 2.4. An example of use is also shown within the context of polygonal surface reconstruction in order to validate the practicality of our method. Appendix B provides more results.

Extraction of prominent structures The persistence-based thresholding (Section 2.4.1) and the scale-based segmentation (Section 2.4.2) tools provide an immediate insight into the most relevant structures of a point cloud at different scales.

We show in Figure 2.8 the extracted components for all our test data, filtered by increasing minimum persistence. The Pisa model is a particularly good example to showcase the capabilities of our method. At the lowest persistence level, all the main roof sections and all the alcoves of the facade are captured. When increasing the persistence threshold, only the larger alcoves persist, until reaching the highest level, at which the largest roof sections are the only highlighted structures. A similar, yet even clearer trend is shown by Cubes, a synthetic model consisting of four nested levels of planar structures. At the lowest persistence value, all planar faces of the boxes appear. As the threshold is increased, the smaller faces progressively disappear, as their features are more and more smoothed out due to the influence of the larger surrounding planes at higher scales of observation.

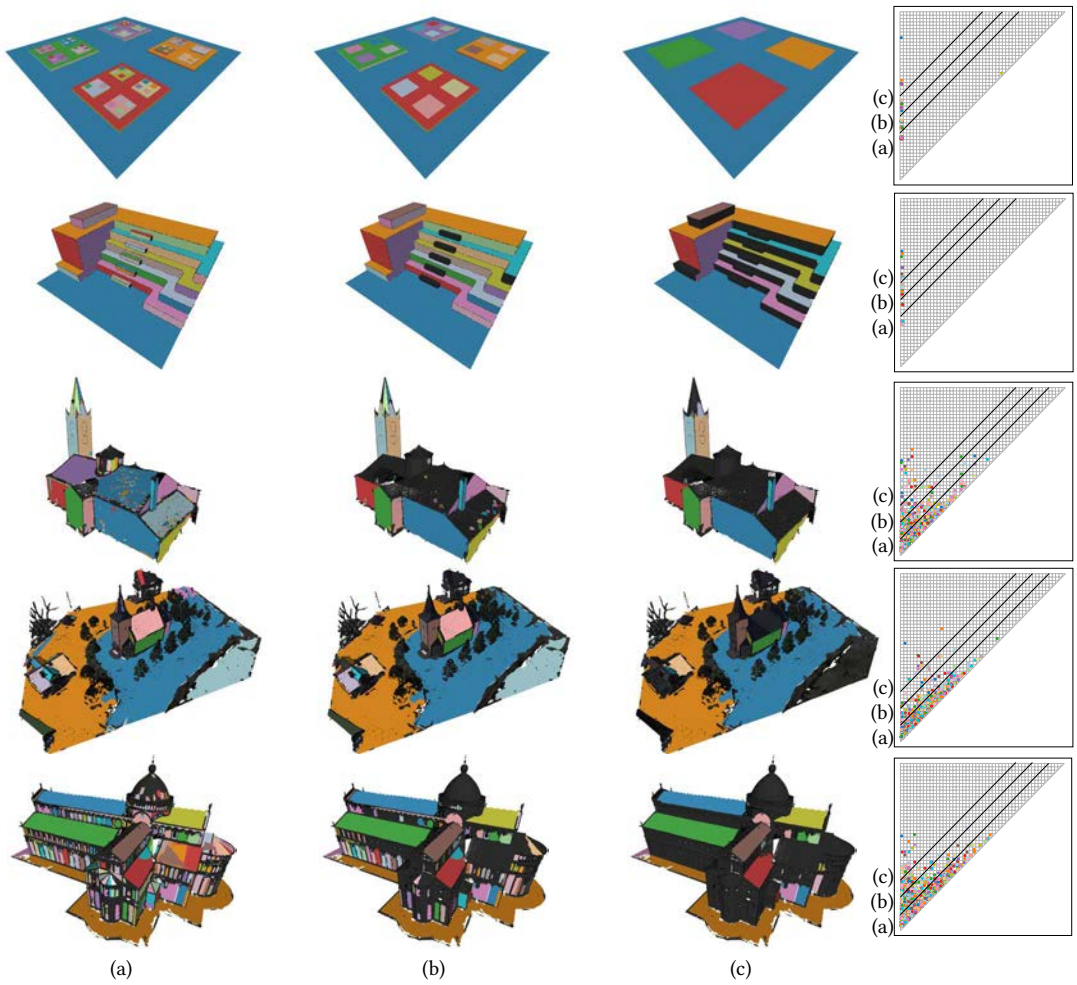


Figure 2.8: Persistence thresholding. Persistent components for five scenes (from top to bottom: Cubes, Stairs, Lans, Church, Pisa) with three increasing persistence thresholds (a, b and c), illustrated on the persistence diagrams (Section 2.4.1).

In Figure 2.15 and 2.16, the structures of the test models are highlighted through the segmentation induced by the persistent components that include a given scale. Note in particular how the individual tiles on the roof of Lans are selected as individual segments at low scale, while for larger scales of interest they are merged into larger individual roof segments and eventually into a single one. Likewise, the alcoves in the lower part of Pisa, which appear as single entities at low and medium scales, are fused into a single structure at the highest scale value considered.

Interactive reconstruction and similarity search Further insights into specific parts of the model can be obtained using the brush-based reconstruction (Section 2.4.3) and similarity search (Section 2.4.4) tools. Using the first tool on the Lans model (Figure 2.17(a)), one can select individual structures on a facade with very rough sketches and replace them with low-complexity polygonal proxies, built based on their most representative associated com-

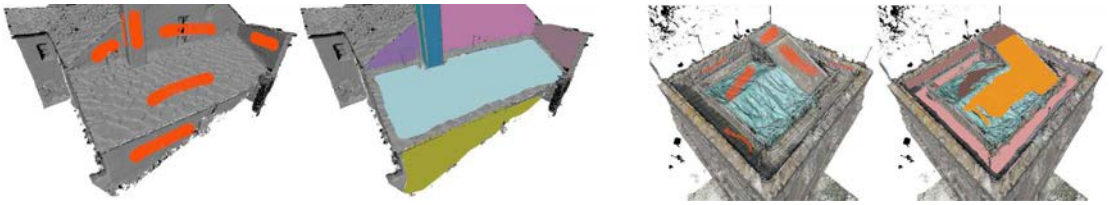


Figure 2.9: Interactive reconstruction. Brush-based reconstruction tool on models Lans (left) and Loudun (right). With just a few rough sketches, the user selects some points (orange strokes) and the tool automatically reconstructs their low-complexity polygonal proxies.

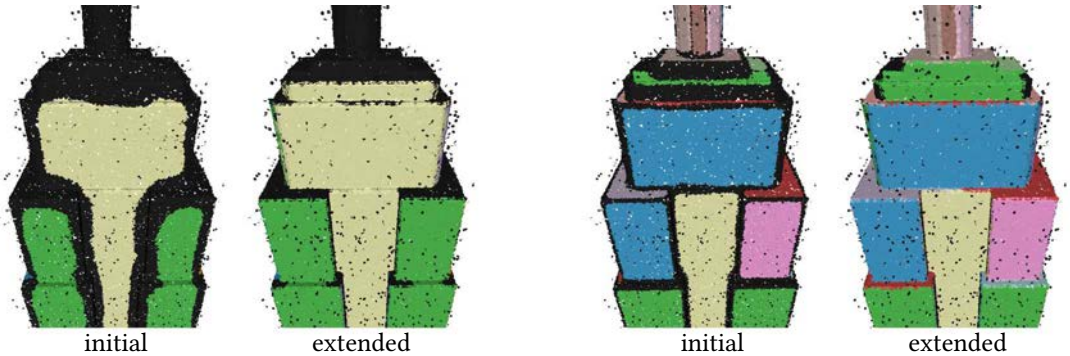


Figure 2.10: Coverage increase. Effect of extending the initial regions at medium scale (left) and high scale (right).

ponents. Thanks to the specific score function used in this process (see Equation 2.2), the proxies approximate well the geometry of the selected structures. Figure 2.9 presents a similar reconstruction for Loudun: remarkably, the reconstructed polygons correctly represent the underlying structures despite the high amount of clutter and outliers.

Alternatively, instead of reconstructing a surface, a selection can be used as a template in the search for matching structures. In Figure 2.17, we show a typical result of this workflow. On the model Lans, the similarity search tool allows to sketch a first selection on a single roof tile (which becomes the query of our search) and then a second one roughly covering the whole roof segment (which represents the domain of the search). By matching the most persistent component underlying the query with the best matching component of the domain, the tool extracts the components representing all the other tiles in the roof (Figure 2.17(a)), without requiring that the user is engaged in a tedious and error-prone selection process. Proceeding in a similar way, one can easily extract the individual steps from the staircase of Stairs (Figure 2.17(b)) and the individual arches in an arcade of Pisa (Figure 2.17(c)).

Point cloud coverage With our technique, large portions of points may not be labeled. This is due to the segmentation of the APSS that tends to shrink at higher scales as the edges get rounded. The regions may be extended to the points that are close to the plane that best fits a region and that has a compatible normal. Figure 2.10 illustrates the coverage obtained with a distance threshold set to 20% of the bounding box diagonal length and an orientation threshold set to 45 degrees.

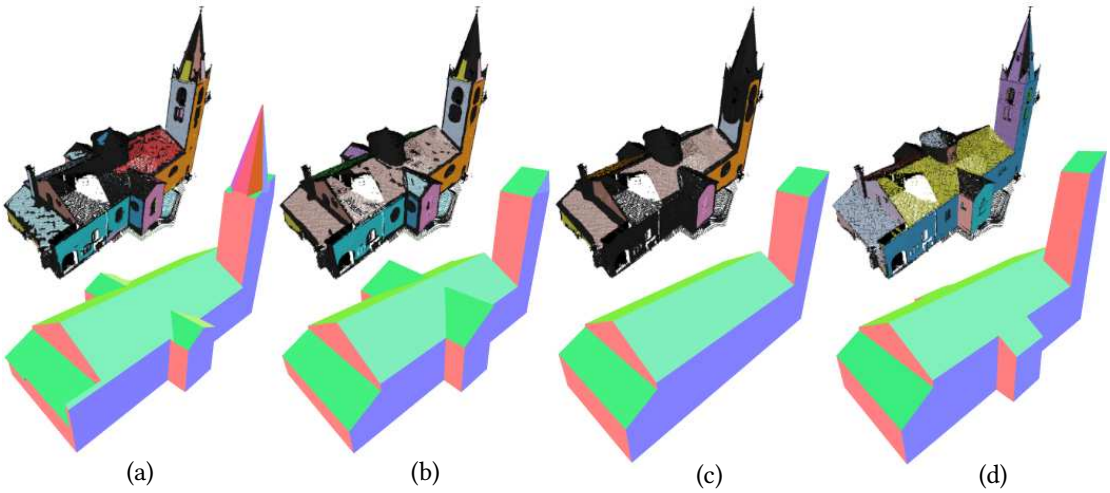


Figure 2.11: Polygonal reconstruction. Polygonal reconstruction of Lans using PolyFit with our planes detected at 3 different scales (a)-(c) and with those detected by Ransac (d). Planar segmentations are shown above the reconstructed meshes.

Application to polygonal reconstruction We considered the sets of planes obtained with our method at three different scales and used them as input for a polygonal reconstruction algorithm (PolyFit) [Nan 2017]. This algorithm reconstructs a watertight polygonal surface from a set of input planes, optimizing three energy terms related to data fidelity, reconstruction complexity and coverage, under hard constraints that ensure that the resulting mesh is manifold and closed. The resulting meshes for the Lans and Empire models are shown in Figures 2.11 and 2.12, respectively. We used the Polyfit implementation provided by CGAL with default parameters, adding an extra plane at the bottom side of the bounding box to obtain a closed surface. We compare our results against those obtained by extracting the input planes using the CGAL implementation of Ransac [Schnabel 2007] with default parameters. Compared to this baseline, using our approach for the input planes selection allows to naturally generate a sequence of meshes at different levels of detail. This is possible thanks to our unique definition of scale, which is not accounted for by Ransac.

2.5.2 Evaluation

Comparisons The fundamental difference between our approach and the multi-scale plane fitting proposed by Fang et al. (2018), Ransac [Schnabel 2007] and RAPter [Monszpart 2015] is that we do not fit the *best* planes considering tolerance, coverage and eventually scale as parameters. We rather find planes faithfully representing the data at different scales. With our approach, one point belongs to zero, one or several planes identified at different range of scales. At a fixed range of scales, the coverage of our planes significantly varies depending on the data and the considered scales. We ran a comparison on the Empire scene, as done by Fang et al. (2018)-Table 3 with Ransac and RAPter. To avoid a corrupted APSS reconstruction at higher scales, we filtered outliers, computing for each point the covariance matrix with a neighborhood ball of radius = 1% of the aabb diagonal and keeping only points with planar neighborhood using standard heuristics (points kept: 79.3%). We have extracted planes (our

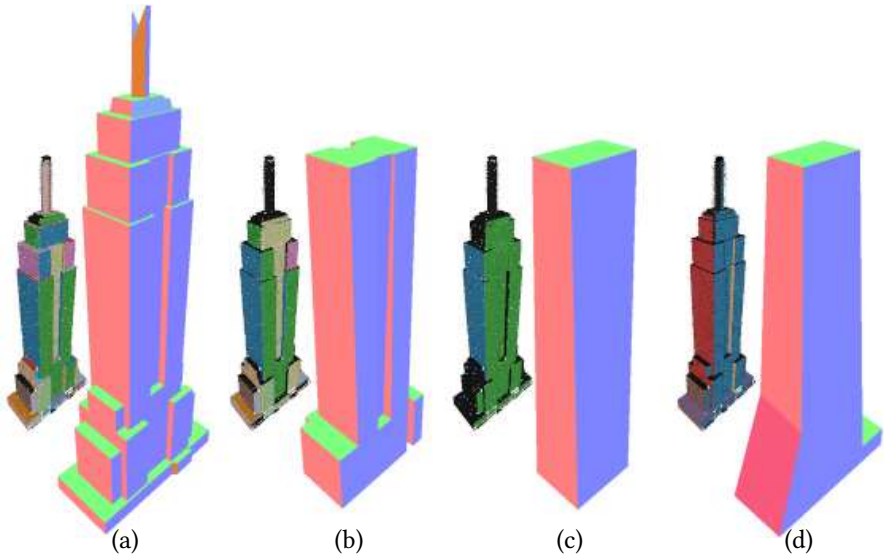


Figure 2.12: Polygonal reconstruction. Polygonal reconstruction of Empire using PolyFit with our planes detected at 3 different scales (a)-(c) and with those detected by Ransac (d). Our planar segmentations (a)-(c) correspond respectively to scales 2, 3 and 4 of Figure 2.13 and Table 2.2.

stable regions) at four different scales. Figure 2.13, as well as the coverage, root mean square error and number of planes presented in Table 2.2 at each scale highlight the difference of our approach: it aims at finding planes that explain the surface at different scales, rather than fitting planes that approximate the points. As such, our tool produces planes with lower coverage and very low geometric error (several orders of magnitude lower than previous work), even at high scales.

Method	# scale	% aabb	coverage	RMS error	planes
[Schnabel 2007]	-	-	0.808	0.034	128
[Monszpart 2015]	-	-	0.817	0.042	163
[Fang 2018]	1	-	0.816	0.017	239
	2	-	0.816	0.290	40
	3	-	0.816	1.030	9
Ours	1	0.452	0.767	0.0001	128
	2	0.890	0.753	0.0002	92
	3	1.018	0.688	0.0006	49
	4	5.880	0.434	0.0060	4

Table 2.2: Qualitative comparison. Comparison of our method with Ransac [Schnabel 2007], RAPter [Monszpart 2015] and Fang et al. (2018) on the Empire scene. For our method, stable regions are extracted at four scales set to different percentages of the axis-aligned bounding box (aabb) (see Figure 2.13). The two last columns give the corresponding Root Mean Square (RMS) error and the number of extracted planes. The coverage, RMS error and number of planes values for Ransac, RAPter and Fang et al. (2018)-Table 3.

Processing time In this chapter, all our experiments are done on an Intel(R) Xeon(R) CPU E5-2640 v4 clocked at 2.40GHz with 40 cores and 128G of RAM. The recorded timings for all

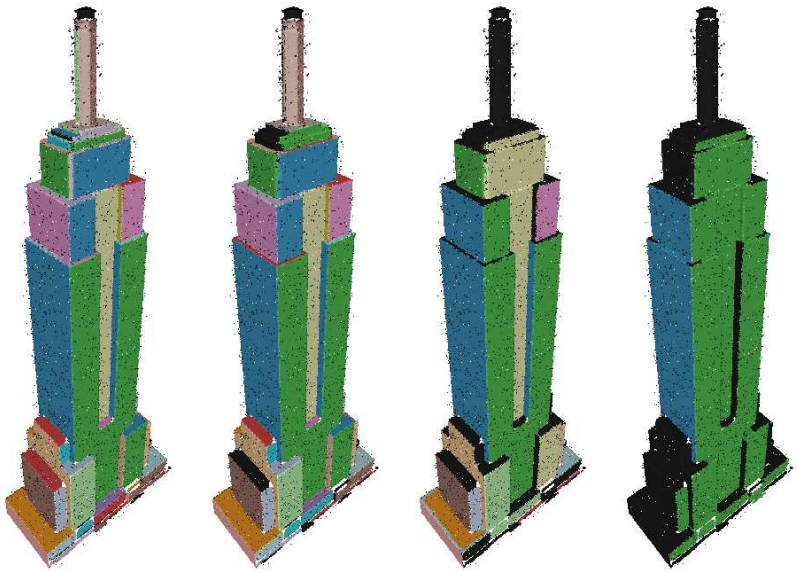


Figure 2.13: Segmentation at four scales. Planes (stable regions) extracted at the four scales presented in Table 2.2. From left to right, scale 1, 2, 3 and 4.

test models are presented in Table 2.1, which provides a detailed breakdown of the individual steps. The total processing time ranges from about 110 seconds for our simplest synthetic model Tri (0.5M points) to about 3.6 hours for Loudun (35.5M points). It is worth noticing that all these steps need to be completed only once to generate the components, which are stored and simply loaded at the beginning of each interactive exploration stage. The step that requires the most computational power is the multi-scale robust APSS surface reconstruction, although it could be speed up by at least a factor 2 thanks to the GPU implementation used in Section 1.5. Note that the method by Fang et al. (2018) and RAPter require respectively 12 minutes and a couple of hours to process 1M points. In contrast, our approach requires around 6 minutes for 1M points, and processes a 35M point cloud in 3.6 hours. In addition, more than 90% of the processing time is spent on the APSS pre-computations, which could be locally recomputed in case of local editing.

Impact of noise We evaluated how noise affects our results by corrupting the synthetic model Cubes with increasing noise and analyzing the corresponding changes in the persistence diagram. In particular, we consider 4 levels of increasing Gaussian additive noise, corresponding to a standard deviation σ_{noise} equal to 0.001%, 0.005%, 0.01% and 0.025% of the diagonal of the axis-aligned bounding box. As shown in Figure 2.14, at low levels of noise the main planar structures emerge already early in the scale-space, since even at the lowest scales the APSS reconstruction is not affected. For this reason, the corresponding components appear on the left of the diagram. As noise is increased, the point-wise surface reconstruction at low scales becomes unreliable. Hence, there is no region that can be detected as associated to those components at those scales, resulting in higher birth levels. Overall, as noise increases, the diagrams become more cluttered with new points (corresponding to noisy structures), which also appear more spread out and generally shifted towards the right. Nevertheless, the main

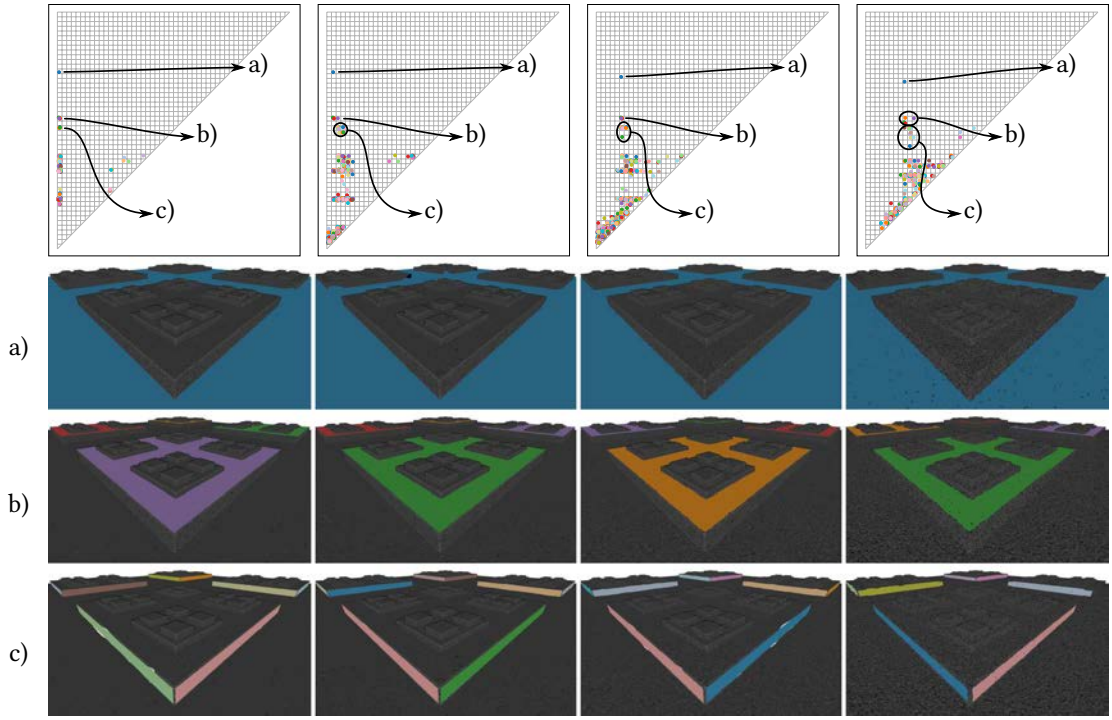


Figure 2.14: Impact of noise. Impact of positional Gaussian noise on the component extraction for the Cubes scene. The standard variation of the noise is a factor of the bounding box diagonal and is set to 1, 5, 10 and 25 times 10^{-5} from left to right.

planes can still be recognized as fairly localized clusters in each diagram.

2.6 Conclusion

This chapter introduces a novel method for the extraction of the meaningful structures of a 3D point cloud at multiple scales. We improve the interactive analysis and exploration of complex acquired data. Our approach is based on computing planar regions with similar differential properties at individual scales. We analyze their stability across the scale-space by studying the topological persistence of a hierarchical graph that stores the regions at different scales. Furthermore, we provide intuitive tools for visualizing the most stable structures discovered, as well as to segment, reconstruct and perform part-based queries on the input model based on these structures. The resulting pipeline is effective and can be applied efficiently to inputs consisting of several millions of unstructured points.

Future work A direct extension of our method would be to handle other geometric primitives than planes. One advantage of our algorithm is its predominant combinatorial and topological aspects. The graph, the components and the persistence analysis can be directly generalized to any other shapes. Indeed, the geometry of the sampled surface is taken into account only during segmentation (Section 2.3.1) where a curvature-based planarity measure defines how seeds are selected, and the normals orientation controls the region growing. This step

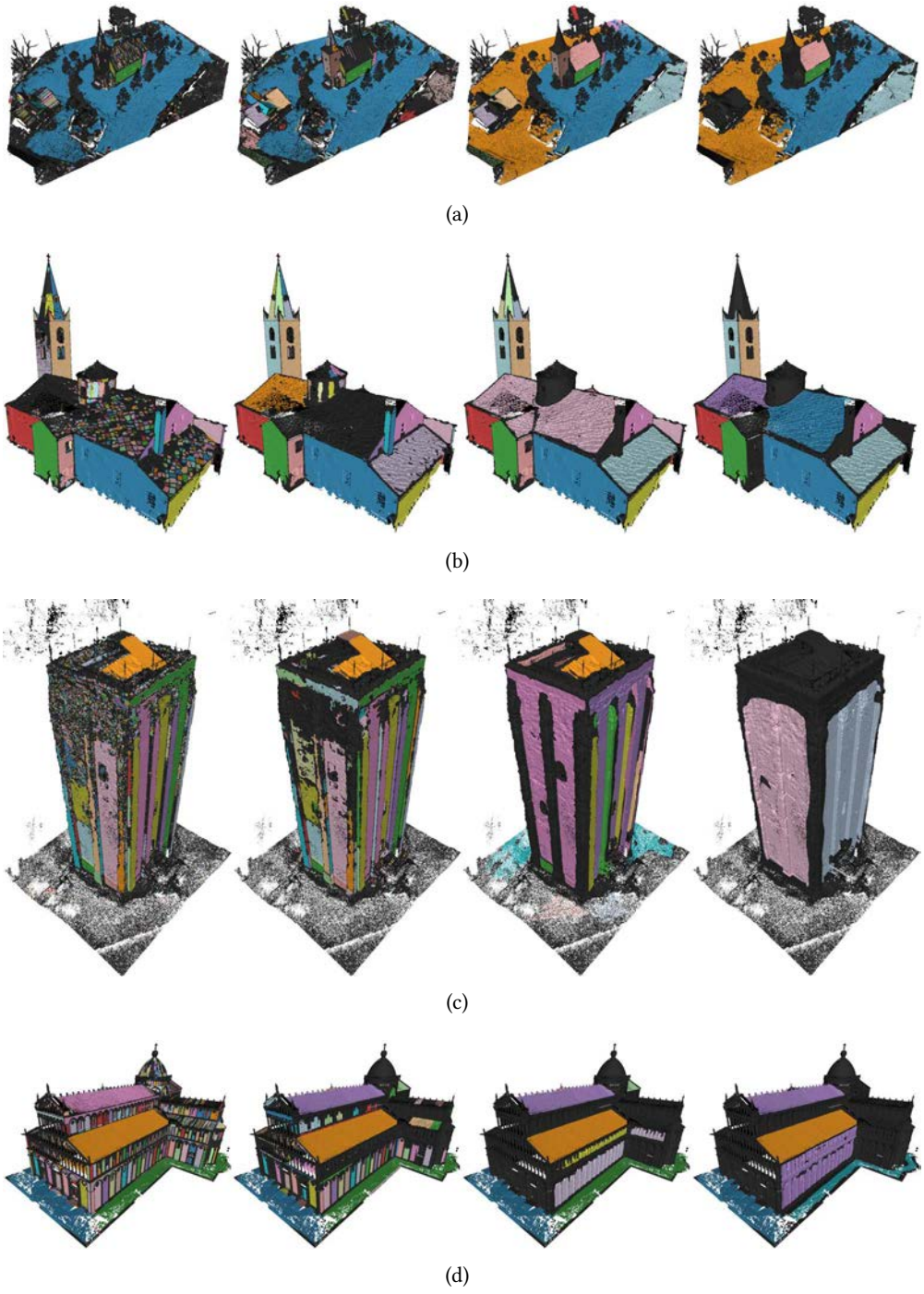
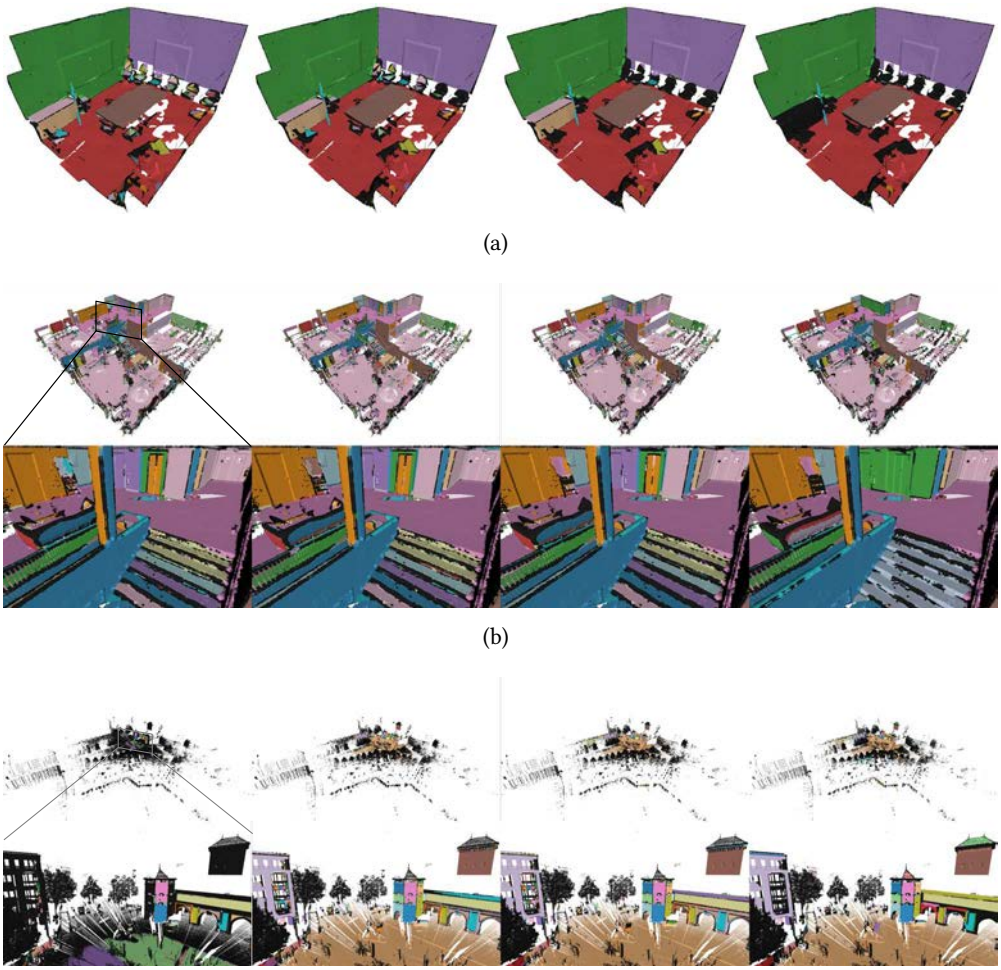


Figure 2.15: Scale thresholding. Segmentation of models Church (a), Lans (b), Loudun (c) and Pisa (d) based on the most persistent components that include scales 5, 15, 20 and 25 out of the 50. (Section 2.4.2).

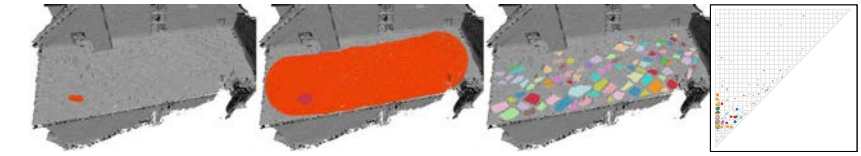


(a)

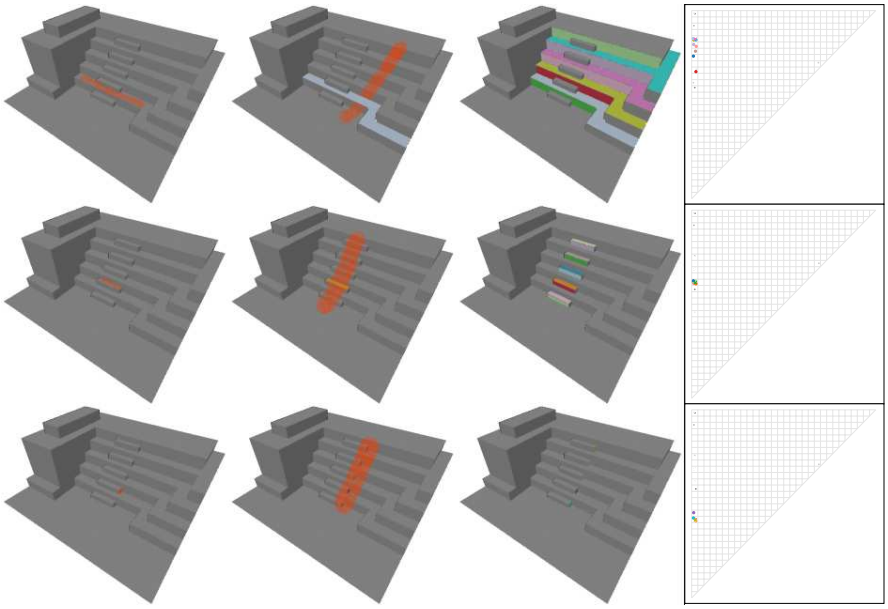
(b)

(c)

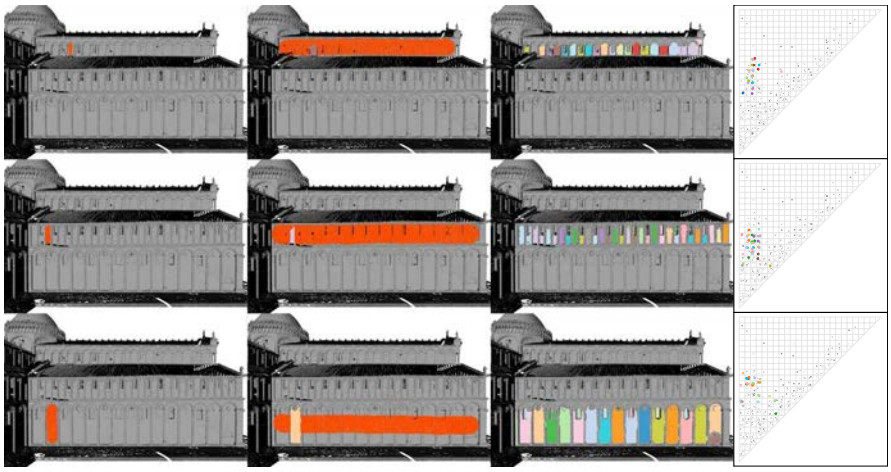
Figure 2.16: Scale thresholding. Segmentation of models Room (a), Euler (b) and Munich (c) based on the most persistent components that include a given scale (with increasing given scale from left to right) (Section 2.4.2).



(a)



(b)



(c)

Figure 2.17: Similarity search. Interactive similarity search tool on models Lans (a), Stairs (b) and Pisa (c): with two simple sketches, the user selects a query part (left) and a larger domain part (middle) and the tool automatically extracts all the components of the domain that match the query. Corresponding diagrams highlight matching components in color.

could include other criteria that involve principal curvatures and their directions to segment spheres, cones and cylinders for instance. The challenge would be to manage different types of segmentation together without increasing too much the overall combinatorics.

Finally, while we illustrate the behavior of our pipeline under both synthetic and realistic real-world noise, we do not estimate specifically the maximum levels of noise and outliers that our approach can tolerate. A theoretical analysis of the robustness to such artifacts makes an interesting future work.

Chapter 3

Anisotropic features detection using curvature lines

Related publications

- E. Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, T. Lejemble, L. Barthe, N. Mellado, C. Romanengo, S. Biasotti, and B. Falcidieno. *SHREC'19 track: Feature Curve Extraction on Triangle Meshes.* Eurographics Workshop on 3D Object Retrieval, 2019.
- S. Biasotti, E. Moscoso Thompson, L. Barthe, S. Berretti, A. Giachetti, T. Lejemble, N. Mellado, K. Moustakas, Iason Manolas, Dimitrios Dimou, C. Tortorici, S. Velasco-Forero, N. Werghi, M. Polig, G. Sorrentino, and S. Hermon. *SHREC'18 track: Recognition of Geometric Patterns over 3D models.* Eurographics Workshop on 3D Object Retrieval, 2018.

Contents

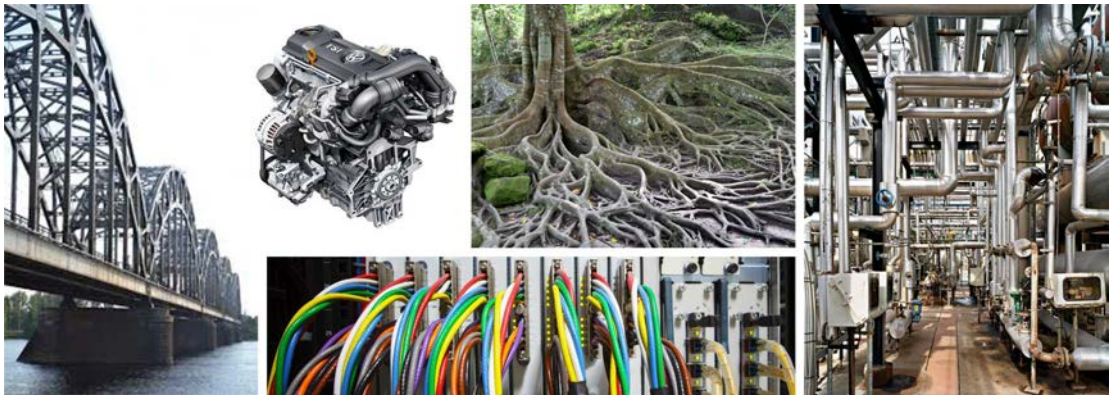


Figure 3.1: Cylindrical shapes. Most of the points on these objects are locally characterized by a high curvature in one direction and an almost zero curvature in the other direction along the cylindrical shape.

3.1 Introduction

Cylindrical shapes are widely present in man-made objects such as pipes, cables and pillars as shown in Figure 3.1. They are mainly characterized by one local principal direction along which the shape does not change much, hence the name anisotropic feature. Similar anisotropic features can be seen in the nature on crests and valleys or along the fractures of a geological site (Figure 3.2). Many archaeological artifacts and CAD objects also show remarkable curves corresponding to contours and sharp edges. One major difference between a pipe and a crest is that the former is smooth while the other corresponds to a discontinuity. Cylindrical shapes are spanned by several regular lines, whereas feature curves are made of salient ones. In any case, both of them share the anisotropy property. They can be highlighted along their entire length by drawing lines following their principal directions. The goal of this chapter is to develop this idea in order to detect these anisotropic shapes in 3D unstructured point clouds acquired from the aforementioned objects.

The question of scale remains an important issue. The cable of Figure 2 can be seen as either one straight vertical cylinder or several curved twisted cylinders. Both observations are true depending on the scale of analysis. As indicated in the state-of-the-art of Section 3.2, few existing methods are capable of jointly detecting these two configurations.

The scale can also help to robustly detect features. A feature detected at more scale than another is probably more pertinent. This idea can be used to extract feature curves on meshes and on point clouds for 3D shape abstraction. High frequency noise and insignificant details are thus avoided.

Finally, feature curves are not necessarily an end in itself. Lines following locally the principal directions of cylinders and feature curves at different scales could greatly improve many geometry processing tasks that are based on characteristic lines such as point cloud meshing [Kalogerakis 2009], quadrangulation [Alliez 2003], symmetry detection [Bokeloh 2009], segmentation [Zhuang 2017], simplification [Gehre 2016] and artistic rendering [Bénard 2019, Section 1.5]. For these reasons, extracting scale-aware characteristic lines on unstructured data is an important challenge we address in this chapter.

Figure 3.2: Feature curves. Many shapes are characterized by apparent curves. Similarly to Figure 3.1, such feature locally shows an an almost zero curvature along the line and a strong curvature in the other direction.

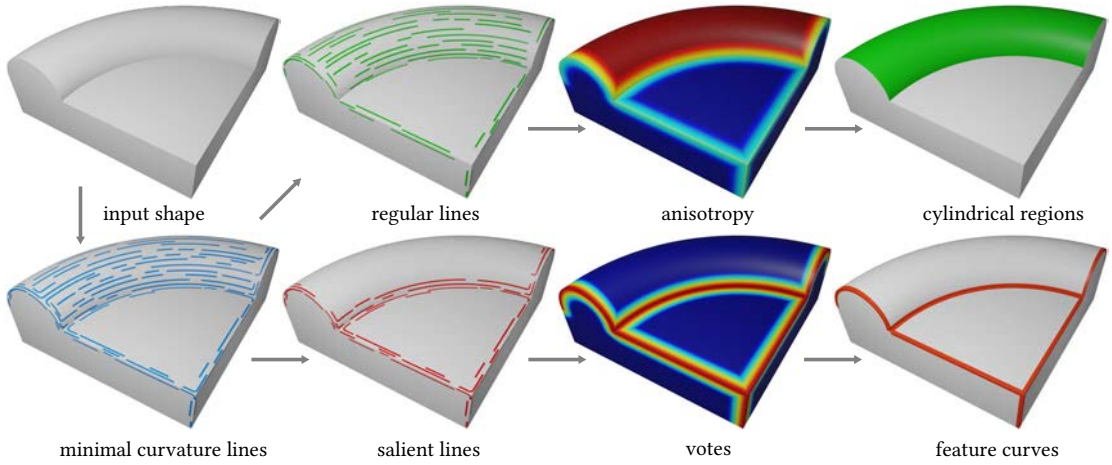


Figure 3.3: Line-based feature detection concept. Regions of the input shape are classified as either cylinders or feature curves. For both of these geometric features, we generate minimal principal curvature lines and filter them to obtain regular or salient lines. Cylinders correspond to regions where the regular lines are mostly aligned, which is measured by a local anisotropy measure. The feature curves are classified based on the votes accumulated locally on the shape from the salient lines.

Key ideas We propose to detect anisotropic geometric features using curvature lines extracted on the sampled surface. As illustrated by Figure 3.3, the idea is to generate many minimal curvature lines scattered over the input shape. Two different filters discard unwanted lines and only keep *regular* or *salient* ones. The first type corresponds to clean and smooth lines, whereas salient lines only span highly curved areas of the surface. The anisotropy measuring the alignment of the regular lines over the shape highlights the cylindrical regions. To detect feature curves, we propose a voting system from the salient lines to the shape, so that regions that are close to many salient lines are classified as feature curves. Note that the general method illustrated in Figure 3.3 can be performed at multiple scales to add robustness the the voting approach, and to detect cylinders of variable size. In the end, we can obtain two different segmentations of the input 3D point cloud or mesh, where each point is classified as part of either a cylindrical shape or a feature curve.

Contributions of this chapter

- In Section 3.3, we propose a new method to extract curvature lines from 3D unstructured point clouds at multiple scales. We mainly rely on the scale-space framework intro-

duced in Chapter 1 by using our shape operator (Section 1.4.2) and our multi-resolution algorithm (Section 1.5.2). The resulting 3D polylines are smooth, scale-aware, and well scattered over the input shape.

- We propose in Section 3.4 a new approach to extract feature curves based on our multi-scale curvature lines. They are first filtered to keep only salient ones. A voting approach at multiple scale and a region growing finally extract regions corresponding to significant feature curves of the input 3D shape. We present a detailed comparison with other existing methods on a specific set of labeled meshes.

- We introduce in Section 3.5 a cylindrical regions segmentation algorithm using minimal curvature lines. After a dedicated filtering, the most regular lines are kept, and a local anisotropy measure quantifies how much the lines are aligned with each other. Thanks to a region growing algorithm, various smooth cylindrical regions can be extracted depending on the scale of analysis. They could be used as input to our persistence analysis framework of Chapter 2.

3.2 State-of-the-art

We first summarize the existing methods that extract various type of characteristic lines from 3D shapes. We then review the problem of cylindrical shapes detection.

3.2.1 Feature curves detection

The problem of abstracting a 3D shape by line-shaped features can be tackled by different approaches. Sharp edges [Koch 2019], crests and valleys (or ridges and ravines) [Ohtake 2004], principal curvature lines [Kalogerakis 2009] and contours [DeCarlo 2003] are definitions referring to the same general idea of a salient anisotropic feature that have one distinguishable principal direction along the surface.

Many methods exist to detect crest and valley lines on triangular meshes [Yoshizawa 2005, Hildebrandt 2005, Cazals 2005b, Weinkauf 2009] among others. Their goal is to find local extrema of principal curvatures along their directions. The usual challenge is to compute third order derivatives (curvature variation) that are often sensitive to noise. But meshes have the crucial advantage that differential quantities can be interpolated on edges and faces, which facilitate zero-crossing localization. This may explain why there exist only a few methods working with unstructured point clouds [Stylianou 2005, Wang 2018a]. They generate relatively sparse and noisy lines even though the input point cloud is fairly clean. Nevertheless, principal curvature lines can be robustly extracted from point clouds under the form of 3D polylines [Kalogerakis 2009], which is the kind of approach we follow in Section 3.3.

Sharp edges in point clouds are widely studied. Local covariance analysis of the points distribution (see Equation 1.1) is a well established approach [Gumhold 2001, Pauly 2003, Mérigot 2010, Bazazian 2015]. Different other strategies exist such as normals clustering [Weber 2010], multi-view image processing [Lin 2015], RANSAC [Ni 2016] and the Hough transform [Torrente 2018] among others. Recently, Hackel et al. (2016) and Yu et al. (2018) proposed to use machine learning in order to avoid any user parameters.

Line recognition can also be cast as a dimensionality detection problem [Brodu 2012, Digne 2018] where the point cloud is segmented in regions of 1, 2 and 3 dimensions corresponding to curves, surfaces and volumes respectively.

Except for a few multi-scale methods [Pauly 2003], the notion of scale is not really taken into consideration when detecting feature lines on 3D shapes. In this work, we take inspiration from the method of [Kalogerakis 2009] to extract curvature lines on 3D point clouds, but we use the Algebraic Point Set Surfaces (APSS) [Guennebaud 2007] as underlying model of surface, which enables multi-scale feature extraction.

3.2.2 Cylinders detection

Detecting cylinders from point clouds is a long-standing problem [Kaiser 2019]. Among many other methods, RANSAC [Schnabel 2007] and the Hough transform [Rabbani 2005] are two widely famous frameworks that produce satisfactory results especially when the data contain outliers. But they are mainly limited to simple cylindrical primitives with very few parameters. Cylinders of varying radii and curved cylinders cannot be correctly detected.

Recognition of tubular structures have been an active area of research this last decade. These structures containing straight and curved cylinders as well as connecting parts such as junctions and elbows usually appear in industrial sites scans for as-build modeling. Various approaches are proposed based on region growing, Hough transform and circle fitting [Patil 2017], curvature-based thresholding and RANSAC [Kawashima 2014], normal-based clustering and circle fitting [Qiu 2014], and region growing and B-spline fitting [Dimitrov 2016]. These processing pipelines are composed of several successive algorithms involving in the end many user parameters which reduce their flexibility. Most of them also contain ad hoc rules to recognize pipes junctions in very specific point clouds. Identifying a known CAD model in a 3D point cloud is also addressed [Bey 2011, Czerniawski 2016], but this knowledge is not always available a priori.

Finding first a skeleton can also help to detect cylindrical shapes [Tagliasacchi 2016]. Zhou et al. (2015) decompose a mesh into generalized cylinders solving en exact cover problem from multiple candidate cylindrical regions. They introduce a cylindricity measure to quantify how much a piece of surface and its skeleton deviate from a right cylinder in terms of quantity of information needed to describe them. Skeleton-based pipes detection from unstructured 3D point clouds can be achieved using Voronoi diagrams [Lee 2013] or accumulated votes in a voxel grid [Kerautret 2015] for instance. Bauer et al. (2009) use local cylinders fitting [Lukács 1998] to reconstruct the spin curve of one single bent tube.

All of the above-mentioned methods analyze the point cloud at only one single scale. To our knowledge, only the Plumber method [Mortara 2004] uses different radii of analysis. However, the scale is used to intersect the mesh with a sphere in order to determine the local topology of the surface, which cannot be easily extended to unstructured data.

3.3 Curvature lines extraction

The key element of the methods presented in this chapter is the extraction of lines of minimal curvature on the surface defined by the APSS. By definition, at each point of such line, the line

tangent is aligned with the direction of minimal principal curvature of the surface. Curvature lines actually correspond to streamlines where the velocity vector field is the field of principal curvature directions. If we denote by s the line parameter (intuitively the time), by $|\kappa_1| > |\kappa_2|$ the two principal curvatures, and by $\mathbf{v}_{\min}$ the direction of minimal curvature κ_2, then the coordinates $\mathbf{q}(s) \in \mathbb{R}^3$ of the line satisfy the following differential equation

$$\partial_s \mathbf{q}(s) = \mathbf{v}_{\min}(\mathbf{q}(s)), \tag{3.1}$$

with the initial condition $\mathbf{q}(0) = \mathbf{q}_0$. Since the APSS is defined by a projection operator ϕ (see Equations 1.2), the additional condition $\phi(\mathbf{q}(s)) = \mathbf{q}(s)$ must also be respected so that the line stays on the surface.

Similarly to a previous work [Kalogerakis 2009], each line is represented as a piecewise linear curves (polylines) using discrete series $\{\mathbf{q}_i\}_{i\in\mathbb{N}}$ of vertices $\mathbf{q}_i \in \mathbb{R}^3$. The principal curvatures directions are computed using our shape operator proposed in Equation 1.38. We generate a line starting from a seed point $\mathbf{q}_0$ and following iteratively the directions $\mathbf{v}_{\min}$ on the APSS

$$\mathbf{q}_{n+1} = \phi\left(\mathbf{q}_n + \delta\, \mathbf{v}_{\min}(\mathbf{q}_n)\right), \tag{3.2}$$

where δ is a step size parameter ($\mathbf{v}_{\min}$ is normalized). It corresponds to a modified forward Euler scheme integrating Equation 3.1 with a projection step led by ϕ (Equation 1.12) that is performed at each iteration to keep the line vertices on the surface. Kalogerakis et al. (2009) use a statistical curvatures estimator [Kalogerakis 2007] and a projection operator [Lipman 2007] that are conceptually different. Instead, we estimate the curvatures of the same surface on which we project the vertices, which leads to a unified framework for curvature lines extraction on unorganized point clouds.

As in the previous chapters, the size t of the neighborhood used to calculate the APSS (see Equation 1.6) defines the scale parameter. Different scales yields various kind of lines as shown by Figure 3.4. Figure 3.5 shows that we successfully highlight the two level of scale of the twisted cable we discussed in the introduction (see Figure 2). A low scale produces lines following details of the surface whereas smoother lines are obtained using a high scale. Note that at high scale, we use our multi-resolution algorithm of Section 1.5 to improve the execution time.

The seed points $\mathcal{P}_{\text{seed}}$ that start the lines drawing are a subset of the input points. We use a Poisson disk sub-sampling (see Section 1.5.2) to select the seeds with a disk radius equal to $r = \beta \cdot t$. Contrary to a uniform random sampling, the Poisson disk sampling better scatters the seeds over the shape and is adapted to the scale. The step size δ of the lines can also be set accordingly to t using $\delta = \gamma \cdot t$. Note that these parameter settings follow the same idea as for the sampling factor used in Equation 1.41. We empirically choose $\beta = \gamma = 0.5$ in all our experiments.

3.4 Feature curves detection

Feature curves are widely present on natural and man-made objects (see Figure 3.2). These anisotropic features correspond to a normal discontinuity that occurs along a demarcation

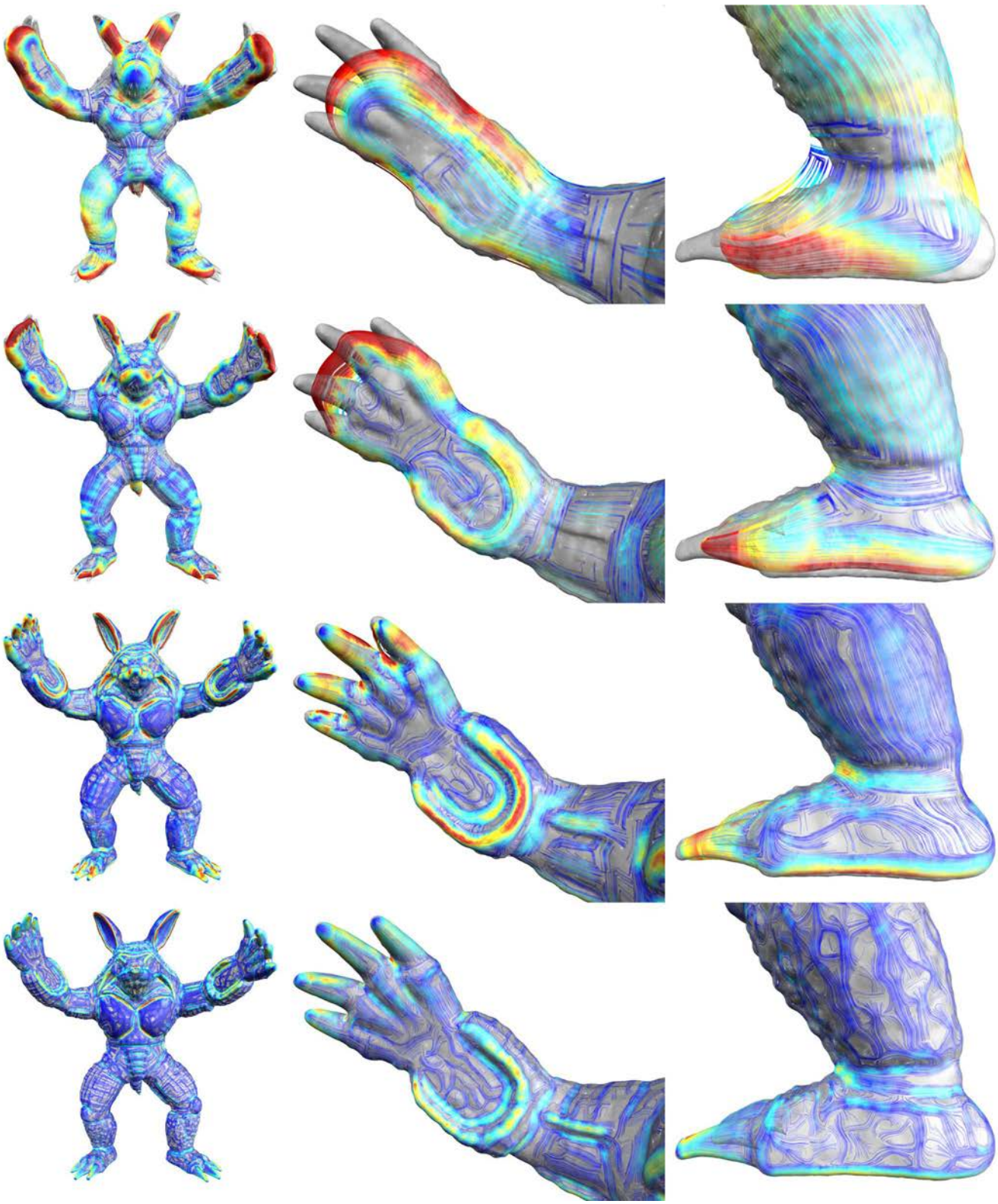


Figure 3.4: Minimal curvature lines. Low to high scale from bottom to top. Colors are based on $|\kappa_1| - |\kappa_2|$ from 0 in blue to 3 in red.

Figure 3.5: Minimal curvature lines. Left: input shape (already introduced in Figure 2). Middle: a low scale produces lines following the small twisted cables. Right: at high scale, the minimal curvature lines follow a vertical straight cylinder. Colors are based on $|\kappa_1| - |\kappa_2|$ from 0 in blue to 3 in red.

line. At each point on such curve, the surface has one principal direction along which the surface varies only a little and a theoretically infinite curvature in the other direction. In practice, we consider the feature curve as a highly curved region that spreads over a relatively significant distance in a given direction. In this way, we take into account not only extremely sharp edges but also smoother feature curves. This more fuzzy definition allows us to process damaged objects that have lost their initial sharpness. It also takes into account the fact that acquisition devices have a limited resolution, and it is quite unlikely that the acquired data contain points exactly on a sharp feature.

Following these ideas, this section presents a new method that was developed in the context of a contest on feature curves extraction from triangle meshes [Thompson 2019]. The goal is to automatically classify the feature curve vertices that have been manually labeled on the 15 meshes shown in Figure 3.7. One challenge is to adapt the recognition to the different anisotropic feature sizes, from the rough and smooth demarcation lines of model 3 to the more detailed curves of model 10. Our minimal curvature lines obtained on the APSS at multiple scales provide a relevant approach to this issue. We first explain our method in Section 3.4.1 and then present the numerical results in Section 3.4.2. For this contest, we consider a triangulated mesh as input and output data, but our method also handles unstructured point clouds data.

3.4.1 Multi-scale curvature lines voting

Our method is illustrated in Figure 3.6 (see also Figure 3.3). It extracts feature curves from meshes by generating a set of 3D polylines in the direction of minimal curvature at multiple scales (Section 3.3). We filter the lines to keep the salient ones, i.e. those that span highly curved regions of the estimated surface. Mesh vertices accumulate votes from neighboring lines, and a region growing process delineates individual set of vertices based on these votes. The resulting feature curves are the set of vertices around which most of the lines of minimal curvature are located. Using multiple scales ensures that all the various prominent feature curves are extracted. The following paragraphs explain in detail the algorithm.

Curvature lines We first sample the initial mesh $\mathcal{M}$ as a dense point cloud $\mathcal{P}$ to avoid potential artifacts caused by an irregular topology. The sampling is uniform and weighted by

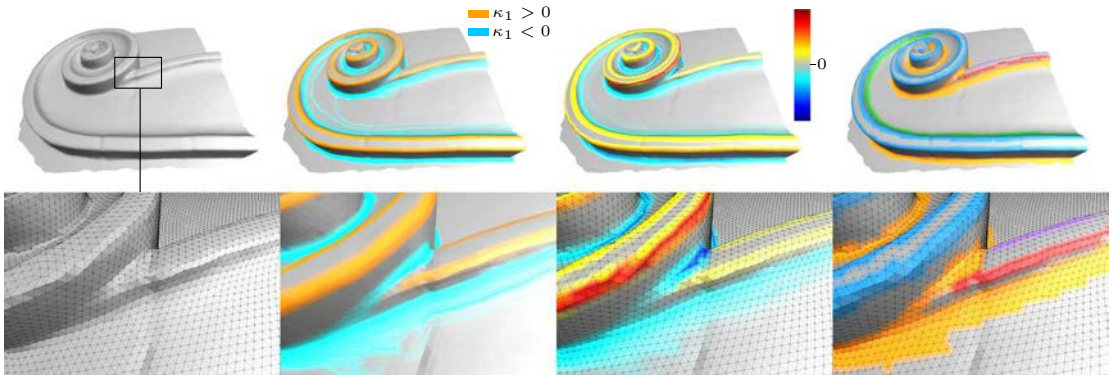


Figure 3.6: Feature curves detection pipeline. (a) Input mesh. (b) Minimal curvature lines extracted as explained in Section 3.3 at 5 scales, filtered, and classified as convex and concave lines in orange and blue. (c) Each line vertices gives a signed vote to its neighboring mesh vertices. (d) Individual feature curves are finally obtained using region growing based on the accumulated votes.

each face area. Sampled points of $\mathcal{P}$ are equipped with a unit normal vector, which is required for the APSS. The normal of a sampled point is the normalized interpolation of the vertices normal in the associated face. The point cloud $\mathcal{P}$ is composed of $P = 100 \cdot V$ points, where V is the vertex count of $\mathcal{M}$. If the input is not a triangle mesh, then $\mathcal{P}$ is simply the input point cloud.

We select 5 scales in $(t_{\min}, t_{\max})$ with $t_{\min} = \overline{e}$ and $t_{\max} = 3\,\overline{e}$, where $\overline{e}$ is the median edge lengths in the initial mesh $\mathcal{M}$, or the average distances to the k-nearest neighbors of each points in $\mathcal{P}$ if there is no edges. With this setting, we are able to handle the variety of feature size of all input meshes displayed in Figure 3.7. The minimal curvature lines are drawn as explained in Section 3.3. They are classified as convex and concave lines depending on the sign of κ_1 as shown in orange and blue in Figure 3.6(b).

We filter the lines vertices to only keep those where $||\kappa_1| - K| / K > 0.5$. The curvature bound K is the 90^{th} centile of maximal curvature in absolute value calculated on $\mathcal{P}$ at scale t_i. The threshold is fixed to 0.5 in all our experiments. In other words, these salient lines propagate only through areas that are curved enough relatively to the most curved location in the point cloud.

Votes accumulation Each vertex of each salient line accumulates a vote in its neighboring vertices of the initial data. The size of the spherical neighborhood is set to $\overline{e}$. The absolute value of a vote ranges from 0 to 1 according to the distance between the salient line vertex that emits the vote and the neighborhing vertex in the input mesh or point cloud that receives and accumulates it. Its sign is negative for concave lines ($\kappa_1 < 0$), and positive for convex lines ($\kappa_1 > 0$). This opposition in sign balances the sum of votes at vertices that are close to both convex and concave lines. In the end, each mesh vertex receives a signed sum of votes as shown by Figure 3.6(c). A feature curve on the mesh that is very significant persists at all the scales and thus receives a lot of votes.

Feature curves segmentation The set of vertices labeled as curve are extracted with a region growing based on the previous sum of votes at each vertex and performed on the mesh or the k-nearest neighbors graph if a point cloud is considered as input. A region grows from a vertex to its neighbor if their sum of votes have the same sign, and if they are either greater than $+v$, or lower than $-v$. The voting threshold is set to $v = 0.05 \cdot V_{\max}$ where $V_{\max}$ is the maximal absolute value of votes on the initial vertices. With this setting, a vertex belongs to a region if its sum of votes is high enough compared to the vertex that received the largest number of votes in absolute value. The result is a set of individual regions corresponding to the feature curves of the mesh (Figure 3.6(d)).

3.4.2 Results

This section reports the results obtained during the contest of feature line extraction on triangle meshes [Thompson 2019].

Input data The dataset displayed in Figure 3.7 consists in 15 meshes characterized by at least one feature curve. Both scanned and synthetic data are considered. Some models come from Aim@Shape [Falcidieno 2004] and the web [Turbosquid 2020]. The original models of the ornaments from which are derived models 4 to 10 are courtesy of the prof. Rodriguez Echavarria. The number of vertices ranges from 5K to 215K.

Groundtruth The definition of a groundtruth for this task is a challenging job since multiple definitions of feature curve on surfaces exists (see Section 3.2). They can be formally defined in terms of curvature derivatives [Ohtake 2004]. Normals angles could also help to define sharp edges [Koch 2019]. Moreover, artistic aspect could also be considered [Cole 2008]. In this contest, the groundtruth shown as colored regions in Figure 3.7 is manually defined by several computer scientists from the IMATI-CNR laboratory (Italy) who where asked to highlight the vertices of each model if, in their opinion, they are part of a feature curve. For a given model M, the groundtruth is a set of curves $f^*_M = \{f^*_{Mi}\}_{i=1\ldots n^*_M}$ where each curve f^*_{Mi} is represented as a set of mesh vertices $f^*_{Mi} = \{\mathbf{v}^*_{Mi,j}\}_{j\ldots n_{Mi}}$ ($*$ denotes the groundtruth).

Evaluation metric Two types of evaluation are conducted. The *overall* comparison does not consider individual lines but their union of vertices over each mesh. In this case, the groundtruth of a model M can be seen as one single curve which is the union of all the feature curves vertices $\cup_i f^*_{Mi}$. The *detailed* comparison compares the results curve-by-curve where each curve produced by a method is matched to its closest curves in the groundtruth. The matching is done manually by the same people who determine the groundtruth. In case a different number of curves are detected than in the groundtruth, only the matched ones are evaluated.

Two evaluation metrics are used to compare a groundtruth curve f^*_{Mi} to a resulting curve f_{Mi}. The Hausdorff distance [Deza 2009, Section 1.5] is a positive distance expressed in global coordinates

$$d_H(f^*_{Mi}, f_{Mi}) = \max\left\{ \max_{\mathbf{v}^*\in f^*_{Mi}} \min_{\mathbf{v}\in f_{Mi}} d(\mathbf{v}^*, \mathbf{v}), \max_{\mathbf{v}\in f_{Mi}} \min_{\mathbf{v}^*\in f^*_{Mi}} d(\mathbf{v}^*, \mathbf{v}) \right\}, \tag{3.3}$$

where $d_H = 0$ occurs for a perfect alignment between the two curves. The Dice coefficient [Deza 2009, Section 17.1] is a value in $(0, 1)$ measuring the overlap of the two sets

$$D(f^*_{Mi}, f_{Mi}) = 2\frac{|f^*_{Mi} \cap f_{Mi}|}{|f^*_{Mi}| + |f_{Mi}|}, \tag{3.4}$$

where $D = 1$ means that the two sets are equal.

Methods The first method to be evaluated is proposed by Hoang-Xuan et al. and is called *point aggregation based on angle and curvature saliency* (PCs). They calculate discrete curvatures at each vertex, and vertices with curvature higher than a fixed threshold finally define a unique feature curve. Two versions are proposed according to how they estimate the curvatures at vertices, using either edges angles (PCs:A) or edges curvatures (PCs:C). This method is not evaluated in the detailed comparison since only one feature curve is extracted for each model.

The second method of Arvanitis and Moustakas adopts a *spectral based saliency estimation* (SBSE). They define a vertex saliency measure as $1/\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}$, where λ_i are the eigenvalues of the covariance matrix of the 15 neighboring normal vectors. Five k-means algorithms are executed on the point-wise normalized saliency with k ranging from 1 to 5. The value of k that maximize the Calinsky-Harabasz index (inter-cluster over intra-cluster variance) produces the resulting k feature curves while ignoring the first two clusters associated to the lowest curvatures.

The last technique proposed by Romanengo et al. is called *feature curve characterization via mean curvature and algebraic curve recognition via Hough transforms* (MHT)[Torrente 2018]. The mesh vertices that have a low minimal curvature and a high maximal curvature are first selected using two thresholds set to 15% and 85% with respect to curvatures distributions. The feature curves are obtained with the DBSCAN algorithm using the k-nearest neighbors graph as underlying topology with $k = 15$ for MHT1 and $k = 4$ for MHT2. Note that the Hough transform is not used for the feature curve extraction but is rather used to compare curves with each other, which is another task.

Numerical results The results of our method are illustrated in Figure 3.8. We summarize the comparison results in Table 3.1 where the average of the two metrics over all models and all curves are reported. The complete comparison results can be found in Appendix C. In the detailed comparison, we excluded all the Hausdorff distances greater than 2 corresponding to 3 outliers of SBSE and 1 outlier of our method for the model 10.

Our method gives the best results on average for the Hausdorff distances, while MHT is better with respect to the Dice coefficient. As we can see in Figure 3.8, almost all the feature curves of the groundtruth are detected by one or more curves with our method. However, our curves are larger than the groundtruth which decreases the Dice coefficient more than it increases the Hausdorff distances. The resulting curves of MHT are more thin and thus more accurate in terms of Dice coefficient, but some of their curves are spatially farther than the groundtruth increasing the average Hausdorff distances.

The other two methods give largely inferior results. PCs only uses a hard and global thresh-

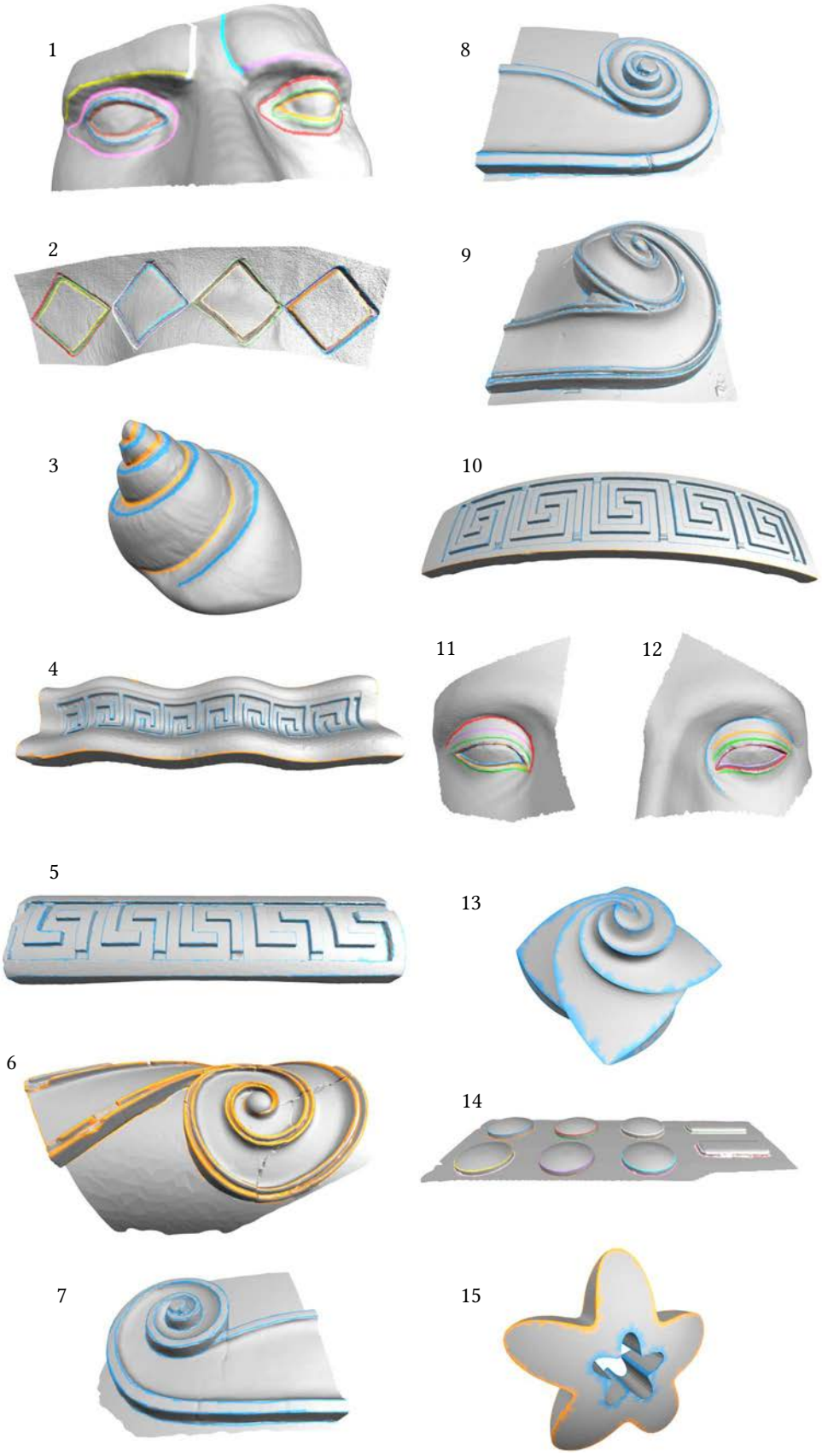


Figure 3.7: Groundtruth. Manually annotated feature curves depicted in different colors.

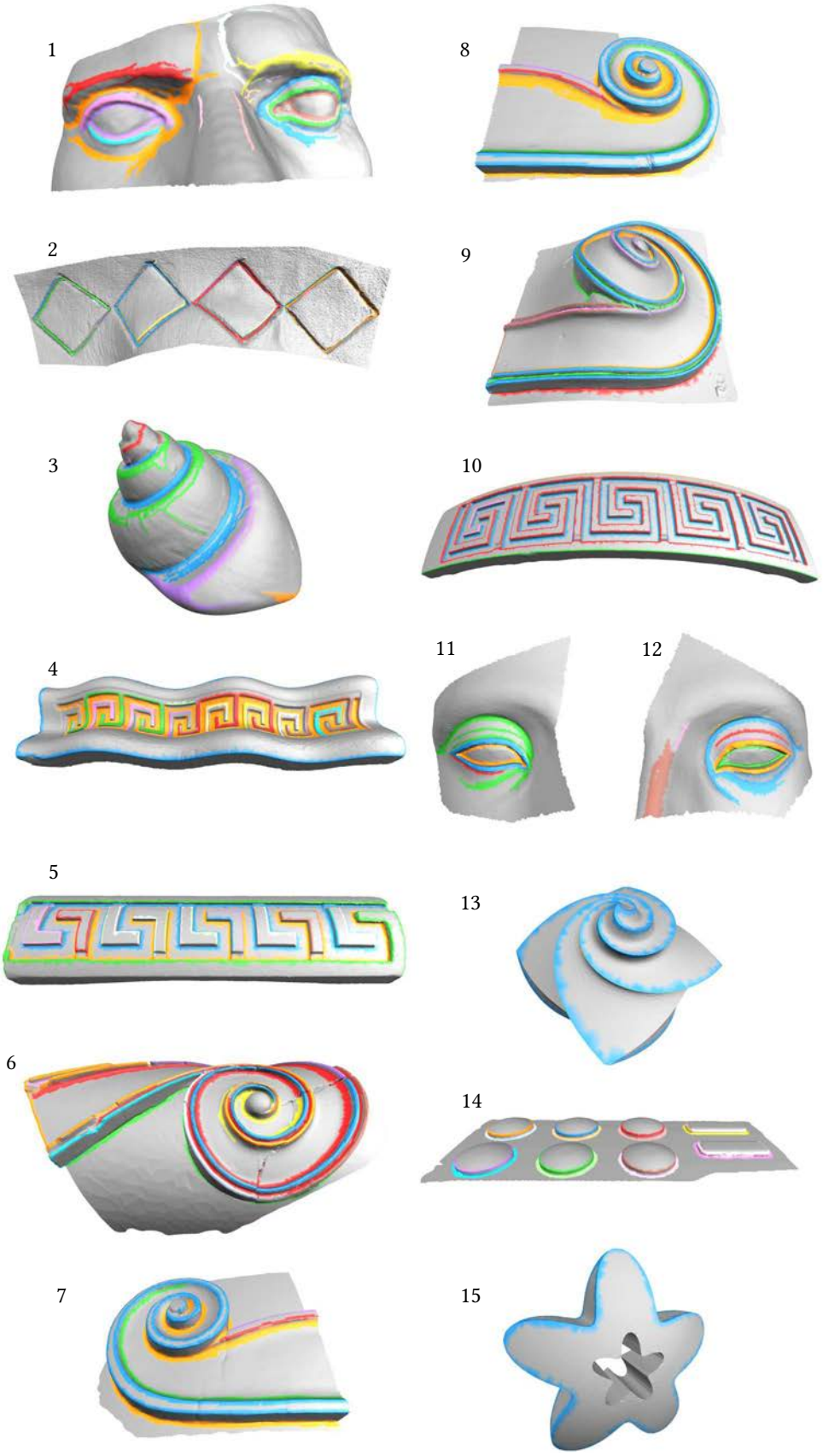


Figure 3.8: Results. The resulting feature curves obtained with our method explained in Section 3.4.1.

Comparison	Metric	PCs:A	PCs:C	SBSE	MHT1	MHT2	ours
Overall	d_H	0.879	0.711	0.966	0.915	0.925	**0.604**
Overall	D	0.518	0.525	0.374	**0.553**	0.552	0.504
Detailed	d_H	-	-	0.448	0.187	0.086	**0.084**
Detailed	D	-	-	0.207	**0.541**	0.530	0.466

Table 3.1: Numerical comparison. Numerical comparison between the results of each method and the groundtruth in overall (curves union) and in details (curve-by-curve) using the Hausdorff distance d_H (0 is better) and the Dice coefficient D (1 is better) averaged over all the 15 models (see Appendix C for the complete comparison).

olding on curvatures, which poorly adapts for each model, and SBSE performs the k-means in curvature space without any locality criterion. On the other hand, MHT and our method both take into account spatial positions of the feature curves thanks to the DBSCAN and our minimal curvature lines extraction. Since our method uses multiple scales to generate the curvature lines, we are able to extract almost all the various feature curves of the meshes.

3.5 Multi-scale cylinders detection

We present a method to detect cylindrical shapes (Figure 3.1) in 3D unstructured point clouds using our minimal curvature lines introduced in Section 3.3. We formally define cylindrical shapes as regions of a 3D surface where the minimal curvature lines are sufficiently regular and aligned with each other. The method illustrated in Figures 3.9 and 3.3 first generates the minimal curvature lines (Section 3.3). In a second step, they are filtered in order to keep the regular ones, where the regularity criterion involves the total length, curvature and scale-space stability of the lines (Section 3.5.1). In order to highlight point cloud regions where curvature lines are well aligned, we propose a local linear anisotropy measure in Section 3.5.2. Finally, Section 3.5.3 explains how the final regions (sets of points) are segmented using a region growing algorithm.

Our method can detect any number of cylinders that do not need to be closed nor straight. The algorithm can be performed at multiple scales so that various kind of cylindrical regions can be detected from detailed to global ones. This approach is also a potential research direction to include cylindrical primitives in our persistence analysis framework presented in Chapter 2.

3.5.1 Curvature lines filtering

We first filter the minimal curvature lines introduced in Section 3.3 in order to keep only the most regular ones, i.e. those which travel along smooth and clean cylindrical regions. For the filtering, we consider the length $\mathcal{L}$, the total curvature $\mathcal{K}$ (sum of vertex angles), and the total

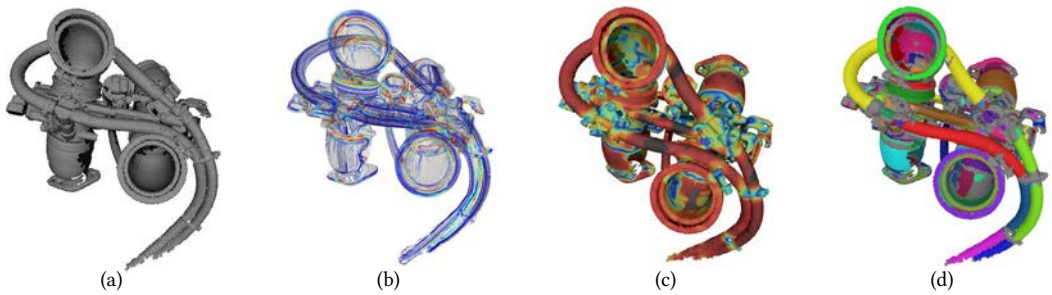


Figure 3.9: Cylinders segmentation pipeline. (a) Input point cloud. (b) Minimal curvature lines introduced in Section 3.3 and filtered in Section 3.5.1. (c) Linear anisotropy (Section 3.5.2). (d) Segmentation (Section 3.5.3). This algorithm can be run at different scales.

geometric variation $\mathcal{V}$ of a line $\{\mathbf{q}_i\}_{i=1\ldots n}, \mathbf{q}_i \in \mathbb{R}^3$

$$\mathcal{L} = \sum_{i=1}^{n-1} \|\mathbf{q}_{i+1} - \mathbf{q}_i\|, \tag{3.5}$$

$$\mathcal{K} = \sum_{i=2}^{n-1} \arccos\left(\frac{(\mathbf{q}_i - \mathbf{q}_{i-1}) \cdot (\mathbf{q}_{i+1} - \mathbf{q}_i)}{\|\mathbf{q}_i - \mathbf{q}_{i-1})\|\|(\mathbf{q}_{i+1} - \mathbf{q}_i\|}\right), \tag{3.6}$$

$$\mathcal{V} = \sum_{i=1}^{n} \tilde{\nu}_i, \tag{3.7}$$

where $\tilde{\nu}_i$ is a normalized version of the the geometric variation ν [Mellado 2012, Equation 5] mapped in $(0, 1)$ using $\tilde{\nu}_i = \tanh(\nu_i)$. Three thresholds $\mathcal{L}_{\min}$, $\mathcal{K}_{\max}$ and $\mathcal{V}_{\max}$ are set by the user depending on the point cloud.

3.5.2 Curvature lines anisotropy

The goal is to quantify how much the minimal curvature lines are aligned near the points of the input point cloud. To do that, we propose to compute the following covariance matrix at scale t and at point $\mathbf{x}$

$$\Sigma_t(\mathbf{x}) = \sum_{\mathbf{q_i} \in \mathcal{N}_t(\mathbf{x})} (1 - \tilde{\nu}_i)\, w_t(\mathbf{x} - \mathbf{q}_i)\, \mathbf{v}_i \mathbf{v}_i^T, \tag{3.8}$$

where w_t is the spatial weighting function defined by Equation 1.6 using support size t (the scale), $\mathbf{v}_i$ is the line tangent at $\mathbf{q}_i$ (minimal curvature direction), and $\mathcal{N}_t(\mathbf{x})$ are the lines vertices within a distance t from $\mathbf{x}$. Σ_t corresponds to a weighted covariance matrix of the lines tangents close to $\mathbf{x}$. The weighting involves the distance between the points $\mathbf{x}$ and $\mathbf{q}_i$ so that the closer a line vertex the greater its influence. The normalized geometric variation $\tilde{\nu}$ also penalizes tangents that are too unstable in the scale-space.

We calculate the linear anisotropy [Peeters 2009, Table 1] to quantify the local lines alignment

$$\sigma = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}, \tag{3.9}$$

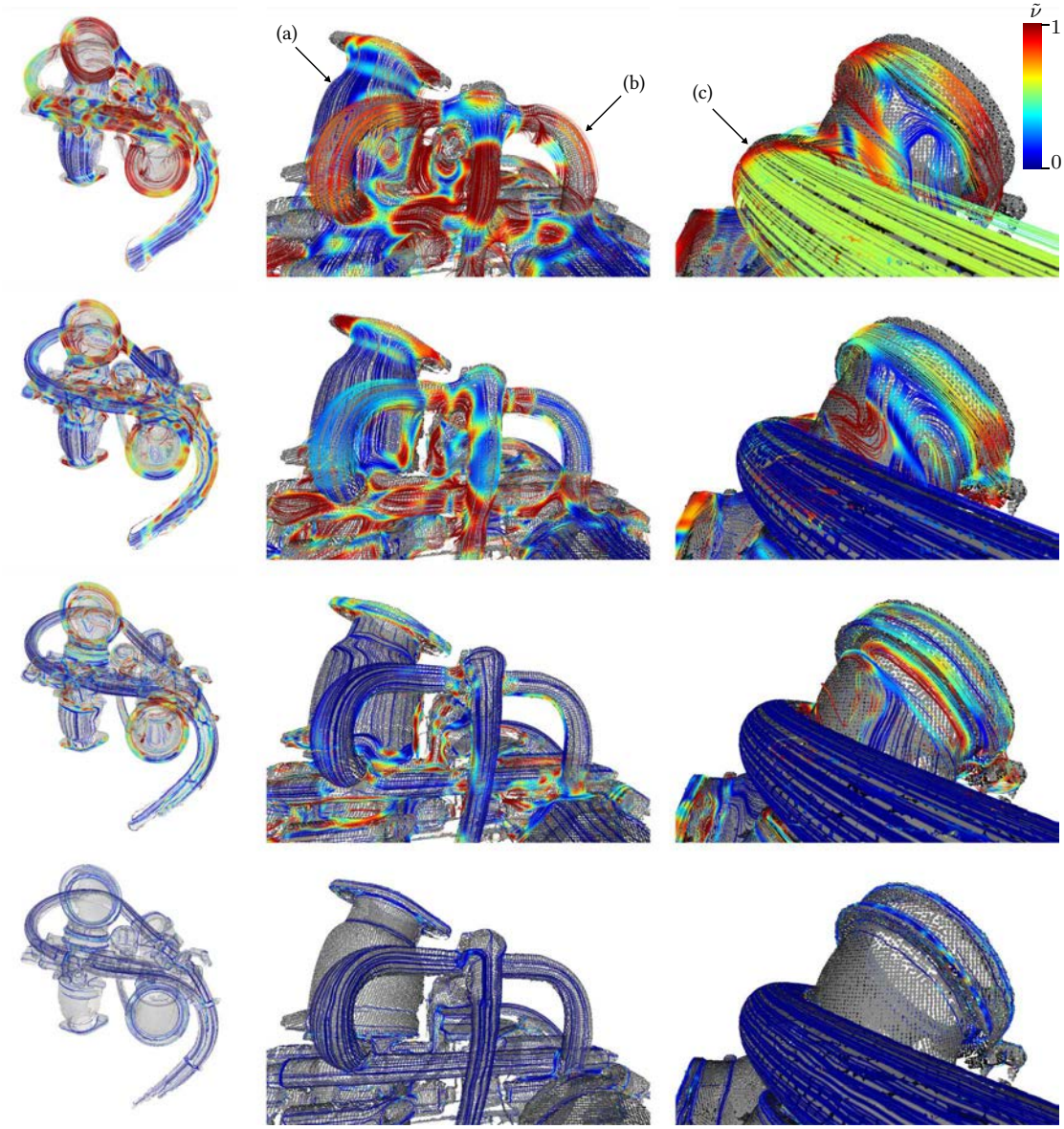


Figure 3.10: Filtered minimal curvature lines. Minimal curvature lines filtered based on attributes of Equations 3.5-3.7 at 4 increasing scales from bottom to top. The large pipe (a) is spanned by stable lines at high scale only. The small pipe (b) is spanned by stable lines at low scale and unstable lines at high scale. The medium pipe (c) is spanned by stable lines at many scale except at the highest one where the lines become unstable.

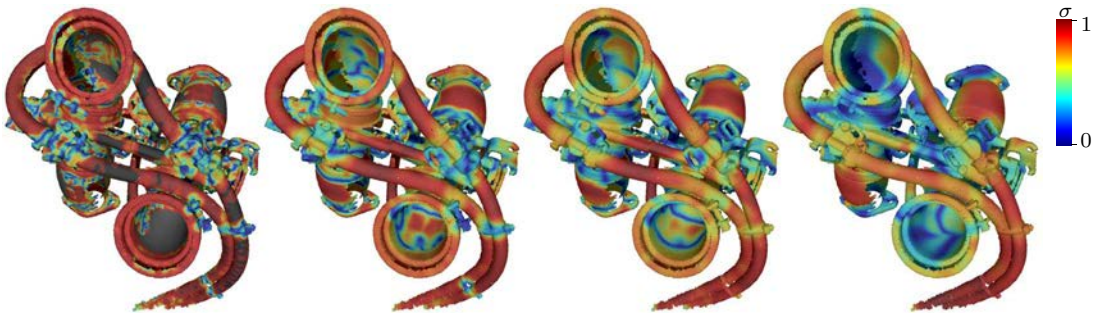


Figure 3.11: Curvature lines anisotropy. Linear anisotropy σ (Equation 3.9) computed at 4 increasing scales from left to right, based on the minimal curvature lines of Figure 3.10.

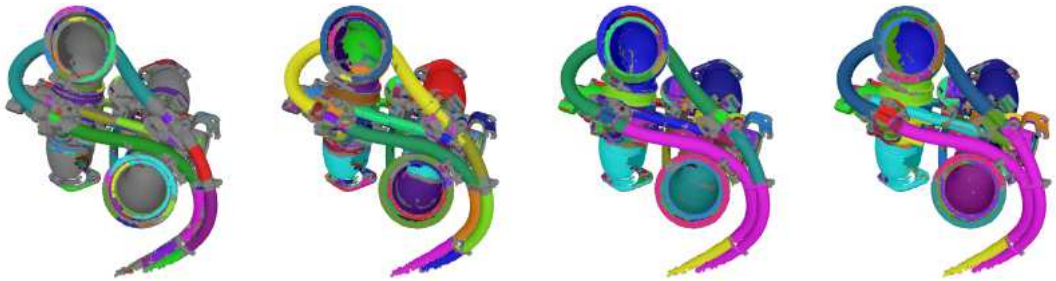


Figure 3.12: Cylinder segmentations. Results of the region growing performed at 4 increasing scales from left to right, based on the linear anisotropy of Figure 3.11.

where $\lambda_1 > \lambda_2 > \lambda_3$ are the (positive) eigenvalues of Σ_t. This scalar coefficient takes its values in $(0, 1)$. If $\sigma = 1$ then all the lines tangents at points $\mathbf{q}_i \in \mathcal{N}_t(\mathbf{x})$ are perfectly aligned meaning that the point $\mathbf{x}$ is likely on a cylindrical region. Figure 3.11 shows the linear anisotropy computed at each input points.

3.5.3 Multi-scale cylinders segmentation

In the end, the input point cloud is segmented in different regions in which all the points share the same linear anisotropy, and the line tangents and normals are smooth. To achieve this, an unseeded region growing is performed on the k-nearest neighbors graph of the 3D point cloud. A region grows from a point $\mathbf{p}_i$ to a neighbor $\mathbf{p}_j$ if and only if

$$\|\sigma_i - \sigma_j\| < \varepsilon \qquad \text{and} \qquad |\mathbf{v}_i \,.\, \mathbf{v}_j| > 1 - \varepsilon \qquad \text{and} \qquad \mathbf{n}_i \,.\, \mathbf{n}_j > 1 - \varepsilon, \tag{3.10}$$

where ε is a unique threshold set to 0.1. Note that the linear anisotropy coefficients σ_i, σ_j and the two dot products appearing in Equation 3.10 are all in $(0, 1)$, so a common threshold ε is well adapted. As we can see on the bottom right of Figure 3.12, two small pipes close to each others are joined in a unique region at high scale whereas they form two distinct regions at a lower scale.

3.6 Conclusion

We propose a method to generate curvature lines from unstructured point clouds in 3D. The originality lies in the intuitive scale parameter coming from the weighting kernel support of the APSS that controls the level of details of the resulting lines. Considering only the salient lines permits to robustly detect feature curves on 3D shapes using a voting approach. We also introduced a point-wise anisotropy measure computed from regular minimal curvature lines. Then, a region growing segments the input point cloud in cylindrical regions. This process can be done at different scales in order to detect a wide range of cylindrical shapes.

Future work Using our detected cylindrical regions in our persistence analysis framework of Chapter 2 is a possible research direction. The segmentation method presented in Section 3.5 could replace the planar segmentation of Section 2.3.1. The resulting cylindrical regions can be used to build a hierarchical graph, and persistent components would represent important anisotropic parts of the point cloud.

A strong limitation that must be addressed is the need of several user parameters to filter the minimal curvatures (see Equations 3.5-3.7). Thresholds are also present in our feature curves detection algorithm. To avoid these issues, different solutions can be considered such as the use of machine learning. It would avoid the trial and error approach implied by the thresholdings.

Memory is another limitation of the methods of this chapter. For now, multi-scale curvature lines are computed and stored before any further processing. The required memory can be significant depending on the number of scales, the number of lines and their resolution. We wonder if their storage is necessary for the anisotropy computation and the voting system. Processing the lines as they are generated is a technical challenge we also would like to address in the future.

Crest and valley lines detected on the surface of the APSS are a possible future work since very few meshless techniques exist to generate such salient lines. The shape operator of Equation 1.38 could be differentiated again in order to obtain principal curvature derivatives. The main question is whether or not these third order differential properties show strong sensitivity to small perturbations as it is usually the case with high order derivatives. Another objective could be to derive theoretical guarantees for such characteristic lines extraction with respect to noise and sampling irregularity. Moreover, the anisotropy measure asymptotically derived from the algebraic sphere regression (see Section 1.3.2) could be further investigated in the context of feature line generation.

We believe that many geometry processing applications could benefit from feature lines drawn on 3D shapes. Recognizing complex geometric patterns over 3D surfaces [Biasotti 2018] is one example. Analyzing, matching and simplifying 3D curves is indeed simpler then processing directly the surface itself thanks to the low dimension and the natural parametrization of curves.

Chapter 4

Point cloud parametrization

Contents

4.1 Introduction

In geometry processing, it is common to assume that the analyzed surface is a 2-manifold embedded in 3D. This hypothesis is motivated by the intuitive observation that a simple object defined by its interior volume has a boundary surface infinitely thin and that looks like a plane from an infinitely close viewpoint. It also has an advantageous practical aspect since data structures and algorithms are often easier to develop than if the surface is considered as non-manifold. The smooth setting is better translated into the discrete one, and special cases near non-manifold regions are thus avoided.

In the context of 3D point cloud analysis, many methods make this hypothesis for the underlying surface represented by the acquired samples. They include the integral invariants [Pottmann 2007], the Osculating Jets [Cazals 2005a] and the WaveJets [Béarzi 2018], the Point Set Surfaces [Alexa 2001] including the Algebraic Point Set Surfaces [Guennebaud 2007], several spectral approaches [Belkin 2009, Liu 2012], many methods computing local geometric features [Pauly 2003, Mérigot 2010, Mellado 2012], most of triangulation algorithms [Amenta 2001, Ohtake 2003, Kazhdan 2006], etc...

Parametrization is a powerful tool to process discrete 2-manifolds as triangle meshes. It provides a map between a 2D parameter domain and the 3D coordinates of the surface. Applications involving parametrizations of 3D shapes are numerous. In rendering, fine details can be efficiently encoded in textures as color and displacement maps. Details transfer, morphing and re-meshing are other examples of edition that usually benefit from parametrization. In addition to shape analysis, it also plays an important role in fabrication and simulation for instance.

Various techniques have been proposed to compute a parametrization of a triangle mesh [Botsch 2010, Chapter 5]. Since they are all based on triangles, they cannot be directly used on unstructured point clouds. In the acquisition pipeline, one solution could be to first triangulate the points, and then compute the parametrization. But the triangulation step remains difficult and could potentially impact the quality of the parametrization. Our goal is thus to directly obtain a mapping from a 2D domain to the input 3D points without any intermediate mesh.

In practice, the goal of a point cloud parametrization method is to find a point $\mathbf{u} \in \mathbb{R}^2$ lying in a flat parameter domain for each point $\mathbf{p} \in \mathbb{R}^3$ of the input point cloud. Depending on the application, interesting properties are often required on the resulting map between $\mathbb{R}^2$ and $\mathbb{R}^3$, such as being fold-free (injective), conformal (angle preserving), authalic (area preserving), isotropic (length preserving) or harmonic (minimizing stretch).

As described in the state-of-the-art (Section 4.2), few methods exist to process a raw point cloud. They are dedicated to either planar or spherical parametrization, but not both. They usually use local Delaunay triangulations in tangent spaces or k-nearest neighbors graph in order to adapt existing mesh parametrization algorithms. These approaches often suffer from strong noise and sampling irregularity that are commonly found in acquired data. To our knowledge, no meshless method exists to measure the distortion of a point cloud parametrization.

Key idea The key element of our meshless parametrization method is the Point Set Surfaces [Alexa 2001] already discussed in Section 1.2. Moving Least Squares (MLS) fitting [Levin 1998] approximate locally and for each point in space a smooth 2-manifold from scattered data. The locality is controlled by a scale parameter defining the neighborhood used for the fitting. Progressively increasing the scale while keeping points onto their evolving MLS surface approximation leads to a situation where all the points are finally projected onto the same global surface. We use this iterative scale-space approach with the Algebraic Point Set Surfaces [Guennebaud 2007] so that points converge toward a common sphere or plane automatically. The result is thus a global planar or spherical 2D parametrization of the input points.

This chapter corresponds to a preliminary work carried out during an ongoing research project. We search for a method that would map any 3D point cloud to a 2D domain. The long term objective is to ensure that the resulting parametrization has useful properties such as preserving angles or areas. In this thesis, we only provide an algorithm that flatten a genus-0 point-sampled surface in 2D, and a method to calculate the parametrization distortions. These distortion measures are a first step toward computing optimal maps of point cloud which is left as future work.

Contributions of this chapter

- We propose in Section 4.3 a new algorithm to automatically flatten a point-sampled surface to a sphere or a plane producing a planar or spherical parametrization. The sampled surface needs to be of genus 0, but it can be closed or open, with any number

of holes boundaries that are implicitly handled. Calculations remain local, so our method can process several millions of points in a reasonable time.

- Meshless parametrization distortion measures are then introduced in Section 4.4. They quantify at any scale the distortion of angles and areas induced by the resulting mapping between the points in 3D and their 2D counterparts. These measures could be used in the future to optimize our results to obtain conformal or authalic parametrizations.

4.2 State-of-the-art

We review in Section 4.2.1 the existing methods that compute a global parametrization of unstructured point clouds. Since our method uses an iterative scheme that makes the point-sampled surface evolve under a specific flow, we also discussed the literature on geometric flows in Section 4.2.2.

4.2.1 Point cloud parametrization

Meshless parametrization methods aim at finding a discrete map from a 2D parameter domain to the 3D input points without using any global triangulation.

Barycentric mapping introduced for graphs [Tutte 1963] and triangular meshes [Floater 1997] can be used to map a point cloud on a plane [Floater 2001]. The input point cloud is splitted into two disjoint sets of interior and boundary points. The boundary points are first mapped to a convex planar polygon, and a sparse linear system is then solved to map the interior points so that each point is a convex combination of its neighbors. However, an ordered and cyclic list of boundary points is required, which is difficult to automatically identify on unstructured point clouds. For closed surface, this method is adapted to map the points on a sphere [Hormann 2002], but the point cloud must be partitioned into several charts, which is also difficult when dealing with complex point-sampled shapes.

Laplacian graph embedding of the k-nearest neighbors graph on the unit sphere is also possible using an iterative procedure [Zwicker 2004]. All the 3D points are first projected onto the unit sphere. Then, each point moves on the sphere toward the center of gravity of its initial k-nearest neighbors. However, the first orthogonal projection on the sphere leads to important folds in the parametrization if the input shape is not already close to a sphere which is often the case. A similar approach makes use of stereographic projections to compute a conformal map [Choi 2016]. Laplace operators recently introduced for point clouds [Sharp 2020] can also help to compute a spectral conformal parametrization [Mullen 2008].

Direction fields can guide the computation of global parametrization for a point-sampled surface of any genus in the context of quadrangulation [Li 2011a]. The point cloud is cut in region topologically equivalent to a disk, and a mixed integer solver guarantees that the resulting parametrization is seamless across the cut lines [Bommes 2009]. The result is dependent on the cross field of principal curvature directions computed from the k-nearest neighbors graph.

Discrete one-forms enable to map a genus-1 point cloud to a toroidal domain [Tewari 2006], taking inspiration from a method applying to meshes [Gu 2003]. On meshes, this approach attempts to preserve edges length under the constraint that edges of a face on the 2D domain form a closed triangle. On unstructured point clouds, the k-nearest neighbors graph edges are considered for the length preservation, and the closedness constraint is modified so that all trivial graph cycles are closed in the parameter domain.

A few machine learning approaches have been proposed such as a Radial Basis Function neural network [Meng 2013] that uses unsupervised self-organizing map to parametrize range images, and AtlasNet [Groueix 2018] that learns how to generate an atlas from a sampled surface in order to reconstruct surfaces from images.

Performance is a common limitation to these techniques since they often involve solving large systems or costly optimization algorithms, which is not compatible with point clouds of several millions of samples. In addition, many meshless algorithms use either a local triangulations in tangent spaces or a k-nearest neighbors graph that are both sensitive to noise, outliers and highly varying densities.

4.2.2 Geometric flows

Several discrete geometric flows morph a surface to a sphere at convergence. The modified mean curvature flow [Kazhdan 2012] updates at each iteration the mass matrix but keeps the initial stiffness matrix in order to avoid any singularities. The volumetric viscous flow [Cohen 2019] takes inspiration from fluid dynamics to move a star-like closed surface on a sphere. Each intermediate surface is included in the previous one creating a foliation of the volume enclosed by the mesh. Another alternative consists in diffusing the mean curvature, and then integrating the underlying triangular mesh to match the current curvature, which corresponds to a Willmore flow [Crane 2013]. A similar method integrates the mesh to match the averaged neighboring faces normal [Zhao 2020]. This flow converges toward a sphere or a plane depending on the geometry of the mesh.

On unstructured point clouds, projecting a point onto a local PCA plane (Equation 1.1) implements a mean curvature flow [Digne 2011]. Digne et al. perform some iterations of PCA projection to produce a smoother point cloud used to generate a triangular mesh with the ball pivoting algorithm [Bernardini 1999]. We take inspiration from this approach to map a point cloud onto a 2D domain with two significant differences. First, the plane fitting is replaced by an algebraic sphere fitting, which enables the surface to converge on a plane or on a sphere. Important singularities are also avoided as already demonstrated in Figure 1.2. Then, we progressively increase the neighborhood size (the scale) to ensure a faster convergence on the final planar or spherical domain.

4.3 Scale-space point cloud parametrization

We propose a new method to automatically flatten a 3D point cloud onto a sphere or a plane, which provides a global parametrization of genus-0 point-sampled surfaces. The idea illustrated in Figure 4.1 consists in iteratively projecting each point onto their

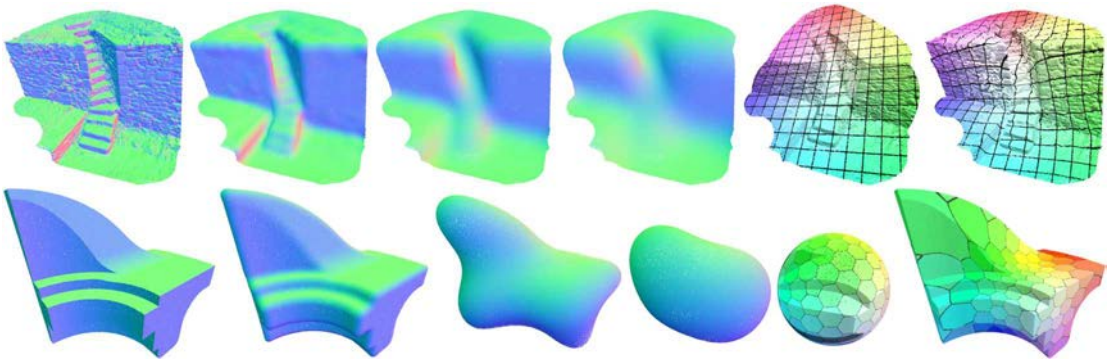


Figure 4.1: Parametrization overview. From left to right: initial point clouds, 3 intermediate projection steps (out of 50), converged points on a plane (top) or on a sphere (bottom), and the initial point clouds colored according to their planar or spherical parametrization.

Moving Least Squares (MLS) approximation [Levin 1998]. We progressively increase the scale (neighborhood size) until the whole point cloud is considered for the local surface fitting. At the end, a last fitting step is performed globally, without any MLS weighting, so that all the points finally lie on a single canonical surface. Thanks to the Algebraic Point Set Surfaces (APSS) [Guennebaud 2007] and its internal algebraic sphere fit procedure already introduced in Chapter 1, the point cloud necessarily converge toward a sphere or a plane depending only on the shape of the sampled surface.

We sample M scale values using the heterogeneous scale-space sampling introduced in Section 1.5.1 in order to handle varying point densities across the point cloud. The scale-space is linear (Equation 1.39) meaning that the scales are uniformly sampled between the local point spacing of each point and the global cloud size. Contrary to the logarithmic sampling used in the previous chapters, it avoids too large gaps between two iterations, decreasing the chances to produce unwanted folds in the resulting parametrization. At each step, the projected points resulting from the previous iteration are used to compute the APSS for the next one. The multi-resolution approach of Section 1.5.2 is used to progressively decimate the point cloud when the scale grows.

Our method does not explicitly manage boundaries since they are well handled by the APSS without any special treatment. Point clouds having several boundaries due to missing acquisition data and holes in the scanned surface are correctly flattened. The underlying surface can be closed or open, with disk or sphere topology, as long as its genus is equal to 0. The main limitation is thus the lack of guarantee about folds and any parametrization distortion minimization.

Results Various point clouds and their flattened version on a plane and on a sphere are shown in Figures 4.2 and 4.3 respectively. We set the number of scales to 50 in all these experiments. As expected, mostly planar point clouds such as those of Figure 4.2 converge toward a plane. But it is not necessarily the case as demonstrated by the fountain point cloud in Figure 4.3-bottom-left. This point cloud is quite planar but is flattened onto a sphere with a high radius. On average, the 50 iterations take 75 seconds per million points using an Intel Xeon CPU 3.70GHz and a NVIDIA GeForce GTX 1080.

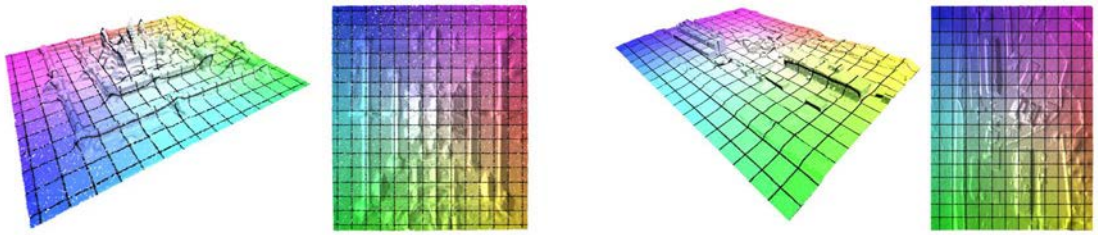


Figure 4.2: Planar parametrization. Point clouds and their planar parametrization.

4.4 Meshless distortion measures

Distortions induced by a map between a parametrization domain and a smooth regular surface can be determined by comparing a unit circle in the first domain and its associated ellipse on the surface. They can be quantified using the eigenvalues λ_1 and λ_2 of the 1st fundamental form of the map. Indeed, a unit circle in the initial domain is mapped to an ellipse on the tangent space of the surface whose semi axis are $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$. An ideal parametrization without any distortion, also called isometric or length-preserving map, associates circles to identical circles, meaning that $\lambda_1 = \lambda_2 = 1$ everywhere on the surface. A parametrization is authalic, or area-preserving, if $\lambda_1\lambda_2 = 1$, and it is conformal, or angle-preserving, if $\lambda_1 = \lambda_2$. When considering the parametrization of meshes, the 1st fundamental form and its eigenvalues can by analytically calculated on a pair of mapped triangles [Sander 2001, Section 3]. Various methods propose different distortion measures based on this eigendecomposition such as the Green-Lagrange deformation [Maillot 1993], the Most Isometric Parametrization of Surface [Hormann 2012], the stretch measure [Sander 2001], the symmetric energy [Sorkine 2002] or the combined energies [Degener 2003]. This does not hold with unstructured point clouds since there is no equivalent of triangle as minimal piece of surface on which these calculations can be performed.

We propose a meshless approach that compares an ellipse representing locally the 3D points distribution in the tangent space with its associated ellipse computed with the same 2D flattened points. To achieve this, we gather the neighboring points indices $\mathcal{N}_i$ near a 3D initial point $\mathbf{p}_i$ in order to compute the two weighted covariance matrices

$$\Sigma_i^{3D} = \frac{1}{\sum_{j\in\mathcal{N}_i} w_{ij}} P_i \sum_{j\in\mathcal{N}_i} w_{ij}(\mathbf{p}_j - \bar{\mathbf{p}}_i)(\mathbf{p}_j - \bar{\mathbf{p}}_i)^T P_i^T, \tag{4.1}$$

$$\Sigma_i^{2D} = \frac{1}{\sum_{j\in\mathcal{N}_i} w_{ij}} \sum_{j\in\mathcal{N}_i} w_{ij}(\mathbf{u}_j - \bar{\mathbf{u}}_i)(\mathbf{u}_j - \bar{\mathbf{u}}_i)^T, \tag{4.2}$$

where $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{u}}_i$ are the weighted average of 3D and 2D points respectively, P_i is the transformation matrix that projects a 3D point onto the 2D tangent plane at $\mathbf{p}_i$, and w_{ij} is a smooth decreasing weight function depending on the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$. The objective is to quantify the deformation between these two ellipses. By denoting $\lambda_1^{3D} \geq \lambda_2^{3D}$ the two eigenvalues of Σ_i^{3D}, we can calculate the anisotropy $\sigma^{3D} = \lambda_2^{3D}/\lambda_1^{3D}$ and the area $a^{3D} = \pi\lambda_1^{3D}\lambda_2^{3D}$ of the ellipse that represents the covariance matrix Σ_i^{3D}. The anisotropy σ^{2D} and the area a^{2D} of the ellipse in 2D are similarly calculated. Intuitively, the map has no distortion at all if these

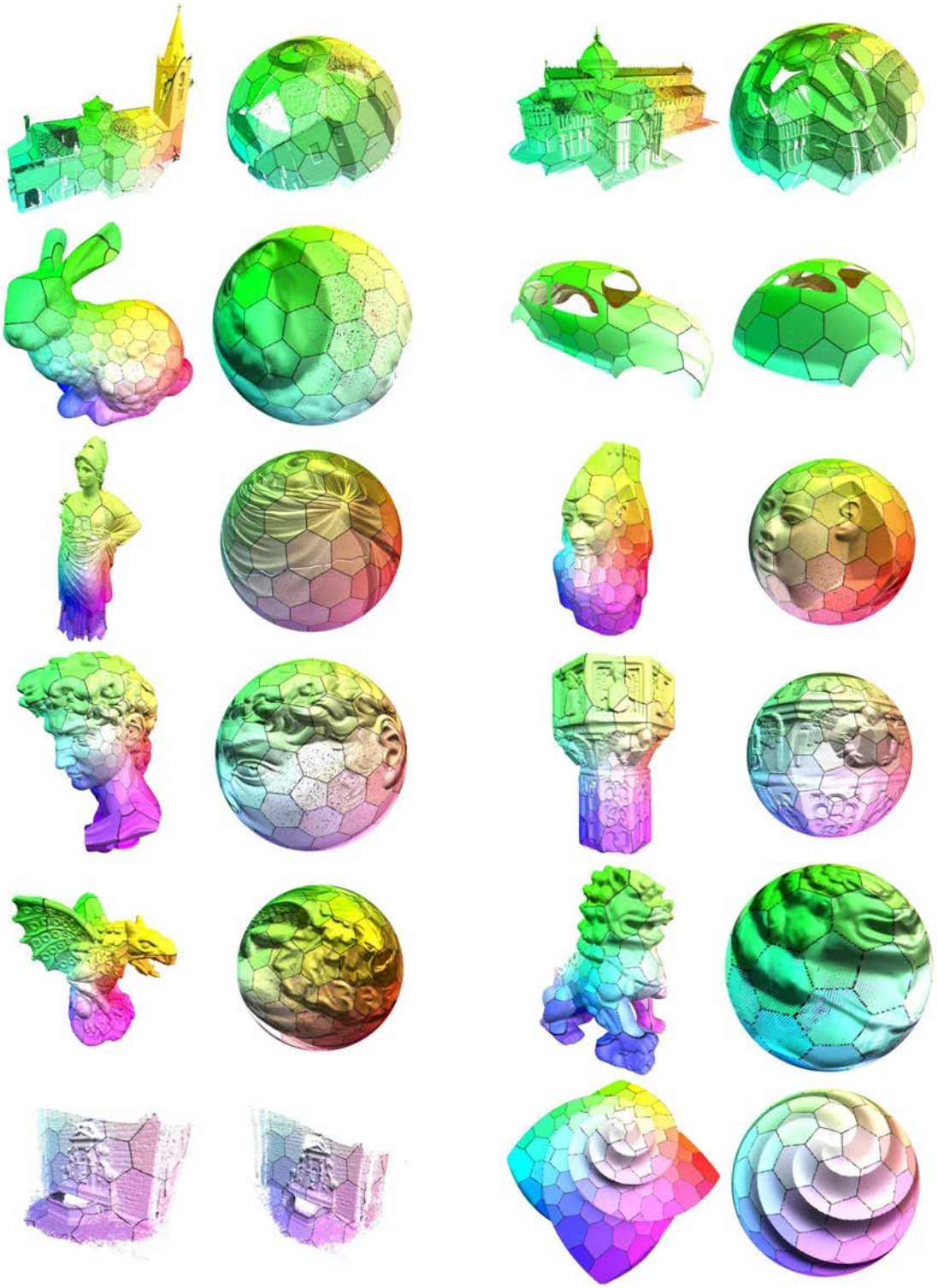


Figure 4.3: Spherical parametrization. Point clouds and their spherical parametrization.

two ellipses are equal. It is conformal if one ellipse is just a scaled and rotated version of the other, so that their anisotropy is the same. Finally, it is authalic if the two ellipses areas are equal. This leads to a conformal and an authalic distortion measures for a map between point clouds

$$\varepsilon_{\text{conf}} = \text{ratio}\left(\sigma^{3D}, \sigma^{2D}\right), \tag{4.3}$$

$$\varepsilon_{\text{auth}} = \text{ratio}\left(a^{3D}, a^{2D}\right), \tag{4.4}$$

where $\text{ratio}(a, b) = \max(a, b)/\min(a, b)$. Each of them is greater or equal to 1, and the map is conformal if $\varepsilon_{\text{conf}} = 1$, and authalic if $\varepsilon_{\text{auth}} = 1$. For instance, if $\varepsilon_{\text{conf}} = 2$, then it means that one ellipse is twice stretched in one direction compared to the other one.

Results Figure 4.4 shows the conformal distortion $\varepsilon_{\text{conf}}$ and the authalic distortion $\varepsilon_{\text{auth}}$ of Equations 4.3 and 4.4 calculated on the two point clouds already shown on Figure 4.1. At low scale, these measures show a lot of variation since a very small neighborhood is used. On the contrary, the ellipses calculated at high scale are likely to be more similar, so measured distortions are less important.

These experiments show that the scale-space parametrization is not authalic nor conformal. Optimizing the resulting parametrization with respect to one of these properties is left as future work. The covariance matrices of Equations 4.1 and 4.2 could help to develop optimization procedures to modify the 3D flattened point cloud so that one of the distortion measures of Equations 4.3 and 4.4 is minimized.

4.5 Conclusion

We propose a meshless parametrization algorithm that flatten a genus-0 point-sampled surface with disk or sphere topology on a 2D domain. Each point is iteratively projected onto a local surface approximation while the neighborhood size increases until reaching the global point cloud size, which finally leads to a planar of spherical global parametrization. Our method automatically handle holes in the surface as well as several boundaries. We also introduce two new multi-scale conformal and authalic measures that quantify the distortion of the resulting map in therms of preservation of angles and areas respectively.

Future work Developing optimization algorithms to minimize our conformal or authalic distortions is a promising future work. Our current results can be used as initial solution to optimize afterward the parametrization by modifying the flattened point cloud in 2D. One challenge is either to choose one appropriate scale, or to consider all of them in the minimization process. Another possibility is to add an optimization step after each APSS projection in 3D on each tangent space, so that distortions are minimized at any iteration of our algorithm.

To achieve this, we consider using the two covariance matrices of Equation 4.1 and 4.2 that reflect the points distribution in the initial 3D point cloud and the 2D flattened one respectively. By representing them as ellipses, we know the transformation that must be applied to one ellipse to match the area or the anisotropy of the other. We could compute this transformation

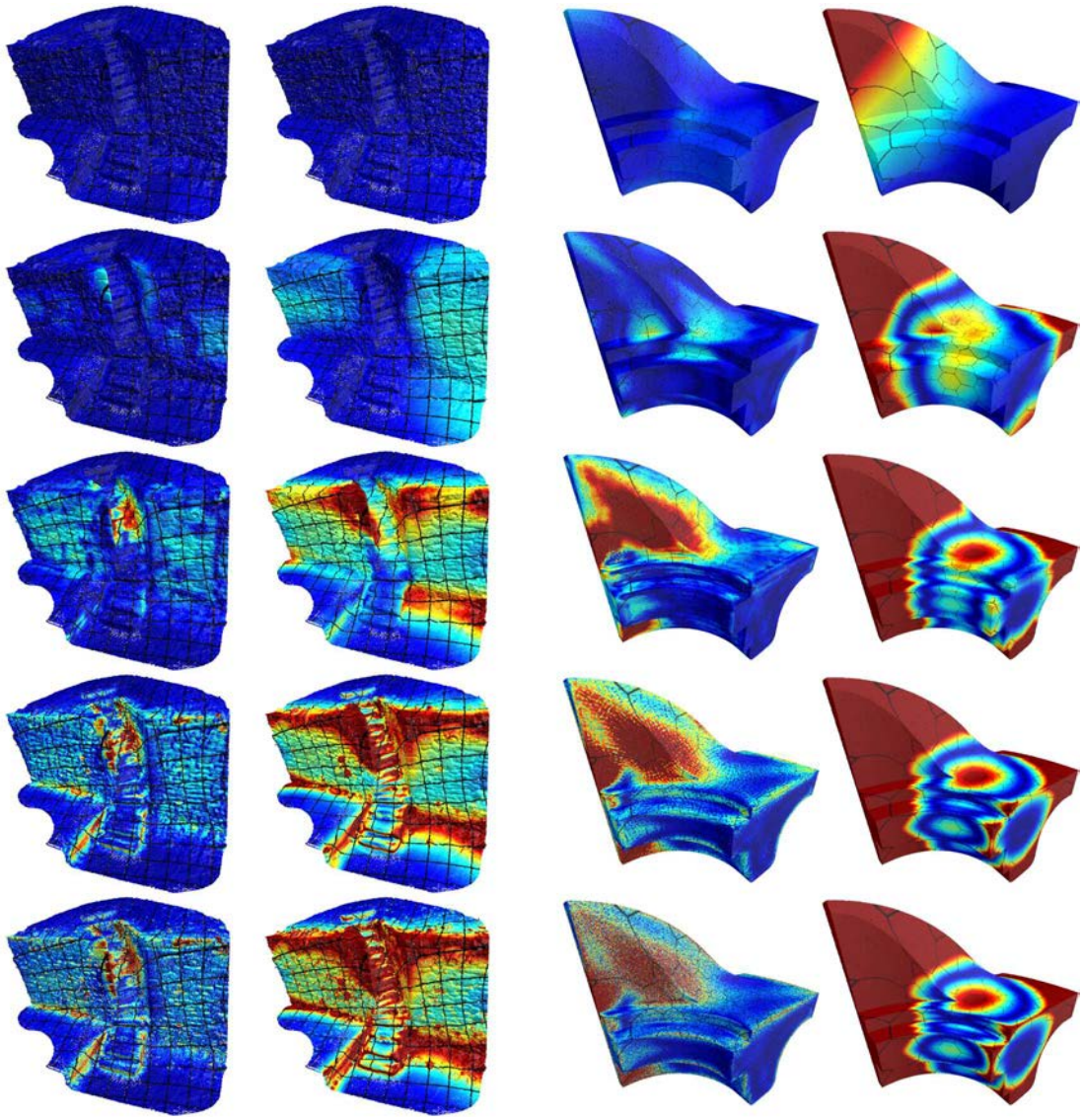


Figure 4.4: Meshless distortions. For the two data: distortions $\varepsilon_{\text{conf}}$ (left) and $\varepsilon_{\text{auth}}$ (right) of Equations 4.3 and 4.4. From bottom to top: low to high scale (50% of the bounding box diagonal length). Values range from 1 in blue (no distortion) to 2 in red.

as two scaling factors along the two eigenvectors at each 2D points and then apply it to its neighborhood. Points in the parametrization domain would move in directions to match the initial distribution leading to a more conformal parametrization in case ellipses anisotropy is considered. Our current experiments show that the distortion are not necessarily minimized and instabilities may occur, so a deeper analysis of this approach is required.

Parameterizing shapes of higher genus and preventing any folds in the parametrization are other challenging research directions. At least, detecting when the surface genus is not 0 or when a fold happens could be helpful. Many properties of the algebraic sphere fit such as its least squares residuals, the geometric variation [Mellado 2012], and its discriminant measuring anisotropy (see Section 1.3) may help to achieve this.

Another limitation we would like to address is the number of iterations that needs to be manually fixed. The scale could be updated locally depending to the shape itself and not according to the pre-sampled scale-space. For instance, if the algebraic sphere fit is reliable enough and the surface is quite flat, then the scale could grow more quickly than situations where the fitted sphere is far from the points and the surface is highly curved and noisy. It could lead to a suitable trade-off between a fold-free but costly parametrization, and a fast but self-overlapping one.

Finally, many 3D shape analysis tasks such as pattern recognition, segmentation, classification and meshing may be improved using a 2D parametrization. The benefit of our global parametrization directly obtained from the unstructured point cloud remains to be demonstrated, especially using more complex scanned data.

Conclusion

In this thesis we propose a multi-scale approach for geometric patterns detection in point clouds. Despite the lack of structure and the presence of many artefacts in the data, we robustly and efficiently estimate differential properties in order to characterize the sampled surface at any scale. We show the theoretical link between the local regression of an algebraic sphere and the surface derivatives including the mean curvature and a measure of shape anisotropy. We also compute accurate and robust principal curvatures from the algebraic sphere fit. A combined multi-resolution and multi-scale representation of the input point cloud can efficiently process several millions of points. This work leads to a fundamental tool that we show to be useful for pattern recognition and promising for meshless surface parametrization. We believe that it could also be valuable for many other geometry processing tasks analyzing large unstructured point clouds.

Point-wise differential measurements are pertinent but remain valid only for infinitesimal points in space and for one value of scale. It makes difficult the description of larger regions on the shape. We thus propose to structure the point cloud using these differential properties. A first solution gathers nearby points that share a similar differential property. To detect planes, a region growing process based on normal vectors generates these planar segmentations. A second solution focusing on cylinders and feature curves generates flow lines. It iteratively follows the principal curvature directions resulting in interesting discrete lines covering the sampled surface.

We generate these structures at many scales to be able to handle a wide range of feature sizes. With the multi-scale segmentations, we link similar regions from scale to scale leading to a hierarchical graph. The persistence analysis we propose to perform on this graph extracts the regions that persist the most. They correspond to meaningful planar regions of the point cloud. From the curvature lines, a voting approach performed at all scales robustly selects the points that most likely belong to the feature curves of the shape.

The lack of structure can also be addressed by directly finding a 2D parametrization of the 3D points. To flatten the input points on a 2D domain, we progressively increase the scale until all the points are globally projected onto a sphere or a plane. Our algorithm is able to map any genus-0 sampled surface, closed or open, without using any mesh topology. Our approach thus provides a continuous link between local and global calculations, from very small neighborhoods to the whole shape size. With more control on the mapping bijectivity and distortions, it could be leveraged for several shape analysis applications as regularity detection, abstraction and meshing by working in 2D instead of 3D.

In general, we find that differential geometry is a powerful tool to characterize 3D point clouds. Calculations are local which improves efficiency and scales up well to process a large amount of data. Links between concrete algorithms and the well understood continuous theory is an important element to better explain numerical behaviors in practice. We overcome the point-wise nature of differential geometry by adding spatial structures to the points such as discrete salient lines and segmentations. They link points in space sharing similar differential properties. In a second time, we link these structures in the scale-space by using a combi-

natorial graph and a vote-based approach. The persistence analysis finally offers a flexible framework to extract meaningful information from these underlying structures.

To conclude, we extend the toolbox available to efficiently and robustly analyze a large amount of 3D acquired data. Our algorithms directly run on point clouds avoiding the difficult step of mesh reconstruction. They do not require structured data which make them practical for a potentially wider type of shape representation as range images, meshes and voxel grids. They could thus increase the interoperability between heterogeneous data coming from different captors which is becoming more needed in practice. Our research in computer science increases the efficiency of 3D acquisition techniques, which shortens the geometry processing pipeline between the acquisition of a real element and its usages in a virtual environment. In a near future, scanning an object and re-printing it in 3D should become as easy as a copy-and-past operation in a text editor. Adding an existing object or place (or even ourselves) in our favorite video game will certainly be open to everyone, without the need of tedious manual editing nor costly automatic process. This thesis contributes to break the separation between our real environment and its digital counterpart, allowing us to better understand the complexity of the world around us.

Future work

Designing a unified multi-scale framework for geometric pattern recognition is a relevant perspective. A first challenge concerns the combination of our different algorithms to simultaneously detect planes, cylinders and feature curves on 3D shapes. Extensions to handle spheres, corners and other basic features could also be interesting to investigate. Another future work could be to describe and recognize at multiple scales more complex patterns as parts assemblies since many man-made shapes are made of these structured primitives. They pose a more complicated problem since they cannot be characterized by just a few point-wise differential quantities. Persistence analysis has shown to be effective and should be further used in the context of pattern recognition. Giving back structures to unorganized point clouds would enable easier processing of acquired 3D shapes for interactive editing, visualization and physical simulation for instance.

Deep learning is another promising research direction. Many neural networks have been recently introduced to process unstructured points [Qi 2017a, Qi 2017b, Boulch 2017, Li 2018, Thomas 2019] to name a few. Our multi-scale and multi-resolution representation proposed in Section 1.5 could be an appropriate tool for machine learning. Differential quantities as a function of the scale produce time series that are well adapted to traditional neural networks. We experimented this kind of approach to classify points belonging to sharp features in point clouds (This work has been submitted to a journal and is under review at the time of writing). For each point, the GLS parameters [Mellado 2012] as well as the principal curvatures computed from our accurate shape operator (Section 1.4.2) produce a one-dimensional multi-channel signal given as input to a simple neural network. This lightweight network trained on a small synthetic dataset quickly gives high quality results on real data as shown by Figure 4.5. Being able to quickly learn from a small set of examples is crucial when computational resources are limited. Our approach could be used for real-time applications, embedded in small

computer architectures, and using few manually annotated data. It contributes to make deep learning affordable as it considerably decreases the buying and maintenance price of hardware and the time required for labeling examples. We wonder if these multi-scale differential properties remain pertinent to describe semantic informations for object classification as they are for low level geometric features like sharp edges.

In addition to deep learning, the differential estimators we propose in Chapter 1 lead to fundamental geometric descriptors that are potentially useful for more applications other than geometric feature detection and parametrization. Indeed, we demonstrate that they are pertinent to describe the geometry of a surface as they asymptotically converge toward its mean curvature among other differential quantities. In practice, they are shown to be efficient, accurate, and stable with respect to noise, which may correspond to the most important qualities of an estimator. In theory, showing that they are provably stable makes an interesting perspective. Keeping the link between differential geometry theory and numerical algorithms is essential to offer strong guaranties and a clear understanding of what the descriptors represent. Examples of application where such multi-scale differential descriptors may be useful include shape matching, retrieval, registration, re-sampling, etc...

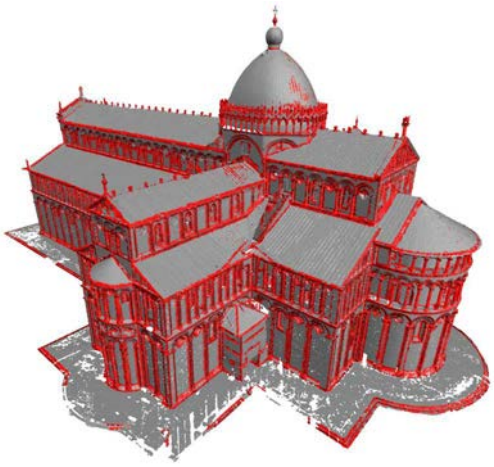


Figure 4.5: Sharp features classification.

In the long term, we wish to bring our analysis tools to higher dimensions. In data analysis, points do not sample the surface of a physical object, but still sample a low-dimensional manifold embedded in a higher dimensional space. The data of interest usually lie onto this manifold that is challenging to extract from the high dimensional point cloud. Manifold learning is a well established scientific domain that may benefit from our point cloud parametrization algorithm that remains valid in any dimension for manifolds of co-dimension 1. Traditional methods often involve costly eigendecomposition of large systems [Tenenbaum 2000, Roweis 2000, Belkin 2003] or expensive optimization algorithms [Zhang 2004, Maaten 2008]. During these last decades, many computational geometry tools have been used in this context in order to analyze the data geometry instead of geometric data [Boissonnat 2017]. Following this transition, our approach could be an efficient and robust alternative with a useful geometric point of view that enables to process massive data. It could open our multi-scale differential methods developed for 3D shapes to even wider scientific fields as medicine and dynamical systems analysis.

List of publications

This thesis led to the following publications.

International journal

T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Persistence Analysis of Multi-scale Planar Structure Graph in Point Clouds.* Computer Graphics Forum, 2020.

J. Shao, W. Zhang, N. Mellado, N. Wang, S. Jin, S. Cai, L. Luo, T. Lejemble and G. Yan *SLAM-aided Forest Plot Mapping Combining Terrestrial and Mobile Laser Scanning.* ISPRS Journal of Photogrammetry and Remote Sensing, 2020.

International workshop contest

E. Moscoso Thompson, G. Arvanitis, K. Moustakas, N. Hoang-Xuan, E. R. Nguyen, M. Tran, T. Lejemble, L. Barthe, N. Mellado, C. Romanengo, S. Biasotti, and B. Falcidieno. *SHREC'19 track: Feature Curve Extraction on Triangle Meshes.* Eurographics Workshop on 3D Object Retrieval, 2019.

S. Biasotti, E. Moscoso Thompson, L. Barthe, S. Berretti, A. Giachetti, T. Lejemble, N. Mellado, K. Moustakas, Iason Manolas, Dimitrios Dimou, C. Tortorici, S. Velasco-Forero, N. Werghi, M. Polig, G. Sorrentino, and S. Hermon. *SHREC'18 track: Recognition of Geometric Patterns over 3D models.* Eurographics Workshop on 3D Object Retrieval, 2018.

International conference poster

T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Multi-Scale Point Cloud Analysis.* Eurographics posters, 2019.

National conference

T. Lejemble, C. Mura, L. Barthe and N. Mellado. *Multi-scale Planar Segments Extraction from Point Clouds.* Journées Françaises d'Informatique Graphique, 2018.

Appendix A

Asymptotic analysis

Contents

This appendix contains all the detailed calculations related to the asymptotic analysis of the algebraic sphere fitting of Section 1.3. Section A.1 details the Taylor polynomials of fundamental quantities such as coordinates and normals that are required for the calculations of the algebraic sphere fit (Equation 1.7-1.9). In Section A.2, we perform the integration of these quantities. Finally, Section A.3 and A.4 compute the asymptotic expansions of the parameters u_c, $\mathbf{u}_\ell$ and u_q, and the projection operator φ, leading to the results of Theorems 1 and 2 respectively.

A.1 Differential quantities

We first give the Taylor polynomials of the coordinates $\mathbf{f}$, the normals $\mathbf{n}$ and their dot product in the local principal frame. Using polar coordinates $(r, \theta) \in (0, t) \times (0, 2\pi)$, these quantities are given in the form of a polynomial of variable r with coefficients depending on variable θ. The coefficients also contain the different derivatives $a_{k,j-k}$ of the surface height defined by Equation 1.15.

Coordinates The surface of Equation 1.14 is expressed in polar coordinates by

$$\mathbf{f}(r, \theta) = \begin{bmatrix} r\cos(\theta) & r\sin(\theta) & z(r, \theta) \end{bmatrix}^T . \tag{A.1}$$

The Taylor expansion of the height field function z in Equation 1.15 is written in polar coordinates as $z(r, \theta) = \sum_{k=2}^{5} r^k b_k(\theta) + o(r^5)$ with the coefficients b_k equal to

$$\begin{aligned}
b_2(\theta) &= \frac{1}{2}\left(\kappa_1 \cos^2(\theta) + \kappa_2 \sin^2(\theta)\right), \\
b_3(\theta) &= \frac{1}{6}\left(a_{30}\cos^3(\theta) + 3a_{21}\cos^2(\theta)\sin(\theta) + 3a_{12}\cos(\theta)\sin^2(\theta) + a_{03}\sin^3(\theta)\right), \\
b_4(\theta) &= \frac{1}{24}\left(a_{40}\cos^4(\theta) + 4a_{31}\cos^3(\theta)\sin(\theta) + 2a_{22}\cos^2(\theta)\sin^2(\theta) + \right.
\end{aligned}$$

$$4a_{13}\cos(\theta)\sin^3(\theta) + a_{04}\sin^4(\theta)\big),$$

$$b_5(\theta) = \frac{1}{120}\big(a_{50}\cos^5(\theta) + 5a_{41}\cos^4(\theta)\sin(\theta) + 10a_{32}\cos^3(\theta)\sin^2(\theta) +$$
$$10a_{23}\cos^2(\theta)\sin^3(\theta) + 5a_{14}\cos(\theta)\sin^4(\theta) + a_{05}\sin^5(\theta)\big).$$

Note that although the order 4 is sufficient to obtain the results of Theorems 1 and 2, one order higher is considered in this appendix. It gives some insight on the higher order terms of every Taylor expansion, which could be certainly useful for future work. The squared height which is required latter is $z(r,\theta)^2 = \sum_{k=4}^{7} c_k(\theta)r^k + o(r^7)$ with coefficients $c_4(\theta) = b_2(\theta)^2$, $c_5(\theta) = 2b_2(\theta)b_3(\theta)$, $c_6(\theta) = b_3(\theta)^2 + 2b_2(\theta)b_4(\theta)$, and $c_7(\theta) = 2b_2(\theta)b_5(\theta) + 2b_3(\theta)b_4(\theta)$.

Tangents Before introducing the normals, the Taylor polynomials of the tangents are required. In the principal frame, they are the partial derivatives of $\mathbf{f}$ with respect to x and y that are given by $\partial_x\mathbf{f}(x,y) = \begin{bmatrix}1 & 0 & \partial_x z(x,y)\end{bmatrix}^T$ and $\partial_y\mathbf{f}(x,y) = \begin{bmatrix}0 & 1 & \partial_y z(x,y)\end{bmatrix}^T$. In polar coordinates, the partial derivatives of z with respect to x and y are $\partial_x z(r,\theta) = \sum_{k=1}^{4} d_{xk}(\theta)r^k + o(r^4)$ and $\partial_y z(r,\theta) = \sum_{k=1}^{4} d_{yk}(\theta)r^k + o(r^4)$ with the following coefficients

$$d_{x1}(\theta) = \kappa_1\cos^2(\theta),$$
$$d_{x2}(\theta) = \frac{1}{2}\big(a_{30}\cos^2(\theta) + 2a_{21}\cos(\theta)\sin(\theta) + a_{12}\sin^2(\theta)\big),$$
$$d_{x3}(\theta) = \frac{1}{6}\big(a_{40}\cos^3(\theta) + 3a_{31}\cos^2(\theta)\sin(\theta) + 3a_{22}\cos(\theta)\sin^2(\theta) + a_{13}\sin^3(\theta)\big),$$
$$d_{x4}(\theta) = \frac{1}{24}\big(a_{50}\cos^4(\theta) + 4a_{41}\cos^3(\theta)\sin(\theta) + 6a_{32}\cos^2(\theta)\sin^2(\theta) +$$
$$4a_{23}\cos(\theta)\sin^3(\theta) + a_{14}\sin^4(\theta)\big).$$

$$d_{y_1}(\theta) = \kappa_2\sin^2(\theta),$$
$$d_{y_2}(\theta) = \frac{1}{2}\big(a_{21}\cos^2(\theta) + 2a_{12}\cos(\theta)\sin(\theta) + a_{03}\sin^2(\theta)\big),$$
$$d_{y_3}(\theta) = \frac{1}{6}\big(a_{31}\cos^3(\theta) + 3a_{22}\cos^2(\theta)\sin(\theta) + 3a_{13}\cos(\theta)\sin^2(\theta) + a_{04}\sin^3(\theta)\big),$$
$$d_{y_4}(\theta) = \frac{1}{24}\big(a_{41}\cos^4(\theta) + 4a_{32}\cos^3(\theta)\sin(\theta) + 6a_{23}\cos^2(\theta)\sin^2(\theta) +$$
$$4a_{14}\cos(\theta)\sin^3(\theta) + a_{05}\sin^4(\theta)\big).$$

The squared partial derivative of z with respect to x in polar coordinates is $\partial_x z(r,\theta)^2 = \sum_{k=2}^{5} e_{xk}(\theta)r^k + o(r^5)$ with coefficients $e_{x2}(\theta) = d_{x1}(\theta)^2$, $e_{x3}(\theta) = 2d_{x1}(\theta)d_{x2}(\theta)$, $e_{x4}(\theta) = d_{x2}(\theta)^2 + 2d_{x1}(\theta)d_{x3}(\theta)$, and $e_{x5}(\theta) = 2d_{x1}(\theta)d_{x4}(\theta) + 2d_{x2}(\theta)d_{x3}(\theta)$. The formula for $\partial_y z$ and its associated coefficients e_{y_k} are the same as $\partial_x z$ and e_{xk} using y subscript instead of x.

Normals We denote by $\mathbf{v}$ a vector orthogonal to the surface $\mathbf{v}(x,y) = \partial_x \mathbf{f}(x,y) \times \partial_y \mathbf{f}(x,y)$, which is equal to $\begin{bmatrix} -\partial_x z(x,y) & -\partial_y z(x,y) & 1 \end{bmatrix}^T$, so that the normal vector $\mathbf{n}$ is given by $\mathbf{n}(x,y) = \frac{\mathbf{v}(x,y)}{\|\mathbf{v}(x,y)\|}$. The squared norm of $\mathbf{v}$ is $\|\mathbf{v}(r,\theta)\|^2 = 1 + \sum_{k=2}^{5} f_k(\theta) r^k + o(r^5)$, with $f_k(\theta) = e_{x_k}(\theta) + e_{y_k}(\theta)$. Using the Taylor expansion of $1/\sqrt{1+X}$, the inverse of the norm is approximated by $1/\|\mathbf{v}(r,\theta)\| = 1 + \sum_{k=2}^{5} g_k(\theta) r^k + o(r^5)$, with $g_2(\theta) = -\frac{1}{2} f_2(\theta)$, $g_3(\theta) = -\frac{1}{2} f_3(\theta)$, $g_4(\theta) = \frac{1}{8}\left(3 f_2(\theta)^2 - 4 f_4(\theta)\right)$, and $g_5(\theta) = \frac{1}{4}\left(3 f_2(\theta) f_3(\theta) - 2 f_5(\theta)\right)$. Finally the normal $\mathbf{n}$ is asymptotically equivalent to

$$\mathbf{n}(r,\theta) = \begin{bmatrix} \mathbf{n}_x(r,\theta) \\ \mathbf{n}_y(r,\theta) \\ \mathbf{n}_z(r,\theta) \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{4} h_{x_k}(\theta) r^k + o(r^4) \\ \sum_{k=1}^{4} h_{y_k}(\theta) r^k + o(r^4) \\ 1 + \sum_{k=2}^{5} g_k(\theta) r^k + o(r^5) \end{bmatrix}, \tag{A.2}$$

with $h_{x1}(\theta) = -d_{x2}(\theta)$, $h_{x2}(\theta) = -d_{x3}(\theta)$, $h_{x3}(\theta) = -d_{x4}(\theta) - g_2(\theta) d_{x2}(\theta)$, $h_{x4}(\theta) = -d_{x5}(\theta) - g_2(\theta) d_{x3}(\theta) - g_3(\theta) d_{x2}(\theta)$, and using similar formula for $h_{y_k}(\theta)$ using y subscript.

Dot products The asymptotic dot product between the coordinates and the normals is $\mathbf{f}(r,\theta) \,.\, \mathbf{n}(r,\theta) = \sum_{k=2}^{5} m_k(\theta) r^k + o(r^5)$ with the following coefficients

$$\begin{aligned}
m_2(\theta) &= \cos(\theta) h_{x1}(\theta) + \sin(\theta) h_{y_1}(\theta) + b_2(\theta), \\
m_3(\theta) &= \cos(\theta) h_{x2}(\theta) + \sin(\theta) h_{y_2}(\theta) + b_3(\theta), \\
m_4(\theta) &= \cos(\theta) h_{x3}(\theta) + \sin(\theta) h_{y_3}(\theta) + b_4(\theta) + g_2(\theta) b_2(\theta), \\
m_5(\theta) &= \cos(\theta) h_{x4}(\theta) + \sin(\theta) h_{y_4}(\theta) + b_5(\theta) + g_2(\theta) b_3(\theta) + g_3(\theta) b_2(\theta).
\end{aligned}$$

The dot product of the coordinates with themselves, which is the squared norm of the positions, is $\|\mathbf{f}(r,\theta)\|^2 = \mathbf{f}(r,\theta) \,.\, \mathbf{f}(r,\theta) = r^2 + z(r,\theta)^2 = r^2 + \sum_{k=4}^{7} c_k(\theta) r^k + o(r^7)$.

A.2 Integration

We now give results of the integration of the previous quantities over the surface patch $\mathcal{P}_t$ (Equation 1.16). In order to calculate these integrals, the integration domain considered is the cylindrical neighborhood instead

$$\mathcal{C}_t = \left\{ (x,y) \in \mathbb{R}^2,\ \|x^2 + y^2\| < t^2 \right\}, \tag{A.3}$$

that gives equivalent results in these asymptotic settings [Digne 2011, Lemma 1]. These calculations are fairly straightforward since they only involve polynomials integrations. Moreover, many integrals containing coefficients of the form $\cos^p(\theta) \sin^q(\theta)$ are discarded if p or q are odd. On the other hand, the coefficients are often tedious to write so we only give the results.

Coordinates The integration over $\mathcal{C}_t$ of the coordinates $\mathbf{f}$ of Equation A.1 results in $\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta = \begin{bmatrix} 0 & 0 & n_4 t^4 + n_6 t^6 + o(t^7) \end{bmatrix}^T$, with $n_4 = \frac{\pi H}{4}$ and $n_6 = \frac{\pi \Delta H}{96}$. The coefficient n_4 agrees with prior work on integral invariants [Pottmann 2007, Theorem 6].

Normals The integration over $\mathcal{C}_t$ of the normals $\mathbf{n}$ of Equation A.2 yields

$$\iint_{\mathcal{C}_t} \mathbf{n}(r,\theta)\, r\, dr d\theta = \begin{bmatrix} p_{x4}t^4 + p_{x6}t^6 + o(t^6) \\ p_{y4}t^4 + p_{y6}t^6 + o(t^6) \\ p_{z2}t^2 + p_{z4}t^4 + p_{z6}t^6 + o(t^6) \end{bmatrix}$$

with the following coefficients

$$\begin{aligned}
p_{x4} &= -\frac{\pi}{8}(a_{30} + a_{12}), \\
p_{y4} &= -\frac{\pi}{8}(a_{03} + a_{21}), \\
p_{x6} &= \frac{\pi}{48}(a_{30}(2H^2 - K + 4\kappa_1^2) + a_{12}(6H^2 - K) - (a_{50} + 2a_{32} + a_{14})/4), \\
p_{y6} &= \frac{\pi}{48}(a_{03}(2H^2 - K + 4\kappa_2^2) + a_{21}(6H^2 - K) - (a_{41} + 2a_{23} + a_{05})/4), \\
p_{z2} &= \pi, \\
p_{z4} &= -\frac{\pi}{8}(\kappa_1^2 + \kappa_2^2), \\
p_{z6} &= \frac{\pi}{192}(144H^2(H^2 - K) + 24K^2 - 4(a_{22} + a_{40})\kappa_1 - 4(a_{22} + a_{04})\kappa_2 \\
&\qquad - 3(a_{30}^2 + a_{03}^2) - 2(a_{12}a_{30} + a_{03}a_{21}) - 7(a_{21}^2 + a_{12}^2)).
\end{aligned}$$

Dot products The last quantities to integrate are the two dot products introduced in the previous section. Their integrals are $\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\,.\,\mathbf{n}(r,\theta)\, r\, dr d\theta = q_4 t^4 + q_6 t^6 + o(t^7)$ and $\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\,.\,\mathbf{f}(r,\theta)\, r\, dr d\theta = r_4 t^4 + r_6 t^6 + r_8 t^8 + o(t^9)$, with the coefficients equal to

$$\begin{aligned}
q_4 &= -\frac{\pi H}{4}, \\
q_6 &= \frac{\pi}{96}(24H^3 - 16KH - \Delta H) \\
r_4 &= \frac{\pi}{2}, \\
r_6 &= \frac{\pi}{24}(3H^2 - K), \\
r_8 &= \frac{\pi}{4068}(3(5a_{40} + 6a_{22} + a_{04})\kappa_1 + 3(a_{40} + 6a_{22} + 5a_{04})\kappa_2 \\
&\qquad + 2(5a_{30}^2 + 9a_{21}^2 + 6a_{30}a_{12} + 6a_{03}a_{21} + 9a_{12}^2 + 5a_{03}^2)).
\end{aligned}$$

A.3 Algebraic sphere fitting (proof of Theorem 1)

The goal of this section is to gather together the previous integrals following the smooth version of Equations 1.7-1.9 to obtain the asymptotic equivalents of u_c, $\mathbf{u}_\ell$ and u_q of the fitted algebraic sphere. These final results are summarized in Theorem 1.

Dot products Before calculating the parameters of the sphere, two intermediate elements need to be developed. The dot product of the coordinates and normals integrals

is $\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta\,.\,\iint_{\mathcal{C}_t} \mathbf{n}(r,\theta)\ r\ dr d\theta = s_6 t^6 + s_8 t^8 + o(t^9)$ with $s_6 = \frac{\pi^2 H}{4}$ and $s_8 = \frac{\pi^2}{96}(-12H^3 + 6KH + \Delta H)$. The second dot product that is applied to the coordinates integral with itself is $\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta\,.\,\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta = u_8 t^8 + u_{10} t^{10} + o(t^{11})$ with $u_8 = \frac{\pi^2 H^2}{16}$ and $u_{10} = \frac{\pi H \Delta H}{192}$.

Quadratic parameter To calculate u_q using Equation 1.9, we rewrite it as a fraction $u_q = \frac{1}{2}\frac{n}{d}$ with n the numerator and d the denominator of u_q (up to the constant $1/2$). In the continuous setting, the numerator of u_q is expressed by

$$n = A_t \iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\,.\,\mathbf{n}(r,\theta)\ r\ dr d\theta - \iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta\,.\,\iint_{\mathcal{C}_t} \mathbf{n}(r,\theta)\ r\ dr d\theta,$$

where $A_t = \pi t^2$ is the area of $\mathcal{C}_t$. Its asymptotic polynomials is $n = v_6 t^6 + v_8 t^8 + o(t^9)$ with $v_6 = -\frac{\pi^2 H}{2}$ and $v_8 = \frac{\pi^2}{48}(18H^3 - 11KH - 2\Delta H)$. The denominator of u_q is

$$d = A_t \iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\,.\,\mathbf{f}(r,\theta)\ r\ dr d\theta - \iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta\,.\,\iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\ r\ dr d\theta,$$

which asymptotically leads to $d = \frac{\pi^2}{2} t^6 \left(1 + w_2 t^2 + w_4 t^4 + o(t^5)\right)$, with the coefficients

$$\begin{aligned} w_2 &= \frac{1}{24}(3H^2 - 2K), \\ w_4 &= 3(5a_{40} + 6a_{22} + a_{04})\kappa_1 + 3(a_{40} + 6a_{22} + 5a_{04})\kappa_2 \\ &\quad + 2(5a_{30}^2 + 9a_{21}^2 + 6a_{30}a_{12} + 6a_{21}a_{03} + 9a_{12}^2 + 5a_{03}^2). \end{aligned}$$

The inverse of the denominator obtained using the Taylor expansion of $1/(1+X)$ is $\frac{1}{d} = \frac{2}{\pi^2 t^6}(1 - w_2 t^2 + o(t^3))$. Finally, the quadratic parameter u_q of the algebraic sphere is asymptotically expressed as

$$u_q = \frac{1}{2}\frac{n}{d} = u_{q0} + u_{q2} t^2 + o(t^3) \tag{A.4}$$

with the following coefficients

$$\begin{aligned} u_{q0} &= -\frac{H}{2}, \\ u_{q2} &= \frac{1}{48}(21H^3 - 13HK - 2\Delta H). \end{aligned}$$

This polynomial truncated at order 1 gives the result shown by Equation 1.17 in Theorem 1.

Linear parameter The linear parameter $\mathbf{u}_\ell$ of the sphere is

$$\mathbf{u}_\ell = \frac{1}{A_t}\left(\iint_{\mathcal{C}_t} \mathbf{n}(r,\theta) - 2u_q \iint_{\mathcal{C}_t} \mathbf{f}(r,\theta)\right) = \begin{bmatrix} \mathbf{u}_{\ell x 2} t^2 + \mathbf{u}_{\ell x 4} t^4 + o(t^4) \\ \mathbf{u}_{\ell y 2} t^2 + \mathbf{u}_{\ell y 4} t^4 + o(t^4) \\ 1 + \mathbf{u}_{\ell z 2} t^2 + \mathbf{u}_{\ell z 4} t^4 + o(t^5) \end{bmatrix}, \tag{A.5}$$

with

$$
\begin{aligned}
\mathbf{u}_{\ell x 2} &= -\frac{a_{30}+a_{12}}{8},\\
\mathbf{u}_{\ell y_2} &= -\frac{a_{03}+a_{21}}{8},\\
\mathbf{u}_{\ell x 4} &= \frac{1}{48}\left(2(a_{30}+3a_{12})H^2-(a_{12}+a_{30})K+4a_{30}\kappa_1^2-(a_{50}+2a_{32}+a_{14})/4\right),\\
\mathbf{u}_{\ell y_4} &= \frac{1}{48}\left(2(a_{03}+3a_{21})H^2-(a_{03}+a_{21})K+4a_{03}\kappa_2^2-(a_{41}+2a_{23}+a_{05})/4\right),\\
\mathbf{u}_{\ell z 2} &= -\frac{H^2-K}{4},\\
\mathbf{u}_{\ell z 4} &= \frac{1}{192}(165H^4-157KH^2+24K^2-4(a_{40}+a_{22})\kappa_1-4(a_{04}+a_{22})\kappa_2\\
&\qquad -3a_{30}^2-2a_{12}a_{30}-7a_{21}^2-2a_{03}a_{21}-7a_{12}^2-3a_{03}^2),
\end{aligned}
$$

which gives the polynomial truncated at order 3 of Equation 1.18 in Theorem 1.

Constant parameter The constant parameter u_c is expressed in the smooth setting by

$$
u_c = -\frac{1}{A_t}\left(\mathbf{u}_\ell \,.\iint_{\mathcal{C}_t}\mathbf{f}(r,\theta)\,r\,dr d\theta + u_q\iint_{\mathcal{C}_t}\mathbf{f}(r,\theta)\,.\,\mathbf{f}(r,\theta)\,r\,dr d\theta\right) \tag{A.6}
$$

Its asymptotic expansion is simply $u_c = u_{c4}t^4 + o(t^4)$ with $u_{c4} = -\frac{1}{96}(9H^3 - 5KH - \Delta H)$ which is the resulting Equation 1.19 of Theorem 1.

A.4 Algebraic sphere projection (proof of Theorem 2)

Once the asymptotic versions of u_c, $\mathbf{u}_\ell$ and u_q are obtained, we determine now the Taylor polynomials of the projection operator φ of Equation 1.10. Since the point $\mathbf{p}$ to be projected is located at the origin of the principal frame, the projection is simplified to

$$
\varphi(\mathbf{p}) = \begin{cases} -\frac{u_c}{\|\mathbf{u}_\ell\|^2}\mathbf{u}_\ell & \text{if } u_q = 0\\ -\frac{\|\mathbf{u}_\ell\|-\sqrt{\Delta}}{2u_q\|\mathbf{u}_\ell\|}\mathbf{u}_\ell & \text{otherwise}\end{cases} \tag{A.7}
$$

In the following we denote by $\bar{\varphi}$ the planar projection (if $u_q = 0$) and by $\tilde{\varphi}$ the spherical one.

Norm of $\mathbf{u}_\ell$ The squared norm of $\mathbf{u}_\ell$ (Equation A.5) is expanded as $\|\mathbf{u}_\ell\|^2 = 1 + \alpha_2 t^2 + \alpha_4 t^4 + o(t^5)$ with

$$
\begin{aligned}
\alpha_2 &= -\frac{H^2-K}{2},\\
\alpha_4 &= \frac{1}{192}\left(2(171H^4-169H^2K+30K^2)-8(a_{40}\kappa_1+a_{04}\kappa_2+2a_{22}H)\right.\\
&\qquad \left.-3(a_{30}^2+a_{03}^2)-11(a_{21}^2+a_{12}^2)+2(a_{21}a_{03}+a_{12}a_{30})\right),
\end{aligned} \tag{A.8}
$$

hence the Equation 1.20. The norm, inverse norm and inverse squared norm of $\mathbf{u}_\ell$ required for the projection are obtained using Taylor expansion of $\sqrt{1+X}$, $1/\sqrt{1+X}$ and $1/(1+X)$ respectively

$$\begin{aligned}
\|\mathbf{u}_\ell\| &= 1 + \frac{\alpha_2}{2}t^2 + \frac{1}{8}(4\alpha_4 - \alpha_2^2)t^4 + o(t^5),\\
\frac{1}{\|\mathbf{u}_\ell\|} &= 1 - \frac{\alpha_2}{2}t^2 + \frac{1}{8}(3\alpha_2^2 - 4\alpha_4)t^4 + o(t^5),\\
\frac{1}{\|\mathbf{u}_\ell\|^2} &= 1 - \alpha_2 t^2 + (\alpha_2^2 - \alpha_4)t^4 + o(t^5).
\end{aligned}$$

Planar projection In this paragraph, the parameter u_q is assumed to be null, so the projection operator we analyze is the planar projection $\bar{\varphi}$. We first compute the intermediate value of the ratio of u_c and $\|\mathbf{u}_\ell\|^2$. It is approximated by $u_c/\|\mathbf{u}_\ell\|^2 = u_{c4}t^4 + o(t^5)$ where u_{c4} is the coefficient of u_c (Equation A.6). Finally the expression of the planar projection is

$$\bar{\varphi} = \begin{bmatrix} \bar{\varphi}_{x6}t^6 + o(t^7) \\ \bar{\varphi}_{y6}t^6 + o(t^7) \\ \bar{\varphi}_{z4}t^4 + o(t^5) \end{bmatrix}, \tag{A.9}$$

with the following coefficients

$$\begin{aligned}
\bar{\varphi}_{x6} &= -\frac{1}{768}(a_{30} + a_{12})(9H^3 - 5KH - \Delta H),\\
\bar{\varphi}_{y6} &= -\frac{1}{768}(a_{03} + a_{21})(9H^3 - 5KH - \Delta H),\\
\bar{\varphi}_{z4} &= \frac{1}{96}(9H^3 - 5KH - \Delta H).
\end{aligned}$$

Since u_q evaluates to zero, all its coefficients in Equation A.4 are null, which implies $H = 0$. Therefore, the truncation of $\bar{\varphi}$ to the order 5 gives the Equation 1.21 in Theorem 2.

Spherical projection We now analyze the projection when $u_q \neq 0$ that corresponds to the projection on a sphere denoted $\tilde{\varphi}$. We first calculate the asymptotic expansion of several intermediate variables before the resulting Equation A.10. The algebraic sphere discriminant Δ defined by $\|\mathbf{u}_\ell\|^2 - 4u_q u_c$ (Equation 1.11) and its square root are

$$\begin{aligned}
\Delta &= 1 + \beta_2 t^2 + \beta_4 t^4 + o(t^5),\\
\sqrt{\Delta} &= 1 + \gamma_2 t^2 + \gamma_4 t^4 + o(t^5).
\end{aligned}$$

The last equation is obtained using the Taylor expansion of $\sqrt{1+X}$. The associated coefficients are $\beta_2 = \mathbf{u}_{\ell z2}$, $\beta_4 = \alpha_4 - 4\, u_{q0}\, u_{c4}$, $\gamma_2 = \beta_2/2$, and $\gamma_4 = (4\beta_4 - \beta_2^2)/8$. They include $\mathbf{u}_{\ell z2}$ (Equation A.5), α_4 (Equation A.8), u_{q0} (Equation A.4), and u_{c0} (Equation A.6). The product of u_q and the norm of $\mathbf{u}_\ell$ as well as its inverse are given by $u_q\|\mathbf{u}_\ell\| = \delta_0 + \delta_2 t^2 + o(t^3)$ and $\frac{1}{u_q\|\mathbf{u}_\ell\|} = \eta_0 + \eta_2 t^2 + o(t^3)$ which is obtained using the Taylor expansion of $1/(1+X)$. Their coefficients are $\delta_0 = u_{q0}$, $\delta_2 = u_{q2} + u_{q0}\, \mathbf{u}_{\ell z2}$, $\eta_0 = 1/\delta_0$, and $\eta_2 = -\delta_2/\delta_0^2$.

Finally, the previous results can be gathered to calculate the spherical projection (in case $u_q \neq 0$), which results in

$$\tilde{\varphi} = \begin{bmatrix} \tilde{\varphi}_{x6} t^6 + o(t^7) \\ \tilde{\varphi}_{y6} t^6 + o(t^7) \\ \tilde{\varphi}_{y4} t^4 + o(t^5) \end{bmatrix}, \tag{A.10}$$

with the coefficient $\tilde{\varphi}_{z4} = \frac{1}{96}(9H^3 - 5KH - \Delta H)$. To conclude, the truncation of Equation A.10 to the order 5 gives the Equation 1.22 in Theorem 2.

Appendix B

Supplemental results of Chapter 2

Contents

This appendix provides more results on the different interactive tools we propose in Section 2.4.

B.1 Persistence exploration

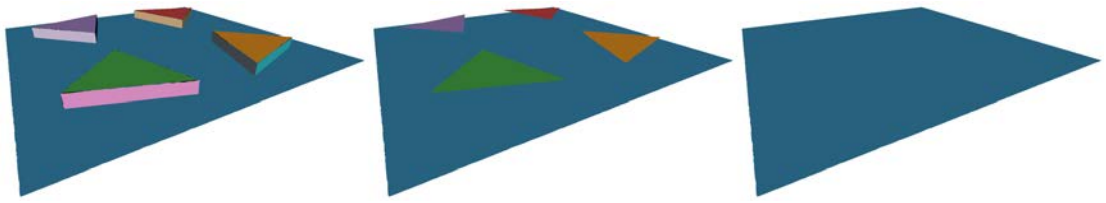


Figure B.1: Tri. Persistence thresholds : 20, 25, 30

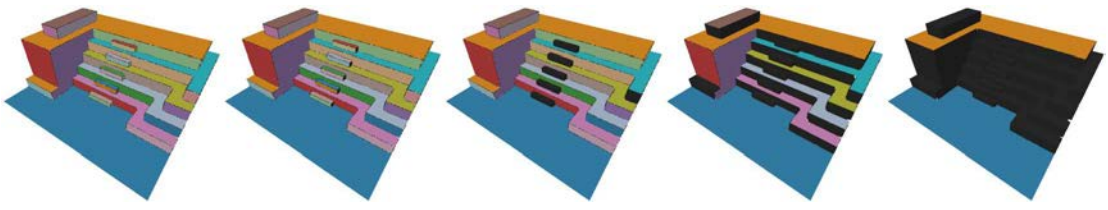


Figure B.2: Stairs. Persistence thresholds : 10, 15, 20, 25, 30

Figure B.3: Lans. Persistence thresholds : 5, 10, 15, 20, 25

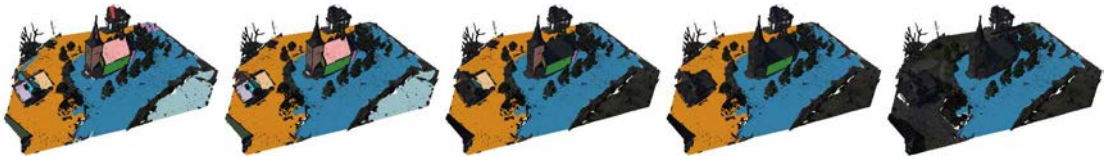


Figure B.4: Church. Persistence thresholds : 5, 10, 15, 20, 25

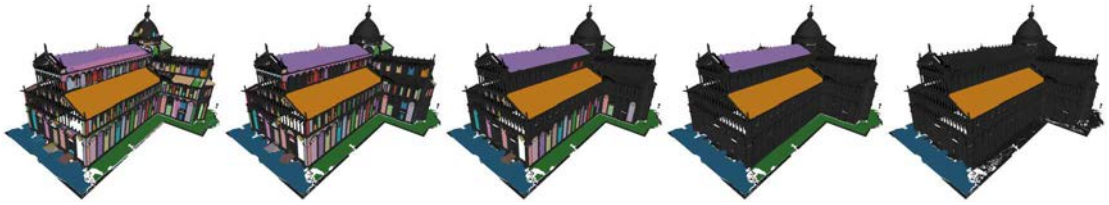


Figure B.5: Pisa. Persistence thresholds : 5, 10, 15, 20, 25

Figure B.6: Euler. Persistence thresholds : 10, 15, 20, 25, 30, 35

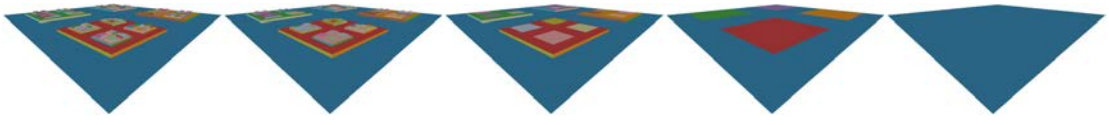


Figure B.7: Cubes. Persistence thresholds : 10, 15, 20, 25, 30

B.2 Scale-Space exploration

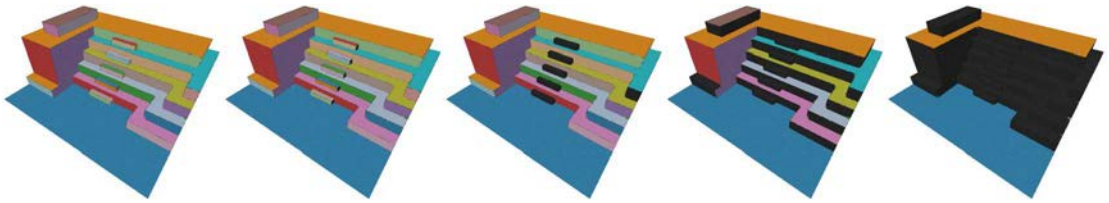


Figure B.8: Stairs. Scale threshold : 0, 15, 20, 25, 30

Figure B.9: Lans. Scale threshold : 5, 10, 15, 20, 25

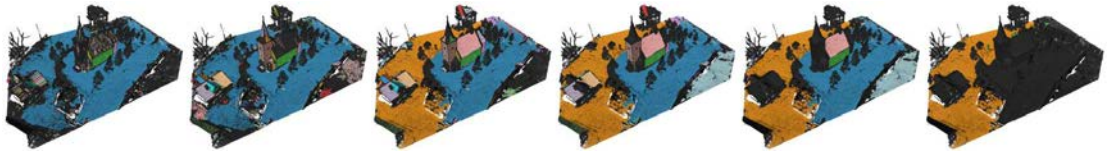


Figure B.10: Church. Scale threshold : 5, 10, 15, 20, 25, 30

Figure B.11: Pisa. Scale threshold : 5, 10, 15, 20, 25

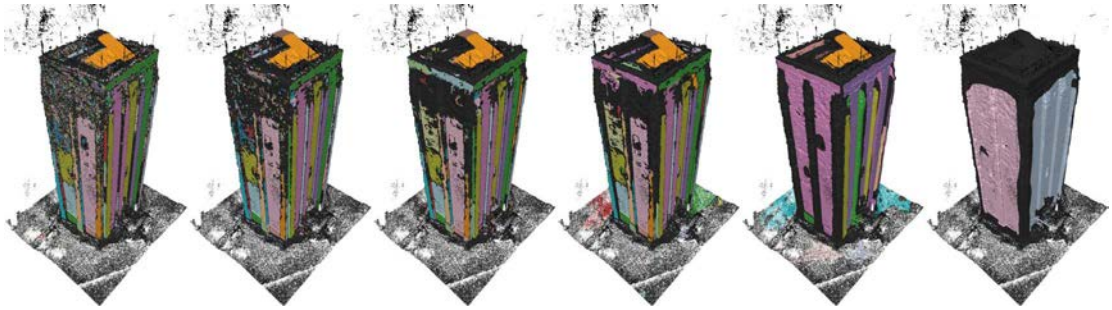


Figure B.12: Loudun. Scale threshold : 5, 10, 15, 20, 25, 30

B.3 Brush Reconstruction

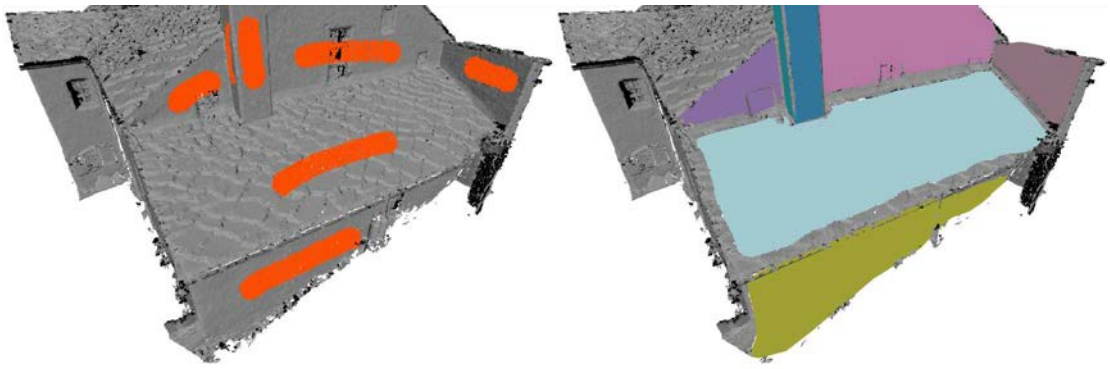


Figure B.13: Lans. Brush reconstruction

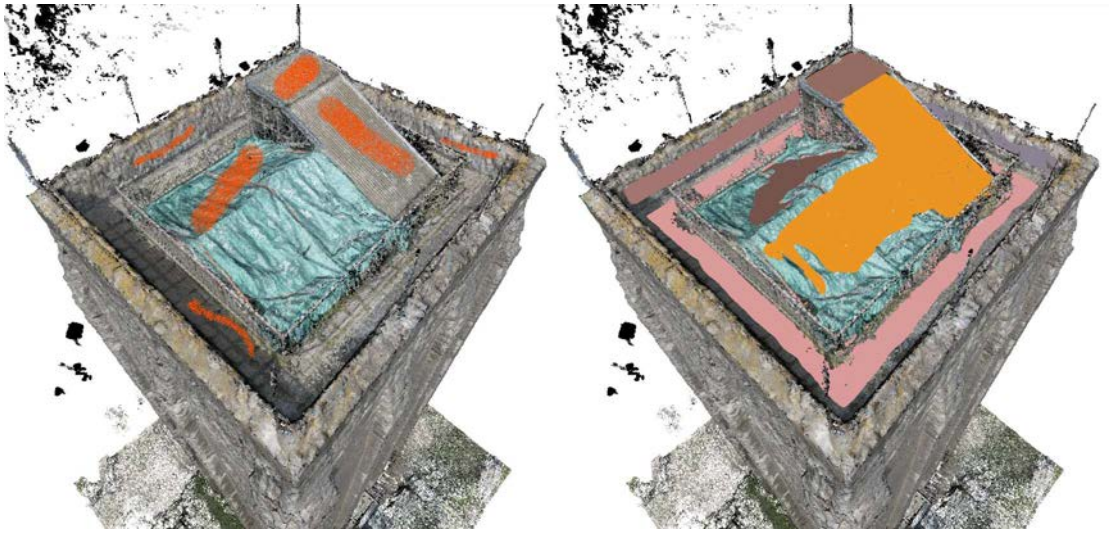


Figure B.14: Loudun. Brush reconstruction

B.4 Similarity Search

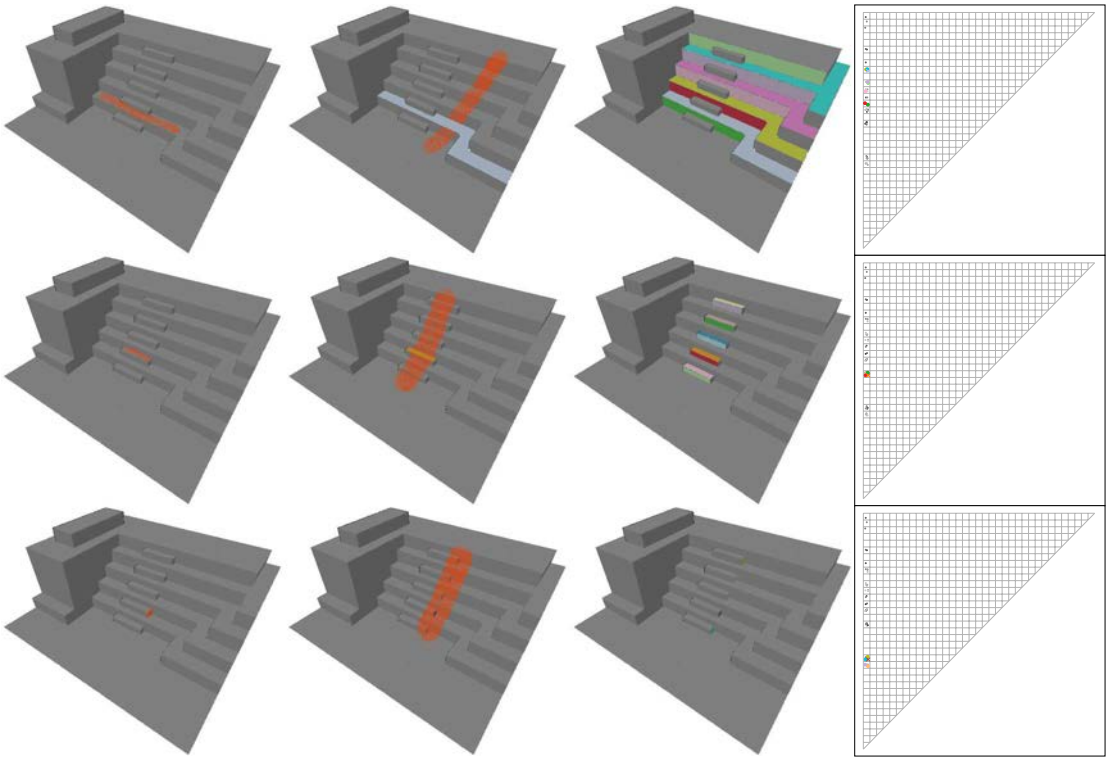


Figure B.15: Stairs. Similarity Search with parameters : 5,5,100,90

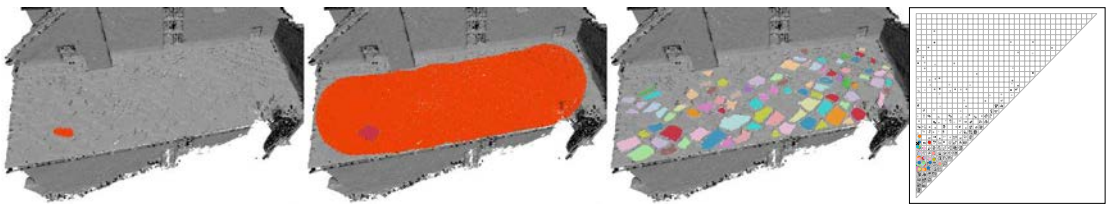


Figure B.16: Lans. Similarity Search with parameters : 5,5,100,5

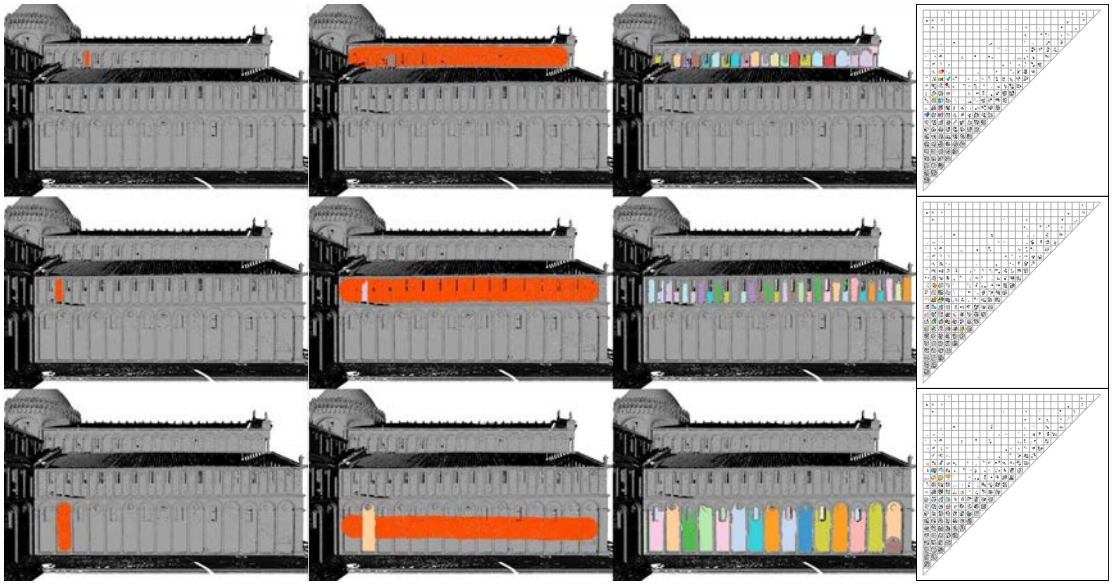


Figure B.17: Pisa. Similarity Search with parameters : 5,5,100,10

Appendix C

Complete comparison of Chapter 3

The following tables detail the numerical comparison of Table 3.1 (see also [Thompson 2019]).

Model	PCs:A	PCs:C	SBSE	MHT1	MHT2	ours
1	5.924	**0.225**	**0.225**	1.570	1.570	0.675
2	0.128	1.643	1.407	0.071	**0.060**	0.079
3	**0.006**	0.184	0.388	0.061	0.061	0.278
4	3.694	2.771	3.047	3.555	3.555	**0.258**
5	1.019	2.100	**0.887**	1.019	1.019	0.921
6	**1.101**	**1.101**	1.399	**1.101**	1.276	2.246
7	0.043	0.031	**0.018**	0.078	0.078	0.026
8	0.039	**0.028**	0.062	0.044	0.044	0.037
9	**0.260**	1.699	4.288	1.442	1.442	1.716
10	0.723	**0.582**	2.427	4.585	4.585	2.348
11	0.043	**0.035**	0.091	0.045	0.045	0.039
12	**0.036**	0.064	0.062	0.040	0.038	0.220
13	**0.060**	0.091	0.068	0.067	0.067	0.070
14	0.019	0.027	0.024	0.010	0.010	**0.008**
15	0.090	0.077	0.090	**0.030**	**0.030**	0.145
Average	0.879	0.710	0.966	0.915	0.925	**0.604**

Table C.1: Overall Hausdorff distances.

Model	PCs:A	PCs:C	SBSE	MHT1	MHT2	ours
1	0.352	0.354	0.345	0.452	0.452	**0.479**
2	**0.494**	0.475	0.421	0.210	0.213	0.482
3	0.492	**0.508**	0.411	0.292	0.292	0.383
4	0.496	**0.513**	0.342	0.449	0.449	0.392
5	**0.586**	0.582	0.427	0.555	0.555	0.563
6	0.446	0.467	0.279	0.445	0.451	**0.525**
7	0.426	0.508	0.306	0.501	0.501	**0.550**
8	0.412	0.498	0.316	0.518	0.518	**0.543**
9	**0.533**	0.502	0.221	0.474	0.474	0.447
10	0.466	0.498	0.425	0.402	0.402	**0.516**
11	**0.579**	0.554	0.389	0.562	0.562	0.562
12	0.711	**0.727**	0.548	0.667	0.637	0.666
13	0.553	0.537	0.298	**0.976**	**0.976**	0.405
14	0.565	0.517	0.304	**0.882**	**0.882**	0.512
15	0.659	0.631	0.584	**0.917**	**0.917**	0.536
Average	0.518	0.525	0.374	**0.553**	0.552	0.504

Table C.2: Overall Dice coefficients.

Model	Curve	SBSE	MHT1	MHT2	ours
1	1	0.604	0.013	0.013	0.014
	2	0.601	**0.034**	**0.034**	0.036
	3	0.575	0.110	0.110	**0.084**
	4	-	0.064	0.064	**0.041**
	5	0.555	**0.035**	**0.035**	0.056
	6	0.527	**0.118**	**0.118**	0.128
	7	0.541	0.062	0.062	**0.027**
	8	-	-	-	**0.049**
	10	0.547	**0.081**	**0.081**	0.082
2	1	0.673	0.674	**0.016**	0.171
	2	0.467	0.468	**0.013**	0.081
	3	0.461	0.461	0.015	**0.013**
	4	0.681	0.681	**0.019**	0.118
	5	0.686	0.688	**0.019**	0.155
	6	0.473	0.475	**0.018**	0.127
	7	0.461	0.462	**0.017**	0.166
	8	0.475	0.476	**0.022**	0.146
	9	0.453	0.454	**0.017**	0.169
	10	0.674	0.674	**0.018**	0.185
	11	0.686	0.686	**0.022**	0.146
	12	0.683	0.685	**0.016**	0.131
3	1	-	-	-	**0.172**
	2	0.414	**0.062**	**0.062**	0.097
4	1	0.048	**0.036**	**0.036**	0.313
	2	0.124	-	-	**0.027**
5	1	0.887	1.086	1.086	**0.470**
6	1	-	-	0.561	**0.079**
	2	**0.060**	1.213	1.213	0.279
7	1	0.070	**0.048**	**0.048**	0.22
8	1	0.064	**0.046**	**0.046**	0.314
9	1	-	**0.071**	**0.071**	0.228
10	1	-	**0.049**	**0.049**	0.15
	2	-	-	-	-
11	1	0.102	**0.033**	**0.033**	0.035
	2	**0.028**	0.048	0.048	0.062
	3	-	0.015	0.136	**0.013**
	4	0.171	0.062	0.062	**0.039**
	5	-	**0.006**	**0.006**	0.028
12	1	0.137	**0.065**	0.143	0.083
	2	-	0.069	0.155	**0.008**
	3	-	**0.019**	0.078	0.023
	4	**0.027**	0.029	0.029	0.051
	5	0.082	0.031	0.097	**0.026**
13	1	**0.035**	0.067	0.067	0.076
14	1	0.716	**0.003**	**0.003**	0.006
	2	0.464	0.011	0.011	**0.006**
	3	0.464	**0.003**	**0.003**	0.007
	4	0.470	**0.006**	**0.006**	0.007
	5	0.672	**0.006**	**0.006**	0.008
	6	0.654	0.015	0.015	**0.007**
	7	0.655	0.010	0.010	**0.007**
	8	0.667	0.069	0.069	**0.006**
	9	0.721	0.012	0.012	**0.007**
	10	0.703	**0.006**	**0.006**	0.007
	11	0.708	**0.004**	**0.004**	0.007
	12	0.500	0.012	0.012	**0.007**
	13	0.494	0.011	0.011	**0.007**
	14	0.513	**0.004**	**0.004**	0.007
	15	0.519	**0.004**	**0.004**	0.007
	16	0.470	0.009	0.009	**0.006**
15	1	0.260	**0.030**	**0.030**	-
	2	0.151	**0.013**	**0.013**	0.022
Average		0.448	0.187	0.086	**0.084**

Table C.3: Detailed Hausdorff distances.

Model	Curve	SBSE	MHT1	MHT2	ours
1	1	0.121	0.636	0.636	**0.650**
	2	0.145	0.413	0.413	**0.450**
	3	0.244	**0.503**	**0.503**	0.435
	4	-	0.445	0.445	**0.499**
	5	0.180	**0.578**	**0.578**	0.434
	6	0.214	**0.481**	**0.481**	0.458
	7	0.041	0.338	0.338	**0.382**
	8	-	-	-	**0.447**
	10	0.126	0.334	0.334	**0.354**
2	1	0.094	0.025	0.048	**0.296**
	2	0.194	0.147	0.449	**0.542**
	3	0.164	0.119	0.332	**0.633**
	4	0.265	0.176	**0.461**	0.457
	5	0.060	0.033	0.082	**0.102**
	6	0.077	0.009	0.029	**0.537**
	7	0.041	0.016	0.035	**0.076**
	8	0.062	0.018	0.044	**0.337**
	9	0.029	0.015	0.019	**0.339**
	10	**0.104**	0.022	0.038	0.032
	11	0.078	0.037	0.044	**0.248**
	12	0.174	0.107	0.282	**0.474**
3	1	-	-	-	**0.443**
	2	**0.586**	0.497	0.497	0.487
4	1	0.300	**0.426**	**0.426**	0.126
	2	0.159	-	-	**0.590**
5	1	0.381	**0.414**	**0.414**	0.304
6	1	-	-	0.046	**0.663**
	2	0.318	**0.516**	**0.516**	0.357
7	1	0.333	**0.528**	**0.528**	0.479
8	1	0.313	**0.533**	**0.533**	0.454
9	1	-	**0.390**	**0.390**	0.270
10	1	-	0.311	0.311	**0.394**
	2	-	-	-	-
11	1	0.497	**0.680**	**0.680**	0.627
	2	0.367	**0.404**	**0.404**	0.393
	3	-	**0.553**	**0.553**	0.546
	4	0.043	0.489	0.489	**0.511**
	5	-	0.705	0.705	**0.778**
12	1	0.190	0.535	0.230	**0.588**
	2	-	0.713	0.131	**0.817**
	3	-	**0.286**	0.237	0.137
	4	0.553	0.514	0.514	**0.572**
	5	0.557	**0.774**	0.450	0.766
13	1	0.574	**0.976**	**0.976**	0.567
14	1	0.227	**0.979**	**0.979**	0.559
	2	0.224	**0.987**	**0.987**	0.529
	3	0.219	**0.978**	**0.978**	0.509
	4	0.002	**0.942**	**0.942**	0.450
	5	0.001	**0.795**	**0.795**	0.489
	6	0.027	0.646	0.646	**0.657**
	7	0.178	**0.739**	**0.739**	0.655
	8	0.153	**0.667**	**0.667**	0.576
	9	0.002	**0.897**	**0.897**	0.478
	10	0.227	**0.919**	**0.919**	0.516
	11	0.004	**0.952**	**0.952**	0.465
	12	0.000	**0.956**	**0.956**	0.446
	13	0.226	**0.954**	**0.954**	0.510
	14	0.225	**0.983**	**0.983**	0.465
	15	0.000	**0.980**	**0.980**	0.464
	16	0.002	**0.907**	**0.907**	0.466
15	1	0.512	**0.913**	**0.913**	-
	2	0.732	**0.921**	**0.921**	0.662
Average		0.207	**0.541**	0.530	0.466

Table C.4: Detailed Dice coefficients.

Bibliography

[Adamson 2003a] Anders Adamson and Marc Alexa. *Approximating and Intersecting Surfaces from Points.* In Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '03, page 230–239, Goslar, DEU, 2003. Eurographics Association. [9 and 11]

[Adamson 2003b] Anders Adamson and Marc Alexa. *Ray tracing point set surfaces.* In 2003 Shape Modeling International., pages 272–279. IEEE, 2003. [11]

[Adamson 2006] Anders Adamson and Marc Alexa. *Anisotropic point set surfaces.* In Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, pages 7–13, 2006. [9]

[Alexa 2001] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin and Claudio T Silva. *Point set surfaces.* In Visualization Conference, pages 21–29. IEEE, 2001. [6, 9, 13, 19, 23, 30, 34, 77, and 78]

[Alexa 2003] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin and Claudio T. Silva. *Computing and rendering point set surfaces.* IEEE Transactions on Visualization and Computer Graphics, vol. 9, no. 1, pages 3–15, 2003. [11]

[Alexa 2004] Marc Alexa and Anders Adamson. *On Normals and Projection Operators for Surfaces Defined by Point Sets.* SPBG, vol. 4, pages 149–155, 2004. [11]

[Alliez 2003] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy and Mathieu Desbrun. *Anisotropic Polygonal Remeshing.* ACM Trans. Graph., vol. 22, no. 3, page 485–493, July 2003. [60]

[Amenta 2001] Nina Amenta, Sunghee Choi and Ravi Krishna Kolluri. *The power crust.* In Proceedings of the sixth ACM symposium on Solid modeling and applications, pages 249–266, 2001. [77]

[Amenta 2004] Nina Amenta and Yong Joo Kil. *Defining Point-Set Surfaces.* In ACM SIGGRAPH 2004 Papers, SIGGRAPH '04, page 264–270, New York, NY, USA, 2004. Association for Computing Machinery. [9]

[Armeni 2016] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer and Silvio Savarese. *3D Semantic Parsing of Large-Scale Indoor Spaces.* In 2016 IEEE Conference on Computer Vision and Pattern Recognition, pages 1534–1543, 2016. [48]

[Attene 2010] Marco Attene and Giuseppe Patanè. *Hierarchical Structure Recovery of Point-Sampled Surfaces.* Computer Graphics Forum, vol. 29, no. 6, pages 1905–1920, 2010. [40]

[Aubry 2011] Mathieu Aubry, Ulrich Schlickewei and Daniel Cremers. *The wave kernel signature: A quantum mechanical approach to shape analysis.* In 2011 IEEE international conference on computer vision workshops (ICCV workshops), pages 1626–1633. IEEE, 2011. [12]

[Avron 2010] Haim Avron, Andrei Sharf, Chen Greif and Daniel Cohen-Or. *ℓ1-Sparse Reconstruction of Sharp Point Set Surfaces.* ACM Trans. Graph., vol. 29, no. 5, November 2010. [9]

[Bartels 1972] Richard H. Bartels and George W Stewart. *Solution of the matrix equation AX+ XB= C [F4].* Communications of the ACM, vol. 15, no. 9, pages 820–826, 1972. [35]

[Bauer 2009] Ulrich Bauer and Konrad Polthier. *Generating parametric models of tubes from laser scans.* Computer-Aided Design, vol. 41, no. 10, pages 719–729, 2009. [63]

[Bazazian 2015] Dena Bazazian, Josep R Casas and Javier Ruiz-Hidalgo. *Fast and robust edge extraction in unorganized point clouds.* In 2015 international conference on digital image computing: techniques and applications (DICTA), pages 1–8. IEEE, 2015. [62]

[Belkin 2003] Mikhail Belkin and Partha Niyogi. *Laplacian eigenmaps for dimensionality reduction and data representation.* Neural computation, vol. 15, no. 6, pages 1373–1396, 2003. [89]

[Belkin 2009] Mikhail Belkin, Jian Sun and Yusu Wang. *Constructing Laplace operator from point clouds in $\mathbb{R}$ d.* In Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms, pages 1031–1040. SIAM, 2009. [12 and 77]

[Bénard 2019] Pierre Bénard and Aaron Hertzmann. *Line drawings from 3D models: a tutorial.* Foundations and Trends in Computer Graphics and Vision, vol. 11, no. 1-2, page 159, September 2019. [60]

[Berger 2017] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf and Claudio T. Silva. *A Survey of Surface Reconstruction from Point Clouds.* Computer Graphics Forum, vol. 36, no. 1, pages 301–329, 2017. [6]

[Berkmann 1994] Jens Berkmann and Terry Caelli. *Computation of surface geometry and segmentation using covariance techniques.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 11, pages 1114–1116, 1994. [8]

[Bernardini 1999] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva and Gabriel Taubin. *The ball-pivoting algorithm for surface reconstruction.* IEEE transactions on visualization and computer graphics, vol. 5, no. 4, pages 349–359, 1999. [80]

[Bey 2011] Aurelien Bey, Raphaelle Chaine, Raphael Marc, Guillaume Thibault and Samir Akkouche. *Reconstruction of consistent 3D CAD models from point cloud data using a priori CAD models.* In ISPRS workshop on laser scanning, volume 1, 2011. [63]

[Biasotti 2018] Silvia Biasotti, E. Moscoso Thompson, Loïc Barthe, Stefano Berretti, A. Giachetti, Thibault Lejemble, N Mellado, Konstantinos Moustakas, Iason Manolas, Dimitrios Dimou, Claudio Tortorici, Santiago Velasco-Forero, Naoufel Werghi, M Polig, G Sorrentino and Sorin Hermon. *SHREC'18 track: Recognition of geometric patterns over 3D models.* Eurographics Workshop on 3D Object Retrieval, 2018. [76]

[Böhm 1984] Wolfgang Böhm, Gerald Farin and Jürgen Kahmann. *A survey of curve and surface methods in CAGD.* Computer Aided Geometric Design, vol. 1, no. 1, pages 1–60, 1984. [9]

[Boissonnat 2017] Jean-Daniel Boissonnat. Géométrie algorithmique: des données géométriques à la géométrie des données. Collège de France. Fayard, 2017. [89]

[Bokeloh 2009] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel and A. Schilling. *Symmetry Detection Using Feature Lines.* Computer Graphics Forum, vol. 28, no. 2, pages 697–706, 2009. [60]

[Bommes 2009] David Bommes, Henrik Zimmer and Leif Kobbelt. *Mixed-Integer Quadrangulation.* In ACM SIGGRAPH 2009 Papers, SIGGRAPH '09, New York, NY, USA, 2009. Association for Computing Machinery. [79]

[Borrmann 2011] Dorit Borrmann, Jan Elseberg, Kai Lingemann and Andreas Nüchter. *The 3D hough transform for plane detection in point clouds: A review and a new accumulator design.* 3D Research, vol. 2, no. 2, page 3, 2011. [40]

[Botsch 2005] Mario Botsch, David Bommes and Leif Kobbelt. *Efficient linear system solvers for mesh processing.* In Mathematics of Surfaces XI, pages 62–83. Springer, 2005. [32]

[Botsch 2010] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez and Bruno Lévy. Polygon mesh processing. A K Peters/CRC Press, 2010. [78]

[Boulch 2017] Alexandre Boulch, Bertrand Le Saux and Nicolas Audebert. *Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks.* 3DOR, vol. 2, page 7, 2017. [88]

[Briggs 2000] William L Briggs, Van Emden Henson and Steve F McCormick. A multigrid tutorial. SIAM, 2000. [32]

[Brodu 2012] Nicolas Brodu and Dimitri Lague. *3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 68, pages 121 – 134, 2012. [8 and 63]

[Bronstein 2010] Michael Bronstein and Iasonas Kokkinos. *Scale-invariant heat kernel signatures for non-rigid shape recognition.* In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1704–1711, 2010. [27]

[Bronstein 2011] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas and Maks Ovsjanikov. *Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval.* ACM Trans. Graph., vol. 30, no. 1, February 2011. [12]

[Béarzi 2018] Yohann Béarzi, Julie Digne and Raphaëlle Chaine. *Wavejets: A Local Frequency Framework for Shape Details Amplification.* Computer Graphics Forum, vol. 37, no. 2, pages 13–24, 2018. [7, 8, 23, and 77]

[Carr 2001] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum and Tim R Evans. *Reconstruction and representation of 3D objects with radial basis functions.* In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 67–76, 2001. [9]

[Cazals 2005a] F. Cazals and M. Pouget. *Estimating differential quantities using polynomial fitting of osculating jets.* Computer Aided Geometric Design, vol. 22, no. 2, pages 121 – 146, 2005. [7, 8, 9, 19, 23, 34, and 77]

[Cazals 2005b] Frédéric Cazals and Marc Pouget. *Topology driven algorithms for ridge extraction on meshes.* Research Report RR-5526, INRIA, 2005. [62]

[Cazals 2006] Frédéric Cazals and Joachim Giesen. *Delaunay triangulation based surface reconstruction.* In Effective computational geometry for curves and surfaces, pages 231–276. Springer, 2006. [7]

[Chauve 2010] Anne-Laure Chauve, Patrick Labatut and Jean-Philippe Pons. *Robust Piecewise-planar 3D Reconstruction and Completion from Large-scale Unstructured Point Data.* In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1261–1268, 2010. [39]

[Chazal 2017] Frédéric Chazal and Bertrand Michel. *An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists.* Submitted, 2017. [45]

[Chen 2013] Jiazhou Chen, Gaël Guennebaud, Pascal Barla and Xavier Granier. *Non-Oriented MLS Gradient Fields.* Computer Graphics Forum, vol. 32, no. 8, pages 98–109, 2013. [34]

[Cheng 2008] Zhi-Quan Cheng, Yan-Zhen Wang, Bao Li, Kai Xu, Gang Dang and Shi-Yao Jin. *A Survey of Methods for Moving Least Squares Surfaces.* In Volume graphics, pages 9–23, 2008. [9]

[Choi 2016] Gary Pui-Tung Choi, Kin Tat Ho and Lok Ming Lui. *Spherical conformal parameterization of genus-0 point clouds for meshing.* SIAM Journal on Imaging Sciences, vol. 9, no. 4, pages 1582–1618, 2016. [79]

[Clarenz 2004] Ulrich Clarenz, Martin Rumpf and Alexandru Telea. *Robust feature detection and local classification for surfaces based on moment analysis.* IEEE Transactions on Visualization and Computer Graphics, vol. 10, no. 5, pages 516–524, 2004. [14]

[Coeurjolly 2013] David Coeurjolly, Jacques-Olivier Lachaud and Jérémy Levallois. *Integral based curvature estimators in digital geometry.* In International Conference on Discrete Geometry for Computer Imagery, pages 215–227. Springer, 2013. [8]

[Coeurjolly 2014] David Coeurjolly, Jacques-Olivier Lachaud and Jérémy Levallois. *Multigrid convergent principal curvature estimators in digital geometry.* Computer Vision and Image Understanding, vol. 129, pages 27 – 41, 2014. Special section: Advances in Discrete Geometry for Computer Imagery. [8]

[Cohen-Steiner 2004] David Cohen-Steiner, Pierre Alliez and Mathieu Desbrun. *Variational Shape Approximation.* ACM Trans. Graph., vol. 23, no. 3, page 905–914, August 2004. [8]

[Cohen 2019] David Cohen and Mirela Ben-Chen. *Generalized volumetric foliation from inverted viscous flow.* Computers & Graphics, vol. 82, pages 152 – 162, 2019. [80]

[Cole 2008] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser and Szymon Rusinkiewicz. *Where Do People Draw Lines?* ACM Trans. Graph., vol. 27, no. 3, page 1–11, August 2008. [68]

[Crane 2013] Keenan Crane, Ulrich Pinkall and Peter Schröder. *Robust Fairing via Conformal Curvature Flow.* ACM Trans. Graph., vol. 32, no. 4, July 2013. [80]

[Czerniawski 2016] Thomas Czerniawski, Mohammad Nahangi, Carl Haas and Scott Walbridge. *Pipe spool recognition in cluttered point clouds using a curvature-based shape descriptor.* Automation in Construction, vol. 71, pages 346–358, 2016. [63]

[DeCarlo 2003] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz and Anthony Santella. *Suggestive Contours for Conveying Shape.* In ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, page 848–855, New York, NY, USA, 2003. Association for Computing Machinery. [62]

[Degener 2003] Patrick Degener, Jan Meseth and Reinhard Klein. *An Adaptable Surface Parameterization Method.* IMR, vol. 3, pages 201–213, 2003. [82]

[Demantké 2011] Jérôme Demantké, Clément Mallet, Nicolas David and Bruno Vallet. *Dimensionality based scale selection in 3D lidar point clouds.* In Proceedings of the ISPRS Workshop Laser Scanning, volume 38, pages 97–102, 01 2011. [8]

[Desbrun 1999] Mathieu Desbrun, Mark Meyer, Peter Schröder and Alan H Barr. *Implicit fairing of irregular meshes using diffusion and curvature flow.* In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 317–324, 1999. [12]

[Deza 2009] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. In Encyclopedia of distances, pages 1–583. Springer, 2009. [68 and 69]

[Digne 2011] Julie Digne, Jean-Michel Morel, Charyar-Mehdi Souzani and Claire Lartigue. *Scale Space Meshing of Raw Data Point Sets*. Computer Graphics Forum, vol. 30, no. 6, pages 1630–1642, 2011. [2, 6, 8, 13, 14, 15, 18, 23, 26, 80, and 95]

[Digne 2014] Julie Digne and Jean-Michel Morel. *Numerical analysis of differential operators on raw point clouds*. Numerische Mathematik, vol. 127, no. 2, pages 255–289, 2014. [13 and 14]

[Digne 2018] Julie Digne, Sébastien Valette and Raphaëlle Chaine. *Sparse geometric representation through local shape probing*. IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 7, pages 2238–2250, 2018. [63]

[Dimitrov 2016] Andrey Dimitrov, Rongqi Gu and Mani Golparvar-Fard. *Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling*. Computer-Aided Civil and Infrastructure Engineering, vol. 31, no. 7, pages 483–498, 2016. [63]

[Do Carmo 1976] Manfredo P Do Carmo. Differential geometry of curves and surfaces. Prentice-Hall, 1976. [15]

[Dong 2006] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci and John C. Hart. *Spectral Surface Quadrangulation*. In ACM SIGGRAPH 2006 Papers, SIGGRAPH '06, page 1057–1066, New York, NY, USA, 2006. Association for Computing Machinery. [12]

[Dong 2018] Zhen Dong, Bisheng Yang, Pingbo Hu and Sebastian Scherer. *An efficient global energy optimization approach for robust 3D plane segmentation of point clouds*. ISPRS Journal of Photogrammetry and Remte Sensing, 2018. [40]

[Edelsbrunner 1983] Herbert Edelsbrunner, David Kirkpatrick and Raimund Seidel. *On the shape of a set of points in the plane*. IEEE Transactions on information theory, vol. 29, no. 4, pages 551–559, 1983. [43]

[Edelsbrunner 2000] Herbert Edelsbrunner, David Letscher and Afra Zomorodian. *Topological persistence and simplification*. In Proceedings 41st annual symposium on foundations of computer science, pages 454–463. IEEE, 2000. [45]

[Falcidieno 2004] Bianca Falcidieno. *Aim@Shape project presentation*. In Proceedings Shape Modeling Applications, 2004., page 329. IEEE, 2004. [48 and 68]

[Fang 2018] Hao Fang, Florent Lafarge and Mathieu Desbrun. *Planar Shape Detection at Structural Scales*. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2965–2973, 2018. [41, 51, 52, and 53]

[Feng 2014] Chen Feng, Yuichi Taguchi and Vineet R Kamat. *Fast Plane Extraction in Organized Point Clouds using Agglomerative Hierarchical Clustering*. In Proc. International Conference on Robotics and Automation (ICRA), pages 6218–6225, 2014. [40]

[Fischler 1981] Martin A Fischler and Robert C Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Communications of the ACM, vol. 24, no. 6, pages 381–395, 1981. [40]

[Fleishman 2003] Shachar Fleishman, Daniel Cohen-Or, Marc Alexa and Cláudio T. Silva. *Progressive Point Set Surfaces*. ACM Trans. Graph., vol. 22, no. 4, page 997–1011, October 2003. [9 and 30]

[Fleishman 2005] Shachar Fleishman, Daniel Cohen-Or and Cláudio T. Silva. *Robust Moving Least-Squares Fitting with Sharp Features.* In ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, page 544–552, New York, NY, USA, 2005. Association for Computing Machinery. [9]

[Floater 1997] Michael S Floater. *Parametrization and smooth approximation of surface triangulations.* Computer aided geometric design, vol. 14, no. 3, pages 231–250, 1997. [79]

[Floater 2001] Michael S Floater and Martin Reimers. *Meshless parameterization and surface reconstruction.* Computer Aided Geometric Design, vol. 18, no. 2, pages 77–92, 2001. [79]

[Gehre 2016] Anne Gehre, Isaak Lim and Leif Kobbelt. *Adapting Feature Curve Networks to a Prescribed Scale.* Computer Graphics Forum, vol. 35, no. 2, pages 319–330, 2016. [60]

[Gelfand 2005] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas and Helmut Pottmann. *Robust Global Registration.* In Proceedings of the Third Eurographics Symposium on Geometry Processing, SGP '05, page 197–es, Goslar, DEU, 2005. The Eurographics Association. [2]

[Groueix 2018] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell and Mathieu Aubry. *AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation.* In Computer Vision and Pattern Recognition (CVPR), 2018. [80]

[Gu 2003] Xianfeng Gu and Shing-Tung Yau. *Computing Conformal Structure of Surfaces.* Communications in Information and Systems, vol. 2, jan 2003. [80]

[Guennebaud 2007] Gaël Guennebaud and Markus Gross. *Algebraic Point Set Surfaces.* ACM Trans. Graph., vol. 26, no. 3, page 23–es, July 2007. [3, 6, 10, 19, 21, 23, 25, 34, 41, 63, 77, 78, and 81]

[Guennebaud 2008] Gaël Guennebaud, Marcel Germann and Markus Gross. *Dynamic Sampling and Rendering of Algebraic Point Set Surfaces.* Computer Graphics Forum, vol. 27, no. 2, pages 653–662, 2008. [10, 11, 14, and 35]

[Guennebaud 2010] Gaël Guennebaud, Benoît Jacob *et al. Eigen v3.* `http://eigen.tuxfamily.org`, 2010. [24 and 47]

[Guerrero 2018] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov and Niloy J. Mitra. *PCPNet Learning Local Shape Properties from Raw Point Clouds.* Computer Graphics Forum, vol. 37, no. 2, pages 75–85, 2018. [35]

[Guillemot 2012] Thierry Guillemot, Andres Almansa and Tamy Boubekeur. *Non local point set surfaces.* In 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pages 324–331. IEEE, 2012. [9]

[Guinard 2019] St'ephane Guinard, Loic Landrieu, Laurent Caraffa and Bruno Vallet. *Piecewise-planar Approximation of Large 3D Data as Graph-structued Optimization.* ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 4, pages 365–372, 2019. [40]

[Gumhold 2001] Stefan Gumhold, Xinlong Wang, Rob S MacLeod *et al. Feature Extraction From Point Clouds.* In 10th International Meshing Roundtable, pages 293–305, 2001. [62]

[Hackel 2016] Timo Hackel, Jan D Wegner and Konrad Schindler. *Contour detection in unstructured 3d point clouds.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1610–1618, 2016. [48 and 62]

[Harris 1988] Christopher G Harris, Mike Stephens *et al. A combined corner and edge detector.* In Alvey vision conference, volume 15, 1988. [8]

[Hildebrandt 2005] Klaus Hildebrandt, Konrad Polthier and Max Wardetzky. *Smooth Feature Lines on Surface Meshes.* In Proceedings of the Third Eurographics Symposium on Geometry Processing, SGP '05, page 85–es, Goslar, DEU, 2005. The Eurographics Association. [62]

[Hoppe 1992] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle. *Surface Reconstruction from Unorganized Points.* SIGGRAPH Comput. Graph., vol. 26, no. 2, page 71–78, July 1992. [8]

[Hormann 2002] Kai Hormann and Martin Reimers. *Triangulating point clouds with spherical topology.* Curve and Surface Design: Saint-Malo, pages 215–224, 2002. [79]

[Hormann 2012] Kai Hormann and Guenther Greiner. *MIPS: An Efficient Global Parametrization Method.* Curve and Surface Design: Saint-Malo, vol. 2000, page 10, 11 2012. [82]

[Hough 1962] Paul VC Hough. *Method and means for recognizing complex patterns*, December 18 1962. US Patent 3,069,654. [40]

[Hu 2018] Shi-Min Hu, Jun-Xiong Cai and Yu-Kun Lai. *Semantic Labeling and Instance Segmentation of 3D Point Clouds using Patch Context Analysis and Multiscale Processing.* IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 7, pages 2485–2498, 2018. [41]

[Huang 2008] Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt and Hujun Bao. *Spectral Quadrangulation with Orientation and Alignment Control.* In ACM SIGGRAPH Asia 2008 Papers, SIGGRAPH Asia '08, New York, NY, USA, 2008. Association for Computing Machinery. [12]

[Huang 2009] Qi-Xing Huang, Martin Wicke, Bart Adams and Leonidas Guibas. *Shape Decomposition using Modal Analysis.* Computer Graphics Forum, vol. 28, no. 2, pages 407–416, 2009. [12]

[Huang 2019] Zhiyang Huang, Nathan Carr and Tao Ju. *Variational Implicit Point Set Surfaces.* ACM Trans. Graph., vol. 38, no. 4, July 2019. [9]

[Iijima 1963] Taizo Iijima. *"Theory of pattern recognition.* Electronics and Communications in Japan, pages 123–134, 1963. [2]

[Isack 2012] Hossam Isack and Yuri Boykov. *Energy-Based Geometric Multi-model Fitting.* Int. Journal of Computer Vision, vol. 97, no. 2, pages 123–147, 2012. [40]

[Jacobson 2018] Alec Jacobson, Daniele Panozzo *et al. libigl: A simple C++ geometry processing library*, 2018. https://libigl.github.io/.

[Kaiser 2019] Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur. *A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data.* Computer Graphics Forum, vol. 38, no. 1, pages 167–196, 2019. [6, 40, and 63]

[Kalogerakis 2007] Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai and Karan Singh. *Robust Statistical Estimation of Curvature on Discretized Surfaces.* In Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07, page 13–22, Goslar, DEU, 2007. The Eurographics Association. [35 and 64]

[Kalogerakis 2009] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari and Karan Singh. *Extracting lines of curvature from noisy point clouds.* Computer-Aided Design, vol. 41, no. 4, pages 282–292, 2009. [60, 62, 63, and 64]

[Kalogerakis 2010] Evangelos Kalogerakis, Aaron Hertzmann and Karan Singh. *Learning 3D Mesh Segmentation and Labeling*. ACM Trans. Graph., vol. 29, no. 4, July 2010. [8]

[Karni 2000] Zachi Karni and Craig Gotsman. *Spectral compression of mesh geometry*. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 279–286, 2000. [12]

[Kawashima 2014] Kazuaki Kawashima, Satoshi Kanai and Hiroaki Date. *As-built modeling of piping system from terrestrial laser-scanned point clouds using normal-based region growing*. Journal of Computational Design and Engineering, vol. 1, no. 1, pages 13–26, 2014. [63]

[Kazhdan 2006] Michael Kazhdan, Matthew Bolitho and Hugues Hoppe. *Poisson Surface Reconstruction*. In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, page 61–70, Goslar, DEU, 2006. The Eurographics Association. [77]

[Kazhdan 2012] Michael Kazhdan, Jake Solomon and Mirela Ben-Chen. *Can Mean-Curvature Flow be Modified to be Non-singular?* Computer Graphics Forum, vol. 31, no. 5, pages 1745–1754, 2012. [80]

[Kerautret 2015] Bertrand Kerautret, Adrien Krähenbühl, Isabelle Debled-Rennesson and Jacques-Olivier Lachaud. *3D geometric analysis of tubular objects based on surface normal accumulation*. In International Conference on Image Analysis and Processing, pages 319–331. Springer, 2015. [63]

[Kim 2013] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi and Thomas Funkhouser. *Learning Part-Based Templates from Large Collections of 3D Shapes*. ACM Trans. Graph., vol. 32, no. 4, July 2013. [8]

[Koch 2019] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin and Daniele Panozzo. *ABC: A big CAD model dataset for geometric deep learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9601–9611, 2019. [62 and 68]

[Koenderink 1984] Jan J Koenderink. *The structure of images*. Biological cybernetics, vol. 50, no. 5, pages 363–370, 1984. [13 and 26]

[Kokkinos 2008] Iasonas Kokkinos and Alan Yuille. *Scale invariance without scale selection*. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2008. [27]

[Kolluri 2008] Ravikrishna Kolluri. *Provably good moving least squares*. ACM Transactions on Algorithms (TALG), vol. 4, no. 2, pages 1–25, 2008. [10]

[Lafarge 2012] Florent Lafarge and Clément Mallet. *Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation*. International journal of computer vision, vol. 99, no. 1, pages 69–85, 2012. [40]

[Lee 2013] Joohyuk Lee, Hyojoo Son, Changmin Kim and Changwan Kim. *Skeleton-based 3D reconstruction of as-built pipelines from laser-scan data*. Automation in construction, vol. 35, pages 199–207, 2013. [63]

[Levallois 2015] Jérémy Levallois, David Coeurjolly and Jacques-Olivier Lachaud. *Scale-space feature extraction on digital surfaces*. Computers & Graphics, vol. 51, pages 177 – 189, 2015. International Conference Shape Modeling International. [13]

[Levin 1998] David Levin. *The approximation power of moving least-squares.* Mathematics of computation, vol. 67, no. 224, pages 1517–1531, 1998. [9, 78, and 81]

[Levin 2004] David Levin. *Mesh-independent surface interpolation.* In Geometric modeling for scientific visualization, pages 37–49. Springer, 2004. [9]

[Li 2011a] Er Li, Bruno Lévy, Xiaopeng Zhang, Wujun Che, Weiming Dong and Jean-Claude Paul. *Meshless quadrangulation by global parameterization.* Computers & Graphics, vol. 35, no. 5, pages 992 – 1000, 2011. [79]

[Li 2011b] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or and Niloy J Mitra. *Globfit: Consistently fitting primitives by discovering global relations.* ACM Transactions on Graphics, vol. 30, no. 4, page 52, 2011. [41]

[Li 2018] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di and Baoquan Chen. *Pointcnn: Convolution on x-transformed points.* In Advances in neural information processing systems, pages 820–830, 2018. [88]

[Liang 1990] Ping Liang and John S. Todhunter. *Representation and Recognition of Surface Shapes in Range Images: A Differential Geometry Approach.* Comput. Vision Graph. Image Process., vol. 52, no. 1, page 78–109, August 1990. [8]

[Lin 2015] Yangbin Lin, Cheng Wang, Jun Cheng, Bili Chen, Fukai Jia, Zhonggui Chen and Jonathan Li. *Line segment extraction for large scale unorganized point clouds.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 102, pages 172–183, 2015. [62]

[Lindeberg 1990] Tony Lindeberg. *Scale-space for discrete signals.* IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 3, pages 234–254, 1990. [13]

[Lindeberg 2013a] Tony Lindeberg. *A computational theory of visual receptive fields.* Biological cybernetics, vol. 107, no. 6, pages 589–635, 2013. [2]

[Lindeberg 2013b] Tony Lindeberg. Scale-space theory in computer vision, volume 256. Springer Science & Business Media, 2013. [2]

[Ling 2015] Ruotian Ling, Jin Huang, Bert Jüttler, Feng Sun, Hujun Bao and Wenping Wang. *Spectral Quadrangulation with Feature Curve Alignment and Element Size Control.* ACM Trans. Graph., vol. 34, no. 1, December 2015. [12]

[Lipman 2007] Yaron Lipman, Daniel Cohen-Or, David Levin and Hillel Tal-Ezer. *Parameterization-Free Projection for Geometry Reconstruction.* In ACM SIGGRAPH 2007 Papers, SIGGRAPH '07, page 22–es, New York, NY, USA, 2007. Association for Computing Machinery. [64]

[Liu 2012] Yang Liu, Balakrishnan Prabhakaran and Xiaohu Guo. *Point-based manifold harmonics.* IEEE Transactions on visualization and computer graphics, vol. 18, no. 10, pages 1693–1703, 2012. [12, 13, and 77]

[Liu 2017] Hsueh-Ti Derek Liu, Alec Jacobson and Keenan Crane. *A Dirac Operator for Extrinsic Shape Analysis.* Computer Graphics Forum, vol. 36, no. 5, pages 139–149, 2017. [13]

[Lowe 1999] David G Lowe. *Object recognition from local scale-invariant features.* In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. IEEE, 1999. [13]

[Lubachevsky 1997] Boris D Lubachevsky and Ronald L Graham. *Curved hexagonal packings of equal disks in a circle.* Discrete & Computational Geometry, vol. 18, no. 2, pages 179–194, 1997. [32]

[Lukács 1998] Gabor Lukács, Ralph Martin and Dave Marshall. *Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation.* In European conference on computer vision, pages 671–686. Springer, 1998. [63]

[Maaten 2008] Laurens van der Maaten and Geoffrey Hinton. *Visualizing data using t-SNE.* Journal of machine learning research, vol. 9, no. Nov, pages 2579–2605, 2008. [89]

[Maillot 1993] Jérôme Maillot, Hussein Yahia and Anne Verroust. *Interactive Texture Mapping.* In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, page 27–34, New York, NY, USA, 1993. Association for Computing Machinery. [82]

[Manay 2004] Siddharth Manay, Byung-Woo Hong, Anthony J Yezzi and Stefano Soatto. *Integral invariant signatures.* In European Conference on Computer Vision (ECCV), pages 87–99. Springer, 2004. [14]

[Mattausch 2014] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung and Renato Pajarola. *Object Detection and Classification from Large-Scale Cluttered Indoor Scans.* Computer Graphics Forum, vol. 33, no. 2, pages 11–21, 2014. [39]

[Mellado 2012] Nicolas Mellado, Gaël Guennebaud, Pascal Barla, Patrick Reuter and Christophe Schlick. *Growing Least Squares for the Analysis of Manifolds in Scale-Space.* Computer Graphics Forum, vol. 31, no. 5, pages 1691–1701, 2012. [2, 11, 14, 16, 19, 20, 21, 26, 73, 77, 86, and 88]

[Mellado 2015a] Nicolas Mellado, Matteo Dellepiane and Roberto Scopigno. *Relative scale estimation and 3D registration of multi-modal geometry using Growing Least Squares.* IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 9, pages 2160–2173, 2015. [14 and 27]

[Mellado 2015b] Nicolas Mellado, Matteo Dellepiane and Roberto Scopigno. *Relative scale estimation and 3D registration of multi-modal geometry using Growing Least Squares.* IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 9, pages 2160–2173, 2015. [48]

[Mellado 2020] Nicolas Mellado, Thibault Lejemble, Gaël Guennebaud and Pascal Barla. *Ponca: a Point Cloud Analysis Library.* https://github.com/poncateam/ponca, 2020. [6, 11, 19, 21, 22, 23, and 24]

[Melzi 2018] S. Melzi, E. Rodolà, U. Castellani and M. M. Bronstein. *Localized Manifold Harmonics for Spectral Shape Analysis.* Computer Graphics Forum, vol. 37, no. 6, pages 20–34, 2018. [13]

[Meng 2013] Qinggang Meng, Baihua Li, Horst Holstein and Yonghuai Liu. *Parameterization of point-cloud freeform surfaces using adaptive sequential learning RBFnetworks.* Pattern Recognition, vol. 46, no. 8, pages 2361–2375, 2013. [80]

[Mérigot 2010] Quentin Mérigot, Maks Ovsjanikov and Leonidas J Guibas. *Voronoi-based curvature and feature estimation from point clouds.* IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 6, pages 743–756, 2010. [35, 62, and 77]

[Meyer 2003] Mark Meyer, Mathieu Desbrun, Peter Schröder and Alan H Barr. *Discrete differential-geometry operators for triangulated 2-manifolds.* In Visualization and mathematics III, pages 35–57. Springer, 2003. [12]

[Mitra 2003] Niloy J. Mitra and An Nguyen. *Estimating Surface Normals in Noisy Point Cloud Data.* In Proceedings of the Nineteenth Annual Symposium on Computational Geometry, SCG '03, page 322–328, New York, NY, USA, 2003. Association for Computing Machinery. [8]

[Monszpart 2015] Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow and Niloy J. Mitra. *RAPter: Rebuilding Man-Made Scenes with Regular Arrangements of Planes.* ACM Trans. Graph., vol. 34, no. 4, July 2015. [39, 41, 48, 51, and 52]

[Mortara 2004] Michela Mortara, Giuseppe Patané, Michela Spagnuolo, Bianca Falcidieno and Jarek Rossignac. *Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies.* In Symposium on Solid Modeling and Applications, pages 339–344, 2004. [63]

[Mullen 2008] Patrick Mullen, Yiying Tong, Pierre Alliez and Mathieu Desbrun. *Spectral Conformal Parameterization.* Computer Graphics Forum, vol. 27, no. 5, pages 1487–1494, 2008. [79]

[Musialski 2013] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. van Gool and W. Purgathofer. *A Survey of Urban Reconstruction.* Computer Graphics Forum, vol. 32, no. 6, pages 146–177, 2013. [6]

[Nader 2014] G. Nader, G. Guennebaud and N. Mellado. *Adaptive multi-scale analysis for point-based surface editing.* Computer Graphics Forum, vol. 33, no. 7, pages 171–179, 2014. [14]

[Najman 2017] Laurent Najman, Pascal Romon *et al.* Modern approaches to discrete curvature, volume 2184 of *Lecture Notes in Mathematics.* Springer, 2017.

[Nan 2017] Liangliang Nan and Peter Wonka. *PolyFit: Polygonal Surface Reconstruction from Point Clouds.* In 2017 IEEE International Conference on Computer Vision (ICCV), pages 2372–2380, 2017. [51]

[Narváez 2006] Esmeide A Leal Narváez and Nallig Eduardo Leal Narváez. *Point cloud denoising using robust principal component analysis.* In GRAPP, pages 51–58, 2006. [8]

[Nasikun 2018] Ahmad Nasikun, Christopher Brandt and Klaus Hildebrandt. *Fast Approximation of Laplace-Beltrami Eigenproblems.* Computer Graphics Forum, vol. 37, no. 5, pages 121–134, 2018. [13]

[Ni 2016] Huan Ni, Xiangguo Lin, Xiaogang Ning and Jixian Zhang. *Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods.* Remote Sensing, vol. 8, no. 9, page 710, 2016. [62]

[Oesau 2016] Sven Oesau, Florent Lafarge and Pierre Alliez. *Planar Shape Detection and Regularization in Tandem.* Computer Graphics Forum, vol. 35, no. 1, pages 203–215, 2016. [41]

[Ohtake 2003] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk and Hans-Peter Seidel. *Multi-Level Partition of Unity Implicits.* ACM Trans. Graph., vol. 22, no. 3, page 463–470, July 2003. [77]

[Ohtake 2004] Yutaka Ohtake, Alexander Belyaev and Hans-Peter Seidel. *Ridge-Valley Lines on Meshes via Implicit Surface Fitting.* ACM Trans. Graph., vol. 23, no. 3, page 609–612, August 2004. [62 and 68]

[Ovsjanikov 2012] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher and Leonidas Guibas. *Functional Maps: A Flexible Representation of Maps between Shapes.* ACM Trans. Graph., vol. 31, no. 4, July 2012. [12]

[Oztireli 2009] Cengiz Oztireli, Gaël Guennebaud and Markus Gross. *Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression.* Computer Graphics Forum, vol. 28, no. 2, pages 493–501, 2009. [9, 41, and 42]

[Patil 2017] Ashok Kumar Patil, Pavitra Holi, Sang Keun Lee and Young Ho Chai. *An adaptive approach for the reconstruction and modeling of as-built 3D pipelines from point clouds.* Automation in construction, vol. 75, pages 65–78, 2017. [63]

[Pauly 2001] Mark Pauly and Markus Gross. *Spectral processing of point-sampled geometry.* In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 379–386, 2001. [12]

[Pauly 2002] Mark Pauly, Markus Gross and Leif P Kobbelt. *Efficient simplification of point-sampled surfaces.* In IEEE Visualization, 2002. VIS 2002., pages 163–170. IEEE, 2002. [8, 13, 27, and 30]

[Pauly 2003] Mark Pauly, Richard Keiser and Markus Gross. *Multi-scale Feature Extraction on Point-Sampled Surfaces.* Computer Graphics Forum, vol. 22, no. 3, pages 281–289, 2003. [2, 6, 13, 26, 27, 30, 38, 62, 63, and 77]

[Pauly 2006] Mark Pauly, Leif P. Kobbelt and Markus Gross. *Point-Based Multiscale Surface Representation.* ACM Trans. Graph., vol. 25, no. 2, page 177–193, April 2006. [14]

[Pauly 2008] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann and Leonidas J. Guibas. *Discovering Structural Regularity in 3D Geometry.* ACM Trans. Graph., vol. 27, no. 3, page 1–11, August 2008. [6]

[Peeters 2009] THJM Peeters, PR Rodrigues, Anna Vilanova and BM ter Haar Romeny. *Analysis of distance/similarity measures for diffusion tensor imaging.* In Visualization and Processing of Tensor Fields, pages 113–136. Springer, 2009. [73]

[Petronetto 2013] F. Petronetto, A. Paiva, E. S. Helou, D. E. Stewart and L. G. Nonato. *Mesh-Free Discrete Laplace–Beltrami Operator.* Computer Graphics Forum, vol. 32, no. 6, pages 214–226, 2013. [12]

[Pham 2016] Trung T. Pham, Markus Eich, Ian Reid and Gordon Wyeth. *Geometrically consistent plane extraction for dense indoor 3D maps segmentation.* In Proc. Intelligent Robots and Systems (IROS), pages 4199–4204, 2016. [40]

[Pintore 2020] Giovanni Pintore, Claudio Mura, Fabio Ganovelli, Lizeth Fuentes-Perez, Renato Pajarola and Enrico Gobbetti. *State-of-the-art in Automatic 3D Reconstruction of Structured Indoor Environments.* Computer Graphics Forum, vol. 39, no. 2, pages 667–699, 2020. [6]

[Poppinga 2008] Jann Poppinga, Narunas Vaskevicius, Andreas Birk and Kaustubh Pathak. *Fast Plane Detection and Polygonalization in Noisy 3D Range Images.* In Proc. Intelligent Robots and Systems (IROS), pages 3378–3383, 2008. [39]

[Pottmann 2007] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai and Shi-Min Hu. *Principal curvatures from the integral invariant viewpoint.* Computer Aided Geometric Design, vol. 24, no. 8, pages 428 – 442, 2007. Discrete Differential Geometry. [2, 6, 8, 14, 15, 23, 34, 77, and 95]

[Pottmann 2009] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang and Yong-Liang Yang. *Integral invariants for robust geometry processing.* Computer Aided Geometric Design, vol. 26, no. 1, pages 37 – 60, 2009. [14]

[Pratt 1987] Vaughan Pratt. *Direct Least-Squares Fitting of Algebraic Surfaces.* SIGGRAPH Comput. Graph., vol. 21, no. 4, page 145–152, August 1987. [10 and 11]

[Qi 2017a] Charles R Qi, Hao Su, Kaichun Mo and Leonidas J Guibas. *Pointnet: Deep learning on point sets for 3d classification and segmentation.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017. [88]

[Qi 2017b] Charles Ruizhongtai Qi, Li Yi, Hao Su and Leonidas J Guibas. *Pointnet++: Deep hierarchical feature learning on point sets in a metric space.* In Advances in neural information processing systems, pages 5099–5108, 2017. [88]

[Qin 2018] Hongxing Qin, Yi Chen, Yunhai Wang, Xiaoyang Hong, Kangkang Yin and Hui Huang. *Laplace–Beltrami Operator on Point Clouds Based on Anisotropic Voronoi Diagram.* Computer Graphics Forum, vol. 37, no. 6, pages 106–117, 2018. [12]

[Qiu 2014] Rongqi Qiu, Qian-Yi Zhou and Ulrich Neumann. *Pipe-run extraction and reconstruction from point clouds.* In European Conference on Computer Vision (ECCV), pages 17–30. Springer, 2014. [63]

[Rabbani 2005] Tahir Rabbani and Frank Van Den Heuvel. *Efficient hough transform for automatic detection of cylinders in point clouds.* Isprs Wg Iii/3, Iii/4, vol. 3, pages 60–65, 2005. [63]

[Rabbani 2006] Tahir Rabbani, Frank van den Heuvel and George Vosselman. *Segmentation of Point Clouds using Smoothness Constraint.* International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 36, no. 5, pages 248–253, 2006. [39 and 42]

[Reuter 2006] Martin Reuter, Franz-Erich Wolter and Niklas Peinecke. *Laplace–Beltrami spectra as 'Shape-DNA'of surfaces and solids.* Computer-Aided Design, vol. 38, no. 4, pages 342–366, 2006. [12]

[Reuter 2009] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè and Michela Spagnuolo. *Discrete Laplace–Beltrami operators for shape analysis and segmentation.* Computers & Graphics, vol. 33, no. 3, pages 381 – 390, 2009. IEEE International Conference on Shape Modelling and Applications 2009. [12]

[Ridel 2015] Brett Ridel, Gaël Guennebaud, Patrick Reuter and Xavier Granier. *Parabolic-cylindrical moving least squares surfaces.* Computers & Graphics, vol. 51, pages 60 – 66, 2015. International Conference Shape Modeling International. [9]

[Roweis 2000] Sam T Roweis and Lawrence K Saul. *Nonlinear dimensionality reduction by locally linear embedding.* science, vol. 290, no. 5500, pages 2323–2326, 2000. [89]

[Sanchez 2020] Julia Sanchez, Florence Denis, David Coeurjolly, Florent Dupont, Laurent Trassoudaine and Paul Checchin. *Robust normal vector estimation in 3D point clouds through iterative principal component analysis.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 163, pages 18–35, 2020. [8]

[Sander 2001] Pedro V Sander, John Snyder, Steven J Gortler and Hugues Hoppe. *Texture mapping progressive meshes.* In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 409–416, 2001. [82]

[Scheidegger 2005] Carlos E. Scheidegger, Shachar Fleishman and Cláudio T. Silva. *Triangulating Point Set Surfaces with Bounded Error.* In Proceedings of the Third Eurographics Symposium on Geometry Processing, SGP '05, page 63–es, Goslar, DEU, 2005. The Eurographics Association. [11]

[Schnabel 2007] R. Schnabel, R. Wahl and R. Klein. *Efficient RANSAC for Point-Cloud Shape Detection.* Computer Graphics Forum, vol. 26, no. 2, pages 214–226, 2007. [40, 51, 52, and 63]

[Schnabel 2008] Ruwen Schnabel, Raoul Wessel, Roland Wahl and Reinhard Klein. *Shape Recognition in 3D Point-Clouds.* In Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2008. [40]

[Sharma 2009] Avinash Sharma, Radu Horaud, David Knossow and Etienne Von Lavante. *Mesh segmentation using Laplacian eigenvectors and Gaussian mixtures.* In AAAI fall symposium on manifold learning and its applications, pages 50–56. AAAI Press, 2009. [12]

[Sharp 2020] Nicholas Sharp and Keenan Crane. *A Laplacian for Nonmanifold Triangle Meshes.* Computer Graphics Forum, vol. 39, no. 5, pages 69–80, 2020. [12 and 79]

[Shen 2004] Chen Shen, James F. O'Brien and Jonathan R. Shewchuk. *Interpolating and Approximating Implicit Surfaces from Polygon Soup.* In ACM SIGGRAPH 2004 Papers, SIGGRAPH '04, page 896–904, New York, NY, USA, 2004. Association for Computing Machinery. [10]

[Silberman 2012] Nathan Silberman, Derek Hoiem, Pushmeet Kohli and Rob Fergus. *Indoor segmentation and support inference from rgbd images.* In European Conference on Computer Vision (ECCV), pages 746–760. Springer, 2012. [40]

[Sinha 2009] Sudipta Sinha, Drew Steedly and Rick Szeliski. *Piecewise planar stereo for image-based rendering.* In 2009 IEEE 12th International Conference on Computer Vision, pages 1881–1888, 2009. [40]

[Sketchfab 2020] Sketchfab. `http://Sketchfab.com`, 2020. [48]

[Sorkine 2002] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal and Dani Lischinski. *Bounded-Distortion Piecewise Mesh Parameterization.* In Proceedings of the Conference on Visualization '02, VIS '02, page 355–362, USA, 2002. IEEE Computer Society. [82]

[Stylianou 2005] Georgios Stylianou and Yiorgos Chrysanthou. *Crest line extraction from point clouds.* GraphiCon 2005 - International Conference on Computer Graphics and Vision, Proceedings, 01 2005. [62]

[Sun 2009] Jian Sun, Maks Ovsjanikov and Leonidas Guibas. *A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion.* Computer Graphics Forum, vol. 28, no. 5, pages 1383–1392, 2009. [12]

[Tagliasacchi 2016] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta and Alexandru Telea. *3D Skeletons: A State-of-the-Art Report.* Computer Graphics Forum, vol. 35, no. 2, pages 573–597, 2016. [63]

[Taubin 1995] Gabriel Taubin. *A signal processing approach to fair surface design.* In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pages 351–358, 1995. [12]

[Tenenbaum 2000] Joshua B Tenenbaum, Vin De Silva and John C Langford. *A global geometric framework for nonlinear dimensionality reduction.* science, vol. 290, no. 5500, pages 2319–2323, 2000. [89]

[Terrell 1992] George R Terrell and David W Scott. *Variable kernel density estimation.* The Annals of Statistics, pages 1236–1265, 1992. [27]

[Tewari 2006] Geetika Tewari, Craig Gotsman and Steven J. Gortler. *Meshing genus-1 point clouds using discrete one-forms.* Computers & Graphics, vol. 30, no. 6, pages 917 – 926, 2006. [80]

[The CGAL Project 2020] The CGAL Project. CGAL user and reference manual. CGAL Editorial Board, 5.1 edition, 2020. [24 and 47]

[Thomas 2018] Hugues Thomas, François Goulette, Jean-Emmanuel Deschaud and Beatriz Marcotegui. *Semantic classification of 3D point clouds with multiscale spherical neighborhoods.* In 2018 International Conference on 3D Vision (3DV), pages 390–398. IEEE, 2018. [8]

[Thomas 2019] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette and Leonidas J Guibas. *Kpconv: Flexible and deformable convolution for point clouds.* In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 6410–6419, 2019. [88]

[Thompson 2019] E Moscoso Thompson, G Arvanitis, K Moustakas, N Hoang-Xuan, ER Nguyen, M Tran, Thibault Lejemble, Loïc Barthe, Nicolas Mellado, C Romanengo *et al. SHREC'19 track: Feature Curve Extraction on Triangle Meshes.* In 12th EG Workshop 3D Object Retrieval 2019, 2019. [66, 68, and 107]

[Torrente 2018] Maria-Laura Torrente, Silvia Biasotti and Bianca Falcidieno. *Recognition of feature curves on 3D shapes using an algebraic approach to Hough transforms.* Pattern Recognition, vol. 73, pages 111–130, 2018. [62 and 69]

[Turbosquid 2020] Turbosquid. *http://Turbosquid.com*, 2020. [68]

[Tutte 1963] William Thomas Tutte. *How to draw a graph.* Proceedings of the London Mathematical Society, vol. 3, no. 1, pages 743–767, 1963. [79]

[Unnikrishnan 2008] Ranjith Unnikrishnan and Martial Hebert. *Multi-scale interest regions from unorganized point clouds.* In 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 1–8. IEEE, 2008. [13]

[Vallet 2008] B. Vallet and B. Lévy. *Spectral Geometry Processing with Manifold Harmonics.* Computer Graphics Forum, vol. 27, no. 2, pages 251–260, 2008. [12 and 13]

[Vo 2015] Anh-Vu Vo, Linh Truong-Hong, Debra F. Laefer and Michela Bertolotto. *Octree-based region growing for point cloud segmentation.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 104, pages 88–100, 2015. [40]

[Wang 2018a] Yinghui Wang, Huanhuan Zhang, Xiaojuan Ning, Wen Hao, Zhenghao Shi, Minghua Zhao, Hongfang Zhou, Liansheng Sui and Ke Lv. *Ridge-valley-guided sketch-drawing from point clouds.* IEEE Access, vol. 6, pages 13697–13705, 2018. [62]

[Wang 2018b] Yu Wang, Mirela Ben-Chen, Iosif Polterovich and Justin Solomon. *Steklov Spectral Geometry for Extrinsic Shape Analysis.* ACM Trans. Graph., vol. 38, no. 1, December 2018. [13]

[Weber 2010] Christopher Weber, Stefanie Hahmann and Hans Hagen. *Sharp feature detection in point clouds.* In Shape Modeling International (SMI 2010), pages 175–186, Los Alamitos, CA, USA, jun 2010. IEEE Computer Society. [62]

[Weinkauf 2009] T. Weinkauf and D. Günther. *Separatrix Persistence: Extraction of Salient Edges on Surfaces Using Topological Methods.* Computer Graphics Forum, vol. 28, no. 5, pages 1519–1528, 2009. [62]

[Witkin 1987] Andrew P. Witkin. *Scale-space filtering.* In Readings in Computer Vision, pages 329 – 332. Morgan Kaufmann, San Francisco (CA), 1987. [2, 13, 26, and 38]

[Yan 2015] Dong-Ming Yan, Jian-Wei Guo, Bin Wang, Xiao-Peng Zhang and Peter Wonka. *A survey of blue-noise sampling and its applications.* Journal of Computer Science and Technology, vol. 30, no. 3, pages 439–452, 2015. [30]

[Yang 2006] Yong-Liang Yang, Yu-Kun Lai, Shi-Min Hu and Helmut Pottmann. *Robust Principal Curvatures on Multiple Scales.* In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, page 223–226, Goslar, DEU, 2006. The Eurographics Association. [26]

[Yang 2007] Pinghai Yang and Xiaoping Qian. *Direct Computing of Surface Curvatures for Point-Set Surfaces.* SPBG, vol. 7, pages 29–36, 2007. [11]

[Yoshizawa 2005] Shin Yoshizawa, Alexander Belyaev and Hans-Peter Seidel. *Fast and robust detection of crest lines on meshes.* In Proceedings of the 2005 ACM symposium on Solid and physical modeling, pages 227–232, 2005. [62]

[Yu 2011] Jin Yu, Tat-Jun Chin and David Suter. *A global optimization approach to robust multi-model fitting.* In Proc. Computer Vision and Pattern Recognition, pages 2041–2048, 2011. [40]

[Yu 2018] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or and Pheng-Ann Heng. *EC-Net: an edge-aware point set consolidation network.* In Proceedings of the European Conference on Computer Vision (ECCV), pages 386–402, 2018. [62]

[Zhang 2004] Zhenyue Zhang and Hongyuan Zha. *Principal manifolds and nonlinear dimensionality reduction via tangent space alignment.* SIAM journal on scientific computing, vol. 26, no. 1, pages 313–338, 2004. [89]

[Zhang 2010] H. Zhang, O. Van Kaick and R. Dyer. *Spectral Mesh Processing.* Computer Graphics Forum, vol. 29, no. 6, pages 1865–1894, 2010. [12]

[Zhao 2020] Hui Zhao, Kehua Su, Chenchen Li, Boyu Zhang, Lei Yang, Na Lei, Xiaoling Wang, Steven J. Gortler and Xianfeng Gu. *Mesh Parametrization Driven by Unit Normal Flow.* Computer Graphics Forum, vol. 39, no. 1, pages 34–49, 2020. [80]

[Zhou 2015] Yang Zhou, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong and Daniel Cohen-Or. *Generalized cylinder decomposition.* ACM Trans. Graph., vol. 34, no. 6, pages 171–1, 2015. [63]

[Zhuang 2017] Yixin Zhuang, Hang Dou, Nathan Carr and Tao Ju. *Feature-aligned segmentation using correlation clustering.* Computational Visual Media, vol. 3, no. 2, pages 147–160, 2017. [60]

[Zokai 2005] Siavash Zokai and George Wolberg. *Image registration using log-polar mappings for recovery of large-scale similarity and projective transformations.* IEEE Transactions on Image Processing, vol. 14, no. 10, pages 1422–1434, 2005. [27]

[Zwicker 2002] Matthias Zwicker, Mark Pauly, Oliver Knoll and Markus Gross. *Pointshop 3D: An Interactive System for Point-Based Surface Editing.* ACM Trans. Graph., vol. 21, no. 3, page 322–329, July 2002. [11]

[Zwicker 2004] Matthias Zwicker and Craig Gotsman. *Meshing Point Clouds Using Spherical Parameterization.* In SPBG, pages 173–180, 2004. [79]

Abstract

3D acquisition techniques like photogrammetry and laser scanning are commonly used in numerous fields such as reverse engineering, archeology, robotics and urban planning. The main objective is to get virtual versions of real objects in order to visualize, analyze and process them easily. Acquisition techniques become more and more powerful and affordable which creates important needs to process efficiently the resulting various and massive 3D data.

Data are usually obtained in the form of unstructured 3D point cloud sampling the scanned surface. Traditional signal processing methods cannot be directly applied due to the lack of spatial parametrization. Points are only represented by their 3D coordinates without any particular order.

This thesis focuses on the notion of scale of analysis defined by the size of the neighborhood used to locally characterize the point-sampled surface. The analysis at different scales enables to consider various shapes which increases the analysis pertinence and the robustness to acquired data imperfections.

We first present some theoretical and practical results on curvature estimation adapted to a multi-scale and multi-resolution representation of point clouds. They are used to develop multi-scale algorithms for the recognition of planar and anisotropic shapes such as cylinders and feature curves. Finally, we propose to compute a global 2D parametrization of the underlying surface directly from the 3D unstructured point cloud.

Résumé

Les techniques d'acquisition numérique 3D comme la photogrammétrie ou les scanners laser sont couramment utilisées dans de nombreux domaines d'applications tels que l'ingénierie inverse, l'archéologie, la robotique, ou l'urbanisme. Le principal objectif est d'obtenir des versions virtuels d'objets réels afin de les visualiser, analyser et traiter plus facilement. Ces techniques d'acquisition deviennent de plus en plus performantes et accessibles, créant un besoin important de traitement efficace des données 3D variées et massives qui en résultent.

Les données sont souvent obtenues sont sous la forme de nuage de points 3D non-structurés qui échantillonnent la surface scannée. Les méthodes traditionnelles de traitement du signal ne peuvent alors s'appliquer directement par manque de paramétrisation spatiale, les points étant explicités par leur coordonnées 3D, sans ordre particulier.

Dans cette thèse nous nous focalisons sur la notion d'échelle d'analyse qui est définie par la taille du voisinage utilisé pour caractériser localement la surface échantillonnée. L'analyse à différentes échelles permet de considérer des formes variées et ainsi rendre l'analyse plus pertinente et plus robuste aux imperfections des données acquises.

Nous présentons d'abord des résultats théoriques et pratiques sur l'estimation de courbure adaptée à une représentation multi-échelle et multi-résolution de nuage de points. Nous les utilisons pour développer des algorithmes multi-échelle de reconnaissance de formes planaires et anisotropes comme les cylindres et les lignes caractéristiques. Enfin, nous proposons de calculer une paramétrisation 2D globale de la surface sous-jacente directement à partir de son nuage de points 3D non-structurés.