Progressive Shape Reconstruction from Raw 3D Point Clouds

Tong Zhao

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Reconstruction progressive de formes à partir de nuages de points 3D

Tong ZHAO

INRIA Sophia-Antipolis Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur

Dirigée par : Pierre ALLIEZ, Directeur de recherche, Inria

Co-dirigée par : Laurent BUSÉ, Directeur de recherche, Inria

Soutenue le : 27 Mars, 2023

Devant le jury, composé de :

Silvia BIASOTTI, Directeur de recherche , CNR-IMATI
Damien ROHMER, Professeur, Ecole Polytechnique (LIX)
Alexandre BOULCH, Ingénieur de recherche, Valeo.ai
Julie DIGNE, Chargée de recherche, CNRS
Géraldine MORIN, Professeur des universités, Toulouse INP
Tamy BOUBEKEUR, Directeur de recherche, Adobe Research
Jean-Marc THIERY, Chargé de recherche, Adobe Research

UNIVERSITÉ CÔTE D'AZUR | 3iA Côte d'Azur Interdisciplinary Institute for Artificial Intelligence

Inria

RECONSTRUCTION PROGRESSIVE DE FORMES À PARTIR DE NUAGES DE POINTS 3D

Progressive Shape Reconstruction from Raw 3D Point Clouds

Tong ZHAO

Jury :

Rapporteurs

Silvia BIASOTTI, Directeur de recherche , CNR-IMATI
Damien ROHMER, Professeur, Ecole Polytechnique (LIX)

Examinateurs

Alexandre BOULCH, Ingénieur de recherche, Valeo.ai
Julie DIGNE, Chargée de recherche, CNRS
Géraldine MORIN, Professeur des universités, Toulouse INP

Directeur de thèse

Pierre ALLIEZ, Directeur de recherche, Inria

Co-directeur de thèse

Laurent BUSÉ, Directeur de recherche, Inria

Membres invités

Tamy BOUBEKEUR, Directeur de recherche, Adobe Research
Jean-Marc THIERY, Chargé de recherche, Adobe Research

Université Côte d'Azur

This dissertation is dedicated in loving memory of my dear mother Ping Chen and my dear grandmother Huiying Huang. They have shaped me into the person I am today.

Reconstruction progressive de formes à partir de nuages de points 3D

Résumé

Avec l'enthousiasme pour les jumeaux numériques dans divers domaines, la reconstruction de surfaces à partir de nuages de points 3D est de plus en plus nécessaire, tout en faisant face à des défis scientifiques multiformes. Les dispositifs d'acquisition de données à faible coût permettent d'obtenir facilement et rapidement des scans 3D, mais peuvent conduire à des nuages de points 3D avec défauts. De plus, les utilisateurs attendent des approches de reconstruction de surface légères et contrôlables avec des critères personnalisés en fonction de leurs applications. Cette thèse vise à aborder ces questions en apportant plusieurs approches au problème de la reconstruction de surface sous deux aspects : (1) l'analyse de nuages de points, et (2) la reconstruction de surface avec des informations a priori.

Nous proposons d'abord une discrétisation progressive (grossier vers fin) du domaine pour les approches globales implicite, leur permettant de gérer divers défauts. Cette méthode est étendue pour être informée par des primitives détectées dans le nuage de points. Ensuite, nous présentons une approche basée sur l'apprentissage profond qui apprend à détecter et à consolider des points appartenant aux singularités (coins, arêtes vives) dans le nuage de points 3D. Ces point singuliers sont ensuite considérés comme une entrée supplémentaire à la méthode de reconstruction progressive. Enfin, nous apportons une méthode de reconstruction variationnelle à partir de nuages de points non-orientés, tirant parti à la fois des métriques d'erreur quadratiques et du clustering via partitionnement variationnel. Le maillage triangulaire de sortie est concis et anisotrope, avec des sommets de maillage répartis sur les arêtes vives.

En résumé, cette thèse vise à contribuer des approches efficientes pour la reconstruction de surfaces dans une perspective globale et progressive. En combinant plusieurs a priori et en prenant en compte différents critères, les méthodes proposées peuvent traiter divers défauts et détails à plusieurs échelles.

Mots-clés : Reconstruction de surfaces, Reconstruction globale, Reconstruction lisse par morceaux, Nuages de points avec défauts, Quadriques d'erreur, Réseaux de neurones.

Progressive Shape Reconstruction from Raw 3D Point Clouds

Abstract

With the enthusiasm for digital twins in various domains, surface reconstruction from raw 3D point clouds is increasingly demanded while facing multifaceted challenges. Low cost data acquisition devices make it possible to obtain 3D scans easily and quickly, but may lead to defect-laden point clouds. In addition, users expect lightweight and controllable surface reconstruction approaches with personalized criteria depending on their applications. This thesis aims at addressing these issues by contributing several approaches to the problem of surface reconstruction from two aspects: (1) point cloud comprehension and (2) surface reconstruction with a priori information.

Concretely, we first propose the notion of progressive discrete domain for global implicit reconstruction approaches that refines and optimizes a discrete 3D domain in accordance with both input and output, and to user-defined criteria. Based on such a domain discretization, we devise a progressive primitive-aware surface reconstruction approach with the capacity to refine the implicit function and its representation, in which the most ill-posed parts of the reconstruction problem are postponed to later stages of the reconstruction, and where the fine geometric details are resolved after discovering the topology. Secondly, we contribute a deep learning-based approach that learns to detect and consolidate sharp feature points on raw 3D point clouds, whose results can be taken as additional inputs to consolidate sharp features for the previous reconstruction approach. Finally, we contribute a variational shape reconstruction method from unoriented point clouds, leveraging both quadric error metrics (QEM) and clustering through variational partitioning. The output triangle mesh is concise and anisotropic, with mesh vertices equidistributed along sharp creases.

In summary, this thesis seeks effective surface reconstruction from a global and progressive perspective. By combining multiple priors and designing meaningful criteria, the contributed approaches can deal with various defects and multi-scale features.

Keywords: Surface reconstruction, Global and piecewise-smooth surface reconstruction, Defect-laden point clouds, Quadric error metrics, Neural networks.

Remerciements

The pursuit of PhD is an adventure. I was always wondering if I could make it happen in the past three years and here the end of this adventure officially comes.

First and foremost, I would like to thank my PhD supervisors : Mr. Pierre Alliez, Mr. Laurent Busé, Mr. Tamy Boubekeur and Mr. Jean-Marc Thiery. My first exposure to the world of computational geometry was in the class of Pierre in 2017. He offered me two internship opportunities and then accepted me as a PhD student, during which we discovered together many interesting topics. He is always creative, proposing plenty of novel ideas during our discussions. Laurent is a mathematician. I have always enjoyed the meetings in which we derived formulas on whiteboards and analyzed the mathematics behind various algorithms. Tamy gave me three months at the beginning of my PhD to read the most classic papers in the domain, that benefited me a lot in my following research. Jean-Marc is always encouraging. He taught me how to find out the way to solve problems properly. He read my codes line by line and verified every details. He was always around when I got into trouble. I extend my gratitude to Mrs. Florence Barbara, the assistant of our team, for preparing all paperwork for me in the last five years.

I would like to express my sincere gratitude to the jury members : Mrs. Silvia Biasotti and Mr. Damien Rohmer for reviewing my manuscript and providing valuable feedbacks ; Mrs. Géraldine Morin for kindly presiding it ; Mrs. Julie Digne and Mr. Alexandre Boulch for kindly examining it. They came to my defense from different cities and different countries, despite the transport strike. Their suggestions deserve careful consideration in my following research.

I want to thank specially Mr. Weidong Qiu for accepting me in his lab when I was a undergraduate student at Shanghai Jiao Tong University. He gave me many valuable advice when I was at the crossroads in my life, leading me to the right path. Another special thank goes to Mr. Mathieu Desbrun, for accepting me as a visiting student at Caltech in 2018. I learnt from Mathieu some basic knowledge of discrete exterior calculus and finite element methods, which are crucial to my research. This great research experience motivated me to pursue my PhD.

I would like to acknowledge my incredible collaborators : Mulin Yu, Mr. Florent Lafatge and Mr. David Cohen-Steiner. We had interesting discussions around many topics and our papers could not have been published without them. I want to thank my colleagues from TITANE team : Mulin Yu, Muxingzi Li, Liuyun Duan, Flora Quilichini, Jean-Philippe Bauchet, Xiao Xiao, Dmitry Anisimov, Rao Fu, Hao Fang, Leman Feng, Nicolas Girard, Cédric Portaneri, Gaétan Bahl, Jean-Dominique Favreau, Onur Tasar, Vincent Vadez, Julien Vuillamy and Daniel Zint. My colleagues from CG Group : Mr. Kiwon Um, Mr. Amal Dev Parakkat, Jiong Chen, Alban Gauthier, Thibault Lescoat, Adrien Kaiser, Elie Michel, Sylvain Rousseau, Corentin Mercier and Chloé Paliard. As

well as my friends : Rudan Xiao, Dingge Liang, Xue Bai, Xue Yang, Nan Li, Wei Li, Jiayi Wei and many other friends in France, in China or in other countries.

I'm also gratuful to 3IA Côte d'Azur - the Interdisciplinary Institute for Artificial Intelligence for funding my PhD. Many thanks to INRIA Sophia Antipolis Méditerranée, Télécom Paris and the École Doctorale STIC de Université Côte d'Azur for hosting me.

In the end, I would like to express my deepest gratitude to my family : my mother Ping Chen, my father Xiaofeng Zhao, my husband Wei Cheng, my dog Mousse and my cat Pudding. My mother was my first teacher and my best friend. She began to teach me English when I was six and my English is still not as good as hers even up till now. She taught me a lot about life, about making hard decisions, about doing what is right, and about always standing up for what I believe in. Studying abroad had been her dream since she was young but she didn't make it happen due to several reasons. And that is why I decided to come to France seven years ago. I met Wei five years ago and we decided to get married one year ago. We complete each other in terms of character and I thoroughly enjoy every day we spend together. Without him, I definitely wouldn't be able to complete my PhD. My gratitude finally goes to my dog and my cat for giving me unconditional loves and making every day full of happiness.

Table des matières

CHAPITRE 1

Introduction

1.1 Background

Computer graphics has been prevalently utilized in many areas including manufacturing, computer-aided design, film and video games, and various specialized applications. In recent years, it is attracting an increasing attention especially with the rising computing power, the rapidly evolving display technology, and many novel applications like self-driving, digital twins, smart city and so on.

In auto industry, many companies make huge investment to the research and development of advanced driver-assistance systems and reliable self-driving cars. Although most of the vehicles are equipped with level 1 driving automation and several companies are producing vehicles with level 2 driving automation in the existing market, there are still major scientific challenges for developing higher level driving automation systems. One of the biggest challenges is to reconstruct rapidly the surrounding scenes and understand the captured 2D and 3D information with the minimum device cost and the maximum reliability.

Film industry and game industry have been growing fast in the last decades. “Avatar”, released in 2009, contains already more than 60% Computer-Generated Imagery (CGI) in 24 fps. The original scenes are captured by a special binocular fusion camera system composed of paired cameras and the 3D stereoscopic scenes are generated in post-production. Their motion capture system produces animated 3D characters by deforming 3D models under the control of the key points of the actors. More impressively, the recently released “Avatar : The Way of Water” brings novel technologies for making lifelike 3D animations underwater, which is considered as a milestone in 3D film history (shown in Figure 1.1). The film is produced in 48 fps in order to recover more details, especially in high speed motion. However, people never stop pursuing more realistic 3D animations with more advanced visual effects and higher resolutions.

Figure 1.1 – *Film clips from “Avatar” (left) and “Avatar : The Way of Water” (right)*

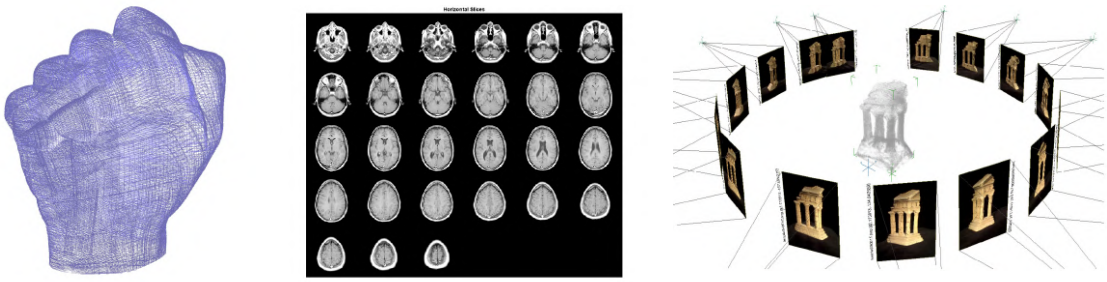
Digital twin, as a virtual conception method proposed in recent years, has also made a significant change in many fields. The target of digital twin is to build virtual representations of intended or actual real-world physical products or systems. In manufacturing industry, digital twins have changed the way they design and maintain industrial products and begins to play a very important

role during the whole product lifecycles. In construction industry, digital twins are able to shorten the design and the construction cycle and reduce costs as well as carbon footprints. In healthcare industry, digital twins are essential to build patient models and datasets, which facilitate the accurate diagnosis and make the remote consultation feasible. Still, there are thus many issues to be solved : How to better model the digital twins ? How to simulate the real processes on the digital twins ? How to interact between the digital twin and the physical twin ?

To sum up, computer graphics is of central importance of many industries. Among all challenges mentioned above, a core challenge that has driven computer graphics is : what is a good way to reconstruct and represent shapes ? And its answer depends on two crucial points : (1) the data acquisition and (2) the various 3D tasks.

1.1.1 Data acquisition

In recent years, abundant 3D scanners are proliferating in our daily life, such as Light Detection And Ranging (LiDAR), industrial computed tomography (CT) scanning, RGB-D sensors, etc. According to Fortune Business Insights [1], the global 3D scanning market size was USD 5.02 billion in 2018 and is projected to reach USD 10.90 billion by 2026, exhibiting a Compound Annual Growth Rate (CAGR) of 10.2%. In addition, 3D scanners are becoming more lightweight and increasingly common. Apple Inc. even equips the latest iPhone models with built-in LiDAR sensors which enable users to create and edit 3D models with their smartphones. In summary, 3D scanners can produce three common data types, as shown in Figure 1.2.



(a) Point cloud (b) 2D slices (c) Multi-view photograph

Figure 1.2 – *Common acquired data representations. (b) Image taken from MathWorks Documentation. (c) Image taken from [FH+15].*

Point cloud. A 3D point cloud is a set of point coordinates in three dimensions. It can contain supplementary information such as colors, normals, intensities, labels, etc. LiDAR is one of the

1. https ://www.fortunebusinessinsights.com/3d-scanning-market-102627

main ways to acquire 3D point clouds. Its non-contact laser emission offers several advantages such as fast measurement speed, high precision, and accurate identification. This makes it a core sensor in many devices, e.g. drones, autonomous cars, robots, and so on. Though point clouds can be used directly for rendering or semantic analysis, they are often converted into meshes for further use in the applications.

2D slices. 2D slices are commonly used in the field of medicine, acquired with e.g. CT, Magnetic resonance imaging (MRI), and X-ray microtomography. They correspond to a stack of 2D slices (i.e., tomograms), the slices giving the x and y coordinates and the slice's serial number indicating the z coordinates. 3D meshes can be reconstructed from such structures to get a deeper understanding and more accurate diagnosis.

Photographs. 3D data can also be acquired and reconstructed from stereo image pairs or RGB-D images without a priori knowledge. Stereoscopic borrows the principle of human eye imaging. Using binocular vision, the depth information can be retrieved from pairs of images given the positions and the focal lengths of the cameras. Reconstruction from images with depth maps highly simplifies the routines of structure from motion (SFM) and can generate high-quality geometric models.

So far, point cloud is one of the most common 3D data representations. Furthermore, 2D slices and photographs are usually converted to point clouds in many cases before feeding to the corresponding 3D algorithms.

1.1.2 3D tasks

3D models have been widely used in various tasks, including animation, physical simulation, modeling and rendering. The requirements of data representations highly rely on the targeted applications.

Animation and simulation. The core objective of both tasks is similar, aiming at modeling and deforming 3D objects given some context. Animation seeks to produce fancy imaginary scenes or objects, while the simulation is designed to restore real-world scenes and objects or simulate physical phenomena. Animation and simulation are powerful tools for many industries such as film production, video games, safety engineering, or training education. One of the most common input data representation for both tasks is 3D mesh, a collection of vertices, edges, and faces that defines the shape of an object or a scene. It can be either a surface or a volume. By discretizing a complex object into well-defined cells (e.g. triangles, quadrangles, tetrahedra, etc.), variables are assigned to each element so that solvers can easily simulate expected behavior.

Modeling. 3D modeling is a powerful tool for developing mathematical coordinate-based representations of any object in three dimensions. It has a wide range of applications in the manufacturing industry, medicine, or gaming. In addition, the popularization of 3D printer also promotes the development of 3D modeling technology. Computer-aided design (CAD) models based on non-uniform rational basis spline (NURBS) surfaces are by far the most common representation for designing 3D models, whereas meshes are widely used for 3D modeling from existing objects or scenes. Meshes are effective in explicitly representing the geometry of a model by describing the relationship between geometric primitives through mesh connectivity. In addition, a wide range of operations is well defined on meshes, such as Boolean logic (Constructive Solid Geometry), editing, smoothing, simplification, subdivision or remeshing.

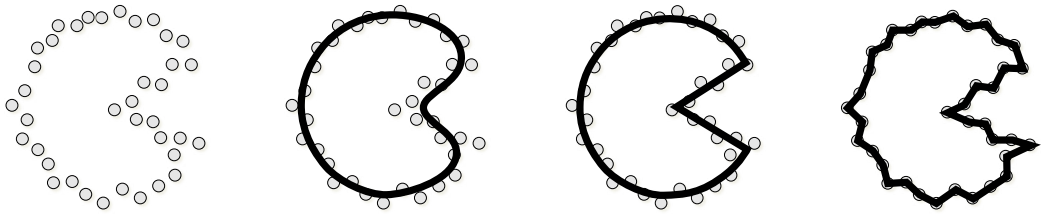
Rendering. Rendering is the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. As the last major step of the graphics pipeline, giving models and animation their final appearance, it is an indispensable technology in many fields. In architectural design, for instance, the architects can make design decisions with the help of 3D rendering, before the building gets physically built. Meshes are an important data source for 3D rendering. In the context of ray tracing-based rendering, their continuity facilitates the computation of intersection between rays and meshes ; the facet normals and orientations make the computation of ray reflections well-defined.

1.1.3 Shape reconstruction problem

3D shape reconstruction solves the problem of converting data acquired from 3D scanners into geometric data structures that can be used in various tasks and applications. This thesis focuses on surface reconstruction from raw 3D point clouds, into triangle surface meshes, by building desired bridges between two data representations.

Point cloud is a subsampling of the real surface of objects, it may suffer from insufficient sampling rates, intrinsic noises of 3D scanners, outliers, missing parts, etc. Although it captures the approximate shapes of objects, it would be unrealistic to recover all details from a discrete representation. In other words, surface reconstruction is an ill-posed problem (see Figure 1.3). Given a point cloud, we can find an infinite number of surfaces that interpolate, i.e. pass through all input points. The approximation is a relevant alternative to interpolation when the input point set is defect-laden. In addition, regularization criteria must be chosen with care for dealing gracefully with the defects, so-called priors. Some assumptions are made before solving the reconstruction problem so as to regularize the missing information of the shapes. Common priors include smoothness prior, primitive prior, data-driven prior and visibility prior.

Triangle mesh is one of the most widely used mesh structures for discretizing 3D shapes. It comprises a set of triangles that are connected by their common edges or vertices. Compared to



(a) Raw point cloud (b) Global smooth reconstruction (c) Piecewise smooth reconstruction (d) Interpolated reconstruction

Figure 1.3 – *The ill-posed nature of surface reconstruction problems. The best solution differs depending on the selected prior and regularization criteria.*

general polygon meshes, triangle meshes are more common due to their simplicity, easing many geometric operators. Secondly, they are easier to generate via many available automatic mesh generators. Thirdly, planar elements facilitate geometric computations.

1.2 General state of the art

As mentioned above, surface reconstruction are guided by various priors. In what follows, we provide an overview of surface reconstruction methods using smoothness priors, primitive priors and data-driven priors. We refer the reader to recent surveys [BTS+14, BTS+17, GXW18, JJC20, HWW+22b] for more details.

1.2.1 Smoothness-based methods

The smoothness prior, including local surface smoothness prior, global surface smoothness prior, and piecewise smoothness prior, produces surface approximations satisfying a certain level of smoothness. It is less limited especially when the input data are non-uniform, incomplete or noisy.

1.2.1.1 Surface representation

Smoothness-based methods often use implicit functions to represent reconstructed surfaces. The latter is defined as the zero-level set of a scalar-valued function defined in Euclidean space : $F : \mathbb{R}^3 \to \mathbb{R}$, i.e. $\mathcal{S} = \{x \in \mathbb{R}^3 | F(x) = 0\}$. Implicit surfaces offer several advantages :

- They allow fast checking of the position of a given point. The function sign provides information on whether it is inside or outside the reconstructed surface.
- One can perform easily local modifications and Boolean operations on implicit surfaces.

- The topology changes can be performed efficiently by modifying the function values at critical points.

1.2.1.2 Discretization domain

In most cases, implicit functions are defined as a sum of basis functions with compact support, thus a discretization domain must be defined at the beginning of the reconstruction process. The simplest choice is to associate the basis functions with the input points. It is widely used in local smoothness-based methods thanks to its efficiency. However, it is hard to adapt in global smoothness-based methods because the complexity of global systems can be extremely high when the number of input points increases.

Octrees are commonly used in global methods. They are sufficiently scalable to large point clouds and their hierarchical structure facilitates the application of multi-grid solvers. Still, there are several limitations of octree-based discretization :

- Beyond its bounding box, the main parameter of an octree, i.e. the maximum depth of the tree, is not directly related to a meaningful criterion that decides the reconstruction accuracy, such as a geometric tolerance error. It is thus not obvious for a user to select a proper depth for a given input point cloud.
- An octree is a data structure made up of cubes aligned to a Cartesian coordinate system. Such an alignment and cube-based isotropy in general do not match the geometry of a given point cloud.
- The peak memory of an octree is hard to control. For instance, an octree of depth $n + 1$ may consume much more memory than one of depth n, and we cannot predict or choose the maximum memory consumption.

Another common choice for global methods is the 3D Delaunay triangulation, whose vertices satisfy the empty sphere property, i.e. the circumsphere of any tetrahedron does not contain any other vertex of the triangulation. Delaunay triangulations are easier to control by adding or removing vertices from the triangulation. In addition, they are not axis-aligned as octrees, and can thus be aligned to any local feature. Compared to octree-based methods, Delaunay-based methods differ mostly in the following aspects : (1) the choice of the basis function, (2) the discretization of the operators, and (3) the solvers.

Basis function. The choice of the basis functions has an important influence on the smoothness of the implicit functions. On a discretized octree, each node is associated with a centered unit-integral basis function with compact support. A common choice is the second-order B-spline basis.

Equation (1.1) provides its definition in $\mathbb{R}^1$ (see Figure 1.4) :

$$\Phi^2(s) = \begin{cases} 1.125 + 1.5s + 0.5s^2 & \text{if } s \in [-1.5, -0.5] \\ -s^2 + 0.75 & \text{if } s \in [-0.5, 0.5] \\ 1.125 - 1.5s + 0.5s^2 & \text{if } s \in [0.5, 1.5] \end{cases} \tag{1.1}$$

and Equation (1.2) provides its definition in $\mathbb{R}^3$:

$$\Phi_o^2(q) = \Phi^2(\frac{q^x - c_o^x}{w_o}) \cdot \Phi^2(\frac{q^y - c_o^y}{w_o}) \cdot \Phi^2(\frac{q^z - c_o^z}{w_o}), \tag{1.2}$$

where o denotes the node, c_o denotes the node center and w_o denotes the node width. As a result, the implicit function at a given query position depends not only on nodes of all depths containing this query, but also nodes of all depths whose supports include this query.

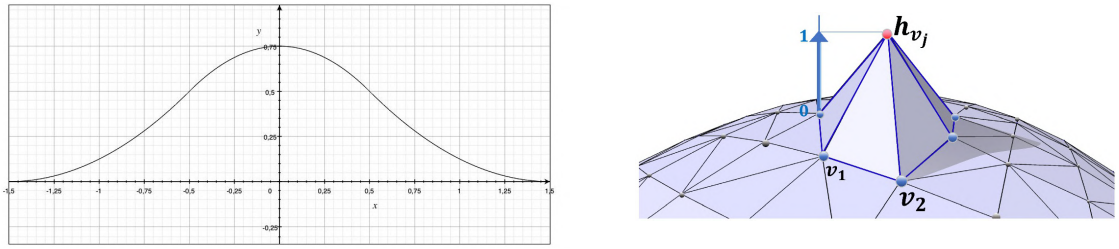


Figure 1.4 – *Basis functions. Left : second-order B-spline function in 1D. Right : hat function on a 2D triangulation (image taken from [NNZ21]).*

On a 3D triangulation, the most common choice is the piecewise linear hat function defined at the triangulation vertices (see Figure 1.4). It has a value of 1 on the central vertex and a value of 0 on its 1-ring neighbors. Given a query position, its function value is the barycentric interpolation of the basis functions of the four vertices of the tetrahedron containing this query. Although higher-order basis functions can be introduced by adding control points, it highly increases the system and meshing complexity. Compared to the basis functions designed for octrees, the ones designed for triangulations are less smooth and are more sensitive to the quality of the tetrahedra.

Operator discretization. Operators on octrees are discretized by using the finite element method (FEM), which considers dot products between pairs of functions over the finite elements. Operators on triangulations can either be discretized by using FEM or using discrete exterior calculus (DEC). DEC-based methods [DHLM05] discretize operators using simplicial chains and cochains, requiring dual Voronoi partitioning. Alexa et al. [AHKSH20] compared both discretizations of the Laplacian operator on 3D triangulations.

Solvers. Several linear solvers are proposed to accelerate the computation on octrees. They benefit from the fact that (1) Octree is a hierarchical structure, (2) Octree nodes have regular shapes, and (3) Efficient data structures facilitate neighbor lookup. Kazhdan et al. [KBH06, KH13, KH19] proposed successively the log-linear cascadic, linear cascadic, and V-cycle solvers. The latter is general-purpose and both time- and memory-efficient.

In general, solvers for 3D triangulations are less efficient than the ones for octrees. Conjugate Gradient solver is a good iterative solver for large-scale linear systems, while its convergence is highly dependent on the condition number of the system. Several preconditioners have been proposed to deal with this issue, for example, Krishnan et al. [KFS13] proposed an algebraic multigrid methods (AMG) style preconditioner, Chen et al. [CSHD21] proposed a multiscale Cholesky preconditioner.

1.2.1.3 Isosurface meshing

Although ray tracking techniques can be used for extracting the final isosurface once the implicit function is solved, it remains an expensive operation of the whole reconstruction pipeline. As a consequence, more methods adopt tessellation-based methods, where meshing is performed in a discretization domain. Lorensen et al. [LC87] proposed the marching cubes algorithm to perform efficient extraction of an isosurface from an implicit function defined on an octree. Many variants have been proposed in the following years, e.g. Dual Marching Cubes [Nie04] proposed by Nielson et al. which extracts quad patches; Dual Marching Cubes [SW04] proposed by Schaefer, which extracts local feature aligned triangular patches on a grid topologically dual to the octree; Dual contouring [JLSW02] proposed by Ju et al., which contours an octree tagged by Hermite data. Marching tetrahedra [DK91], proposed by Doi et al. is an alternative to marching cubes and it applies not only on triangulations but also on octrees whose nodes can be split into 6 tetrahedra. In addition, it is shown to avoid some ambiguous cases that appear in the marching cubes algorithm.

1.2.1.4 Local smoothness-based reconstruction

Point Set Surfaces (PSS) were introduced by Alexa et al. in 2001 [ABCO+01]. As they approximate the surfaces by moving least squares (MLS) methods with high flexibility and controllability, this approach received a lot of attention. Many more variations have been introduced in the following years : Implicit MLS (IMLS) proposed by Shen et al. in 2004 [SOS05], which integrates constraints over polygons to allow exact interpolations; Algebraic PSS (APSS) proposed by Guennebaud et al. in 2007 [GG07] (see Figure 1.5), which is based on moving least squares fitting of algebraic spheres; and Hermite PPS (HPPS) proposed by Alexa et al. in 2009 [AA09], which allows interpolating normal constraints in a stable way. Recently, Mercier et al. [MLR+22] proposed a method that extended the APSS approach, which approximates algebraic spheres using

non-compact kernels and introduces an adaptive progressive octree refinement scheme driven by the resulting implicit surface. This approaches properly captures the reconstructed geometry even far away from the input samples.

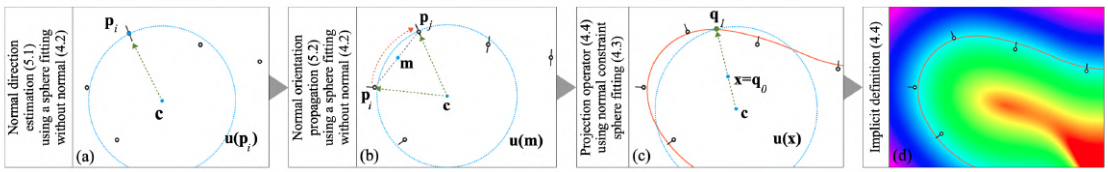


Figure 1.5 – *Algebraic point set surface (APSS). Overview in 2D (Image taken from [GG07]).*

The multi-level Partition of Unity (MPU) was proposed by Ohtake et al. in 2003 [OBA+03]. It is an adaptive method based on local shape functions, partitions of unity, and an octree hierarchy. The octree is constructed following a top-down scheme controlled by a single specified accuracy parameter. In other words, the cells are subdivided recursively until a user-defined error tolerance is met. Fuhrmann et al. [FG14] proposed in 2014 a method extending MPU, called floating scale surface reconstruction, which takes into account both normal and scale information of the input point clouds (see Figure 1.6).

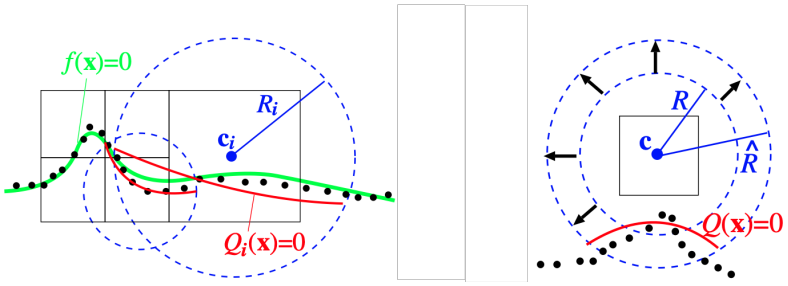


Figure 1.6 – *Multi-level Partition of Unity (MPU). Overview of the approach in 2D (Image taken from [OBA+03]). Left : adaptive subdivision coupled with least-squares fitting. Right : enlarging the spherical domain for the local approximation to make it more robust.*

Local smoothness-based approaches have a good capability to generalize for various shapes and acquisition devices, and are usually both memory and computation efficient. However, they are insufficiently smooth when dealing with severe artifacts. In addition, there is no guarantee that the resulting isosurface is watertight or hole-free.

1.2.1.5 Global smoothness-based reconstruction

Most global smoothness-based methods follow the same process : given a point cloud with oriented or unoriented normals, they discretize a geometric domain on which they solve a global energy optimization problem, and then extract the zero-set of the obtained implicit function to

yield a surface triangle mesh. The existing approaches mostly differ in the way that they (1) define the scalar function with respect to the inferred surface (e.g. smoothed indicator [KBH06], signed distance [CT11, HCJ19], signed robust Wasserstein distance [MdGD+10]), (2) define the objective function (often trading data fidelity for regularity) and (3) solve for it (linear system [KBH06, CT11], generalized eigenvalue solver [ACTD07]).

RBF-based approaches. Among the pioneering approaches, the global smoothness-based methods rely upon globally-supported basis functions discretized either on the input point cloud or on a regular grid. Carr et al. [CBC+01] introduced in 2001 an interpolating method based on radial basis function (RBF). However, it is slow and compute-intensive for large point clouds, and the linear system is badly conditioned due to the application of global supported RBF associated with input points and off-surface points.

Poisson-based approaches. Kazhdan [Kaz05] proposed in 2005 a Fourier series-based method solved on a voxel grid, which consumes a large amount of memory due to the choice of the discretization. Based on their previous method, Kazhdan et al. [KBH06] proposed the popular Poisson surface reconstruction (PSR) method in 2006 (see Figure 1.7), which is still considered as the state-of-that algorithm when seeking high performance surface reconstruction. It is designed to solve for an indicator function whose gradient best fits a vector field defined by the oriented normals of the point cloud. To solve this problem, locally supported basis functions are assigned to an octree discretized by the input point cloud, and then a spatially adaptive multiscale approach is utilized to accelerate and reduce the memory footprint of the solver.

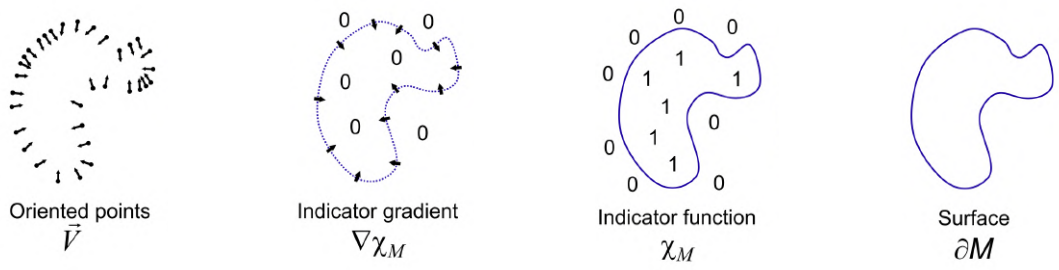


Figure 1.7 – *Poisson reconstruction. An intuitive illustration in 2D (Image taken from [Kaz05]).*

Several improvements based on PSR have been proposed in the following years : Screened Poisson surface reconstruction [KH13] (SPR) which adds a data fitting term that encourages the isosurface to pass through the input points; an adaptive and efficient multi-grid solver [KH19] for solving linear systems on octrees; an improved PSR algorithm that incorporates a constraint envelope of the input point cloud as a Dirichlet constraint within the global Poisson formulation in order to better reconstruct regions with missing data. Despite all the above advantages of the

Poisson-based approaches, they are still limited by the fact that they require oriented point clouds as input and the quality of normals highly impacts the final results.

In order to solve this issue, Peng et al. [PJL+21] introduced a differentiable formulation of PSR allowing backpropagation via neural networks. They optimize simultaneously the point coordinates and corresponding normals by minimizing the Mean Square Error (MSE) loss between the predicted and the ground truth indicator function. The function is discretized on a uniform grid and the Poisson equation is solved using spectral methods to offer a fast GPU implementation. Although this method achieves good performance for small scenes, it does not scale to large scenes because the memory consumption grows cubically with respect to the grid resolution. Recently, Hou et al. [HWW+22a] improved PSR by updating iteratively the vector field of an unoriented input point cloud (see Figure 1.8). They observe that a ray originating from the interior of a solid and pointing to infinity crosses the isosurface an odd number of times. A proper average of the isosurface normals can thus correct the vector field used to define the Poisson equation. This method inherits the scalability and robustness features of PSR. It can deal with both sparse and dense raw point sets, but cannot deal with missing data and sharp features.

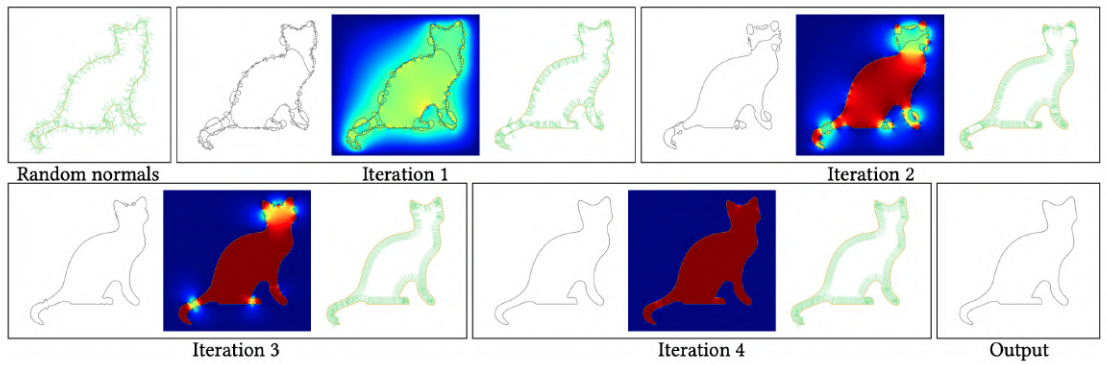


Figure 1.8 – *Iterative Poisson reconstruction. An intuitive illustration in 2D (Image taken from [HWW+22a]).*

Delaunay-based approaches. Several methods are discretized and solved over Delaunay triangulations. Alliez et al. [ACTD07] proposed a spectral surface reconstruction approach that computes an implicit function such that its gradient is best aligned with the principal axes of a covariance tensor field defined by the estimated normals (see Figure 1.9). It is formulated as a maximization of the anisotropic Dirichlet energy (i.e. the Laplacian operator computed in the norm of the tensor field) over the unit ball defined by a linear combination of the Dirichlet energy (i.e. the Laplacian operator) and the biharmonic energy (i.e., the bi-Laplacian operator). This method removes the need of normal orientations, at the cost of solving a generalized eigenvalue problem instead of a simpler linear system. We observe that the bi-Laplacian operator is highly dependent on

the quality of the triangulation, leading to numerical instabilities when the triangulation contains badly-shaped tetrahedra.

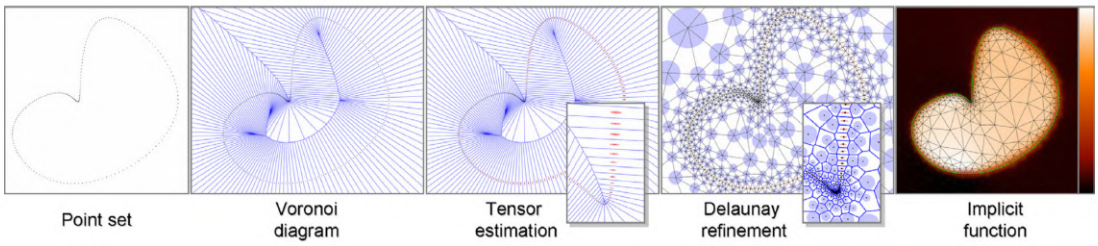


Figure 1.9 – *Spectral surface reconstruction approach in 2D (Image taken from [ACTD07]).*

Mullen et al. [MdGD+10] proposed in 2010 a method that first approximates locally an unsigned distance field from the input unoriented point cloud, before proceeding to a global stochastic signing of the function. It performs an adaptive domain discretization by using an octree only for the initialization, combined with Delaunay refinement before solving. This method offers robustness to unstructured outliers.

Other approaches. Another method quite similar to the Poisson approach is the Smooth Signed Distance surface reconstruction method (SSD) proposed by Calakli et al. [CT11] in 2011 (see Figure 1.10). Instead of formulating the implicit function as an indicator function, SSD seeks an implicit function that approximates a signed distance field of the oriented input point cloud. While PSR encodes the smoothness prior to the divergence of the input normal field, SSD minimizes directly data fitting and normal fitting on the input points, under a soft constraint that encourages the Hessian matrix of the implicit function to have zero norms. The smoothed vector field and divergence operator are no longer needed, which facilitates the operator discretization. SSD follows the same pipeline as PSR and it also benefits from the advantages of a multi-grid solver. It seeks higher-order smoothness than PSR at the price of a longer computation time because the linear system contains much more non-zero values.

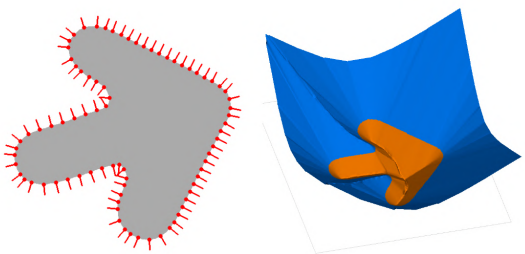


Figure 1.10 – *Smooth signed distance reconstruction in 2D (Image taken from [CT11]). Left : oriented point cloud. Right : signed distance function.*

Variational implicit point set surfaces (VIPSS) have been proposed by Huang et al. [HCJ19] in 2019 (see Figure 1.11). Unlike other MLS-based methods, VIPSS formulates surface reconstruction as a global quadratic optimization problem and seeks an implicit function that minimizes the data fitting term under a soft constraint encouraging the function to have a unit gradient field. Discretization is not needed during optimization but is indispensable for meshing. This method is very well suited to sparse, non-uniform, and unoriented point clouds, but it cannot deal with large point clouds since the memory consumption grows cubically with the number of points.

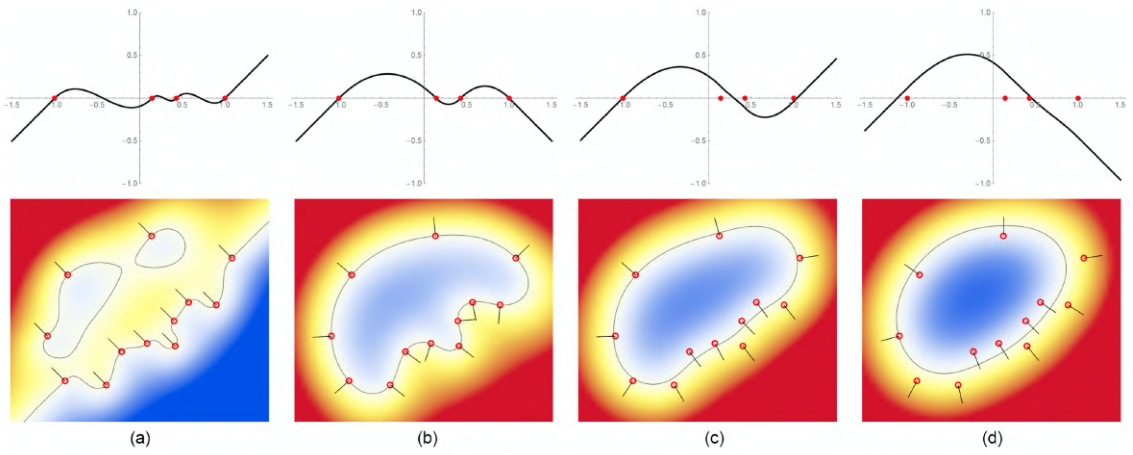


Figure 1.11 – *Variational implicit point set surfaces (VIPSS). Examples of Duchon's interpolants that interpolate scattered points in 1D and 2D for different choices of the Hermite data (Image taken from [HCJ19]).*

1.2.1.6 Piecewise smoothness-based reconstruction

We now classify the popular piecewise smoothness-based reconstruction approaches by distinguishing when the piecewise smoothness prior is introduced, i.e. before, during, or after the reconstruction step.

Before reconstruction. Avron et al. [ASGCO10] proposed in 2010 a two-step $L1$ optimization approach for reconstructing point set surfaces. They first solve a global sparse minimization problem to consolidate normal orientations, which are further used to consolidate point positions. The final surface mesh is extracted using the Ball Pivoting algorithm [BMR+99] from the consolidated point set. Huang et al. [HWG+13] proposed an edge-aware point set resampling technique for upsampling point sets with noise-free normals and preservation of sharp features. The regions away from the edges are resampled, and then the remaining ambiguous areas are resampled progressively to approach the edge singularities. Introducing priors before reconstruction is made flexible,

but the quality of sharp feature recovery is hard to control since it is highly dependent on the selected reconstruction solver.

During reconstruction. Several MLS-based approaches are proposed for piecewise smooth reconstruction with sharp features. The robust moving least-squares approach (RMLS) [FCOS05] introduced by Fleishman et al. in 2015 differs from previous MLS-based approaches in two aspects : (1) The point cloud is segmented into multiple outlier-free smooth regions before reconstruction, and (2) The projection operator projects query points to the closest smooth region or the closest sharp feature depending on their relative locations to the nearby smooth regions. Öztireli et al. [ÖGG09] proposed the Robust implicit moving least-squares variant (RIMLS). It combines the IMLS surface with a robust local kernel regression kernel using Ψ-type M-Estimators, leading to a differentiable MLS surface that better preserves sharp features.

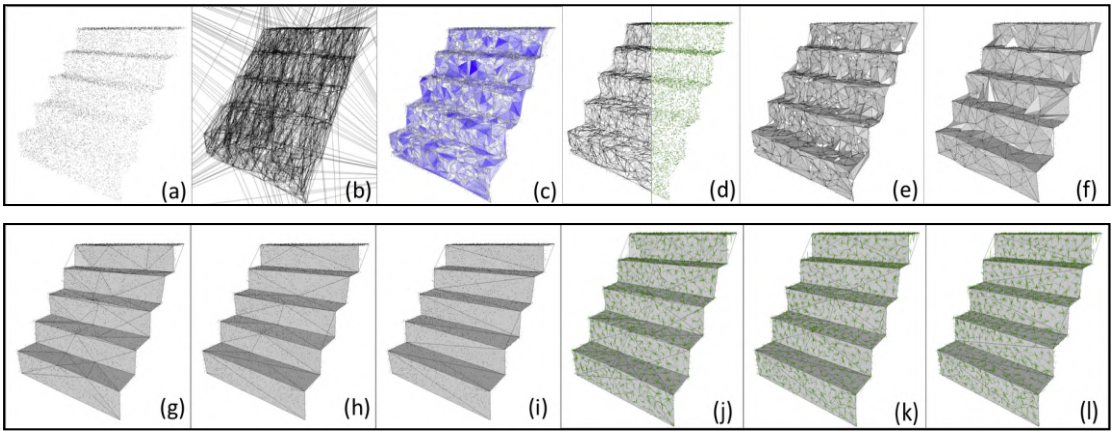


Figure 1.12 – *Feature-preserving surface reconstruction and simplification (Image taken from [DCSA+14]). (a) Point set. (b) Initial 3D Delaunay triangulation. (c) Initial reconstruction surface. (d) Initial transport plan. (e-f) Intermediary decimation steps. (g-i) Reconstruction results with 100, 55 and 22 vertices. (j-l) Final transport plan with 100, 55 and 22 vertices.*

After reconstruction. Some approaches regularize and simplify an initial surface guess with respect to sharp features. Hoppe et al. [HDD+94] proposed a piecewise smooth surface reconstruction approach that takes as input a reconstructed and optimized dense mesh and fits an accurate and concise piecewise-smooth subdivision surface. Wang et al. [WYZC13] pioneered a reconstruction pipeline that first removes outliers and eliminates noises from the input point cloud, then passes the so-consolidated point cloud to PSR. The sharp features are recovered by post-processing of a bilateral filtering of face normals and a normal-guided point position optimization. Digne et al. [DCSA+14] introduced a sharp feature-aware surface reconstruction and simplification ap-

proach, which starts with an initial surface guess and decimates it via an optimal transportation error metric (see Figure 1.12).

1.2.2 Primitive-based methods

Surface reconstruction with primitive priors usually consists of two steps : primitive detection and primitive assembly. They are widely used in large-scale indoor scenes or outdoor scenes reconstruction owing to their good scalability and the low complexity of the output meshes, even if the meshes are in most cases polyhedral surfaces instead of triangular surfaces.

1.2.2.1 Primitive detection

Primitive detection has received considerable attention for more than 40 years, and many approaches have been proposed to solve this problem in various contexts. For more details, we refer to a recent survey authored by Kaiser et al. [KZB19].

The basic geometric primitives include planes, spheres, cylinders, cones and torie. One or more primitive types are detected depending on the design of each approach. The planar primitive is the most common choice, not only because it is easy to estimate in the detection phase, but also because it is efficient to intersect planes for assembling primitives.

Fischler et al. [FB81] introduced in 1981 a paradigm for fitting a model to experimental data, called RANdom SAmple Consensus (RANSAC). The algorithm starts with selecting randomly a minimal set of candidate control points. A hypothesis is formulated according to the candidate set, followed by a score function evaluation that decides whether the candidate is kept or discarded. Schnabel et al. [SWK07] proposed the efficient RANSAC approach that improves the original method by extending RANSAC to all types of primitives and accelerating it during the score evaluation phase. The algorithm is robust against noises and outliers, and it scales well with respect to the size of the input point cloud and the number of detected primitives.

(a) Point Cloud (b) RANSAC (c) Region Growing

Figure 1.13 – *Comparison between RANSAC and Region Growing on a 3D point cloud. (Image taken from [OVJ+22]).*

Rabbani et al. [RVDHV06] proposed a region growing based approach clustering points based on unoriented normals and local connectivity of the points. A local hypothesis grows in a spatial neighborhood of a seed point until no points can be merged under a user-defined normal threshold.

The process stops once all points are segmented. Compared to efficient RANSAC, this approach is less resilient to outliers but provides better quality outputs for large scenes with many small details, at the cost of higher computation time.

Recently, Yu et al. [YL22] proposed an energy-based method that refines an initial planar primitive configuration (obtained from e.g. RANSAC, region growing, etc.) by optimizing an objective function relating to high fidelity, simplicity, and completeness. The said objective function is explored by performing repeatedly five local geometric operators : transfer, exclusion, insertion, merging, and splitting. A greedy search strategy is adopted based on a modifiable priority queue. This approach is shown to be able to largely improve an initial configuration provided by aforementioned primitive detection methods.

1.2.2.2 Primitive assembling

We roughly divide the primitive assembling methods into two types : connectivity-based and partition-based approaches.

Connectivity-based assembling. A popular method proposed by Chen et al. [CC08] first recognizes planar regions based in the input point clouds, then intersects the planes to form creases and corners, before generating closed polyhedra.

Partition-based assembling. Most of the partition-based approaches partition space into a discrete domain, then label each cell as *inside* or *outside*.

Schnabel et al. [SDK09] proposed in 2009 a graph-cut based method to reconstruct and complete surfaces from a set of detected primitives. The graph is constructed on the vertices of a regular grid bounding the input point cloud and the graph cut algorithm is performed iteratively to minimize an objective function trading completeness for fidelity, followed by a violation detection of the cut edges in order to update edge costs in the graph. The output surface is extracted by an extended version of marching cubes which preserves the sharp feature edges deduced from the inside/outside labels. Lafarge et al. [LA13] proposed a hybrid approach that first constructs a structured point set and then reconstructs the hybrid surface by combining consolidated primitives and the remaining unconsolidated points. A novel efficient min-cut formulation discretized on a Delaunay triangulation was designed to combine structure, geometry and visibility in order to guarantee that the output meshes are intersection-free and 2-manifold surfaces. Chauve et al. [CLP10] introduced in 2010 a lightweight method discretized on a 3D space decomposition of polyhedral cell complexes. Bauchet et al. [BL20] proposed an efficient kinetic data structure for partitioning the 3D space into convex polyhedra, that largely reduces the computation cost of previous approaches and improves the quality of the output 3D polyhedral partitions.

1.2.3 Learning-based methods

With the undisputed success and rapid growth of neural networks and deep learning in many domains, more and more learning-based reconstruction approaches using data-driven priors have been proposed in the last five years. Several well-rounded 3D datasets with quality labels (e.g. ModelNet Dataset [CFG+15], ABC Dataset [KMJ+19]) have been published as well, stimulating the use of data-driven priors and enabling new benchmarks and baselines for the surface reconstruction problem. Depending on the output, we can roughly classify the learning-based methods into three types : signed distance function (SDF), occupancy, and explicit mesh.

Signed distance function. Park et al. [PFS+19] proposed DeepSDF in 2019, a learned continuous signed distance function representation from point clouds. They adopted an encoder-decoder structure, which encodes at first the input shape into a 512-dimensional shape code, then the decoder takes both the code and a query position and gives the predicted signed distance value. Though DeepSDF shows interesting results in several 3D tasks, it lacks the capacity of representing the details of shapes. Boulch et al. [BM22] overcomes this issue by computing latent vectors at each input point (see Figure 1.14). The final latent code for a query position is produced by a learning-based interpolation on nearest neighbors. Their approach can represent not only a single object but also large-scale scenes, while it is time and memory-consuming. What's more, there is no guarantee that the implicit function is continuous or differentiable. Liu et al. [LGP+21] proposed DeepIMLS, which predicts an octree as a scaffold for generating MLS points with learned priors. As a result, the implicit function benefits the advantages of MLS and can be applied in differentiable tasks.

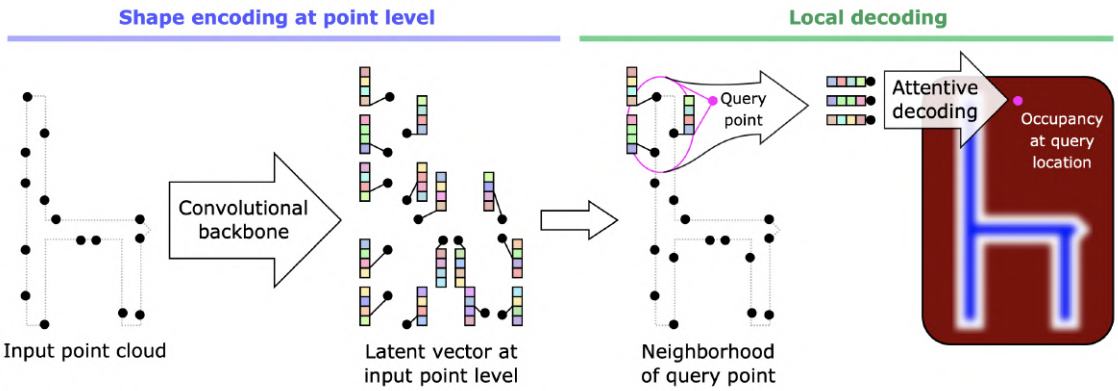


Figure 1.14 – *POCO (Image taken from [BM22]).*

Occupancy. The surface reconstruction problem can be reformulated as an occupancy prediction problem on regular grids or octrees. Several approaches have been proposed since 2017 : Oc-

tree Generating Network [TDB17], O-CNN [WLG+17], Minkovski engine [CGS19], Occupancy network [MON+19], etc. A detailed review is provided in Section 2.5.

Explicit mesh. Some approaches directly predict an explicit mesh as the reconstructed surface. Hanocka et al. [HMGC20] proposed Point2Mesh, which learns to deform iteratively an initial mesh in order to minimize the distance between the input point cloud and the sampled point cloud from the current mesh. Rakotosaona [RGA+21] et al. proposed the Delaunay Surface Element (DSE) reconstruction approach that trains one network for selecting geodesic neighbors from Euclidean neighbor sets and another network for creating log map projection from geodesic neighbors to 2D embeddings (see Figure 1.15). The final mesh is synchronized from Delaunay triangulation surface patches of the learned 2D embeddings. Chen et al. proposed neural marching cubes [CZ21] in 2021 and then neural dual contouring [CTFZ22] in 2022. Taking as input a regular grid with either corner signs (for signed distance field) or edge intersection flags (for unsigned distance field), it predicts for each grid cell a Hermite point for reconstructing the final surface mesh. These methods can also be used for extracting sharp features and open boundaries for both oriented and unoriented point clouds.

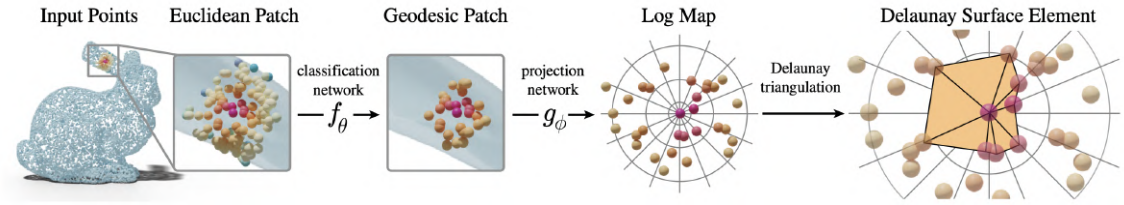


Figure 1.15 – *Delaunay surface element reconstruction (Image taken from [RGA+21]).*

Existing problems. Although many learning-based approaches have been proposed, most of them have one or several limitations :

- The resolution of the final surface is limited by the high memory consumption, hence the scalability of these approaches remains a major issue in real-world applications.
- The method can be overfitted to one or several shape classes in the training dataset, and the generalization to other classes is not easy to achieve, which restricts the applications of the trained models.
- Training and inference can take a long time even on compute-intensive devices and well-annotated datasets are not always available.

1.3 Focus and positioning

We address two main problems in this thesis : (1) How to reconstruct good surfaces from defect-laden point clouds ; (2) How to approximate the surfaces to achieve a satisfactory balance between complexity and accuracy. These problems pose several challenges :

Q1. How to deal with point clouds with latent defects ?

In real-world applications, point clouds scanned by acquisition equipments usually contain various defects such as missing data, noises, outliers, misalignment among devices, etc. Therefore, the reconstruction process has to be carried out under defective information that may lead to inaccurate surfaces, or even outrageous surfaces. There are several reasons. The first one is the popularization of low-cost scanners and sensors. Such devices sacrifice accuracy in order to reduce production costs. The second one is the wide usage of medium-range and long-range scanners, which pay more attention to high-level geometry instead of fine details. The last one is due to the use of handheld laser scanning systems, which introduce more errors during the registration step.

Only very few existing methods can deal with a wide range of defects, thus the choice of a proper reconstruction method for a given defect-laden point cloud is highly empirical and requires domain specific knowledge. A possible solution is to design both **input- and output-aware** reconstruction algorithms instead of being only input-aware. Another way to handle this issue is to introduce **data-driven priors** with sufficient data augmentation.

Q2. How to deal with point clouds with multi-scale features ?

We wish to address the increasing demand for modeling complex objects and large-scale scenes. As a result, reconstruction with multi-scale details must be taken into consideration during the algorithm design phase. A sufficient sampling rate is a key guarantee to reconstructing areas with high curvature and fine details, while the reality is that the sampling rate is rather low in such regions.

Introducing **an appropriate combination of priors** can help solving this issue : smoothness priors guarantee good consistency for free-form parts ; primitive priors reconstruct easily regions with simple canonical primitives, and data-driven priors retrieve information from training datasets for ambiguous regions.

Q3. How to derive meaningful parameters for controlling surface reconstruction algorithms ?

Most reconstruction algorithms require some user-defined controlling parameters. While meaningful parameters (e.g. the ones related to error tolerance or complexity) are widely used in local algorithms, it remains a difficult problem to devise intuitive ones for controlling global

smoothness-based reconstruction algorithms. For instance, the two main parameters of screened Poisson reconstruction are the depth of the octree and the screening weight. SSD reconstruction has an additional Hessian weight to control the smoothness of the final implicit function. All such parameters are not directly related to a meaningful criterion that governs the reconstruction accuracy. It is thus not obvious for a user to select them given a point cloud. In addition, the optimal choices may change for different point clouds.

Distance- and normal-related parameters are good default options for users. Introducing such parameters into global smoothness-based reconstruction algorithms makes the latter more flexible. In addition, using a **parameter field** instead of a single threshold can further improve the controllability of the output surface.

Q4. How to achieve a good trade-off between complexity and accuracy ?

The reconstruction accuracy is highly influenced by the solver quality. The improvement of the quality always depends on more expensive operators. For global smoothness-based methods, dense discretizations play a critical role to ensure good condition numbers and approximate solutions. For local smoothness-based methods, global kernel functions help reconstruct surfaces with enough smoothness. For primitive-based methods, accurate detection of primitives is indispensable for the assembling step. However, such operators are limited by the memory and computing power of the computing devices.

To achieve a good trade-off between complexity and accuracy, we should provide "just enough" degrees of freedom (DoF) during the computation. One way to tackle this issue is to introduce some amount of **progressiveness** into the reconstruction algorithms.

1.4 Contributions

Focusing on surface reconstruction from raw 3D point clouds, we introduce three surface reconstruction approaches suitable for different situations, together with a sharp feature point consolidation approach which can be used as a pre-processing for the proposed reconstruction approaches.

Contribution 1 : Progressive discrete domain for global implicit reconstruction.

Most global smoothness-based reconstruction approaches follow the same pipeline : they discretize a geometric domain on which they solve a global energy optimization problem, and then they extract the zero-set of the obtained implicit function to get a triangle surface mesh. Although many different energies and meshing approaches have been proposed in the past years, the domain discretization step received fewer attention in the literature.

We first underline in this thesis the importance of the domain discretization step in the global implicit reconstruction pipeline. It has a direct influence on the smoothness of the implicit function, the complexity of the global system and the quality of the extracted isosurface. We then devise a progressive coarse-to-fine approach that jointly refines the implicit functions and its representation domain, through iterating solver, optimization and refinement steps applied to a 3D Delaunay triangulation. It can be combined with many global implicit solvers. Our discretized domain has several properties : (1) It is both input and output aware. The domain is denser near the isosurface, and even denser where it exhibits small local feature size. Just enough degrees of freedom are allocated in order to ensure a good complexity-accuracy tradeoff. (2) The domain has a high quality to improve the solver conditioning. (3) The isosurface extracted from the domain also has a high quality.

Our experiments show that our discrete domain can largely improve the reconstruction quality on real scanned point clouds with various defects, such as noises, missing data, variable resolution, or variable feature size.

Contribution 2 : Primitive-guided implicit reconstruction on progressive discrete domain.

Global implicit reconstruction approaches offen struggle to reconstruct canonical primitives. There are two reasons : (1) The global implicit function is a weighted sum of all basis functions with compact supports instead of a simple shape parameterization. (2) Hard constraints cannot be integrated into global solvers on octree-based discretization. For instance, the screened term in PSR is added into the energy function as a soft constraint, as well as the hessian term in SSD. As a result, we can never expect that a Poisson function equals to zero at all input points or a SSD function is a real distance function.

Based on the proposed discrete domain, we explores a progressive implicit reconstruction approach able to consider detected primitive configurations. We leverage an initial primitive detection step to constrain the implicit function on non-ambiguous canonical primitive areas, then interleave global implicit solvers with refinement and optimization of the 3D tetrahedron mesh used to represent an implicit function respecting detected primitives.

Although we are still improving this approach in order to produce high-quality isosurface with smooth transitions between constrained and unconstrained areas, our approach shows already several benefits : (1) Our approach is capable of combining global smoothness prior with primitive prior and it has the possibility to involve data-driven prior to reconstruct sharp features. (2) Our approach explores that global implicit reconstruction approaches can be controlled by known information, which eliminates some defects around ambiguous regions.

Contribution 3 : Sharp feature consolidation via displacement learning

Point clouds, especially those scanned from man-made objects, often contain sharp features. The latter gives crucial semantic information of a shape. However, recovering sharp features during smoothness-based reconstruction is hard. First, the normals of the sharp feature points are ambiguous or ill-posed. Smoothing their normals destroys the sharpness around such region. Second, there are few points sampled exactly on sharp features due to the sampling techniques. Last, introducing smoothness prior further hinders recovering sharp features from raw point clouds.

In this thesis, we present a deep learning-based network that learns to detect and consolidate sharp feature points on raw 3D point clouds. We devise a multi-task neural network architecture that identifies points near sharp features and predicts displacement vectors toward the local sharp features. The so-detected points are thus consolidated via relocation. A point-to-feature oracle is designed to improve the data augmentation quality so as to prevent our models from overfitting.

Our proposed approach is noise-resilient and flexible. It outperforms the state-of-the-art approaches both quantitatively and qualitatively on the popular ABC dataset. It is also a powerful preprocessing tool for surface reconstruction approaches. We are working on introducing it into our progressive implicit reconstruction pipeline as data-driven prior.

Contribution 4 : Coarse-to-fine piecewise smooth reconstruction via clustering quadric error metrics.

At the end of the thesis, we tackle the hard problem of reconstructing piecewise smooth surfaces with sharp features from raw 3D point clouds. Most previous piecewise smooth-based reconstruction approaches solve it by treating three separate sub-tasks in a certain order : (1) separating sharp features from smooth patches, (2) reconstructing smooth patches and (3) stitching them with sharp features. Although it facilitates the algorithm design, it does not take into account the interaction among the sub-tasks and thus either leads to defective results or increases the algorithm complexity.

We devise a coarse-to-fine approach which reconstructs and simplifies simultaneously a piecewise smooth surface from an unoriented raw 3D point cloud via clustering quadric error metrics (QEM). Local geometries are captured by point cloud QEM estimations and the points are clustered via a variational shape approximation-type region growing process. An oriented mesh is extracted using a binary integer problem solver from a connectivity graph guided by the adjacency of the clusters.

The proposed approach is evaluated on challenging point clouds both qualitatively and quantitatively. It is shown that our approach produces oriented output meshes whose densities are adapted to local feature sizes and local curvatures with just enough degree of freedoms. It is computationally efficient and is scalable to large point clouds.

1.5 Thesis overview

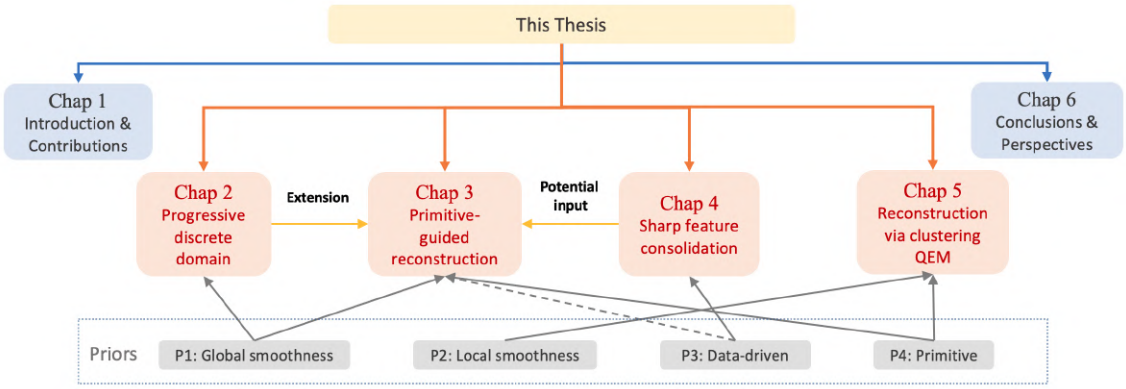


Figure 1.16 – *Outline of the thesis.*

The overview of this thesis is shown in Figure 1.16, and its parts are organized as follows :

Chapter 2 presents our first contribution on progressive discrete domains for global implicit reconstruction methods.

Chapter 3 presents our current progress on primitive-guided implicit reconstruction on progressive discrete domains.

Chapter 4 presents the proposed deep learning framework for sharp feature consolidation.

Chapter 5 presents our fourth contribution on coarse-to-fine mesh reconstruction via clustering quadric error metrics.

Chapter 6 provides conclusive remarks on all contributions of this thesis. We also discuss future work and potential breaking points in these research directions.

1.6 Publications

[ZAB+21] Tong Zhao, Pierre Alliez, Tamy Boubekeur, Laurent Busé, Jean-Marc Thiery. Progressive Discrete Domains for Implicit Surface Reconstruction. Computer Graphics Forum 40 : 143-156, 2021.

[ZYAL23] Tong Zhao, Mulin Yu, Pierre Alliez, Florent Lafarge. Sharp Feature Consolidation from Raw 3D Point Clouds via Displacement Learning. Computer Aided Geometric Design : 102204, 2023.

[ZBCS+23] Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, Pierre Alliez. Variational Shape Reconstruction via Quadric Error Metrics. ACM SIGGRAPH Conference Proceedings : 2023.

CHAPITRE 2

Progressive discrete domains for global implicit reconstruction methods

Many global implicit surface reconstruction algorithms formulate the reconstruction problem as a volumetric energy minimization, trading data fitting for geometric regularization. As a result, the output surfaces may be located arbitrarily far away from the input samples. This is amplified when considering (i) strong regularization terms, (ii) sparsely distributed samples or (iii) missing data. This breaks the strong assumption commonly used by popular octree-based and triangulation-based approaches that the output surface should be located near the input samples. As these approaches refine, during a pre-process, their cells near the input samples, the implicit solver deals with a domain discretization not fully adapted to the final isosurface. We relax this assumption and devise a progressive coarse-to-fine approach that jointly refines the implicit function and its representation domain, through iterating three steps (solver, optimization and refinement) applied to a 3D Delaunay triangulation. There are several advantages to this approach : the discretized domain is adapted near the isosurface and optimized to improve both the solver conditioning and the quality of the output surface mesh contoured via marching tetrahedra. At the end of this chapter, we also describe an attempt for discretizing an octree from a defect-laden point cloud, via a deep learning-based approach. We then analyze the remaining problems and describe some directions to improve this approach in a future work.

2.1 Introduction and related work

Assuming input measurement data provided as an unorganized 3D point set, surface reconstruction is the process of recovering shapes or entire scenes that fit these data, while dealing with defect-laden or missing data. The reconstruction problem is inherently ill-posed as an infinite number of shapes may fit the data. The common wisdom consists of reducing the search space, i.e. regularizing the problem via adding predetermined priors deriving from assumptions about geometry, semantics, acquisition or structure. A geometric prior may relate to trivial topology, absence of boundaries, canonical shape primitives or smoothness. Furthermore, data fitting is only half the problem, as satisfactory complexity-distortion tradeoffs are also sought after. In addition, other properties are desirable for downstream use of the reconstructed discretized surfaces (often triangle meshes), such as well-shaped elements and adaptive density.

Global implicit surface reconstruction methods commonly hinge upon global solvers, possibly multi-scale. These solvers, commonly tailored to discrete differential operators, yield implicit functions that trade data fidelity for geometric prior matching, where the prior commonly favors closed smooth surfaces. Such global solvers require discretizing both the 3D domains where the implicit function is defined and the aforementioned operators, and contouring the implicit function to extract the final meshes. Ideally, the above discretization should provide (1) just-enough degrees of freedom near the reconstructed surface for the solver, (2) well-shaped volumetric elements everywhere to ensure good numerical conditioning for the solver, (3) geometric regularity when inferring smooth surfaces, and (4) well-shaped elements of the output surface mesh.

2.1.1 Previous work

As surface reconstruction approaches are already reviewed in Section 1.2, we restrict next our review to implicit global reconstruction methods with an emphasis on discretization issues. As our progressive approach utilizes tetrahedron mesh optimization and refinement principles, we also review the related work.

Implicit surface reconstruction. The type of domain discretization used for representing the implicit function differs for each method. The popular Poisson reconstruction method and its variants [KBH06, KH13, KCRH20] refine an octree before solving for the implicit function. It utilizes smooth basis functions defined on the octree elements, and requires diffusing the input normals in order to compute the divergence operator. On areas with missing data, such approaches can generate spurious surfaces and coarse output meshes where the cells of the octree are not refined. A recent variant [KCRH20] adds a close envelope and constraints the solver to generate level sets only inside this envelope, which significantly reduces the undesired spurious surfaces, while the envelope is not always available in real applications. The Smooth Signed Distance (SSD) reconstruction approach proposed by Calakli and Taubin [CT11] also utilizes an octree before sol-

ving, but does not require diffusing the normals or discretizing the divergence operator. A spectral approach [ACTD07] removes the need for oriented point sets as input points, at the cost of solving for a generalized eigenvalue problem instead of a simpler linear system. It uses a 3D Delaunay triangulation as domain discretization, refined before solving as in previous work. The "signing-the-unsigned" approach [MdGD+10] signs an outlier-robust distance function. Adaptive domain discretization is performed by using an octree only for the initialization, combined with Delaunay refinement before solving.

In summary, most common implicit-based approaches rely upon a predetermined domain discretization : they determine a data structure (often an axis-aligned bounding volume hierarchy such as an octree) from the local density of input samples, a resolution for the reconstructed surfaces, and then utilize a global solver to infer an implicit surface. Refining the data structure *a priori*, and only near the input samples, can either overlook areas where the solver completes missing data or over-refines where the reconstructed surface is flat. In addition, the axis-aligned nature of the above data structure is too rigid : it produces output meshes that may contain badly shaped elements and the overall approach is not intrinsically invariant to rotations. We are thus left with a chicken-and-egg problem, as knowing the reconstructed surface requires solving, and ideal solving conditions require knowing the final reconstructed surface. The difficulty of sequencing actions where each seems to depend on others calls for a progressive approach in which the solver is used iteratively, as a means of consolidating hypotheses emitted in previous iterations, and interleaved with (isosurface-driven) optimization and refinement of the discretization. Several approaches already proceed coarse-to-fine during reconstruction (e.g., [OBA+03, SLS+06] to cite a few), but do not jointly refine and optimize both the implicit function and its representation domain. Optimization herein should be understood as adapting the domain geometry around the isosurface.

Tetrahedron mesh optimization. Quality meshes commonly refer to tetrahedron elements with controlled size and shape, which offer both accuracy and conditioning for discrete operators such as Laplacian, Hessian, or divergence [She02]. Some approaches optimize via local topological transformations [LCS09, CZZ+17]. Other approaches relocate the vertices, either locally [JWZ11] or globally [VWP13]. Klingner and Shewchuk use an even broader repertoire of mesh transformations [Kli08]. These approaches are insufficient for our specific context where both the discretized domain (shape of tetrahedra) and the discretized isosurface (shape of triangles extracted via marching tetrahedra) must be optimized jointly.

Tetrahedron mesh refinement. Mesh refinement is a relevant paradigm for improving the complexity-distortion tradeoff of finite element simulations [BKK20]. Mesh adaptation requires estimating the simulation error [GBA+17]. Delaunay refinement is a popular greedy refinement method for isotropic mesh generation and shape approximation [CDS13, Si15]. The Delaunay re-

finement paradigm is also relevant for discretizing the 3D domain in our context, by inserting circumsphere centers of tetrahedra. However, we also need a specialized refinement scheme around the inferred surface (discretized by marching tetrahedra), tailored to preserve the shape of isosurface triangles.

2.1.2 Positioning and contributions

Departing from most implicit surface reconstruction methods that discretize the domain *a priori* based on the input data, we propose an output-sensitive progressive coarse-to-fine approach that jointly refines the implicit function and its representation domain, while discovering its isosurface (see Figure 2.1). The motivation for such an approach stems from the observation that the global surface reconstruction problem is two-fold : the domain should be adapted both for the solver to capture adequately the variations of the solution as a scalar field ($\{f(x), x \in \mathbb{R}^3\}$) and then specifically around an isosurface to be extracted (without loss of generality, $S = f^{-1}(0)$). Dedicated refinement and optimization approaches must be developed for these two entangled sub-problems. Our method is intended to deal with implicit reconstruction methods that can be discretized on tetrahedron meshes.

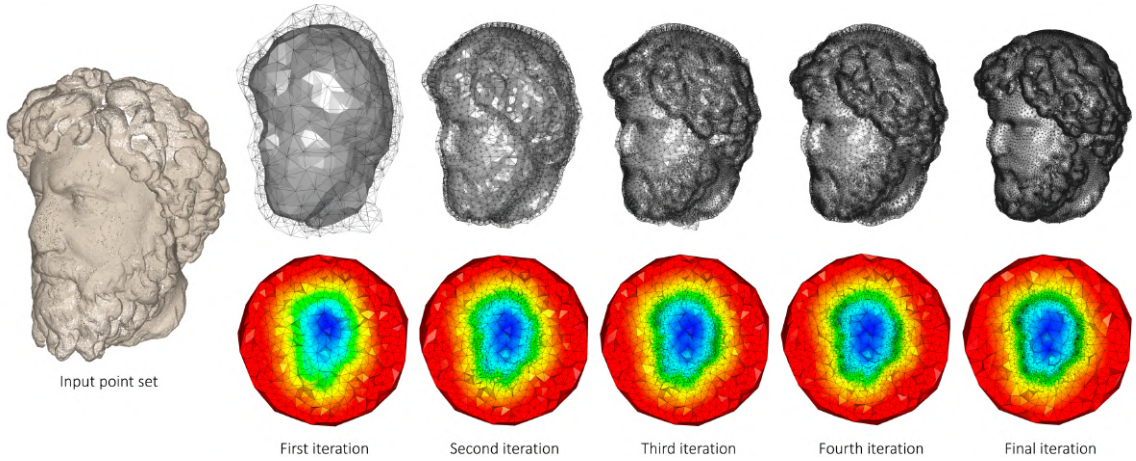


Figure 2.1 – *Progressive discrete domain for surface reconstruction. Left : input point set. Right : our approach jointly refines and optimizes the implicit function (bottom) and its discretized domain (a 3D Delaunay triangulation) around the refined isosurface (top). In such a progressive approach, the implicit solver is used iteratively, as a means of consolidating hypotheses emitted in previous iterations. Top : the isosurface and only the set of tetrahedra intersected by the isosurface are shown. Bottom : the implicit function (piecewise-linear over the 3D triangulation) is depicted on the facets intersected by a clipping plane.*

Our key insights and technical contributions are :

— *The domain discretization should be of high quality*, so that the differential operators used to solve for the implicit function can perform reliably. Meanwhile, the discretization should be economical, to allow for fast updates of the solution between two iterations. We contribute a sparse refinement scheme that allocates degrees of freedom where most needed.

— *The local discretization density should be adapted to the target surface* : denser near the isosurface, and even denser where it exhibits a small local feature size. Without prior knowledge about the output surface (i.e., the locus where $\{f(x) = 0\}$), rendering pre-allocation of an adaptive structure around input samples inadequate, we adopt a progressive approach that allocates additional degrees of freedom that are necessary to improve the accuracy and quality of the isosurface.

— *The isosurface should be of high-quality*. Given a current isosurface contoured by marching tetrahedra, we optimize the tetrahedron mesh so that the isosurface intersects the tetrahedra preferably through their edge midpoints. Our approach differs from the recent contribution of Hass and Trnkova [HT20] in that we also optimize for the shape of tetrahedra.

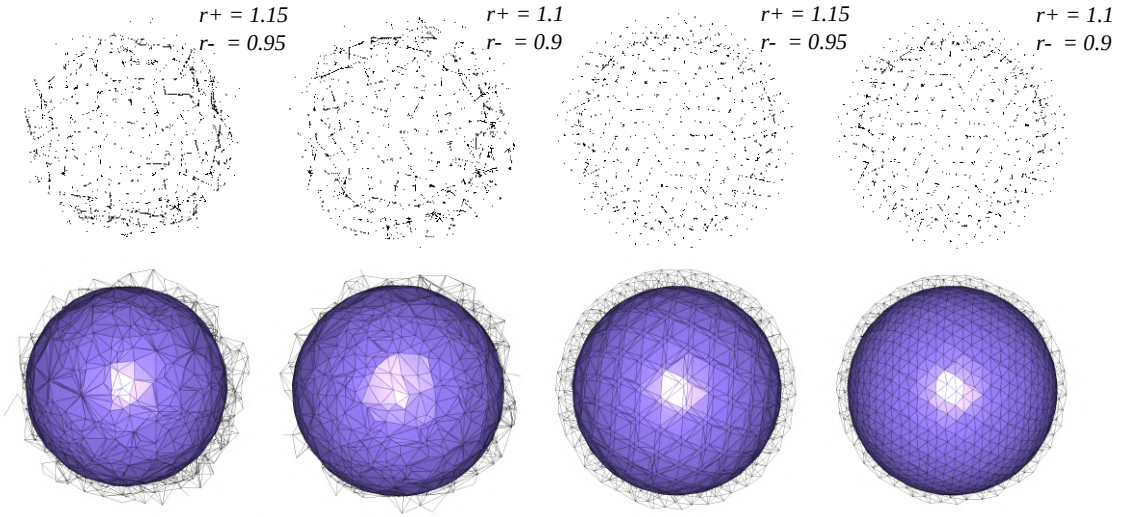
— Our method is intended to be generic, making little assumptions about the implicit function, solver and surface extraction methods, so that existing solvers can be used as black boxes. We demonstrate the relevance of our method by instantiating it on three popular method : Poisson [KH13], smooth signed distance (SSD) [CT11] and spectral [ACTD07] reconstruction.

2.2 Technical background

2.2.1 Implicit surfaces

An implicit surface, referred to as isosurface in the sequel, is defined as the level-set of a function $f : \mathbb{R}^3 \to \mathbb{R}$. Some conditions such as non-vanishing gradients are required to ensure that the isosurface is a surface. In our context where we perform a progressive implicit reconstruction of surfaces from point samples, we distinguish between three main cases, locally : (1) The isosurface passes near the input points : the original objective of faithfully approximating the input data is met ; (2) The isosurface locally passes far away from the input points : either it fills a hole or the domain discretization is too coarse, or a high regularization term creates a high tension of the isosurface ; (3) Some points are isolated, i.e., not locally approximated by the isosurface : these points are outliers or the domain discretization is too coarse.

Once the implicit function is computed, the isosurface can be contoured by the *marching tetrahedra* approach. Since the function is linearly interpolated inside each tetrahedron, in general (omitting degenerate cases), it extracts linear surface elements (quadrangles or triangles) inside the tetrahedra whose vertices have function values with opposite signs. The isosurface is thus a hybrid quad-triangle surface mesh.



(a) Biased, Noisy (b) Unbiased, Noisy (c) Biased (d) Unbiased

Figure 2.2 – *Contouring a discretized domain. (a) A "biased" triangulation is generated by two layers of randomly-generated vertices located on two concentric spheres with an inner radius of 0.95 and an outer radius of 1.15. (b) A noisy yet unbiased triangulation is generated by two layers of random vertices sampled on two concentric spheres (outer radius 1.1, inner radius 0.9). The resulting isosurface contains badly shaped triangles as well. (c) A biased triangulation is generated by two layers of evenly-placed vertices sampled on two concentric spheres. Although the tetrahedra are well shaped (isotropic, i.e., nearly-equilateral), isosurface mesh contains many skinny triangles which correspond to triangulated quadrangle elements connecting well-shaped triangles of different sizes due to the biased triangulation. (d) A unbiased triangulation is generated by two layers of evenly-placed vertices sampled on two concentric spheres. Most triangles of the isosurface are well-shaped and uniformly sized.*

2.2.2 Discretization

What is a good tetrahedron discretization for the global implicit reconstruction problem is a central question in our context. The tetrahedron elements should be well-shaped to ensure good conditioning of the solver. In addition, the 3D triangulation should exhibit denser elements only near the isosurface, where the inferred surface has a small local feature size (equivalently : large curvature, small thickness or small separation distance). Furthermore, the quality of the isosurface mesh elements has a close relationship with the discretized domain used to represent the implicit function, for the tetrahedra intersecting the isosurface. Figure 2.2 depicts four different discretizations of an implicit function approximating a unit sphere, and the resulting isosurface meshes. The implicit function is defined analytically in order to eliminate the role of the solver, and all quadrangles are triangulated so as to maximize the smallest triangle angle. This illustrates that contouring well-shaped tetrahedron elements through their edge midpoints lends to isosurface

meshes containing mostly high-quality triangles, with controlled size. To summarize, we seek for a discretized domain whose elements are well-shaped, denser near detailed isosurfaces, and sparser far away, with one layer of well-shaped tetrahedra intersecting the isosurface in their edge midpoints.

2.3 Approach

2.3.1 Overview

Our algorithm takes as input a 3D point set with oriented or unoriented normals (depending on the implicit solver) and generates as output a discretized reconstruction domain, in accordance to the pipeline depicted by Figure 2.3. We assume that the point set is sampled on a closed 2-manifold smooth surface, i.e., on the boundary of a solid. Measurement noise as and missing data are tolerated.

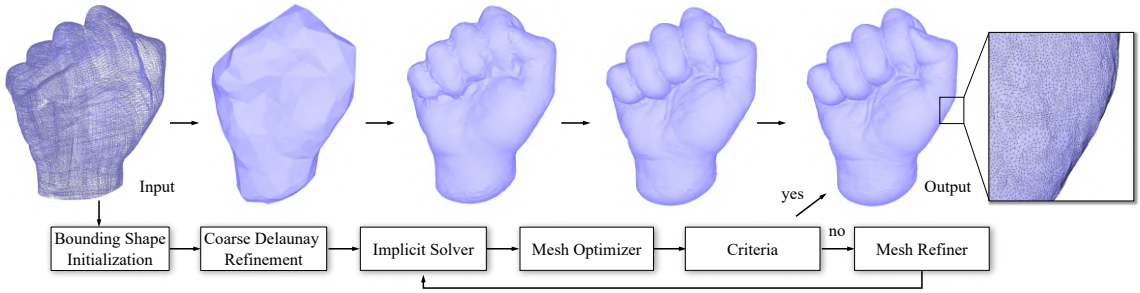


Figure 2.3 – *Overview. The domain boundary is initialized by a loose sphere bounding the input point samples, refined by coarse Delaunay refinement. The algorithm then iterates through the solver, optimizer and refiner. The isosurface, contoured by marching tetrahedra, is generated as output once user-specified criteria are met.*

We aim at generating and adapting a tetrahedron mesh discretizing the domain, serving as a support of an implicit function computed via a user-defined implicit surface reconstruction solver. We then extract an isosurface in the form of a surface triangle mesh. Possible choices for the solver include Poisson (screened or not) [KBH06, KH13], smooth signed distance [CT11] and spectral [ACTD07].

Our algorithm outputs an isosurface that fits well the input data set while exhibiting the following properties : (1) It represents an approximation of the smooth inferred surface, with a uniform or adaptive sizing; (2) It completes missing data (holes) with piecewise linear approximations of smooth surface patches, in accordance with the regularity requested for the solver; and (3) Its triangles are well shaped, i.e. isotropic.

The main user parameter of our algorithm is the aforementioned sizing field, which provides indirect control over the final mesh complexity. In the uniform case, a target triangle area is requi-

red. In the adaptive case, a variable sizing field is required. We propose such a sizing field based on a local curvature estimate.

Notations. The input point set is denoted by $\mathcal{X} = \{(p_1, n_{p_1}), \ldots, (p_N, n_{p_N})\}$. The loose bounding 3D volume mesh is denoted by $\mathcal{T} = \{V, E, T\}$ where $V = \{v_1, \ldots, v_M\}$ denotes the vertex set, $E = \{e_1, \ldots, e_P\}$ denotes the edge set and $T = \{t_1, \ldots, t_Q\}$ denotes the tetrahedron set. The implicit function is denoted by a piecewise linear scalar function $f : \mathbb{R}^3 \to \mathbb{R}$ defined onto $\mathcal{T}$ and the isosurface is denoted by a set of oriented triangles $\mathcal{S} = f^{-1}(0) = \{(a_1, n_{a_1}), \ldots, (a_R, n_{a_R})\}$. A uniform (constant) sizing field is denoted by s, and an adaptive sizing field is defined as a scalar function $s(x)$ from $\mathbb{R}^3 \to \mathbb{R}$.

Assumptions made on the solver. Our method requires the solver to be discretizable on tetrahedron meshes. This is the case for all solvers requiring discrete differential operators such as the Laplacian, gradient, divergence and Hessian. For solvers that take as input parameters a kernel for diffusing attributes from the input samples (e.g., normals for Poisson reconstruction), we require as parameter the kernel size in order to ensure a minimal vertex sampling around the input point samples. SSD and spectral reconstruction do not suffer from this constraint. We assume that the solver provides as output a piecewise-linear scalar implicit function defined on the tetrahedron mesh vertices, and that the output surface is the zero level set of this function.

2.3.2 Initialization

Given the input point set $\mathcal{X}$, we compute its bounding sphere whose center is denoted as $c_{\mathcal{X}}$ and radius denoted as $r_{\mathcal{X}}$. The geometric domain is initialized by inserting into a 3D Delaunay triangulation 100 points uniformly sampled on the surface of a sphere centered at $c_{\mathcal{X}}$ with radius $1.4r_{\mathcal{X}}$. We then perform 3D Delaunay refinement according to the circumradius-to-shortest edge ratio (threshold set to 1.3 by default), which ensures an initial quality 3D triangulation. The initialization has little impact on the final reconstructed surface when the enlarged ratio (1.4 by default) is large enough. Its role is to bootstrap the refinement algorithm after obtaining an initial implicit function from the solver. The domain is then further discretized gradually in the following process.

2.3.3 Optimization

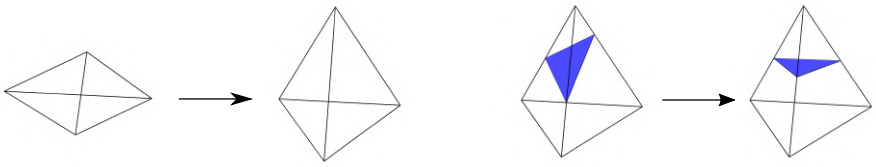
Given the current triangulation $\mathcal{T}$, implicit function and relating isosurface $\mathcal{S}$, the optimization step is designed to relocate the triangulation vertices so as to find a balance between improving the shape of tetrahedra (making them as isotropic as possible), favoring that tetrahedra intersected by $\mathcal{S}$ are intersected through their edge midpoints (creating a layer of tetrahedra "sandwiching" the isosurface), while preserving the boundary of the initial bounding sphere.

We formulate the optimization as a function-minimization process for the variable vertex positions $\{v_i\}$, in which the objective function comprises three terms : (1) the As-Similar-As-Possible (ASAP) term E_a rewards isotropic tetrahedra, weighted by a coefficient λ_a, (2) the mid-edge term E_m rewards midpoint isosurface intersection, weighted by a coefficient λ_m and (3) the damping term E_d penalizes large-scale vertex relocation far away from the isosurface.

Denoting by $T_{\mathcal{S}}$ the set of tetrahedra intersecting $\mathcal{S}$, the objective function $E(\mathcal{T})$ is defined as :

$$E(\mathcal{T}) = \lambda_a \sum_{t \in \mathcal{T}} w_a(t) E_a(t) + \lambda_m \sum_{t \in \mathcal{T}_{\mathcal{S}}} w_m(t) E_m(t) + \sum_{v \in V} w_d E_d(v). \tag{2.1}$$

Figure 2.4 depicts the two principal terms of the optimized objective function.



(a) ASAP energy (b) Mid-edge energy

Figure 2.4 – *Objective function. (a) The as-similar-as-possible (ASAP) term favors the tetrahedron to become equilateral under a volume constraint. (b) The mid-edge term favors the midpoint of edges with opposite function value signs to pass through the isosurface (blue).*

ASAP objective. The goal is to deform each tetrahedron t to make it as congruent as possible to an equilateral tetrahedron, referred to as reference tetrahedron, under a scaling operator that preserves a given volume, while preserving its center. We define the reference tetrahedron t' as the centered unit regular tetrahedron and find the optimal transformation by minimizing the following function :

$$E_a(t) = \min_{\mathbf{S}} \sum_{i=0}^{3} \|(c(t) + \mathbf{S} v_{t'_i}) - v_{t_i}\|^2, \tag{2.2}$$

where $\mathbf{S}$ denotes an arbitrary isotropically stretched rotation matrix (a similarity matrix), $c(t)$ denotes the center of t, and t_i (resp. t'_i) denotes the i^{th} corner of t (resp. t'). To find the minimizer matrix $\mathbf{S}^*$, we borrow ideas from the local/global optimization approach [SA07]. We start from translating the centroid of the tetrahedron $c(t)$ to the origin, and then compute the 3×3 covariance matrix defined as

$$\mathbf{C}(t', t - c(t)) = t'(t - c(t))^T. \tag{2.3}$$

We then perform a singular value decomposition (SVD) onto the covariance matrix :

$$\mathbf{C} = \mathbf{U} \Sigma \mathbf{V}^T. \tag{2.4}$$

To determine the scaling factor of $\mathbf{C}$, we first compute the volume of all tetrahedra T and then smooth the volume values by averaging the values of adjacent tetrahedra. This decreases the sizing gradation among adjacent tetrahedra and better conditions the system. Given $\overline{vol}(t)$, the smoothed volume of the tetrahedron t, we set the modified singular value matrix Σ' to be :

$$\Sigma' = \sqrt[3]{\frac{\overline{vol}(t)}{vol(t')}} \cdot \mathbf{I}_3. \tag{2.5}$$

The minimizer matrix $\mathbf{S}^\star$ is set to :

$$\mathbf{S}^\star = \mathbf{V}\Sigma'\mathbf{U}^T. \tag{2.6}$$

If the determinant of $\mathbf{S}^\star$ is negative, we invert the sign of the last singular vector of $\mathbf{U}$ and then recompute $\mathbf{S}^\star$. The objective function can then be written as :

$$E_a(t) = \sum_{i=0}^{3} \|(c(t) + \mathbf{S}^* v_{t_i'}) - v_{t_i}\|^2, \tag{2.7}$$

where the vertex locations v_{t_i} denote the variables under optimization.

Mid-edge objective. This term favors that the isosurface intersects tetrahedra through their edge midpoints. Combined with the ASAP objective, contouring the intersected tetrahedra yields well-shaped triangles with locally uniform sizing, where the length should be expressed within the norm of the sizing field.

Given a tetrahedron t intersected by isosurface $\mathcal{S}$, we consider only the edges intersecting $\mathcal{S}$, i.e. the edges with alternating signs of the implicit function value. In essence, this term encourages the midpoint of all edges crossing $\mathcal{S}$ to be mapped to it, which in turn encourages the isosurface $\mathcal{S}$ to be located in the middle of two offset surfaces composed of the faces of $T_\mathcal{S}$ that do not cross $\mathcal{S}$.

$$E_m(t) = \sum_{e \in t, f(v_{e_1}) \cdot f(v_{e_2}) < 0} \left(\left(\frac{v_{e_1} + v_{e_2}}{2} - p_e \right) \cdot n_{a_t} \right)^2, \tag{2.8}$$

where a_t denotes the isosurface inside tetrahedron t (a quadrangle, resp. a triangle, if the number η_t of edges of t crossing $\mathcal{S}$ is 4, resp. 3), p_e is the intersection of edge e and a_t, and n_{a_t} is the normal of a_t. The variables v_{e_1} and v_{e_2} refer to the vertex locations of the vertices of edge e.

Damping objective. To prevent the boundary of the triangulation $\mathcal{T}$ from expanding or shrinking during optimization, we add a damping term to restrict the movement of the vertices near $\partial\mathcal{T}$. More specifically, we first iterate over each vertex v and compute the minimum (resp. maximum) function values $f_{\min}$ (resp. $f_{\max}$), and the minimum (resp. maximum) function values at each vertex adjacent to a tetrahedron crossing the isosurface $f^{\mathcal{S}}_{\min}$ (resp. $f^{\mathcal{S}}_{\max}$).

The weight of vertex $g_d(v)$ is defined as :

$$g_d(v) = \begin{cases} \dfrac{f(v) - f^{\mathcal{S}}_{\min}}{f_{\min} - f^{\mathcal{S}}_{\min}}, & \text{if } f(v) < f^{\mathcal{S}}_{\min} \\ \dfrac{f(v) - f^{\mathcal{S}}_{\max}}{f_{\max} - f^{\mathcal{S}}_{\max}}, & \text{if } f(v) > f^{\mathcal{S}}_{\max} \\ 0, & \text{otherwise.} \end{cases} \tag{2.9}$$

Large holes may generate open isosurfaces with boundary on the domain boundary $\partial\mathcal{T}$. In order to avoid shrinking the isosurface's boundary, we compute the distance from each vertex v to $\partial\mathcal{T}$ and set $g_d(v) = 1$ for vertices located near $\partial\mathcal{T}$ to ensure that they remain unchanged. Once the weight $g_d(v)$ is determined, the objective is computed as :

$$E_d(v) = g_d(v) \cdot \|\tilde{v} - v\|^2, \tag{2.10}$$

where $\tilde{v}$ denotes the vertex position at the previous iteration.

There are several reasons motivating this design choice : **(i)** We observed that large sliver tetrahedra are generally located at the boundary of $\mathcal{T}$, and while they contribute in practice very little to the accuracy of the solution around the target isosurface $\mathcal{S}$, penalizing their shape's distortion comes at the much higher price of preventing optimization of the tetrahedra around $\mathcal{S}$. **(ii)** Formulating the damping weights as a function of the scalar field f itself allows us to favor shape improvement of the tetrahedra around $\mathcal{S}$ without having to walk explicitly on the triangulation (making it possible to set them in linear time), or updating and querying the nearest neighbor structure, which would be necessary if the weights were defined in terms of distance to $\mathcal{S}$. **(iii)** For all solvers under consideration, the *uncertainty* of $\mathcal{S}$ is better expressed in terms of variation of the scalar field around 0 than in terms of distance to $\mathcal{S}$ (consider the Poisson solver for example, for which large holes in the data are filled with an extremely slowly-varying scalar field f).

Weighting functions. To make the optimization invariant to rigid motion and scaling of the input point set $\mathcal{X}$ and triangulation $\mathcal{T}$, we define weighting functions for the three terms, respectively. Denote by $r(\mathcal{T})$ the diameter of the triangulation (largest distance between pairs of vertices). For the ASAP term we define $w_a(t) = vol(t)/r(\mathcal{T})^3$; for the mid-edge term we define $w_m(t) = area(a_t)/r(\mathcal{T})^2$; for the damping term we define a high coefficient, by default $w_d = 100/r(\mathcal{T})$, which makes it a hard constraint to $E(\mathcal{T})$.

Linear solver. Since the three above terms are quadratic, the total objective function $E(\mathcal{T})$ is quadratic and can be minimized by solving a linear system with $3|V|$ variables. We initialize the solution as the unoptimized vertex coordinates and utilize an iterative linear solver. As all vertices are relocated, we update the implicit function values at the vertices by linear interpolation over

the triangulation computed by the previous solver, since the cost of running a solver on a large triangulation is substantially more expensive than the one of the optimizer.

2.3.4 Refinement

We now describe a progressive and parsimonious refinement process for the 3D triangulation of the domain, in order to provide "just enough" degrees of freedom around the isosurface $\mathcal{S}$ for further computation. More specifically, we propose two refinement schemes tailored to induce a local subdivision of the triangle facets of the isosurface with variable granularity, that preserves the shapes of both the facets of the isosurface and tetrahedra of the 3D triangulation.

We observe the two possible configurations of a tetrahedron intersecting the isosurface, i.e. with both positive and negative function values at its vertices (see Figure 2.5). For case 1, we have either 3 positive values and 1 negative value, or vice-versa : the isofacet a_t is a triangle. For case 2, we have 2 positive and 2 negative values : the isofacet a_t is a quadrangle. In order to triangulate a_t, we split the quadrangle by its shortest diagonal.

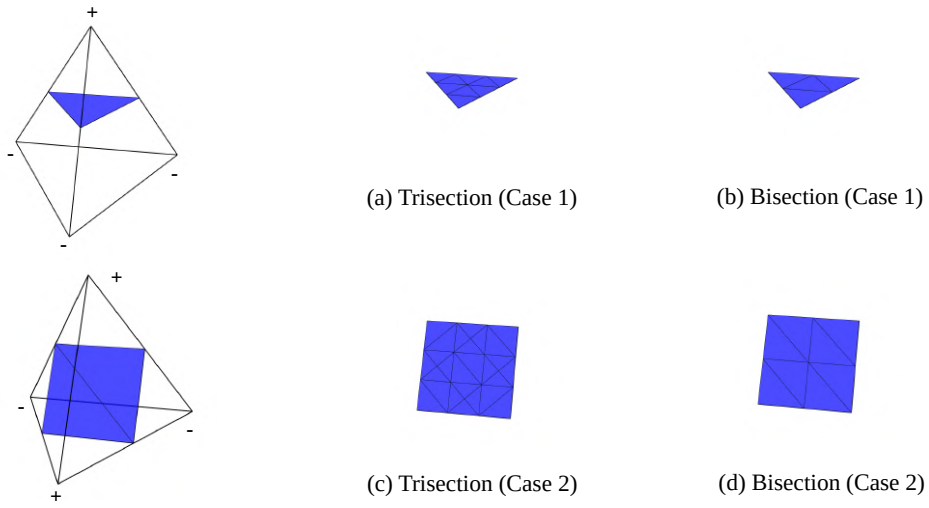


Figure 2.5 – *Subdivided facets of the isosurface. Left : two configurations of a tetrahedron containing the isosurface. Middle : trisection refinement. Right : bisection refinement. For case 1 (1/3 vertices), the number of triangles is multiplied by a factor 9 after trisection refinement (a) and by 4 after bisection refinement (b). For case 2 (2/2 vertices), the number of triangles is multiplied by a factor between 9 and 15 after trisection refinement (c) and by 4 after bisection refinement (d).*

Trisection refinement. Given a tetrahedron t to be refined, we insert the trisection points of each edge and the centroid of each face into the domain triangulation (see Figure 2.6(b)). While for case 1 the isofacet is always divided into 9 sub-triangles (Figure 2.5(a) and 2.6(e)), it is not true

for case 2, in which the number of sub-triangles depends on the shape of t. Empirically, the number is between 9 and 17, with an average of 12, which we use to define our refinement algorithm.

Bisection refinement. We insert the midpoint of each edge with the same value signs (Figure 2.6(c)(d)). In both cases, the isofacet is divided into 4 sub-triangles (Figure 2.5(g-h)).

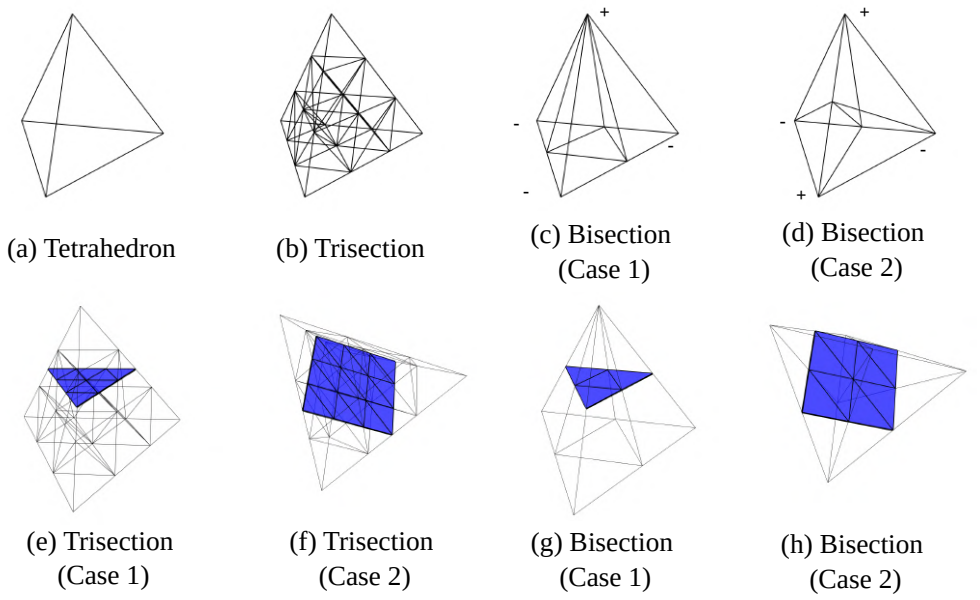


Figure 2.6 – *Refinement schemes. (a) Tetrahedron before refinement. (b) Refinement by crossing edge trisection : we insert three points uniformly sampled on each edge and the centroid of each face. (c-d) Refinement by bisection : we insert the midpoint of each edge with similar value signs (two cases). (e-f) Refined isosurfaces for both cases after trisection refinement. (g-h) Refined isosurfaces for both cases after bisection refinement.*

Refinement strategy. The adaptive mesh refinement step utilizes the above refinement schemes as follows. Consider a tetrahedron t intersecting the isosurface into a facet f_t. We evaluate the target area at the centroid of f_t from the input sizing field, denoted as $s(f_t)$. The area ratio r_t is defined as the ratio between the current area of f_t and the target area $s(f_t)$. If r_t is smaller than 1, we deduce that the local sizing criterion is already satisfied, and we do not refine t. Otherwise, we check the following conditions and adopt the following schemes :

1. If t belongs to case 1 and $r_t > 9$, we trisect t.
2. If t belongs to case 2 and $r_t > 12$, we trisect t.
3. If t belongs to case 1 and $4 \leq r_t < 9$, we bisect t.
4. If t belongs to case 2 and $4 \leq r_t < 12$, we bisect t.
5. If $1 < r_t < 4$, we bisect t with a certain probability.

When $1 < r_t < 4$, we expect r_t to approach 1 while avoiding over-refining t. Suppose we have N tetrahedra intersecting isosurface with average area $\overline{a}$ before refinement. We bisect u percent of tetrahedra and the average area after the refinement is $\overline{a}'$. In order to preserve the total area of the isosurface, we have :

$$\overline{a}' = \frac{N\overline{a}}{4uN + (1-u)N} \quad \rightarrow \quad u = \frac{\overline{a} - \overline{a}'}{3\overline{a}'}$$

For a specific tetrahedron t, we evaluate a uniform random variable $c \in [0, 1]$ and bisect the tetrahedron only if $c < \frac{1}{3}(r_t - 1)$. Otherwise, we do not refine it.

Adaptive sizing field. While the target area is intuitive to deduce for a uniform sizing field, the adaptive sizing field is more delicate. Users can come with their own sizing fields, but we provide a reasonable choice guided by the local curvature estimate of the inferred surface. Our sizing field is dense near the region with high curvatures and sparse otherwise.

We start by evaluating the minimum curvature $c_{\min}$ and the maximum curvature $c_{\max}$ onto the input point cloud $\mathcal{X}$ by fitting a Monge form [CP05].

Then, before beginning each refinement step, we construct a smoothed curvature map for tetrahedra intersecting the isosurface. Given a tetrahedron t intersecting the isosurface, we consider the k-nearest neighbors of the centroid of a_t in $\mathcal{X}$ and estimate the local curvature by computing the maximum absolute value of the two principal curvatures of the neighbors. This curvature map is smoothed following the same process of the volume field in ASAP energy and then clipped within $[c_{\min}, c_{\max}]$.

When refining a tetrahedron t, we deduce the target sizing from its smoothed curvature c. A mapping function is applied to c so that users can control the grading of the sizing field :

$$c' = \left(\frac{c - c_{\min}}{c_{\max} - c_{\min}}\right)^{\lambda} \cdot (c_{\max} - c_{\min}) + c_{\min}.$$

Denote by d a user-controlled parameter that defines the expected distance tolerance from the isosurface to the point cloud. The target sizing s is computed as :

$$s = 4\sqrt{3} \cdot \left(\frac{2d}{c} - d^2\right).$$

Finally, s is clipped within $[s_{\min}, s_{\max}]$ and used for guiding the above refinement process.

2.3.5 Solvers

We now detail our framework at work using global implicit reconstruction solvers applied to tetrahedron meshes.

Discretization elements. We use piecewise-linear basis functions $\{\phi_i\}_i$, $\phi_i(v_j) = \delta_i^j$ for all vertices v_j of $\mathcal{T}$. For a point x inside a tetrahedron $t = (t^0, t^1, t^2, t^3)$, $\{\phi_{t^k}\}_{k=0}^3(x)$ are its barycentric coordinates, and given scalar values $\{f_i\}$ associated with the vertices $\{v_i\}$ of $\mathcal{T}$, $f(x)$ is defined as $f(x) = \sum_{k=0}^3 \phi_{t^k}(x) f_{t^k} = \sum_i \phi_i(x) f_i$. Similarly, the gradient of f at x is defined as $\bigtriangledown f(x) = \sum_{k=0}^3 \bigtriangledown \phi_{t^k}(x) f_{t^k}$. We note $G \in \mathbb{R}^{3|\mathcal{T}|\times|\mathcal{V}|}$ the gradient matrix whose $(3t, 3t+1, 3t+2)$ rows contain the four gradients of the barycentric coordinate functions of the tetrahedron t, Div the (integrated) divergence matrix, $L \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ the (integrated) Laplacian matrix, $B := LM_{\mathcal{V}}^{-1}L \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ the (integrated) bi-Laplacian matrix, and $H \in \mathbb{R}^{9|\mathcal{V}|\times|\mathcal{V}|}$ the (integrated) Hessian matrix. We use the construction presented by Stein et al. [SGWJ18] : Noting $G_x, G_y, G_z \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{V}|}$ the gradient matrices of the (x, y, z) coordinates, H is defined as

$$\begin{aligned} H :=& \tilde{D}^T \tilde{M}_{\mathcal{T}} \tilde{G}, \text{ with} \\ \tilde{G} := \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix}, \tilde{D} :=& \begin{bmatrix} G_x & G_y & G_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_x & G_y & G_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & G_x & G_y & G_z \end{bmatrix}. \end{aligned} \tag{2.11}$$

Denoting the integrated Laplacian matrix, L is defined using the standard cotangent formula as :

$$\begin{cases} L_{ij} & = \sum_{t \ni (i,j)} l_{ij} \cot(\gamma_{ij}^t)/6 \qquad \forall j \in N_1(i) \\ L_{ii} & = -\sum_{j \neq i} L_{ij}, \end{cases} \tag{2.12}$$

where l_{ij} denotes the length of edge (i, j), and γ_{ij}^t denotes the dihedral angle opposite to edge (i, j) in the tetrahedron t.

Screened Poisson solver. We minimize the objective function :

$$\mathcal{E}_{SP} := \int_{\mathcal{T}} \| \bigtriangledown f(x) - \tilde{n}(x) \|^2 dx + \frac{\alpha}{|\mathcal{X}|} \sum_p f(p)^2 \rightarrow \min, \tag{2.13}$$

where $\tilde{n}(x)$ denotes a smooth approximation of the normal field n defined on $\mathcal{X}$. Note that it is common to shift the solution after minimization, in order to find the isosurface best approximating the input samples $\mathcal{X} : f \leftarrow f - \text{median}(\{f(p)\}_{p \in \mathcal{X}})$. If α is set to 0, one obtains the original Poisson solver, in which case it is necessary to add at least one constraint as otherwise the system is underconstrained. Minimizing this objective function on tetrahedron meshes amounts to solving for the following linear system :

$$\left[L + \frac{\alpha}{|\mathcal{X}|} \Phi_{\mathcal{X}}^T \Phi_{\mathcal{X}} \right] F = \text{Div} \tilde{N}, \tag{2.14}$$

where $\Phi_{\mathcal{X}}$ denotes the matrix stacking the barycentric coordinates of $\mathcal{X}$.

Smooth signed distance solver. We minimize the objective function :

$$\mathcal{E}_{SSD} := \frac{\alpha}{|\mathcal{X}|}\sum_p f(p)^2 + \frac{\beta}{|\mathcal{X}|}\sum_p \| \bigtriangledown f(p) - n_p\|^2 + \frac{\gamma}{|\mathcal{T}|}\int_{\mathcal{T}} \|Hf(x)\|^2 dx \rightarrow \text{min.} \tag{2.15}$$

Minimizing this objective function on tetrahedron meshes amounts to solving the following linear system :

$$\left[\frac{\alpha}{|\mathcal{X}|}\Phi_{\mathcal{X}}^T\Phi_{\mathcal{X}} + \frac{\beta}{|\mathcal{X}|}G^T S_{\mathcal{X}}^T S_{\mathcal{X}} G + \frac{\gamma}{|\mathcal{T}|}H^T \tilde{M}_{\mathcal{V}}^{-1} H\right] F = \frac{\beta}{|\mathcal{X}|}G^T S_{\mathcal{X}}^T N_{\mathcal{X}},$$

where $N_{\mathcal{X}} \in \mathbb{R}^{3|\mathcal{X}|}$ stacks the normals of $\mathcal{X}$, $\tilde{M}_{\mathcal{V}} \in \mathbb{R}^{9|\mathcal{V}|\times 9|\mathcal{V}|}$ denotes the mass matrix of the vertices repeated 9 times along the diagonal, and $S_{\mathcal{X}}$ denotes the selection matrix whose $(3i, 3i+1, 3i+2)$ rows contain the Identity matrix at columns $(3t, 3t+1, 3t+2)$ if vertex i lies inside the tetrahedron t.

Spectral solver. We minimize the objective function :

$$\mathcal{E}_{Spec} := \sum_p \bigtriangledown f(p)^T \cdot C_p \cdot \bigtriangledown f(p) \rightarrow \text{max, such that} \quad \frac{\alpha}{|\mathcal{X}|}\sum_p f(p)^2 + \beta\int_{\mathcal{T}} \| \bigtriangledown f(x)\|^2 dx + \gamma\int_{\mathcal{T}} \triangle f(x)^2 dx = 1. \tag{2.16}$$

Maximizing this objective function on tetrahedron meshes amounts to finding the largest eigenvalue and related eigenvector of the following generalized eigenvalue problem :

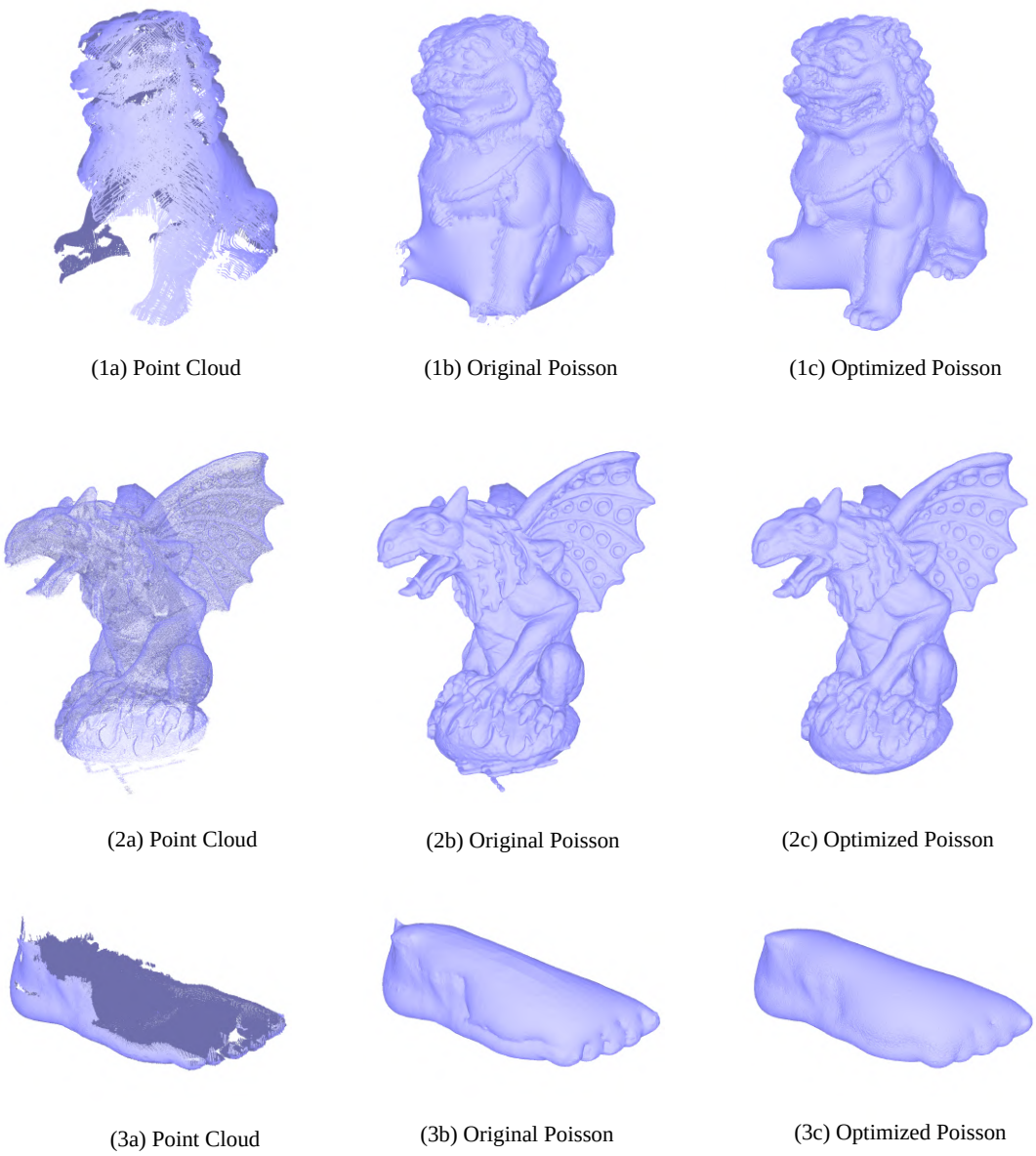
$$\left[G^T S_{\mathcal{X}}^T C_{\mathcal{X}} S_{\mathcal{X}} G\right] F = \lambda \left[\frac{\alpha}{|\mathcal{X}|}\Phi_{\mathcal{X}}^T\Phi_{\mathcal{X}} + \beta L + \gamma B\right] F, \tag{2.17}$$

where $C_{\mathcal{X}}$ denotes the block diagonal matrix whose i^{th} block is the anisotropy matrix $C_i := I + \mu n_i n_i^T$, n_i denotes the (unoriented) normal of input point $p_i \in \mathcal{X}$ and μ controlling the anisotropy favoring alignment between the gradient and the unoriented normal at p_i. Compared to the above solvers, this one is oblivious to the orientation of the input normals. This comes at the cost of solving for a generalized eigenvalue problem designed to yield a signed implicit function.

2.4 Experiments

Our framework is implemented in C++, using the CGAL library for 3D triangulations and geometric computations [The21], the Eigen library for linear algebra and solvers [GJ+10], the Spectra library for solving generalized eigenvalue problems [Qiu21] and OpenMP for multithreading ac-

celeration [DM98]. The experiments are conducted on a MacBook Pro with a 2,9 GHz Quad-Core Intel Core i7 CPU and 16GB memory.



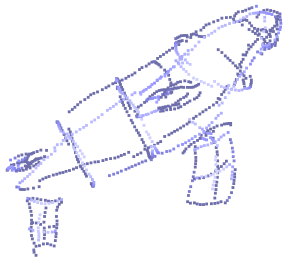
(1a) Point Cloud (1b) Original Poisson (1c) Optimized Poisson

(2a) Point Cloud (2b) Original Poisson (2c) Optimized Poisson

(3a) Point Cloud (3b) Original Poisson (3c) Optimized Poisson

Figure 2.7 – *Reconstruction gallery (part 1). We compare our reconstruction results (right) with the one solved using common input-sensitive approaches (middle).*

Figure 2.7 and 2.8 depict a gallery of reconstructed surfaces. (4) is a very sparse point set with oriented normals (from [HCJ19]). Common octree-based methods cannot yield a smooth surface since the octree is refined only near the input point set, while our progressive approach yields smooth surfaces. The rest depict other scanned point sets with holes and noise. Our progressive



(4a) Point Cloud

(4b) Original SSD

(4c) Optimized SSD

(5a) Point Cloud

(5b) Original SSD

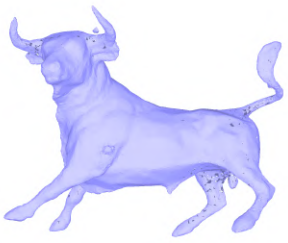
(5c) Optimized SSD

(6a) Point Cloud

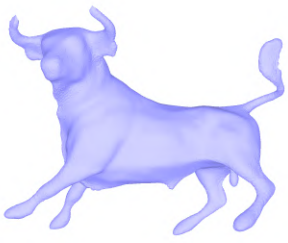
(6b) Original Spectral

(6c) Optimized Spectral

(7a) Point Cloud



(7b) Original Spectral



(7c) Optimized Spectral

Figure 2.8 – *Reconstruction gallery (part 2). We compare our reconstruction results (right) with the one solved using common input-sensitive approaches (middle).*

approach reconstructs smooth surfaces in accordance to the regularity parameter of the selected solver.

2.4.1 Adaptivity

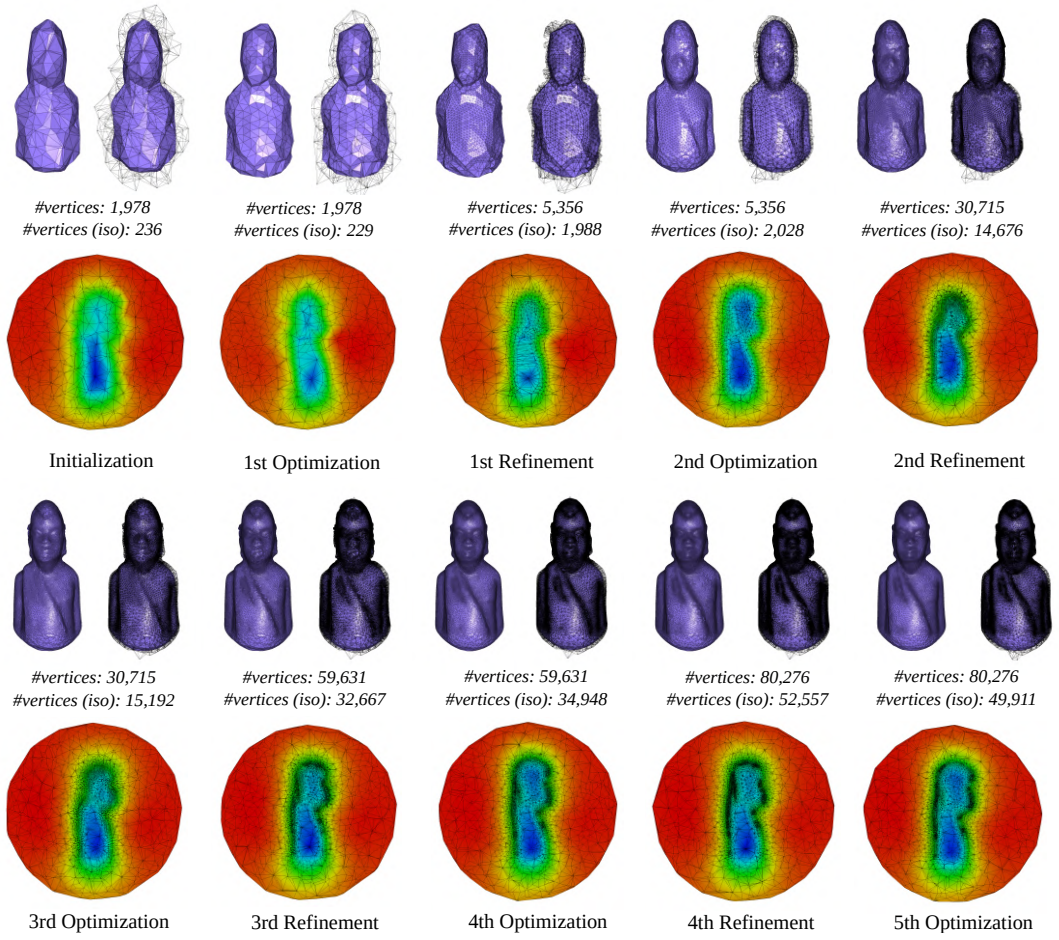


Figure 2.9 – *Reconstruction process for the Massai Model. The first and third rows depict the isosurface and the tetrahedra intersected by the isosurface during the reconstruction process. The second and fourth rows depict the implicit function clipped by a plane.*

Figure 2.9 depicts the progressive reconstruction process on the *Massai* point set. The optimizer and the refiner jointly improve the quality of the discretized domain and isosurface mesh. The discretized domain is getting denser and denser while sandwiching the reconstructed isosurface. The ratio between the number of vertices near the isosurface and of the entire triangulation increases rapidly, showing that we allocate more degrees of freedom where needed. The implicit functions depicted in a cutting plane highlight that the triangulation is denser around the high curvature area, which helps reducing the interpolation error.

2.4.2 Progressive refinement

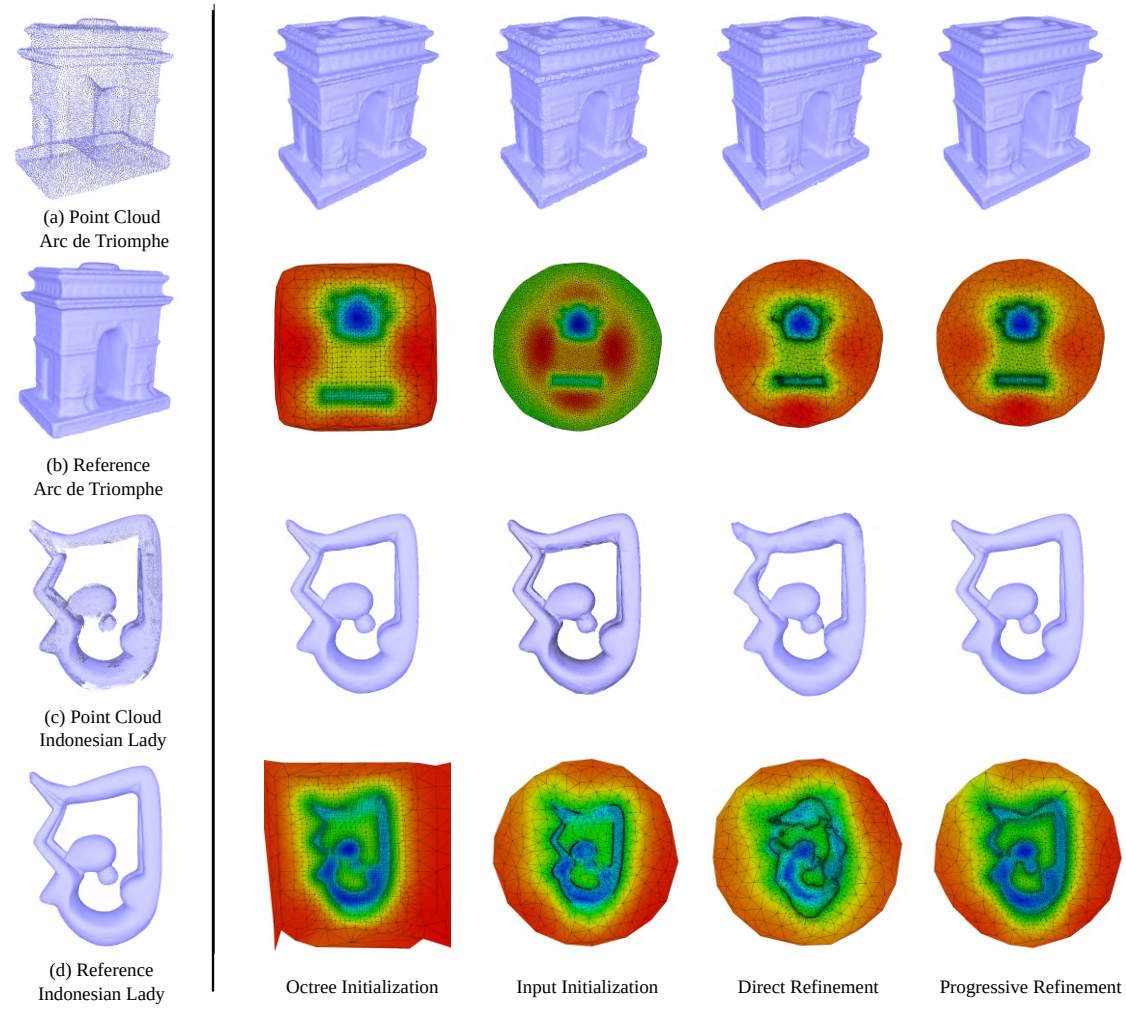


Figure 2.10 – *Reconstructed surfaces and discretized domains of the four above approaches, clipped by a cutting plane. We utilize the SSD solver for both point clouds and use a uniform sizing field to guide the refinement. Our optimized domain adapts to the isosurface.*

We validate the relevance of the proposed progressive domain approach, by comparing four different approaches for generating the discretized geometric domain and reconstructing the output isosurface :

1. **Octree-based initialization** : we discretize the domain by inserting into the 3D triangulation all nodes of a dense octree refined from the local density of the input point set.
2. **Input-based initialization** : we first insert all input points into the 3D triangulation, then perform dense Delaunay refinement inside a loose bounding box of the input point set until all tetrahedra are well-shaped.

3. **Direct refinement** : we initialize the 3D triangulation by a sparse point set sampled on the loose bounding box of the input point set, followed by Delaunay refinement until all tetrahedra are well-shaped. We then launch the solver to obtain an initial isosurface and refine the 3D triangulation by using our refinement process until the sizing criteria are satisfied, without using any optimizer or solver. We run a final solver to generate the final isosurface.
4. **Progressive refinement** : we perform the proposed progressive algorithm through "solver-optimizer-refiner" iterations.

Our goal is not to evaluate the output result by common criteria such as e.g. the average distance from the points to the reconstructed surface, or deviation of normals, etc. Instead, we wish to verify the relevance of our domain discretization, given a solver and its regularization parameters. Once combined, they trade data fidelity for regularization, and can thus deviate largely from perfect data fidelity.

TABLE 2.1 – *Reconstructing the Arc de Triomphe and Indonesian Lady models*

Measures		Methods				
		Octree	Input	Direct	Progressive	Reference
Arc de Triomphe	Hausdorff	1.622006	2.463567	3.124351	1.504585	-
	Mean (Method - Ref)	0.054016	0.085333	0.053248	0.033301	-
	Mean (Ref - Method)	0.049102	0.080371	0.048665	0.031906	-
	RMS (Method - Ref)	0.093774	0.184979	0.111763	0.055931	-
	RMS (Ref - Method)	0.080871	0.162403	0.08594	0.051933	-
	Timing	78.27	805.42	758.92	2532.42	31493.71
	# Isovertices (1)	116232	53340	232447	331968	741211
	# Vertices around iso (2)	57643	24361	115574	153514	364041
	# Vertices (3)	117160	309478	238417	231673	1419823
	Parsimony (4)	0.492002	0.078716	0.484756	0.662632	0.256399
Indonesian Lady	Hausdorff	0.024654	0.036668	0.041566	0.022067	-
	Mean (Method - Ref)	0.000430	0.000838	0.002920	0.000596	-
	Mean (Ref - Method)	0.000612	0.001235	0.002925	0.000588	-
	RMS (Method - Ref)	0.001444	0.001785	0.005649	0.001324	-
	RMS (Ref - Method)	0.001801	0.002647	0.005501	0.001185	-
	Timing	615.17	1222.79	702.13	1502.67	27806.79
	# Isovertices (1)	249969	1576551	88207	312517	455312
	# Vertices around iso (2)	123723	754457	46928	144366	229912
	# Vertices (3)	280335	974507	272648	254091	1947880
	Parsimony (4)	0.441340	0.774194	0.172119	0.568167	0.118032

(1) Number of vertices on the remeshed isosurface. (2) Number of triangulation vertices around the isosurface. (3) Number of vertices of the entire triangulation. (4) Parsimony $= (2)/(3)$.

In order to evaluate and compare the results, we compute a "ground truth" reference isosurface from a point set, by running the given solver and regularization parameters on a densely discretized domain. We generate the domain as follows : (1) Compute the isosurface using a dense octree-based SSD, (2) sample very densely the isosurface, duplicate the resulting point set and offset the two point sets along the negative and positive local normal directions, (3) insert all offset points into a 3D Delaunay triangulation, (4) optimize the 3D triangulation by minimizing the ASAP energy, and (5) solve via SSD to generate the reference isosurface.

Figure 2.10 (the first column) shows the selected point clouds and their corresponding ground truth isosurfaces. Figure 2.10 (the second to the fifth columns) and Table 2.1 depict and record the four aforementioned reconstructions and related statistics. For both input point sets, using our progressive approach we obtain more than 55% vertices of the triangulation near the final isosurface, which validates the parsimony of our approach. We compare the 4 above approaches in terms of (1) the output surface (2) the discretized domain (3) the distances from the reconstructed surface to the reference surface computed using Metro [CRS98], (4) the parsimony, defined as the number of vertices adjacent to tetrahedra intersecting the isosurface, divided by the total number of vertices of the triangulation.

2.4.3 Robustness

Different densities. We sample the kitten model with different densities and compare the results of our algorithm using the SSD solver and uniform area criteria with the results of the original octree-based SSD algorithm, see Figure 2.11.

Variable resolution. We generate a point set of the Kitten with two different resolutions : dense on the head and sparse elsewhere. Figure 2.11 compares our optimized one with the original SSD.

Noise. We compare our results with the original octree-based SSD on a model with increasing levels of noise ($\sigma \in \{0.005, 0.01, 0.05, 0.1\}$). The results are shown by Figure 2.12. When increasing σ, the original SSD becomes more and more sensitive to solver parameters and the isosurface becomes less and less smooth.

Holes. We verify the capability of our approach to fill holes on the two Indonesian models, see Figure 2.13 and Figure 2.14, which are two laser scans with large holes and imperfect normals. Albeit filling holes is an ill-posed problem, our method seems to fill the holes more gracefully. The artifacts of the octree-based SSD are getting more evident when the depth of the octree increases, while there are no such artifacts when we solve SSD on an optimized domain. Note that these artifacts (bumps with high curvature variations) are in contradiction with what is expected from an SSD solution on a smooth input with strong Hessian penalization. This indicates that the allocated

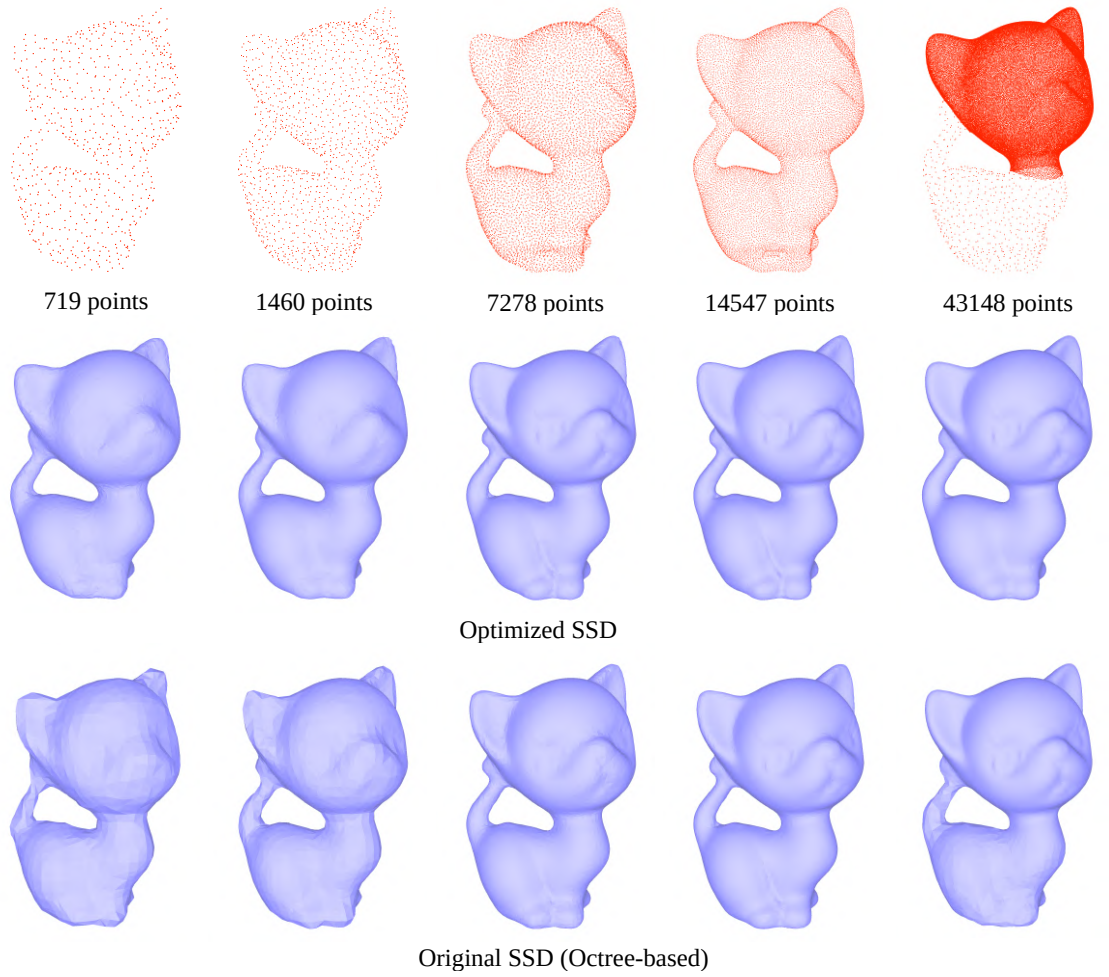


Figure 2.11 – *Reconstruction results of Kitten model with different sampling densities. Top row : input point clouds. Middle row : SSD solved on our optimized geometric domain. Bottom row : SSD solved on an octree. It fails to discover more details when the point cloud is sparse.*

structure prevents the solver from yielding an accurate result, which an a posteriori remeshing approach would not help fixing.

2.4.4 Solver conditioning

The ASAP energy is the key component for improving the solver conditioning. It improves the quality of the tetrahedral elements of the 3D triangulation to achieve this goal. We start by comparing the number of iterations to make the linear solver attain a fixed tolerance error (1e-10) before and after a pure ASAP optimization (without mid-edge energy). All the solvers are initialized with a zero vector solution to make it a fair comparison. From Table 2.2 and Figure 2.15,

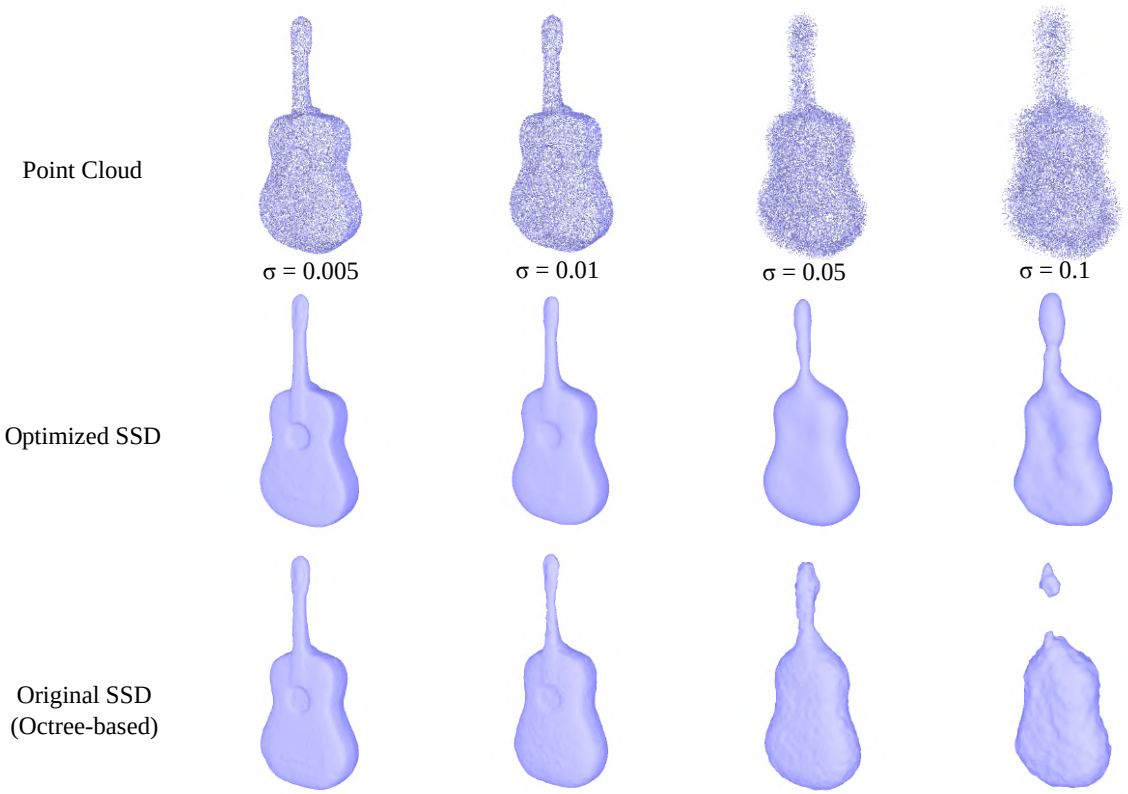


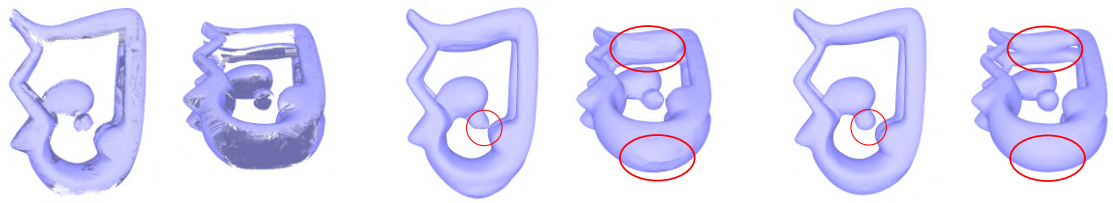
Figure 2.12 – *Reconstruction results of the Guitar model with different noise levels. Top row : input point clouds. Middle row : SSD solved on our optimized geometric domain. Bottom row : SSD solved on an octree. The parameters of the original SSD are chosen to make the isosurface as smooth as possible.*

we observe that the number of iterations decreases and the quality of the tetrahedra improves after each optimization step.

Combining with mid-edge energy, the optimizer makes a trade-off between the quality of the triangulation and the quality of the remeshed isosurface. In practice, we find that $\lambda_a = \lambda_m = 3$ is a good choice for most of the cases. For challenging cases, for example, when the point cloud has many salient features, λ_m can be increased to prevent the failure of the solver.

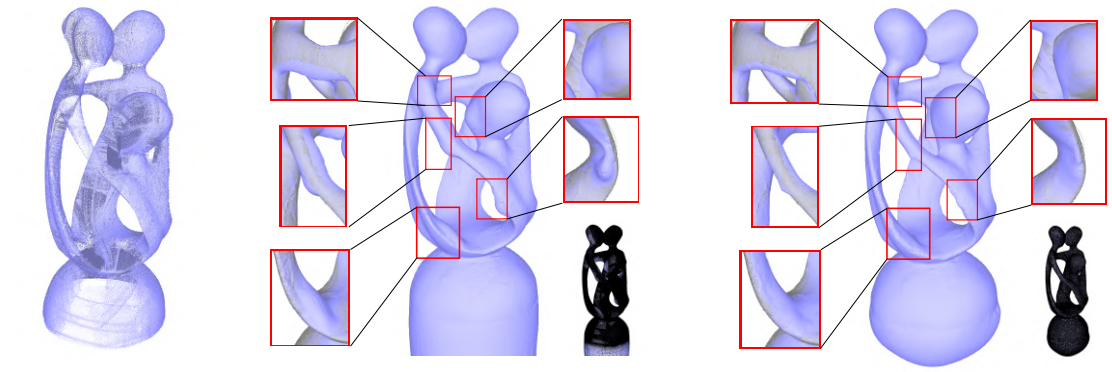
2.4.5 Ablation study

We show the impact of the optimizer by removing one or several components from our approach and compare the produced isosurfaces on the *Horse* point cloud. The following options are tested :



(1a) Indonesian Lady 1 (1b) Original SSD (Octree-based) (1c) Optimized SSD

Figure 2.13 – *Reconstructing the first Indonesian Lady model, with large holes between the legs and near the stomach. The second has many small holes.*



(2a) Indonesian Lady 2 (2b) Original SSD (Octree-based) (2c) Optimized SSD

Figure 2.14 – *Reconstructing the second Indonesian Lady model, with many small holes.*

1. **Non-optimized strategy** : Iterating over solver and refiner without any optimization. One step of ASAP optimization is applied before the solver, otherwise the solver fails to converge.
2. **Mid-edge strategy** : Iterating over solver, optimizer and refiner, while disabling ASAP energy. One step of ASAP optimization is applied before the solver, otherwise the solver fails to converge.
3. **ASAP strategy** : Iterating over solver, optimizer and refiner, while disabling mid-edge energy.

We notice that the ASAP energy is indispensable to the convergence of the solver. The reconstructed isosurfaces and the histogram of their qualities are shown in Figure 2.16. Together with mid-edge energy, they improve a lot the quality of the isosurface, eliminate the influence of outliers and fill the holes with a smooth surface.

TABLE 2.2 – *Reconstructing the Tiki model. Performing several iterations of ASAP optimizations results in faster solver convergence rates, indicating empirically that our optimization improves the conditioning of the solver for a fixed number of vertices.*

Num of ASAP Optimization	0	3	5	10
Num of Solver Iterations	13542	8403	2944	2444
Solver Error	8.67667e-11	9.88958e-11	9.96412e-11	9.94334e-11
Solver Time (s)	2.82103	1.59871	0.569161	0.46091

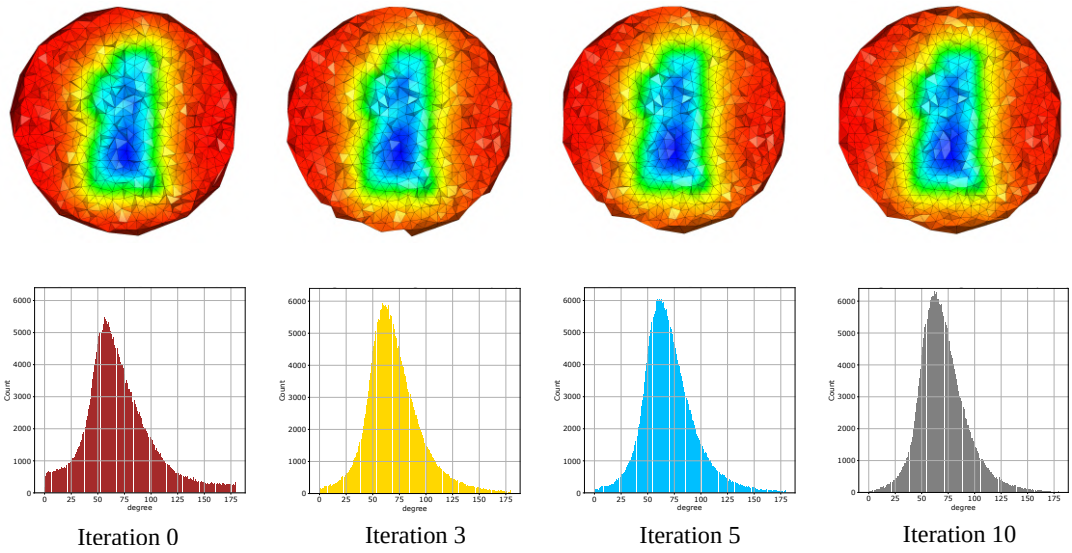


Figure 2.15 – *Reconstructing the Tiki model. The first row shows the clipped domain at iteration {0, 3, 5, 10} and the second row plots the distribution of dihedral angles of the triangulation tetrahedra.*

2.4.6 Timings

Figure 2.17 records the execution times for each model shown. Each color represents one iteration and the final pink color indicates the time for the final solver, which produces the final isosurface on the optimized geometric domain. Compared to octree-based solvers, our algorithm takes more time. However, it is scaling fairly well with the number of input points, and the execution time mostly depends on the target sizing field.

2.4.7 Limitations and future work

With no a priori knowledge about the curvature and local feature size (lfs) of the inferred isosurface, our approach may fail to reconstruct fine details due to insufficient discretization. Figure 2.18 depicts one failure case of our discretized domain. The book in the hand of *Ignatius* is an area with two layers of points with opposite normals. In order to reconstruct it correctly, this

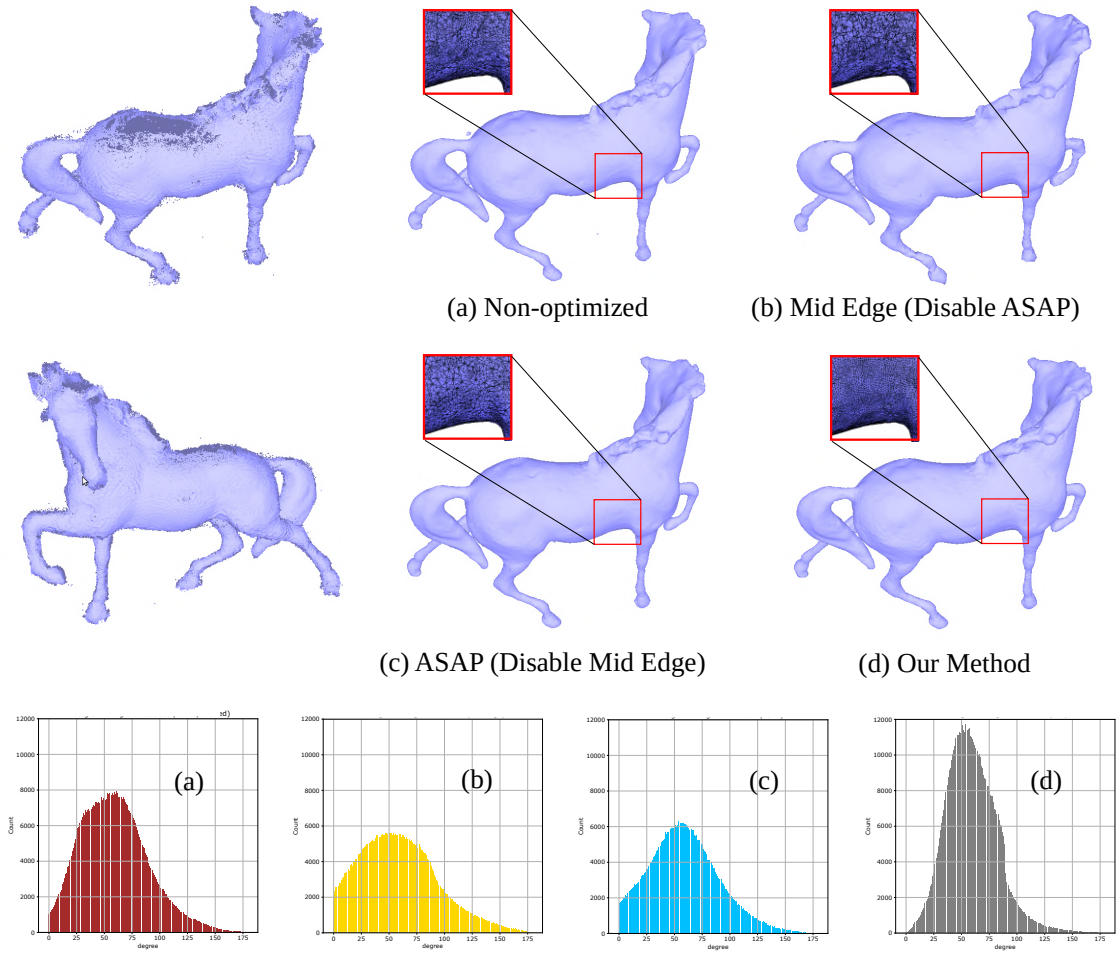


Figure 2.16 – *Reconstruction results of Horse model by disabling one or several components of the proposed approach. The third row plots the distribution of the angles of all isofacets (triangles).*

region should be densely refined to offer enough degrees of freedom. Given that the initial solution does not capture this region, it is not sufficiently refined. As the reconstruction progresses, the region gets denser, but this is insufficient to yield a good solution around this region.

Globally speaking, our current approach presents several limitations. Firstly, the ability of our method to discover salient geometric events - and adapt the reconstruction domain to them - remains bounded to the actual performance of the underlying reconstruction algorithm used at each iteration. Secondly, deriving sufficiency conditions for ensuring convergence remains to be done, and we believe that per-solver approaches could first be designed before addressing the more generic case. Thirdly, our approach focuses on improving the quality of the solvers' outputs at the cost of longer execution timings, but we envision that simple approaches could be used to reduce those. For instance, we rely on Delaunay triangulations, whose structures can change

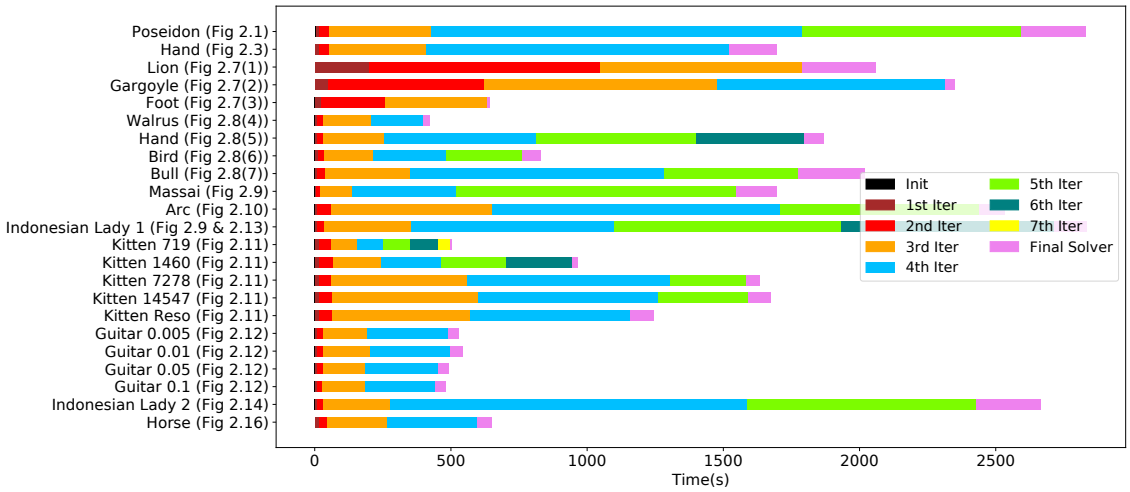


Figure 2.17 – *Timeline of our reconstruction for all models shown. Each color corresponds to one iteration and pink corresponds to the final solver running on the optimized domain.*

unexpectedly when relocating vertices, requiring the solvers' algebraic structures to be updated accordingly even with a fixed vertex count. Adopting other tetrahedron mesh structures could help addressing this problem, while lowering the amount of slivers present in the triangulation, thus improving the conditioning of the solvers. Lastly, we discretize our solvers using piecewise linear elements. In future work, we plan to explore a higher-order variant of this approach, in which the tetrahedron elements of the domain are the support of a non-linear implicit function. We also wish to address piecewise-smooth surfaces with boundaries and non-manifold features.

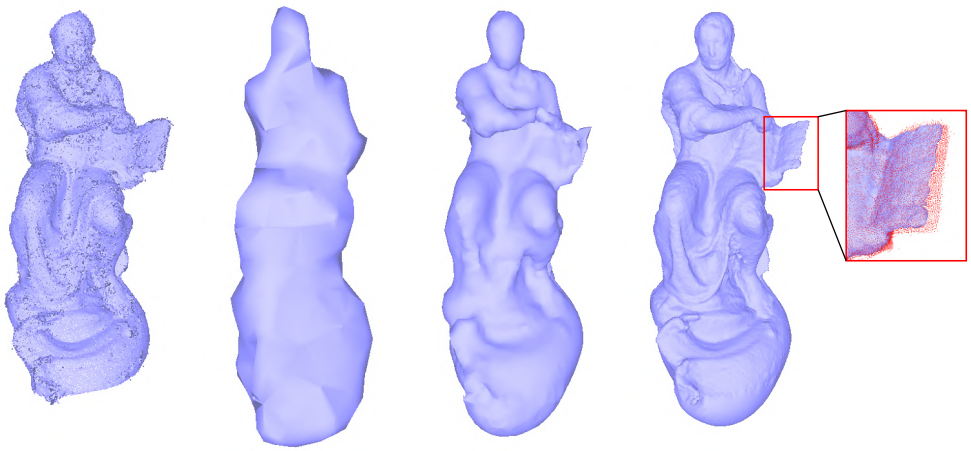
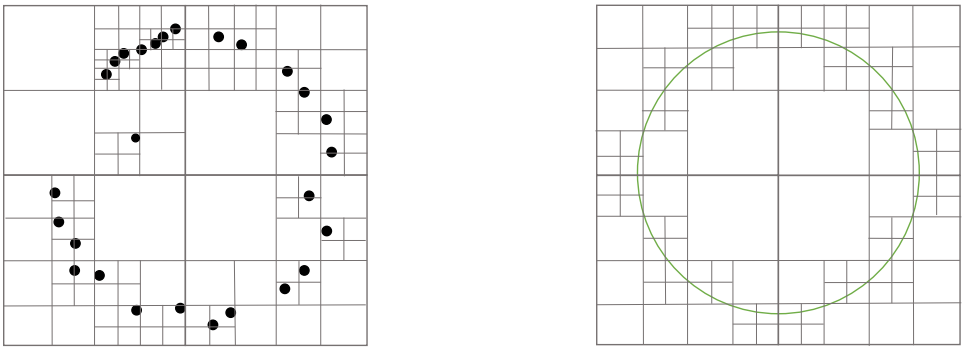


Figure 2.18 – *Reconstruction process for the Ignatius Model. The final isosurface fails to completely reconstruct the book.*

2.5 Octree-based approach

Inspired by the performance of the progressive domain discretization on 3D Delaunay triangulations, in this section we describe an attempt to devise a similar approach for octree-based domain discretization.

Common octree discretization approaches are controlled by two user-defined parameters : (1) maximum tree depth and (2) minimum bucket size. Given a point cloud, the octree stops to refine a node if it is already at maximum tree depth or if it contains fewer input points than the minimum bucket size. As a result, it suffers from the same problems mentioned in Section 2.1.1 when dealing with defect-laden inputs. Figure 2.19 depicts a naive quadtree (2D version of octree) discretized on a noisy circle with missing parts, compared with a quadtree discretized on a perfect full circle. Solving a reconstruction problem on such a naive quadtree leads to a badly discretized implicit function and badly extracted isosurface.



(1) Common octree discretized on point cloud

(2) Ideal octree

Figure 2.19 – *A naive quadtree discretization on a 2D noisy circle and a perfect quadtree discretization on a full circle.*

Compared to Delaunay triangulations, octrees have fewer degrees of freedom. A binary decision must be made for each octree node : refine or stop. Motivated by the successful application of neural networks in various classification tasks, we devise a learning-based approach for performing progressive octree discretization. Our neural network predicts a binary occupancy prediction for a node given the information of its parent and the neighbors of its parent, along with a feature descriptor that will be used as prior information of its children. Instead of taking the whole octree as the network's input as in previous work, our neural network acts on each node, regardless of its depth.

2.5.1 Related work

Voxel-based neural networks ([ZT18, SZT19, LTLH19], to cite a few) have been well explored in the last five years. Convolutional neural networks can be easily generated from 2D images to 3D voxels by adding a new dimension. However, such an extension highly restricts the scalability of voxel-based neural networks since we represent three-dimensional binary sparse matrices by real value dense matrices. There are two common solutions : (1) replacing the grid with octrees, or (2) introducing new sparse operators for grids.

Octree. A careful design must be considered for octree-based neural networks. Different from grids, octrees are usually implemented by a series of pointers, which complicates the querying of neighbor nodes and the implementation of layers.

Riegler et al. [RUG17] proposed OctNet in 2017, which implements an octree as a collection of shallow octrees placed along a regular grid. The position of each node is encoded into bit-strings. Three efficient operators are defined based on this data structure : convolution, pooling and unpooling. This approch is evaluated on the following tasks : 3D classification, point cloud orientation and 3D segmentation. Wang et al. [WLG+17, WSLT18] proposed successively O-CNN and Adaptive O-CNN. The octants of each level are associated with shuffle keys and then sorted in ascending order, which facilitates storing octant features in hashmaps. An encoder-decoder with a skip connection is proposed for their data structure. Tatarchenko et al. [TDB17] proposed an Octree Generating Network (OGN), which learns occupancy values of octants to guide the refinement of the octree. Similarly, their octree is stored using hashmaps of index-value pairs. A feature propagation step is performed for octants labeled as “mixed” states.

In conclusion, existing octree-based methods take commonly an octree as a data structure, which requires complex network structures to perform neighbor querying and GPUs with large memory. Instead, our approach operates on octants, making it flexible and simple.

Sparse operator. Choy et al. [CGS19] proposed and implemented Minkovski Engine, an auto-differentiable PyTorch library for convolving spatially sparse tensors, which fits well 3D grids and octrees. The resulting multi-threaded generalized sparse convolution operator, implemented on GPU, highly improves the efficiency of convolutional operators on sparse tensors.

2.5.2 Method

Our approach takes a raw 3D point cloud as input and a maximum octree depth as parameter, and generates a discretized octree as output. Specifically, it is composed of three main components : octree refinement, neighbor search and neural occupancy prediction. For each octant, the neighbor search component finds the features of its neighbors at the parent level and sends the information to the neural occupancy prediction component. The prediction component learns a

high-dimensional feature from the neighbor information and the inside input points of the current octant, and predicts an occupancy prediction. The octree refinement component refines the current octant if the prediction is above a threshold and then stores the feature in a hashmap for further computation. We provide next a detailed description of each component.

2.5.2.1 Neighbor search

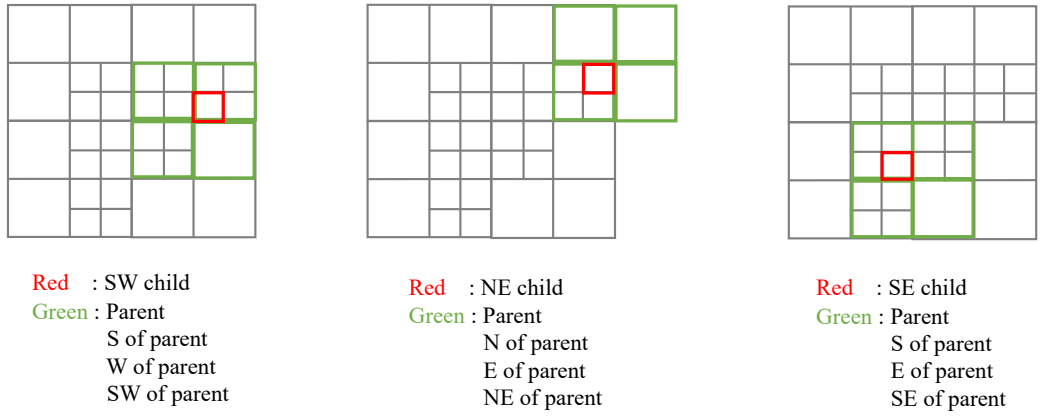


Figure 2.20 – *Given a query octant, the corresponding directions of the neighbors are indicated by the relative position of the octant and its parent.*

We design a neighbor search approach based on the relative position of a query octant and its parent. For one query, there are at most eight surrounding octants whose depths are smaller than the current octant, including its parent octant. We define the vector $\overrightarrow{k} = sign(c_{n_i} - c_{p(n_i)})$ where c_{n_i} denotes the coordinate of the query center, and $c_{p(n_i)}$ denotes the coordinate of its parent. Given the width of the query w_{n_i}, we then search for the nodes (with smaller depth than the query) containing the following set of points :

$$m_j = c_{n_i} + w_{n_i} \cdot \overrightarrow{k} \cdot \Big((j \mid 4) \bmod 2, (j \mid 2) \bmod 2, j \bmod 2\Big), \quad 0 \leq j \leq 7. \tag{2.18}$$

If a node is outside of the bounding box of the octree, we set it to a null pointer. We show three examples on a 2D quadtree in Figure 2.20 and four examples on a 3D octree in Figure 2.21. The time complexity of the neighbor searching step is $O(n)$ where n denotes the maximum octree depth.

2.5.2.2 Occupancy prediction

The prediction component is composed of three sub-networks : a pretrained and froze PointNet++ [QSMG17] that extracts an initial feature from the input points inside the query octant, a

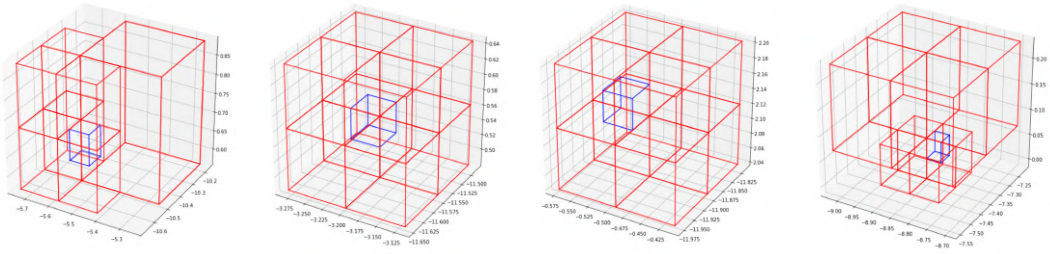


Figure 2.21 – *Some examples of the located octant neighbors (in red) given the query octants (in green).*

convolutional feature merging network which combines the initial feature and the features of its neighbors, and a decision maker which predicts an occupancy probability based on the combined feature. Figure 2.22 depicts the detailed network structure.

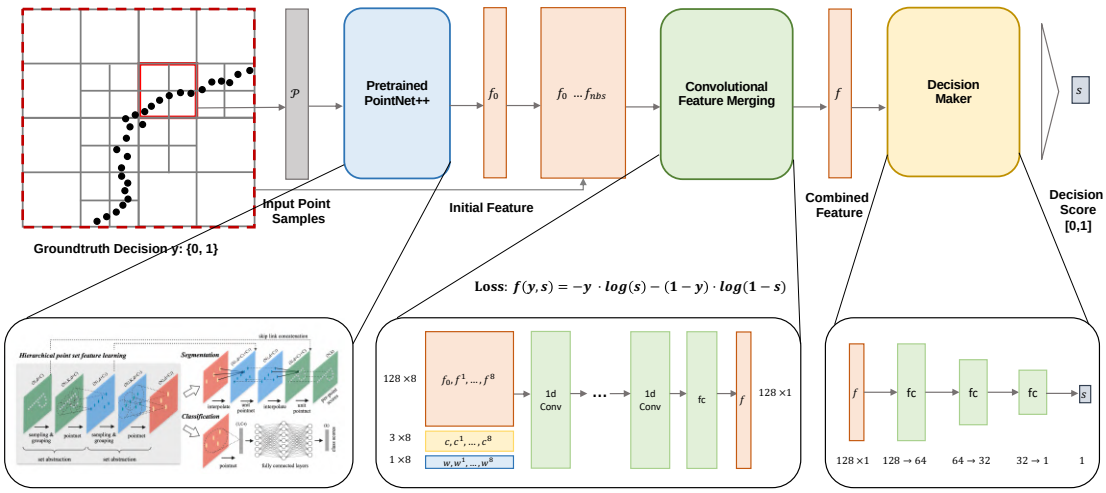


Figure 2.22 – *Network structure of the neural occupancy prediction component.*

As the cardinality of the input points for the pre-trained PointNet++ is fixed to be 1024, the points inside the query octant must be subsampled if it contains more than 1024 points, and upsampled otherwise. Subsampling is implemented via farthest point sampling (FPS) and upsampling is performed by duplicating the points with a small percent of random noise. The output of the subnetwork is a 128-dimensional feature vector that describes the shape of the inside points.

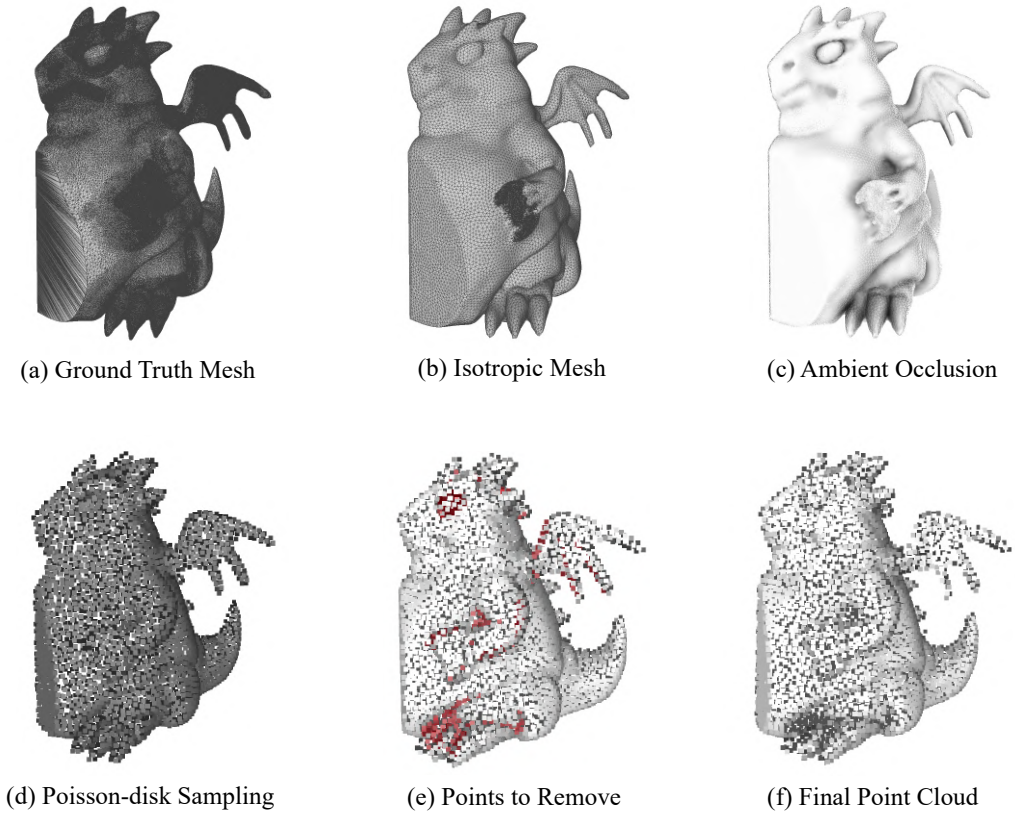
2.5.2.3 Octree refinement

The octree is first fully refined to depth 2. We then process the octants by levels using the prediction component. We adopt a naive refinement choice that refines all octants whose prediction

probabilities are larger than a threshold after all octants of a certain level are processed. More choices are provided in the analysis and discussion section.

2.5.3 Experiments

Dataset. We select 20 models from the Thingi10k Dataset [ZJ16]. We simulate a virtual scanner by first estimating the ambient occlusion of the isotropically remeshed groundtruth surface, and then perform a Poisson-disk sampling to generate a 3D point cloud. After transferring the ambient occlusion from the mesh to the sampled point cloud, we select points whose ambient occlusion values are greater than a user-defined threshold. The resulting point cloud has a denser sampling rate around regions exposed to cameras and a sparser sampling rate around regions under occlusion. A small percentage of noise is added to the point cloud when it is loaded. The groundtruth octree is discretized to a user-defined depth on the surface. Figure 2.23 provides an example.



(a) Ground Truth Mesh (b) Isotropic Mesh (c) Ambient Occlusion

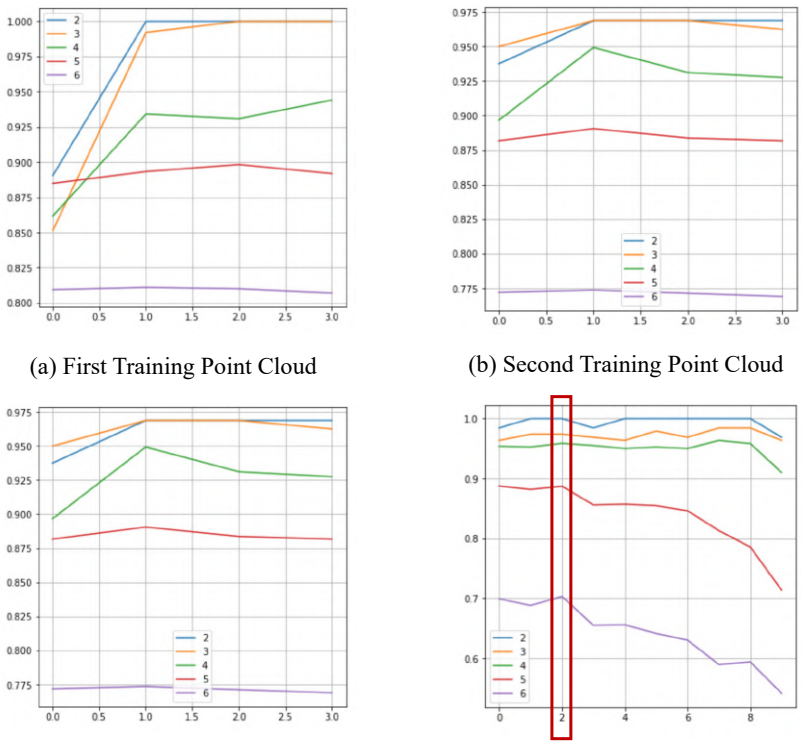
(d) Poisson-disk Sampling (e) Points to Remove (f) Final Point Cloud

Figure 2.23 – *Illustration of our virtual scanner for generating point clouds with defects from ground truth mesh.*

Training approach. The network is trained mesh by mesh and layer by layer. Training one mesh at a time reduces the memory footprint, and we porocess each layer successively since the

children's features are dependent on the ones of their parents. For each pair of input point clouds and surface mesh, we extract initial features for depth 2 and then train the network from this depth to the deepest non-leaf depth. The network saves all combined features until the prediction accuracy reaches a certain threshold or a maximum number of iterations.

Experiments. We train our network on 3 point clouds, using an NVIDIA GeForce GTX 1070 Ti 8G, and validate the model on one point cloud. Training takes 10 hours for 10 epochs. Figure 2.24 plots the training and validation accuracy. The training criteria quickly improve during the first two epochs and then remain stable in the following epochs. However, the validation accuracy decreases rapidly, indicating that the network is overfitting the training dataset.



(a) First Training Point Cloud

(b) Second Training Point Cloud

(c) Third Training Point Cloud

(d) First Validation Point Cloud

Figure 2.24 – *Running accuracies on the training point clouds and the validation point cloud. Note that the x-axis is the number of epochs and the y-axis is the accuracy. Different colors correspond to different depths of the octree (from 2 to 6).*

Figure 2.25 depicts our neural network at work on another point cloud. We observe that the network achieves a good accuracy for shallow depths, but accuracies for deeper depths are unsatisfactory. In addition, the network remains slow, making it hard to enlarge the training dataset.

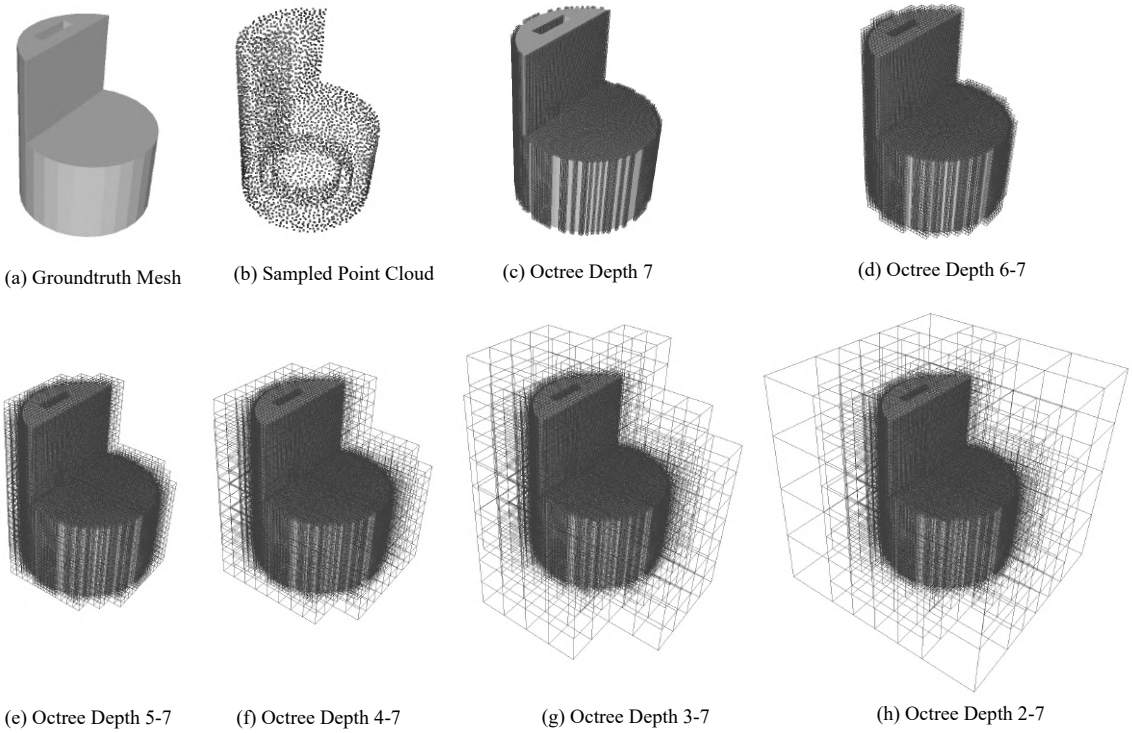


Figure 2.25 – *Nodes of the discretized octree on a test point cloud by the trained neural network with different depths.*

2.5.4 Analysis and discussion

We now discuss the remaining issues with the existing method and potential directions for improving it.

Firstly, the errors introduced in earlier stages of the refinement process have an irreversible impact on the following stages. If one node is not refined, it can never be refined later. A possible direction is to assign a loss weight that decreases with when the octree depth increases. In addition, a better octree refinement approach could help improve this issue. For example, we can introduce a smoothing function that enforces some octants to be refined if the probabilities of their neighbors satisfy several conditions. However, both mentioned propositions add soft constraints to the problem thus this issue cannot be fully solved.

Secondly, the current network is limited in terms of scalability. The number of octants of an octree can easily be over 500,000, especially when it goes deeper. As a result, training a neural network onto a large dataset and constructing an octree is both memory- and compute-intensive. Introducing parallelization may alleviate the problem but is still not sufficient for trainings on very large datasets.

Thirdly, PointNet++ is not satisfactory for computing node features, though it has many successful applications of extracting point cloud features. When the octree goes deeper, the number

of inside points decreases rapidly and its shape is far from a complete point cloud. Duplicating points with random noise makes the extracted feature even more unstable. To deal with this issue, we intend to discretize the query octant using a small-size grid and then train a neural network structure for 3D dense tensors from scratch.

Lastly, we use the same neural network for all octree depths while the patterns for discretizing a node change. It is possible to add a branch of the sub-network which learns a depth approximation of the current octant to the root and unifies the local features according to this approximation, whereas a proper loss function and a suitable network structure must be carefully chosen.

2.6 Conclusion

In this chapter, we proposed two progressive domain discretization approaches for implicit surface reconstruction approaches : triangulation- and octree-based. The triangulation-based approach takes an initial 3D Delaunay triangulation of the domain and an implicit solver. It iterates over three main steps (solve, optimization, adaptive refinement), all steps being designed to cooperate with each other and improve the conditioning of the solver, and the quality and complexity-distortion tradeoff of the output isosurface mesh. In such a progressive approach, the implicit solver is no longer used once as in previous work, but iteratively as a means to discover more and more details for the isosurface.

In the octree-based approach, we make an attempt to discretize an octree from coarse to fine using a deep neural network. A node feature is first extracted from a pre-trained point-based neural network and is then merged with the features of its neighbors. Based on the merged feature, a binary decision is made by an MLP which decides whether the node is refined.

The benefits of optimizing a discrete domain for global implicit surface reconstruction approaches are that the global system has a better conditioning and that the output mesh is adapted to the intrinsic geometric complexity of the reconstructed surface, instead of to the input point set density, as in previous work. An improved discretization can also be leveraged for recovering sharp features and dealing with a wide range of defects of the input 3D point cloud.

CHAPITRE 3

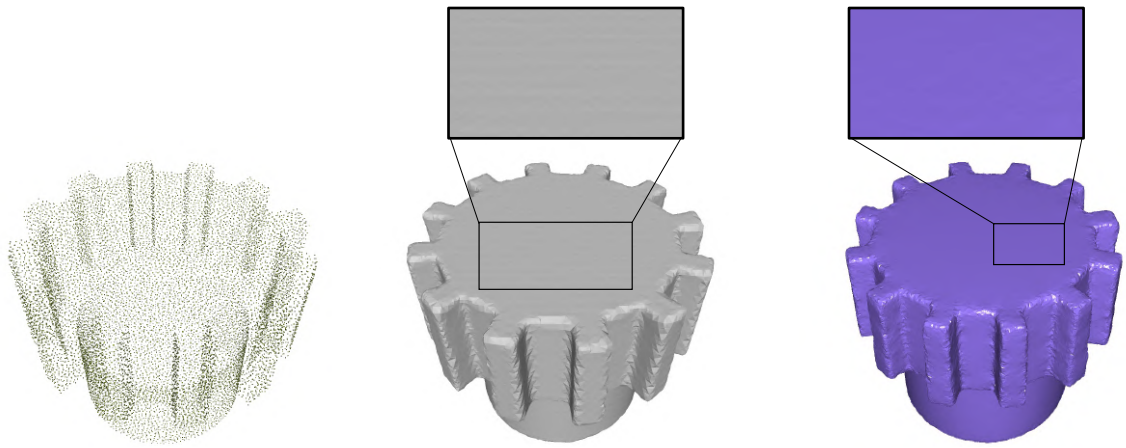
Primitive-guided implicit reconstruction on progressive discrete domains

In this chapter, we explore a progressive implicit reconstruction approach with the capability to refine the implicit function and its representation, where the most ill-posed parts of the reconstruction problem are postponed to later stages of the reconstruction, and where the fine geometric details are resolved after discovering the topology. More specifically, we leverage an initial primitive detection step to constrain the implicit function on canonical primitive areas, then interleave global implicit solves with refinement and optimization of a 3D tetrahedron mesh. The latter is used to represent an implicit function that is constrained to respect the detected primitives. The resulting discrete domain is adapted automatically to the point set : coarser near detected primitives to reduce memory consumption and denser near free-form areas to improve accuracy for curved shapes.

3.1 Introduction and related work

3.1.1 Introduction

Most surface reconstruction approaches rely on a single prior to regularize the problem. Surface reconstruction methods based on global smoothness have been widely used in many applications since it is versatile especially when data are nonuniform, incomplete or noisy. They always guarantee global consistency but the cost of reconstructing even a simple canonical primitive may be prohibitive. Undesired artifacts may appear near primitives due to the discretization domain and the meshing approach (see Figure 3.1). Fewer alternative approaches have been proposed to support global smooth reconstruction with other priors (e.g., boundary information is taken into consideration in [KCRH20]). Primitive-based surface reconstruction approaches easily reconstruct the detected primitives but finding the relationships between primitives is ill-posed and difficult.



(a) Point Cloud (b) Screened Poisson (c) Progressive Poisson

Figure 3.1 – *Reconstructed surfaces of screened Poisson [KH13] and the proposed progressive Poisson in Chapter 2 on a point cloud containing primitives.*

In this chapter, we propose a progressive approach that combines a global implicit function with primitive-based functions, represented on a data-adapted 3D Delaunay triangulation. Canonical primitives are first detected using an efficient RANSAC-based approach before reconstruction. Starting from an initial coarse triangulation, we repeat the three following steps until a user-specified tolerance error is satisfied : (1) adaptive refining of the triangulation, (2) solving for the implicit function that fits both the canonical primitive and the input points located on free-form parts, and (3) optimizing vertex positions so as to improve the fitting of an isosurface of the implicit function.

3.1.2 Related work

Our approach is related to both global smoothness-based reconstruction and primitive-based reconstruction, which are discussed in Section 1.2. In this section, we focus on a very relevant approach : a structured point set surface reconstruction [LA13] proposed by Lafarge et al. in 2013. The output surface triangle mesh is reconstructed in two steps : (1) regularization and denoising of the input 3D point cloud to yield a structured point set, and (2) extraction of the output mesh by labeling tetrahedra in a 3D Delaunay triangulation via a min-cut formulation. More specifically, the approach takes as input an unoriented 3D point cloud with a configuration of planar primitives. It first associates the adjacent primitive information with each input point and then deduces four sets from the point cloud : planar set, crease set, corner set and clutter set. Points in the planar set are sampled uniformly on each planar primitive. Points in the crease set are sampled uniformly on the intersections of adjacent primitives. Points in corners are found by finding 3-cycles in the primitive adjacency graph.The clutter set contains the remaining input points that are not associated to any planar primitive. They are possibly preprocessed by an outlier removal step. Figure 3.2 depicts an example of a structured point set obtained by this approach.

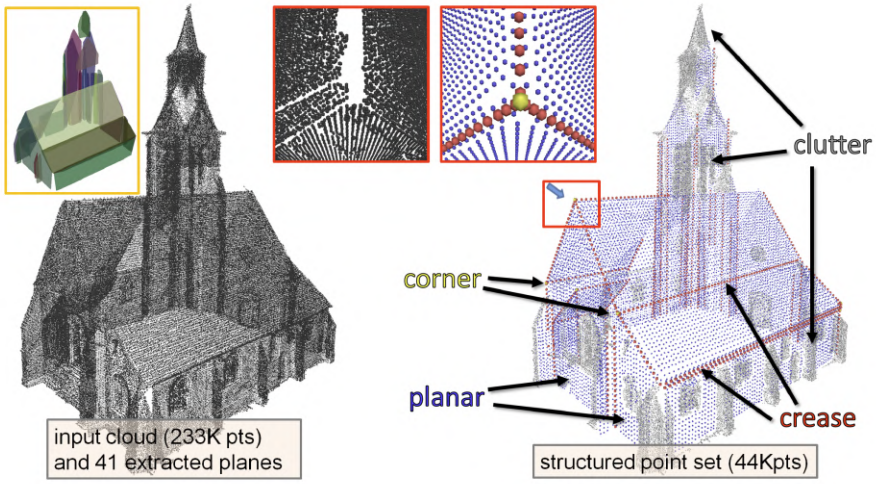


Figure 3.2 – *Image taken from [LA13].*

To extract the output surface mesh, a 3D Delaunay triangulation is initialized inside an enlarged bounding box of the input points and discretized by inserting all points from the structured point set. The set of the tetrahedra is then divided into two non-overlapping subsets : inside or outside tetrahedra, by a graph cut considering both the facet quality and a penalty relating to visibility prediction. Figure 3.3 depicts a reconstruction result on the point cloud of the Statue of Liberty.

This approach is efficient and scales to large 3D point clouds. Nevertheless, we notice the following issues which are not considered in this approach : (1) Only planar primitives are considered ; (2) It cannot complete missing data ; and (3) It cannot deal with shapes with complex occlusions or

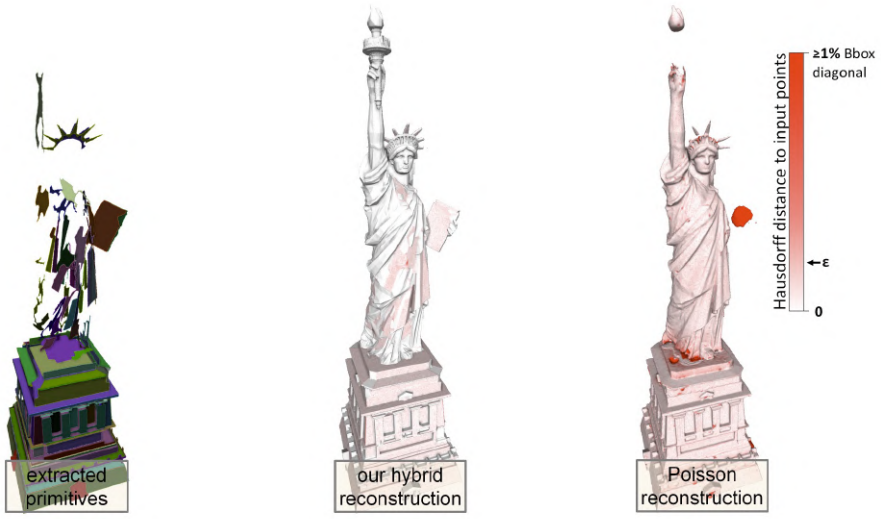


Figure 3.3 – *Image taken from [LA13].*

invisible parts from the scanning directions, since the objective function of the min-cut algorithm highly depends on visibility.

3.1.3 Positioning and contribution

Following the same concept introduced in Chapter 2 but with a departing method, we propose a surface reconstruction approach for oriented 3D point clouds. We extend and enrich the progressive discrete domain approach described in the previous chapter, with primitive information. Our approach offers the following advantages :

- It can deal with multiple types of primitives. We consider three primitives (planes, spheres, cylinders), and other primitives can be handled as well, as long as we can compute the projections and oriented normals of the primitive.
- It benefits from the controllability of global smooth implicit reconstruction approaches, by introducing primitive priors as input. The primitive priors simplify the reconstruction around canonical parts of the model, while the global smoothness priors discover the topology and fuses the canonical with the free form areas.
- It is resilient to both noise and missing data since we take advantage of both priors : all components of our approach are adapted to both free-form and canonical parts.

3.2 Approach

Our approach takes a 3D point set with oriented normals $\mathcal{X} = \{(p_1, n_{p_1}), \ldots, (p_N, n_{p_N})\}$ as input and generates as output a surface triangle mesh $\mathcal{S} = f^{-1}(0) = \{(a_1, n_{a_1}), \ldots, (a_R, n_{a_R})\}$

deduced from an implicit function $f : \mathbb{R}^3 \to \mathbb{R}$ following the pipeline shown in Figure 3.4. The primitive configuration can be either user-provided or generated by our complete pipeline. We assume that the input point set is sampled around a closed 2-manifold smooth surface, i.e., the boundary of a solid. Some measurement noise as well as missing data are tolerated. Similar to our approach detailed in the previous chapter, the main user parameter is a sizing field that provides indirect control on the final mesh complexity.

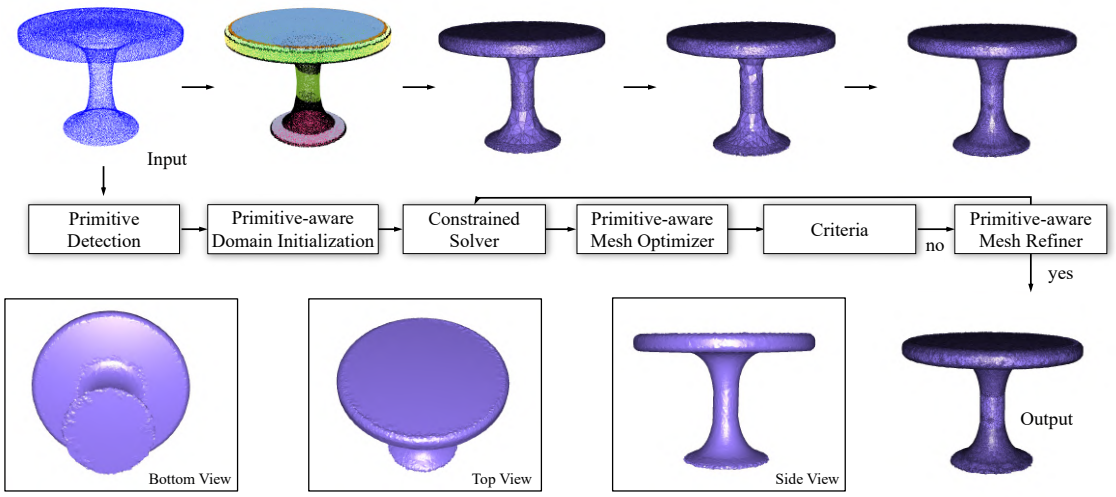


Figure 3.4 – *Overview of the proposed reconstruction approach.*

3.2.1 Primitive detection

We formulate the primitive configuration as a list of labels $\mathcal{L} = \{l_{p_1}, \ldots, l_{p_N}\}$ that indicates the associated primitive labels and a list of primitives $\mathcal{C} = \{C_1, \ldots, C_M\}$. Each primitive should be represented such that the corresponding projection and its normal on the primitive can be returned given any query point in the domain. When this information is not provided by the user, a primitive detection step is performed prior to the progressive reconstruction. Many approaches have been proposed for the detection of canonical geometric primitives from 3D data [KZB19]. We utilize the efficient RANSAC [SWK07] as a default detection method, but it can be replaced by other primitive detection approaches based on region growing [AB94], Hough transforms [DDSD03] or more recent deep learning based approaches [LSD+19].

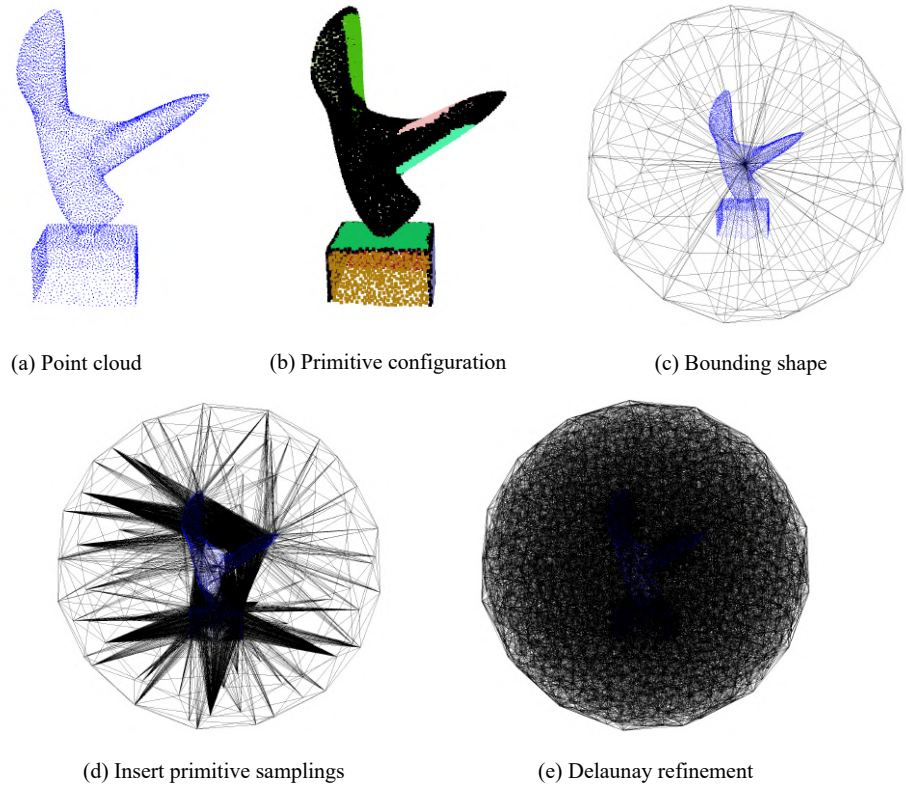
The Random Sample Consensus (RANSAC) algorithm was proposed by Fischler et al. in 1981 [FB81], aiming at estimating the parameters of a mathematical model from a set of observed data that contains outliers. While RANSAC offers relevant properties such as robustness, generality and simplicity, its scalability is limited due to a very large search space and high intrinsic computational complexity. Efficient RANSAC [SWK07] improves significantly the performance

of RANSAC on large 3D point clouds by a hierarchically structured sampling approach for candidate shape generation and a lazy cost function evaluation scheme, which rejects useless primitives at an early stage of the detection process. It is controlled by the following parameters :

1. Maximum angle deviation between the normal of a point and the corresponding normal of its projection onto the primitive ;
2. Maximum distance tolerance between a point and a primitive ;
3. Maximum tolerance of Euclidean distance among all points considered connected on a primitive ;
4. Minimum number of inliers for a primitive ;
5. Probability to miss the largest candidate primitive.

3.2.2 Primitive-aware domain initialization

The domain boundary is discretized by inserting 100 uniformly sampled points of a loose bounding sphere around the input 3D point cloud. Before iterating over the "solver-optimizer-refiner" loop, we initialize the inside of the 3D Delaunay triangulation according to the given primitive configuration.



(a) Point cloud (b) Primitive configuration (c) Bounding shape

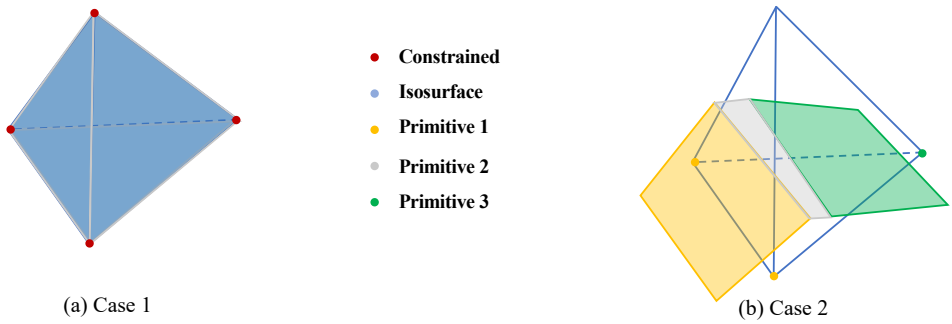
(d) Insert primitive samplings (e) Delaunay refinement

Figure 3.5 – *Primitive-aware domain initialization for an input 3D point cloud.*

In order to guarantee a good isosurface while keeping the mesh complexity reasonable, we perform a subsampling of each primitive. While prior work [LA13] achieves this by computing a binary occupancy 2D grid projected onto the primitive, this operation is limited to planar primitives. Instead, we perform a farthest point sampling of the inliers of each primitive and regularize the sampled points by projecting them on the relating primitive. All sampled and regularized points are inserted into the 3D Delaunay triangulation, and the related vertices are labeled as "constrained" during the whole reconstruction process. A coarse 3D Delaunay refinement according to the circumradius-to-shortest edge ratio [She98] is then performed in the whole domain, in order to obtain a proper initialization for the solver. This choice has several advantages : (1) The areas without ambiguity are determined before reconstruction ; (2) The unknown areas are left to the later stages of the reconstruction ; and (3) The topology and relationships between the areas are discovered progressively throughout the main loops. Figure 3.5 shows the domain initialization for an input 3D point cloud.

3.2.3 Primitive-aware solver

During domain discretization, the vertices of the triangulation are separated into two sets : constrained and unconstrained vertices. Constrained vertices are always placed onto the primitives where the expected implicit function value is zero. Instead of adding a soft constraint rewarding the function values to be zero, we design a constrained solver where the function values of all constrained vertices are enforced to zero, and the corresponding variables are removed from the linear system before solving. We experimented with either Poisson [KH13] or SSD [CT11] solvers, and observed that the Poisson solver is less sensitive to the quality of the 3D triangulation and complexity of the constrained areas. Note that the constrained vertices must lie on the isosurface. Although constraining the (signed distance) function values of the vertices of an edge determines the location of the isosurface locally, it is quite delicate to merge it with either a smoothed indicator function (in the case of Poisson reconstruction) or a signed smooth distance function (in the case of SSD reconstruction). Merging functions with different natures introduces bad constraints on adjacent vertices of constrained vertices and may cause failure of the primitive-aware solver.



(a) Case 1 (b) Case 2

Figure 3.6 – *Illustration of bad cases in the triangulation.*

Before solving, we address two bad cases in the triangulation (depicted Figure 3.6). First, a tetrahedron cell should not have its four vertices constrained, otherwise such cell translates into an iso-solid instead of an iso-surface. Second, a cell should contain either zero or one primitive, not more. Otherwise there are not enough degrees of freedom to recover the relations between primitives. For instance if the cell in Figure 3.6(b) is not refined, the gray primitive can never be discovered in the following steps. To eliminate the cells violating the two above rules, we repeatedly iterate over all cells and insert the circumcenter of the violating cells. Since such a cell is immediately broken into 4 cells once the circumcenter is inserted as unconstrained vertex, this operation terminates in a finite number of steps when assuming sufficient separation between the detected primitives.

Generally speaking, removing a constrained variable x_i (corresponding to constrained vertex v_i) from a linear system $\mathbf{AX} = \mathbf{B}$ requires subtracting from B_j the amount $\mathbf{A}_{ij}\mathbf{x}_i$ for all adjacent vertices v_j (as shown in Figure 3.7). Seeing all the constrained variables are set to 0 in our case, we simply constrain the solver by removing the rows and the columns of the constrained variables from the linear system. This operation has two advantages : (1) the assembling of the linear system is accelerated and (2) the size of the linear system is reduced. Note that octree is not eligible for this operation because that the value of the implicit function becomes a weighted sum of the values of the variables in all octree depths, thus the variables cannot be determined before solving the linear system.

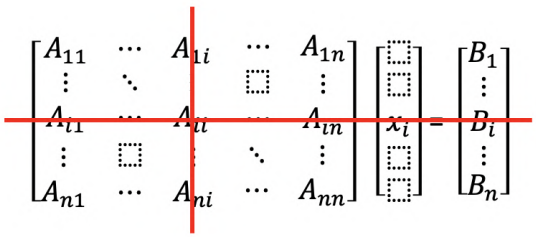


Figure 3.7 – *Removing variables from the constrained linear system.*

3.2.4 Primitive-aware optimization

The primitive-aware optimizer proceeds similarly to the one described Section 2.3.3, with two modifications to the mid-edge and damping terms of the objective function.

Mid-edge objective Instead of encouraging the isosurface to intersect all tetrahedra through their edge midpoints, the isofacets containing three constrained vertices are ignored such that there are two layers of tetrahedra both on the positive and negative sides of the implicit function, which intersect on the primitive-aware isofacets. The tetrahedra containing two constrained vertices, one negative vertex and one positive vertex, are considered as belonging to a transition area,

i.e., the constrained vertices remain unmovable and the other iso-point intersects the midpoint of the edge with opposite function values. The tetrahedra intersected by the isosurface without any constrained vertex are considered as belonging to a free-form area and are optimized as in the previous chapter. In summary, the only modification for the mid-edge objective concerns the edges with both positive and negative function values. Figure 3.8 depicts an example before and after optimization.

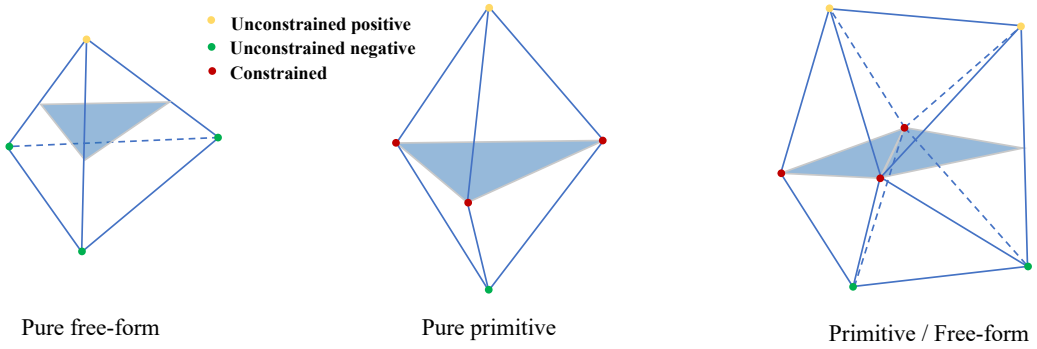


Figure 3.8 – *Three cases for primitive-aware optimization.*

Damping objective To prevent the constrained vertices from moving after the optimization, we increase to a large extent (100 times) the damping energy for all constrained vertices.

3.2.5 Primitive-aware refinement

The primitive-aware refinement step builds upon the refiner described Section 2.3.4, with one modification. Instead of considering all tetrahedra intersecting the isosurface, only the ones containing at most two constrained vertices are considered and refined according to the user-defined sizing field.

3.3 Experiments

The proposed approach is implemented in C++, using the CGAL library for 3D triangulations and geometric computations [The21], and the Eigen library for linear algebra and solvers [GJ+10]. The experiments are conducted on a 3.40GHz Intel Xeon CPU E3-1231 using a single CPU thread.

Figure 3.9 shows the progressive reconstruction process on the “Arc de Triomphe” 3D point cloud. The primitive areas are well reconstructed from the beginning, then the more ambiguous areas are progressively discovered and reconstructed throughout the "solver-optimizer-refiner" loop.

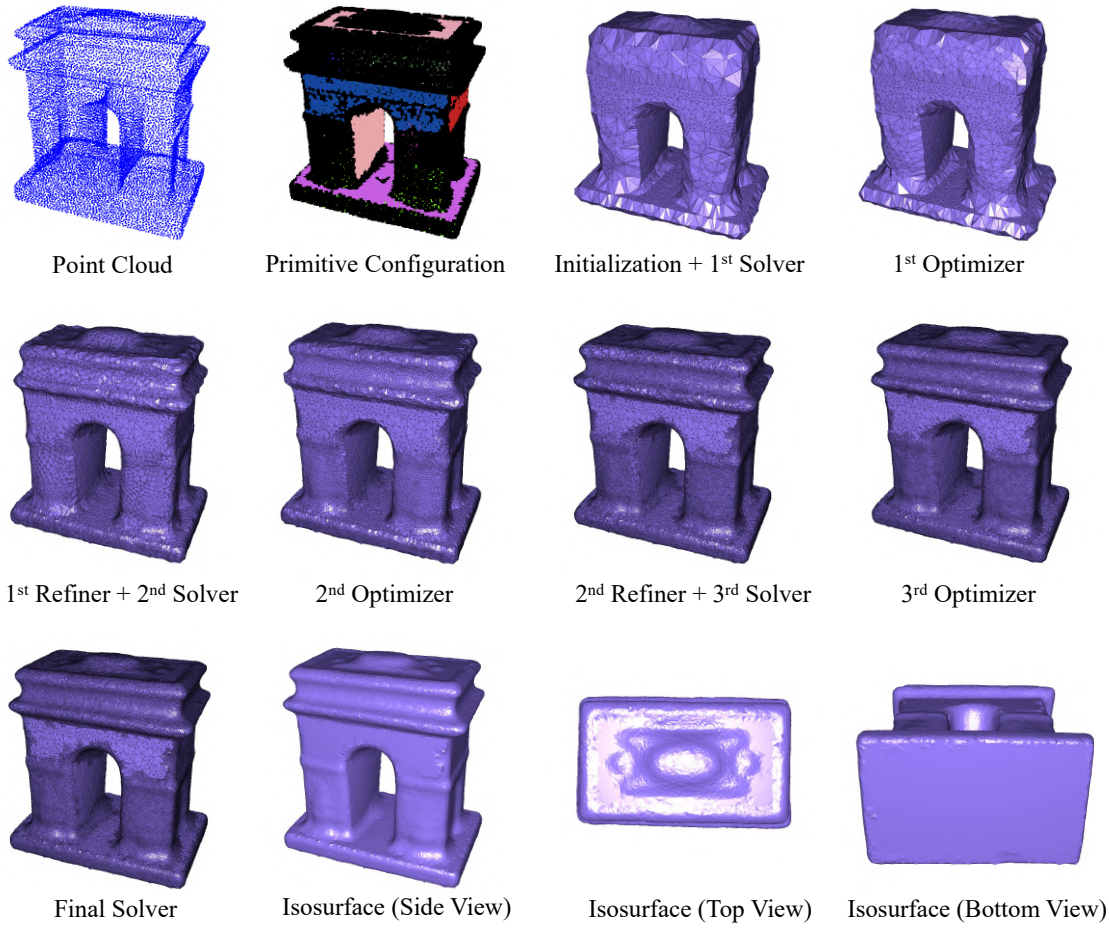


Figure 3.9 – *The whole reconstruction process applied to an input 3D point cloud.*

Figure 3.10 depicts a gallery of reconstructed surfaces. The proposed approach is particularly suitable to reconstructing hybrid shapes mingling free-form and canonical parts, such as sculptures, architectures or man-made objects. The primitives are faithfully reconstructed. The topology and free-form areas are well recovered by the constrained Poisson solver. Nevertheless, we observe that the results exhibit artifacts on transition areas between primitives, or between primitives and free-form areas. We discuss the origin of such artifacts and potential improvements in Section 3.4.

The screened Poisson reconstruction (SPR) approach [KH13] is often struggling to reconstruct nearby surface sheets with opposite orientations. The proposed constrained solver largely improves the reconstruction quality for such cases by setting constraints from primitives. Figure 3.11 depicts a comparison of our approach and SPR on a point cloud with oriented normals, sampled on a pinched sphere. The reconstruction is constrained by four detected cylinder primitives around the central parts of the pinched sphere, guaranteeing a good separation of two sheets of isosurfaces. SPR is not able to find an isosurface with the correct topology there, regardless of the chosen

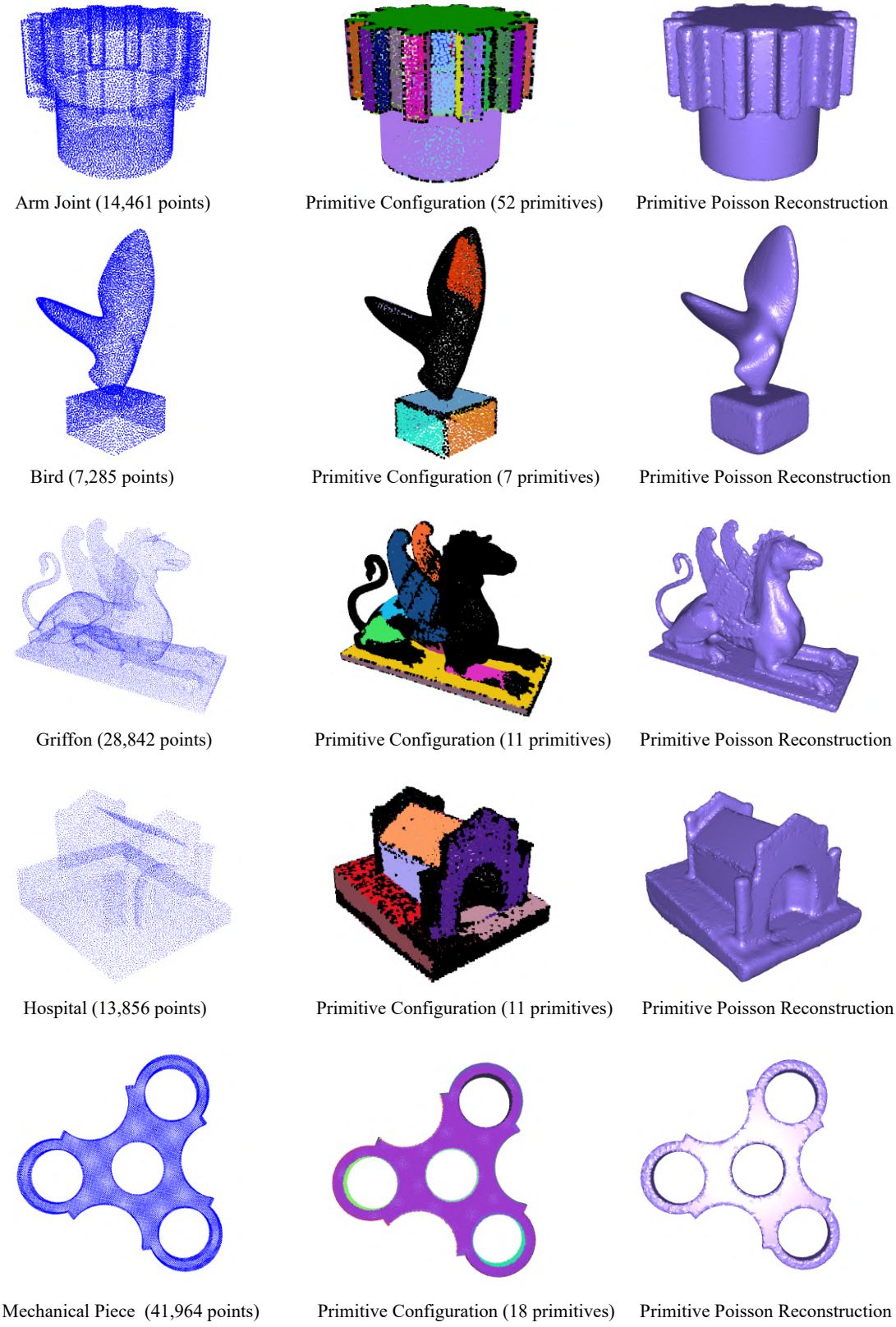


Figure 3.10 – *Reconstruction gallery.*

parameters. The capability to reconstruct nearby sheets is relevant and can render the reconstruction task more predictable depending on the detected primitives. Even in cases where no primitive information is detected, users can manually select few points located on the expected surface and then reconstruct while matching the relating constraints.

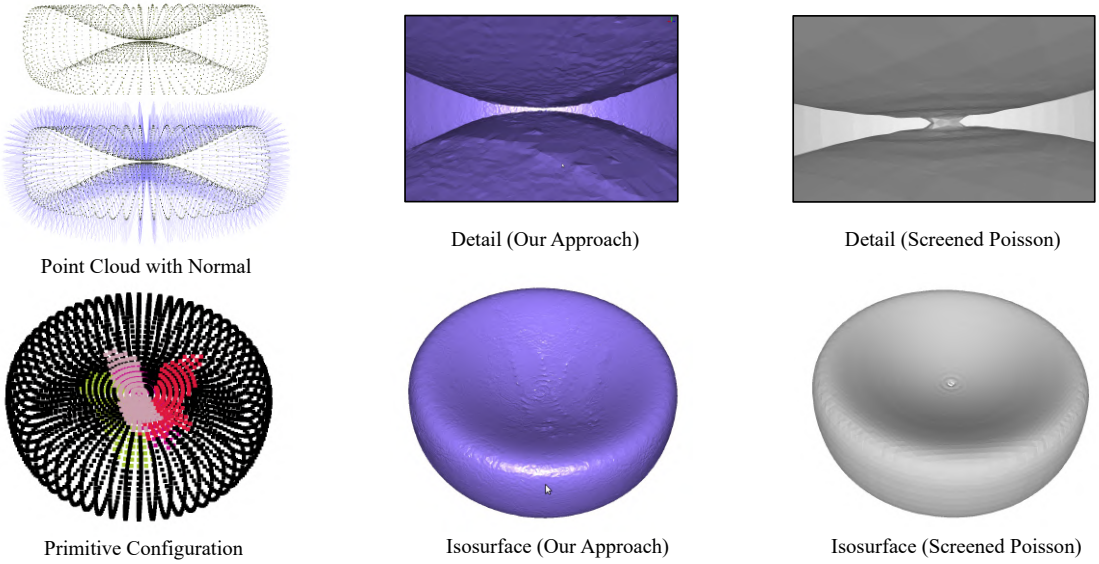


Figure 3.11 – *Reconstructing a pinched sphere with oriented normals.*

Figure 3.12 compares our primitive-guided approach with progressive Poisson reconstruction approach proposed in Section 2. We observe that the primitive one reconstructs smooth primitive regions with much fewer isofacets. The computation is highly accelerated compared to the progressive one because of two reasons : (1) The reconstruction needs fewer iterations to converge. (2) The constrained solver has fewer variables (i.e. free vertices in the discrete domain).

3.4 Discussion

In this chapter, we proposed a primitive-guided global smooth implicit reconstruction approach for oriented point clouds. We leverage an initial primitive detection step to initialize the discrete domain and constrain the implicit function on canonical primitive areas. We then interleave constrained global implicit solves with refinement and optimization of the 3D tetrahedron mesh used to represent an implicit function that respected the detected primitives. Despite the achieved improvements, three issues must be faced to improve our approach.

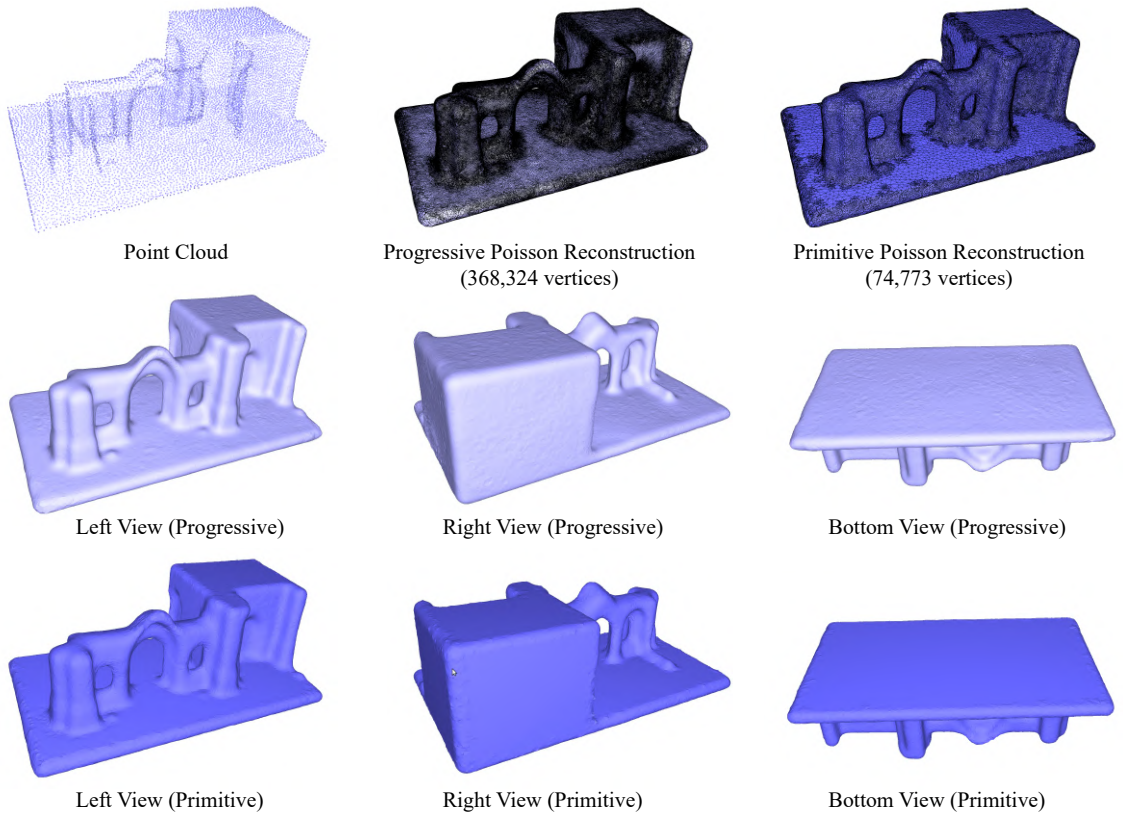


Figure 3.12 – *Comparison between progressive Poisson reconstruction and primitive Poisson reconstruction on a point cloud.*

Issue 1 : How can we better blend constrained with smooth functions ?

Currently, blending of the constrained and smooth functions is handled by the constrained refiner. However, refinement is insufficient as the constrained function matches the primitive prior while the smooth function matches the smooth prior. More specifically, decreasing the data fitting parameter in SPR lends to a smoother function while the constrained function remains unchanged. Figure 3.13 depicts irregular artifacts caused by blending around the intersections between different primitives (black cluster in the primitive configuration). Solving this issue requires introducing a buffer area around such areas, where the two functions are properly blended to achieve a good balance.

Issue 2 : How can we preserve the constrained region during reconstruction ?

Another artifact is depicted by Figure 3.14. When the sampling density of the primitive areas is low, the constrained parts are contaminated by adjacent free-form areas during the progressive reconstruction. Conversely, when the sampling density is high (as in above examples), there is

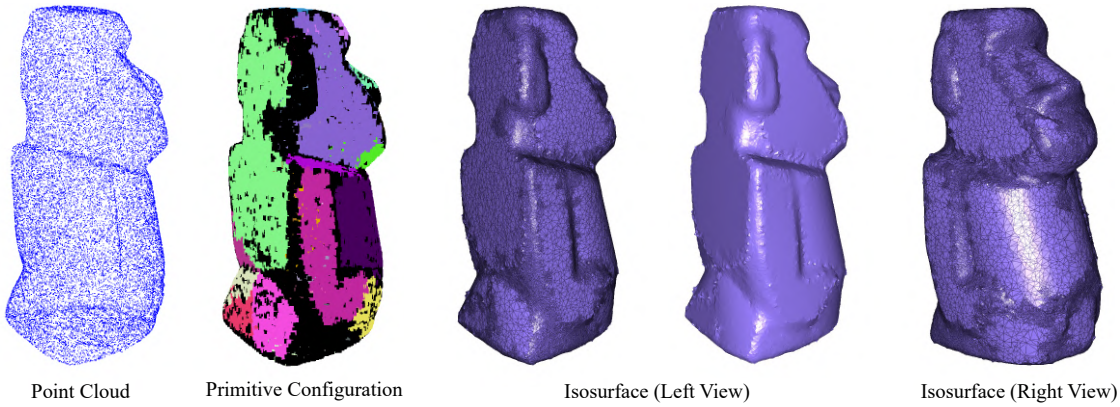


Figure 3.13 – *Reconstruction of the Tiki model.*

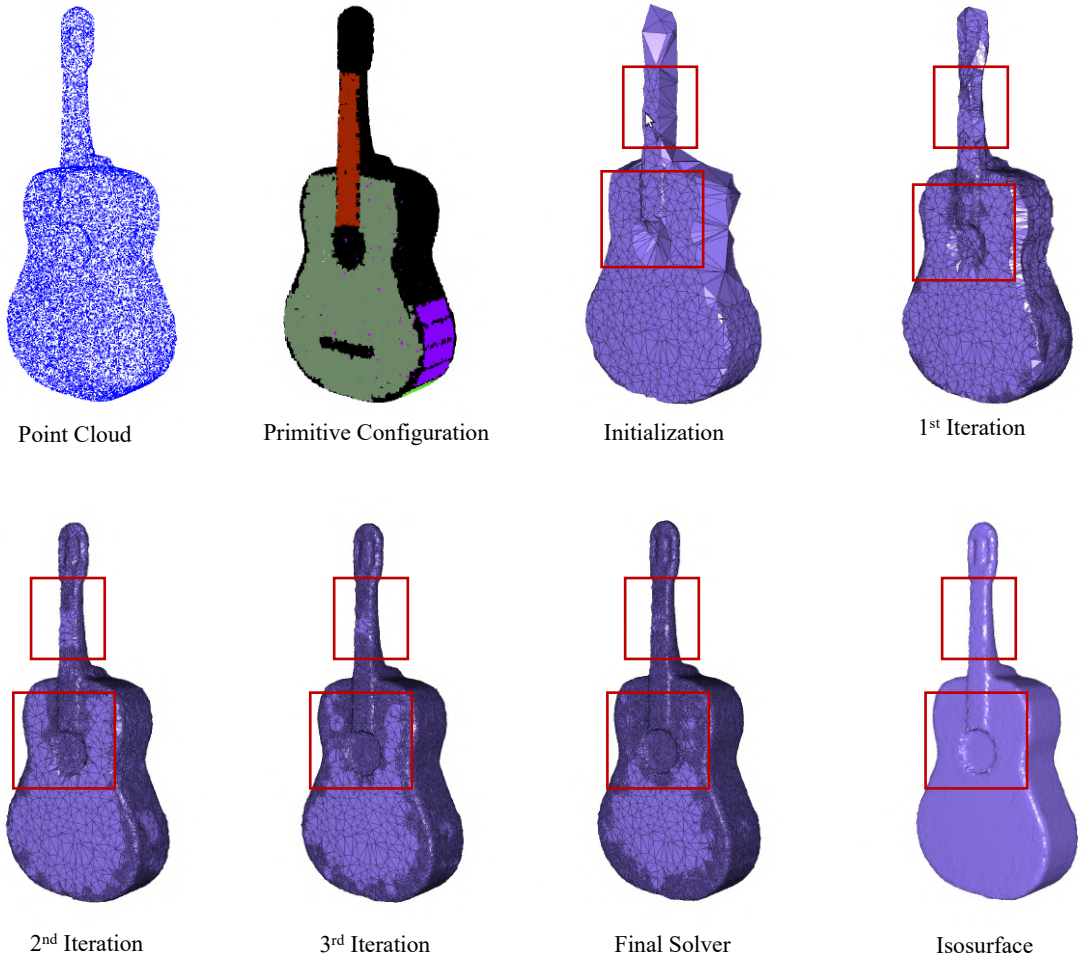


Figure 3.14 – *Reconstruction of the Guitar model.*

no guarantee that all constrained triangles are not destroyed by the refinement step. The reason is that during refinement, we break badly shaped tetrahedra by inserting their circumcenter into the triangulation. When a circumcenter is outside the tetrahedron, other constrained triangles may also be refined. We must devise a better refinement approach that avoids such cases.

Issue 3 : How can we retrieve sharp features ?

Although we have introduced primitive priors into the reconstruction, our output implicit function remains globally smooth. Reconstructing sharp features is indispensable for reconstructing piecewise smooth isosurfaces. We can reconstruct sharp features by inserting constrained vertices located on the sharp features and connecting them with primitive constrained vertices. In other words, it remains to detect sharp feature points from the input 3D point cloud. We propose next a deep learning based sharp feature consolidation approach, which can be considered as an additional input to this approach, in order to reconstruct piecewise-smooth surfaces.

CHAPITRE 4

Sharp feature consolidation via displacement learning

Detecting sharp features in raw 3D point clouds is an essential step for designing efficient priors in several 3D Vision applications. This chapter presents a deep learning-based approach that learns to detect and consolidate sharp feature points on raw 3D point clouds. We devise a multi-task neural network architecture that identifies points near sharp features and predicts displacement vectors toward the local sharp features. The so-detected points are thus consolidated via relocation. Our approach is robust against noise by utilizing a dynamic labeling oracle during the training phase. The approach is also flexible and can be combined with several popular point-based network architectures. Our experiments demonstrate that our approach outperforms the previous work in terms of detection accuracy measured on the popular ABC dataset. We show the efficacy of the proposed approach by applying it to several 3D vision tasks.

4.1 Introduction and related work

4.1.1 Introduction

For scanned or Computer-Aided Design (CAD) 3D models, a sharp feature usually refers to creases and corners where the surface is not smooth. Recognizing such sharp features from raw point cloud is a preliminary step for several point cloud processing tasks such as surface reconstruction [ÖGG09, GSH+07, ABK98, GG07, SYM10], extraction of feature graphs [LB16, NLNZ16] or semantic segmentation [GXT+21, LAK20]. With the development of point cloud acquisition techniques and the release of well-annotated 3D datasets [KMJ+19], many data-driven approaches have been proposed in recent years. Most previous approaches formulate sharp feature detection as a classification problem, which leads to several issues : (1) Feature points are not located exactly on sharp features in general, which leads to incomplete features ; (2) To deal with this issue, several approaches introduce a distance threshold that defines the maximum distance between a feature point to the nearest sharp feature. Using such a distance threshold in common classification problems highly impacts the final prediction and it is delicate to choose (see Figure 4.1) ; (3) Noise has a large impact over the classification results ; and (4) The network can be easily overfitted given patches or clouds with ground truth labels.

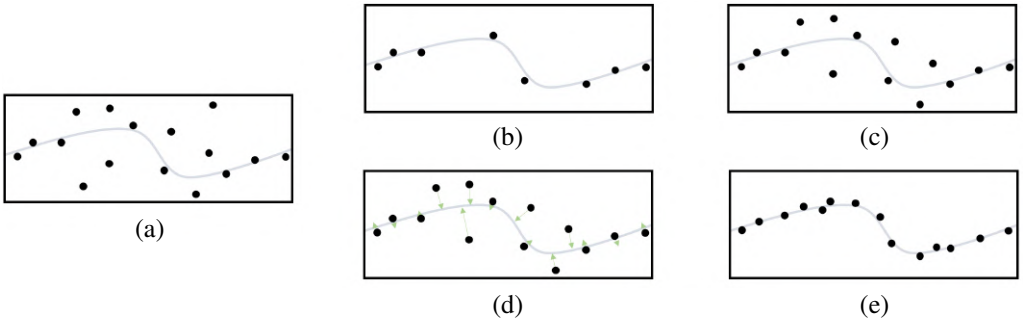


Figure 4.1 – *Impact of distance thresholds and displacement vectors on sharp feature detection. (a) Noisy point cloud around a sharp feature. (b) Given a small distance threshold, existing classification-based methods detect few accurate sharp feature points, which do not capture well the entire sharp feature. (c) Given a large distance threshold, the detected sharp feature points become vague. (d) Our method can detect the sharp feature points within a large distance threshold, and the learned displacement vectors relocate the points closer to the sharp feature. (e) Resulting accurate detection.*

Inspired by displacement-based point cloud denoising methods, we propose a deep learning framework for detecting and consolidating sharp features from a raw 3D point cloud, coined SFC-Net. Our framework contains two main components : (1) A point-to-feature oracle that largely improves the data augmentation quality so as to prevent models from overfitting ; (2) A multi-task

neural network architecture that predicts altogether a binary label (sharp/smooth) for each input point and a displacement vector for each sharp feature point. Our approach outperforms popular unsupervised methods and supervised methods on the ABC dataset [KMJ+19]. We also evaluate our method on noisy point clouds and real scanned models. Quantitative and qualitative results are provided in Section 4.3.

In summary, our insights and technical contributions are :

— We proposed a novel end-to-end multi-task neural network architecture that performs accurate sharp feature detection and consolidation ;
— Our neural network has a flexible structure that can be combined with many popular backbones and adapted to many datasets ;
— We proposed a point-to-feature oracle that largely improves the data augmentation quality during training to prevent models from overfitting ;
— Our approach has a good resilience to noise.

4.1.2 Related work

In this section, we first review the popular neural network structures for extracting geometric features from raw point clouds. We next review related work on sharp feature detection and consolidation from raw point clouds, including unsupervised methods and supervised methods. We then review the pioneering work on feature-preserving mesh denoising, which can also be utilized for sharp feature detection and consolidation from point clouds, when combined with an extra surface reconstruction step. We also briefly review point cloud denoising methods, focusing on displacement-based methods that motivate our displacement learning approach.

4.1.2.1 Learning-based geometric feature extraction

While convolutional neural networks yield impressive results for many image-related tasks, point cloud processing remains a challenge due to the unstructured and irregular nature of point clouds. The PointNet approach proposed in 2017 by Qi et al. [QSMG17] quickly became a popular learning approach for point cloud processing, and was applied to many tasks. It extracts point features from point coordinates by applying input and feature transformations and then aggregates global point features by max pooling. Qi et al. then proposed PointNet++ [QYSG17], which improves over previous work by considering local geometries with sampling and grouping layers. Guerrero et al. proposed another variant of PointNet, referred to as PCPNet [GKOM18]. Instead of taking the whole point cloud as input, PCPNet deals with patches to learn specific local point features. More recently, Qian et al. proposed ASSANet [QHL+21], a novel separable set abstraction module that further improves PointNet++.

Another family of methods is spatial-based. Wang et al. introduced a graph neural network, referred to as DGCNN, which learns features on KNN neighbors using successive EdgeConv

layers [WSL+19]. The So-Net [LCL18] is devised to model the spatial distribution of the input points. The SPLATNet [SJS+18] computes hierarchical and spatially-aware features of input points with sparse and efficient lattice filters. All the above networks can be utilized as a backbone of our framework, as long as they produce pointwise descriptors.

4.1.2.2 Sharp feature detection and consolidation

Unsupervised methods. Several unsupervised methods rely upon Moving Least Squares (MLS) surface reconstruction [Lev04]. Fleishman et al. [FCOS05] proposed a Moving Least Squares method for reconstructing a piecewise smooth surface from a point cloud. Sharp features are identified as points at the intersection of multiple iteratively fitted surfaces. Daniels et al. [DHOS07] improved over this method by using an adaptive threshold. They define a projection operator which takes into consideration edge points, followed by a smoothing filter. However, the performance of these methods highly depends on the quality of surface reconstruction results, which may fail in noisy cases. Moreover, most of these methods are compute-intensive and usually inefficient.

Huang et al. [HWG+13] contributed an edge-aware point cloud resampling technique built upon a normal estimation step followed by a robust projection operator. The final outcome highly depends on the quality of the normal estimation step.

Another line of methods detects sharp features via local geometric descriptors. Weber et al. [WHH10] rely upon point-sampled geometries. They first eliminate planar points by performing a flatness test. Clustering over the Gauss map is then performed to find sharp feature points. Mérigot et al. [MOG10] utilize convolved covariance matrices of Voronoi cells to characterize point properties and then filter out smooth points by a fixed threshold. Bazazian et al. [BCRH15] accelerated this approach by computing tensors directly from the nearest neighbors of points. The above methods are computationally efficient but defining relevant thresholds is a trial-and-error process.

Supervised methods. With the development of neural networks, more and more supervised methods are proposed for point cloud processing. Yu et el. [YLF+18a] proposed EC-net in 2018, in which they extend PU-Net [YLF+18b], a network designed to upsample and perform an edge-aware point cloud consolidation. Despite its success, its results are impacted by the way the patches are created. Wang et al. [WXX+20] contributed an end-to-end neural network, called PIE-Net, that is trained for parameter inference of feature edges over a 3D point cloud, where the output consists of one or more parametric curves. However, it is limited in terms of scalability. Loizou et al. [LAK20] devised a convolutional neural network based on DGCNN [WSL+19], to detect the boundaries of parts in 3D point clouds. Recently, Himeur et al. [HLP+21] introduced PCED-Net, a lightweight neural network that takes as input a series of multi-scale differential geometric descriptors of points and predicts three-class labels for a sharp feature and its surroundings. In

contrast, our framework learns displacement while detecting sharp features, which offers higher genericity and robustness to noise. Matveev et al. [MRA+22] introduced recently DEF, a learning-based framework which estimates a distance-to-feature field for sampled 3D shapes. A set of depth images are generated on local patches and passed to an image-based neural network. The output distance prediction of an individual point is blended from the overlapping patches. Compared to point cloud-based networks, DEF needs a more sophisticated data annotation procedure and the predictions are influenced by feature visibility.

4.1.2.3 Feature-preserving mesh denoising

Many filtering-based approaches have been proposed in the last two decades since its successful applications in image-related tasks. Fleishman et al. [FDC03] first proposed a bilateral mesh denoising approach by filtering mesh vertices in the normal direction, iteratively. Jones et al. [JDD03] introduced another single-pass bilateral filtering approach that can deal with arbitrary triangle soups. Wang [Wan06a] further improved the previous approach by integrating the filter with remeshing operators to iteratively recover sharp features. Zheng et al. [ZFAT10] introduced a novel bilateral filter on the facet normal field, taking into consideration both spatial difference and signal difference into bilateral weighting . Zhang et al. [ZDZ+15] devised a two-stage approach in which they apply a joint bilateral filter on face normals guided by an estimated normal field before updating the vertex position. Instead of filtering vertices, Attene et al. [AFRS03] proposed the Edge sharpener that first filters chamfer triangles and then subdivides them by inserting new vertices located on sharp features. Wang [Wan06b] further improved this incremental approach by introducing a new sharp-fold detector and a skeleton-guided vertex relocation step.

Another family of approaches are optimization-based. He et al. [HS13] introduced a mesh denoising approach via L_0 minimization. The vertex locations are optimized by minimizing the L_0 norm of the signal gradients. Wang et al. [WYL+14] devised a two-stage approach in which they compute at first a smoothly regularized mesh and then recover sharp features from the residual between the regularized mesh and the original mesh via l_1 analysis. With recent trend of data-driven techniques, Zhao et al. [ZLZ+19] proposed NormalNet, a learning-based normal filtering approach for mesh denoising. They followed the same iterative scheme of Zhang et al. [ZDZ+15] and replaced the estimated guidance field by a deep neural network.

While sharp feature detection and consolidation can be performed by combining surface reconstruction approaches and feature-preserving mesh denoising approaches, the former step may introduce extra errors into the pipeline and the latter step may depend on the quality, the density, or the manifoldness of the reconstructed mesh.

4.1.2.4 Displacement-based point cloud denoising

As the displacement property indicates the mismatches between the true model and the sampling, it can be utilized to reduce or even eliminate the intrinsic noise in point clouds. Rakotosaona et al. proposed PointcleanNet [RLBG+20], which pioneered the idea of displacement learning for point cloud denoising. They designed a two-step neural network based on PCPNet [GKOM18] that first eliminates outliers before estimating displacement vectors for all inliers. Instead, our method uses a shared backbone for both classification and regression, which significantly reduces the number of parameters. Pistilli et al. [PFVM20] introduced a fully convolutional network predicting displacement vectors. The success of the above methods shows the potential of displacement learning for sharp feature detection.

4.2 Approach

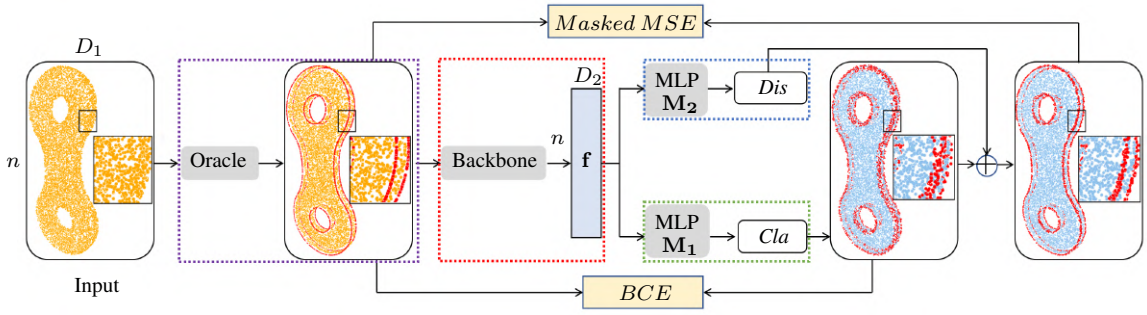


Figure 4.2 – *Overview of the proposed learning architecture. It comprises four modules : point-to-feature oracle (purple dash), point descriptor extractor (red dash), sharp feature detector (green dash) and displacement predictor (blue dash). Point-to-feature oracle module is only used for training. It introduces noise to the input, computes the ground truth classification labels and the ground truth displacement vectors. A backbone is used to extract high dimensional representations* $\mathbf{f}$ *(size :* $n \times D_2$*) in the point descriptor extractor module.* $\mathbf{MLP}$ *stands for multi-layer perceptron. Cla and Dis represent binary sharp feature labels and displacement vectors, respectively.* $\bigoplus$ *adds the corresponding predicted displacement vectors to detected sharp feature points to make them lie closer to the ground truth sharp features.* BCE *and* $Masked\ MSE$ *are two losses calculated base on Cla, Dis and the ground truth generated by the oracle.*

During training, our method takes as input a ground truth mesh $\mathcal{M}$, a set of ground truth sharp features $\mathcal{F}$ (with parametric representations) and an initial point cloud $\mathbf{P} \in \mathbb{R}^{c \times 3}$ sampled near $\mathcal{M}$, where $\mathbf{n}$ is the number of points and 3 represents the $\{x, y, z\}$ coordinates of the points. During inference, our method only needs a 3D point cloud $\mathbf{P}$. It outputs a set of binary classification labels $Cla \in \{0, 1\}^{\mathbf{n}}$ and a set of displacement vectors $Dis \in \mathbb{R}^{\mathbf{n} \times 3}$, simultaneously.

Figure 4.2 provides an overview of our architecture with four modules : *point-to-feature oracle*, *point descriptor extractor*, *sharp feature detector* and *displacement predictor*.

4.2.1 Point-to-feature oracle

Most data-driven methods use pre-defined labels for point clouds during the training session. This operation would probably lead to overfitting, or designing complicated loss functions to compute gradients, which makes the network hard to adapt to other datasets or architectures. Instead, we propose a point-to-feature oracle. It updates dynamically the ground truth classification labels and the ground truth displacement vectors according to the current noisy point cloud and the ground truth mesh before forwarding inputs to the network. Our network can deal with, but is not limited to, four types of sharp creases : segments, circles, ellipsoids and B-splines.

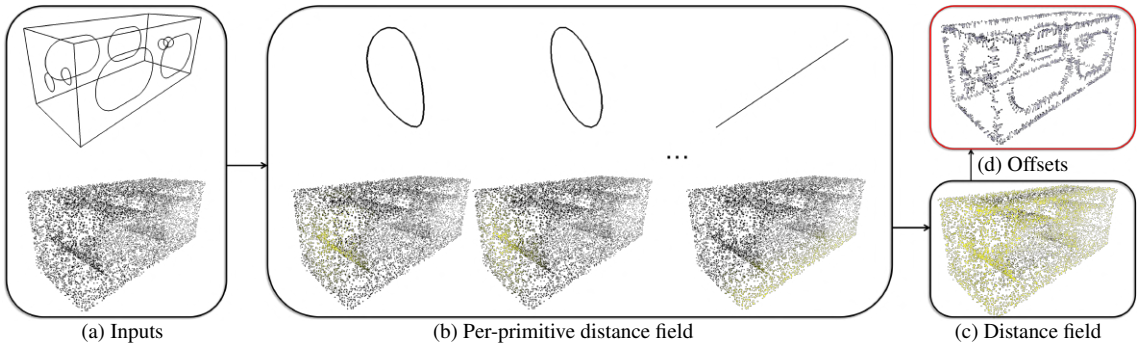


Figure 4.3 – *Principle of the point-to-feature oracle. (a) Our oracle takes a point cloud and a set of ground truth sharp features as input. (b) We compute the point-to-feature distances and the nearest projections between the point cloud and each primitive, respectively. Yellow denotes* $d = 0$ *and black denotes* $d \geq d_{\max}$ *in the color maps. (c) Each point is then mapped to the closest sharp feature. The ones with a point-to-feature distance smaller than* $d_{\max}$ *are labelled as sharp feature points. (d) The offset vectors are computed for all sharp feature points.*

Figure 4.3 shows the principle of our oracle. It takes two user-define parameters, a noise level n and a distance threshold $d_{\max}$, whose influences are well studied in Section 4.3. Given as input a sampled point cloud $\mathbf{P}$ from a CAD model and its associated ground truth sharp features in analytical expression, we first add a uniform random noise with the user-defined noise level (1% length of the bounding box diagonal by default) on $\mathbf{P}$. We then use the oracle to find, for each point p, its nearest ground truth sharp feature, and project p onto this feature. p is labeled as a sharp feature point if the distance between p and its projected point is smaller than the user-defined distance threshold, which is set to 0.03 times the length of the bounding box diagonal in our experiments.

For segments and (open or closed) circles, we compute the point's projection onto sharp features by algebra computation. To simplify the computation for (open or closed) ellipsoids and

B-splines, we generate a dense point sample on the sharp features and find the nearest one to the query point p as its projection.

4.2.2 Point descriptor extractor

The so-called backbone is used to extract pointwise descriptions $\mathbf{f}$ for point clouds. We experimented with DGCNN [WSL+19] and PCPNet [GKOM18] backbones, but users can plug their own backbone as long as it produces pointwise descriptors.

DGCNN is a graph-based convolutional neural network that takes as input a point cloud of fixed cardinality $\mathbf{N}$. It yields a global descriptor of size 1,024 for the whole point cloud and a multi-scale local descriptor of size 448 for each point. By concatenating global and local descriptors, each point has a descriptor of size 1,472. Since $\mathbf{N}$ is fixed for DGCNN, the point cloud must have the same cardinality during inference, which is its main limitation.

PCPNet is a PointNet-based neural network that takes as input a center (query) point along with its k-nearest neighbors and produces a local descriptor of size 1,024. While the cardinality of point clouds is not restricted, the inference is substantially slower than that of the DGCNN backbone.

4.2.3 Network architecture

After generating the features, two multi-layer perceptrons take $\mathbf{f}$ as input and output *Dis* and *Cla*, respectively. Both $\mathbf{M_1}$ and $\mathbf{M_2}$ comprise three layers. *Cla* is a set of binary classification labels, indicating whether a point is near a sharp feature. *Dis* is a set of pointwise vectors that represent the relocation from sharp feature points to their closest sharp features. The displacement vectors for smooth points are ignored by applying masks to the loss function.

4.2.4 Loss function and training

Our loss function is composed of two parts, a Binary Cross Entropy (BCE) with logits loss for sharp feature classification and a masked Mean Squared Error (MSE) loss for displacement prediction. The two losses cooperate in order to yield both a more accurate classification and a more precise displacement regression.

BCE with logits loss. Given a point cloud with $\mathbf{n}$ points, the ground truth labels $y = \{y_1, \ldots, y_\mathbf{n}\}$, the predictions $x = \{x_1, \ldots, x_\mathbf{n}\}$ from $\mathbf{M_1}$ where $x_i \in [-\infty, +\infty]$, and the sigmoid function σ, the loss is computed as :

$$l_c(x, y) = -\frac{1}{\mathbf{n}} \sum_{i=1}^{\mathbf{n}} [y_i \cdot \log \sigma(x_i) + (1 - y_i) \cdot \log(1 - \sigma(x_i))] \tag{4.1}$$

By combining the Sigmoid layer with BCE with logits loss, the back-propagation is numerically more stable by taking advantage of the log-sum-exp formulation, since the computation converts to :

$$l_c(x_i, y_i) = \begin{cases} (1 - y_i) \cdot x_i + \log(1 + e^{-x_i}) & \text{if } x_i > 0 \\ -x_i \cdot y_i + \log(e^{x_i} + 1) & \text{otherwise} \end{cases} \tag{4.2}$$

Masked MSE loss. Given ground truth displacement vectors $u = \{u_1, \ldots, u_{\mathbf{n}}\}$ and the predicted vectors $v = \{v_1, \ldots, v_{\mathbf{n}}\}$ from $\mathbf{M_2}$, the masked MSE is :

$$l_d(y, u, v) = -\frac{1}{\mathbf{n}} \sum_{i=1}^{\mathbf{n}} \mathbb{1}_1(y_i) \cdot \|v_i - u_i\|^2 \tag{4.3}$$

The mask makes the network concentrate on regressing displacement vectors for points around sharp features.

Training. The total loss energy is defined as :

$$l(x, y, u, v) = w_c \cdot l_c(x, y) + w_d \cdot l_d(y, u, v) \tag{4.4}$$

where w_c and w_d denote user-defined coefficients to balance the two tasks. An appropriate balance between these two losses is recommended to obtain a good cooperation of two tasks. During training, we first set a large w_c and a small w_d ($w_c = 1$ and $w_d = 0.01$ for both backbones) during the first 10 epochs, because a good classification result is required for learning displacement vectors. We then set a smaller w_c and a larger w_d ($w_c = 0.01$ and $w_d = 100$ for DGCNN-backbone; $w_c = 1$ and $w_d = 20$ for PCPNet-backbone) until the network converges to ensure accurate displacement prediction.

4.2.5 Cascaded model

In most cases, the proportion of sharp feature points in a point cloud is small, especially when the sharp threshold is low. Consolidating sharp feature points by adding displacement vectors thus becomes inadequate. To solve this issue, we propose a cascaded model based on the one presented in Figure 4.2.

The architecture is depicted by Figure 4.4. Given a point cloud $\mathbf{P}_i \in \mathbb{R}^{N \times 3}$, a pre-trained single model *SFCNet* computes the binary labels and displacement vectors, followed by a *Point Enricher*. The enriched point cloud $\mathbf{P}_{i+1} \in \mathbb{R}^{N \times 3}$ is composed of three sets from $\mathbf{P}_i$: the detected sharp feature points displaced by their corresponding displacement vectors, the detected sharp feature points and a random selection of smooth points. Suppose that n_s points are detected as sharp feature points in $\mathbf{P}_i$, $\mathbf{P}_{i+1}$ contains $2n_s$ sharp feature points and $N - 2n_s$ smooth points in the ideal case. This step is repeated until n_s satisfies a user-defined maximum number of sharp

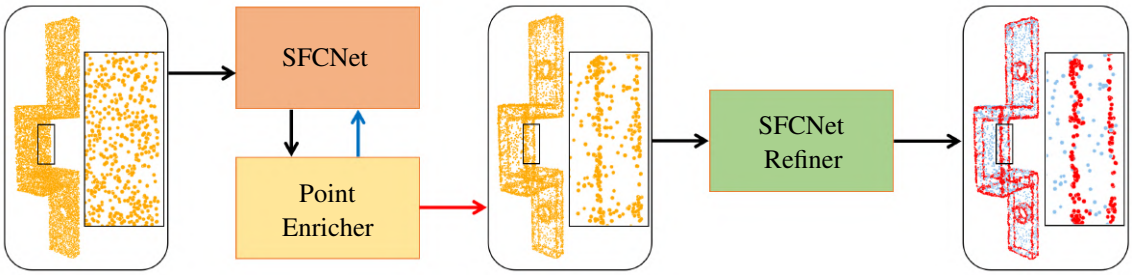


Figure 4.4 – *Cascaded learning architecture. SFCNet is a model trained with the uniformly sampled point clouds. Point Enricher inserts displaced detected sharp feature points to the input point cloud of SFCNet and erases the same amount of detected smooth points. The enriched point cloud is either passed to SFCNet as the blue arrows, or passed to SFCNet-Refiner as the red arrows, decided by the user-defined maximum number of sharp feature points and the maximum number of iterations. SFCNet-Refiner is another model trained with the enriched point clouds shown in the middle box.*

feature points or the iterated time t reaches the maximum number of iterations, set by default to $8,000$ and 3 in our experiments. The point cloud $\mathbf{P}_t$ is then sent to *SFCNet-Refiner* model, with a similar structure than *SFCNet*, but trained with the enriched point clouds.

The cascaded model can be viewed as an end-to-end network with a frozen pre-trained *SFCNet*. *SFCNet-Refiner* is indispensable since the distributions of sharp feature points and smooth points before and after iterative enrichment may be very different.

4.3 Experiments

We implemented our approach using the *Geomdl* library [BK19] for B-spline distance computation and the PyTorch deep learning framework [PGM+19]. We first introduce the setup of our experiments. We then study the robustness of our approach to the hyper-parameters, and carry on an ablation study for each component of our approach to demonstrate their individual relevance. We show both the quantitative and the qualitative results, as well as comparisons with unsupervised and supervised methods.

Dataset. The ABC dataset [KMJ+19] is a collection of CAD models with parameterized curves and annotated surfaces. We selected 5,093 models which contain at least one sharp crease. We split them into three non-overlapping sets used for training (3,623 models), evaluating (471 models) and testing (999 models).

Metrics. We selected two lines of quantitative metrics to evaluate our approach : (1) Common metrics used for binary classification and (2) Distance metrics used for regression. Similarly to

previous approaches [HLP+21, LAK20, WXX+20], we adopt the following metrics for classification : Precision, Recall, F1-score, Accuracy and Intersection over Union score (IoU). Besides, we estimate two metrics based on the Euclidean distance. More specifically, we compute the distance between detected sharp feature points and continuous ground truth sharp features, by taking advantage of the annotation of the ABC dataset and our point-to-feature oracle, which evaluates the correctness of the sharp feature points. We refer to this distance with *PtF* distance. We then estimate the Chamfer distance between the detected sharp feature points and a set of uniformly-sampled points on the ground truth sharp features, which evaluates the completeness of the detected features.

Setups. We experimented with two backbones to validate our approach : PCPNet [GKOM18], an architecture based on PointNet ; and DGCNN [WSL+19], an architecture based on graph neural networks. PCPNet takes a point cloud with arbitrary cardinality, while DGCNN takes a point cloud with a fixed cardinality (set by default to 10k in our experiments). We refer to our model with PCPNet backbone as SFCNet-P and the one with DGCNN backbone as SFCNet-D. The point cloud is normalized inside the bounding box from $[-1,-1,-1]$ to $[1,1,1]$ and centered at the origin. By default, we add a random uniform noise with a noise level set to 0.01 during training. To compare different methods, we add a similar noise level in the test set. The default backbone is DGCNN and the default sharp feature threshold is 0.03. Figure 4.5 shows several results produced by our default model on the test set.

4.3.1 Robustness study

We evaluate the sensitivity of our approach to the sharp distance threshold, defined as the maximum point-to-feature distance for sharp feature points, and to the noise level of the point clouds.

Distance threshold $d_{\max}$. A large sharp distance threshold helps the framework detect more sharp feature points, but displacement learning becomes more challenging, as shown in Figure 4.6. We trained four SFCNet-D networks using the same parameters, except for the distance threshold. Table 4.1 records the measured metrics for $d_{\max} \in \{0.02, 0.03, 0.05, 0.1\}$. We observe that the model with $d_{\max} = 0.1$ achieves good scores for most of the classification criteria while the distance criteria are much worse than $d_{\max} = 0.03$. Figure 4.6 shows the visualization results on one model. The right one detects more than 5k sharp feature points while we can see from the figure that some of the detected sharp feature points can not be well displaced to the sharp features.

Noise level n. This parameter has a strong relationship with $d_{\max}$ thus we cannot fix $d_{\max}$ for all experiments. We choose to train a series of SFCNet-D networks with different pairs of $(\mathrm{n}, d_{\max})$ and then compare the distance metrics. Table 4.1 records the quantitative results, which indicates

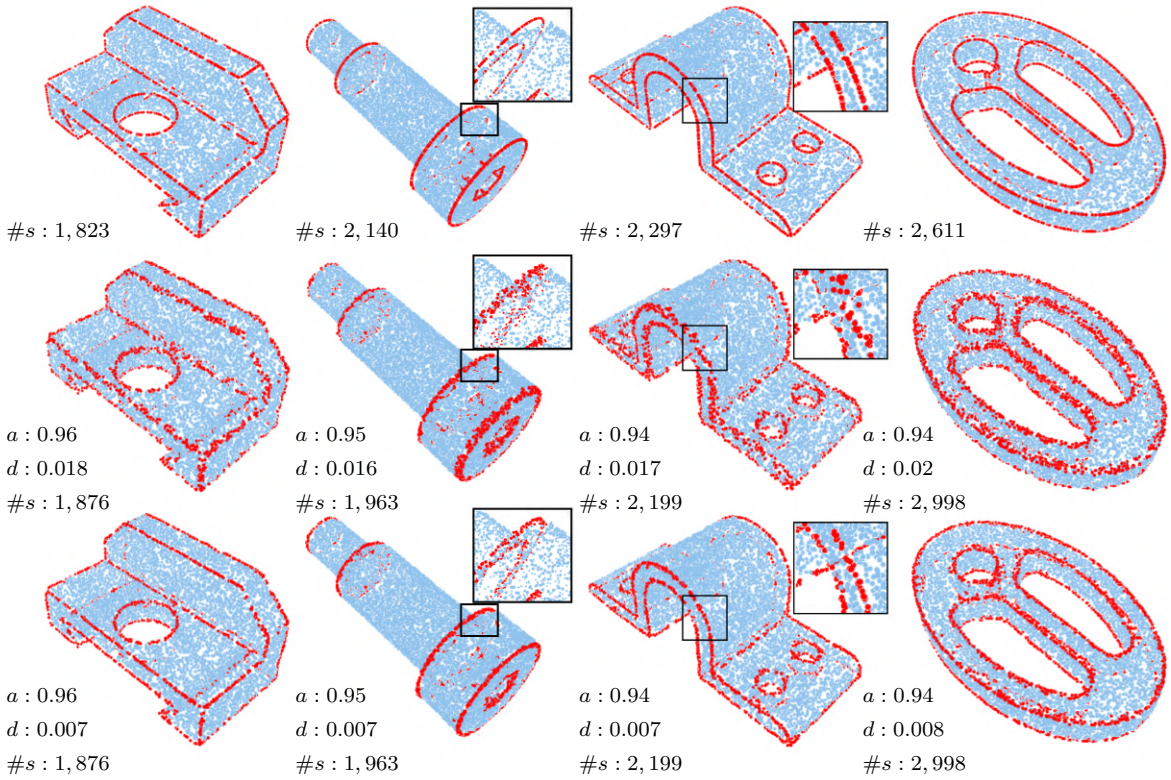


Figure 4.5 – *3D models from the ABC dataset. Our method displaces the detected sharp feature points closer to the ground truth. Top row : ground truth. Middle row : detected raw sharp feature points. Bottom row : displaced sharp feature points. Sharp feature points are depicted in red, and smooth points are depicted in blue. a, d and $\#s$ denote accuracy, point-to-feature distance and number of sharp feature points, respectively.*

that our method is robust to noise when the distance threshold is larger than the noise level. When the distance threshold is smaller than the noise level, the sharp feature points can not be accurately detected, as shown in the eighth row of Table 4.1 and in the fourth model of Figure 4.7 where few sharp feature points are detected.

4.3.2 Ablation study

We evaluate the role of the different components by removing one or several components from our approach and comparing the quantitative results on the test set.

Point-to-feature oracle. To show the importance of the dynamic labeling oracle, we consider four pairs of contrast experiments : D1 versus D2, D3 versus D4, P1 versus P2 and P3 versus P4 as shown in Table 4.1. The results show that the oracle can efficiently improve both the classification and the regression performance. In addition, the oracle helps the network converge faster and

		Classification(%)					Distance($\times 10^{-2}$)	
		Recall ↑	Precison ↑	IoU ↑	F1 ↑	Accuracy ↑	Chamfer ↓	*PtF* ↓
Threshold	$(0.01, 0.02)$	76.20	66.31	54.74	68.29	**93.05**	5.43	2.77
	$(0.01, 0.03)$	79.65	82.68	67.65	78.98	92.65	**4.55**	**2.65**
	$(0.01, 0.05)$	80.58	84.53	69.88	80.77	90.87	5.16	3.26
	$(0.01, 0.1)$	**86.15**	**89.51**	**78.14**	**86.69**	89.96	5.84	4.17
Noise	$(0, 0.03)$	**87.27**	70.91	63.96	76.37	91.83	4.81	3.02
	$(0.01, 0.03)$	79.65	82.68	67.65	78.98	92.65	**4.55**	**2.65**
	$(0.03, 0.03)$	71.34	65.50	51.26	65.63	90.48	6.87	3.73
	$(0.06, 0.03)$	45.32	67.22	36.52	50.71	**93.63**	11.42	3.81
	$(0.06, 0.1)$	84.3	**83.67**	**71.63**	**81.89**	85.88	7.82	5.31
Components	D1 : DGCNN+CLA	76.43	79.03	60.26	72.45	87.58	7.99	4.40
	D2 : DGCNN+ORACLE+CLA	80.62	80.54	67.40	78.38	92.34	6.41	3.61
	D3 : DGCNN+CLA+DIS	**87.32**	74.91	66.44	77.68	89.11	5.72	4.01
	D4 : SFCNet-D	79.65	**82.68**	**67.65**	**78.98**	**92.65**	**4.55**	**2.65**
	P1 : PCPNet+CLA	**96.01**	70.82	68.88	79.79	92.15	7.09	5.36
	P2 : PCPNet+ORACLE+CLA	88.96	87.20	79.14	86.92	95.54	5.97	3.69
	P3 : PCPNet+CLA+DIS	9.24	60.10	8.01	14.14	82.20	51.6	6.04
	P4 : SFCNet-P	88.47	**88.58**	**79.86**	**87.38**	**95.74**	**4.31**	**2.63**

TABLE 4.1 – *Quantitative results for robustness study and ablation study. Quantitative results with different distance thresholds, noise levels and components are recorded. Multiple pairs of* $(noise, threshold)$ *are used to study the user-defined parameters. We also compared our SFCNet-P model and SFCNet-D model with different combinations of the proposed modules. D and P are abbreviations for DGCNN and PCPNet. 1 to 4 indicate the experiment numbers.*

better. In our experiments, networks without oracle have much more chances to diverge (see P3) than the ones with oracle.

Displacement learning. The following four pairs of contrast experiments show the improvement by combining displacement learning with classical binary classification learning : D1 versus D3, D2 versus D4, P1 versus P3 and P2 versus P4 as shown in Table 4.1. Displacement learning boosts not only the distance metrics, but also the classification metrics in most cases. However, the comparison between P1 and P3 clearly shows that when the oracle is disabled during the training phase, the displacement learning introduces instability into the framework, since both MLPs share the same backbone. By combining our oracle with displacement learning, our network achieves the best scores in terms of most metrics.

Backbone. We conducted two groups of experiments : D1-D4 for DGCNN backbone and P1-P4 for PCPNet backbone. For both backbones, our proposed approach performs best within the

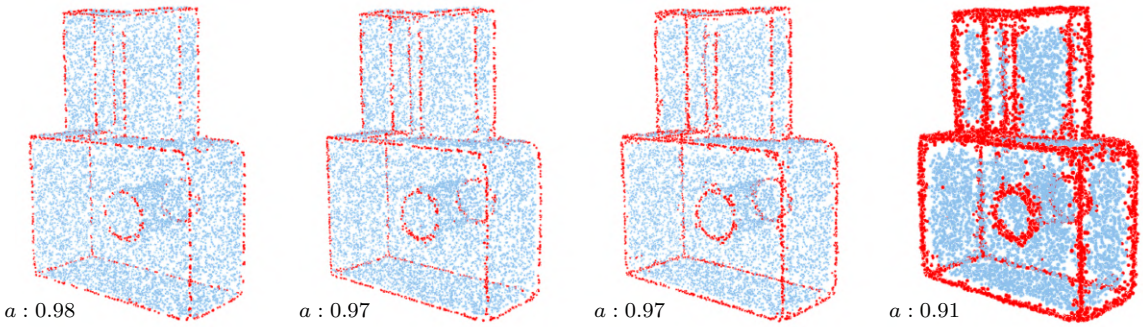


Figure 4.6 – *Visual comparison for different distance thresholds. From left to right : distance thresholds set to 0.02, 0.03, 0.05 and 0.1. The one with* $d_{\max} = 0.1$ *misclassified some smooth points as sharp feature points, while the ones with* $d_{\max} \in \{0.03, 0.05\}$ *keep a good trade-off between the classification accuracy and the distance criterion.*

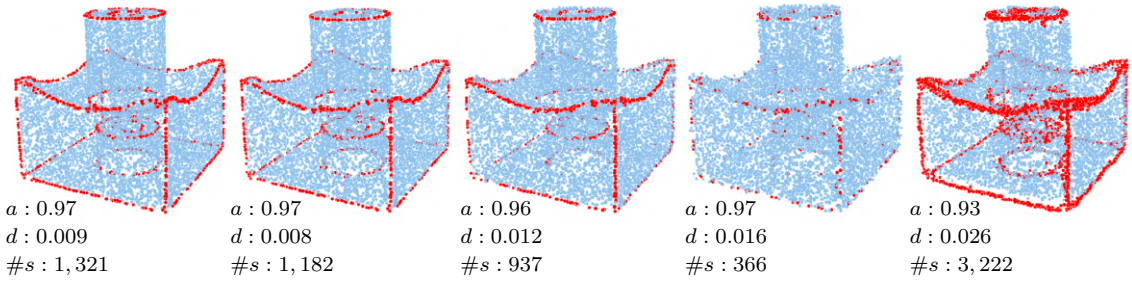
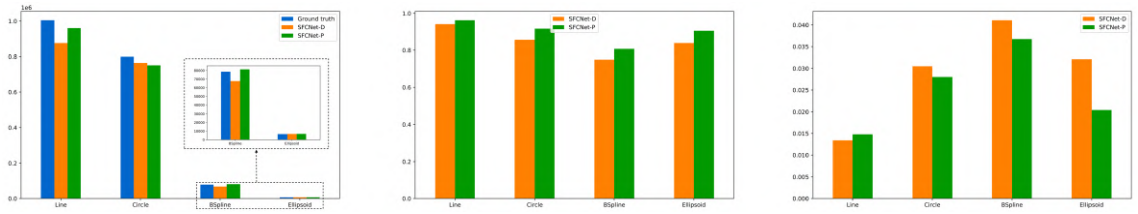


Figure 4.7 – *Visual comparison with different pairs of sharp distance thresholds and noise levels. From left to right, the pairs of sharp distance thresholds and noise levels have the same order as in Table 4.1. An incompatible choice of* $d_{\max}$ *and* n *(the 4th model) leads to a bad result.*

group. Generally speaking, PCPNet backbone has a better performance than DGCNN backbone, while the price is a much longer testing time for a given point cloud.

Sharp feature type We consider four types of sharp features in our experiments : segments, circles, ellipsoids and B-splines. In order to evaluate the capacity of handling all these types, we first computed the distribution of sharp feature types on the testing set and then compared it with the distributions of detected sharp feature points by both SFCNet-D and SFCNet-P. We observe a similarity among the distributions, while our models find slightly less sharp feature points than the ground truth. We then compute respectively the accuracy of detected sharp feature points and the average *PtF* distance of consolidation sharp feature points for each type. Our models achieve a good and balanced results on the four types, while the performance of B-splines is slightly worse than the other types due to its high complexity. Note that our models can not directly predict the

type of feature points, the type of each predicted sharp feature point is obtained by finding the closest ground truth feature.



(a) Distributions (b) Accuracy (c) PtF distance

Figure 4.8 – *The statistics of consolidated sharp feature points by our approach on the test set. Ground truth : computed by our oracle on noisy point clouds. SFCNet-D : D4. SFCNet-P : P4. (a) The distributions of sharp feature points belonging to each sharp feature type. (b) The accuracy of detected sharp feature points for each sharp feature type. (c) The PtF distance of consolidated sharp feature points for each sharp feature type.*

4.3.3 Comparisons

Point-based approaches We compare our approach with PB-DGCNN [LAK20] (re-trained from scratch on our dataset for 50 epochs, since there is no released pre-trained model), EC-Net [YLF+18a] (using the released pre-trained model), PIE-Net [WXX+20] (using the released pre-trained model), together with two unsupervised methods : Feature edge estimation via Voronoi Covariance Measure (VCM) provided by the CGAL Library [MOG10] and Covariance Analysis (CA) [BCRH15]. Table 4.2 records the error metrics on our test set. We observe that both of our networks have smaller Chamfer distances and smaller *PtF* distances compared to other methods. SFCNet-P performs best on the various metrics. Figure 4.9 and Figure 4.10 offer a visual comparison of all methods. VCM and CA are sensitive to user-defined parameters : we find it hard to define through trial-and-error parameters that fit all point clouds in our test set. Compared to PB-DGCNN, our SFCNet-D achieves similar scores for classification metrics while performing better in terms of distance, owing to the displacement learning. EC-Net seems to have more missing parts than our models and both distance criteria are worse than the ours. The results show that PIE-Net is not noise-resilient and the detected sharp feature points may distribute over the whole point cloud. While the paper mentions that it also predicts 3D offset vectors to relocate points onto edges, such a prediction step is commented out in the code and no pre-trained model contains such a relocation operation.

We evaluate the average inference time on our test set. VCM [MOG10], implemented in C++, takes around 1.877s for 1 model with 10k points. All the other methods are implemented in Python, and the approximate test times are : 0.134s for CA [BCRH15], 1.26s for PB-DGCNN [LAK20], 2.4s for EC-Net [YLF+18a], 0.177s for PIE-Net [WXX+20], 0.174s for SFCNet-D and 5s for SFCNet-P.

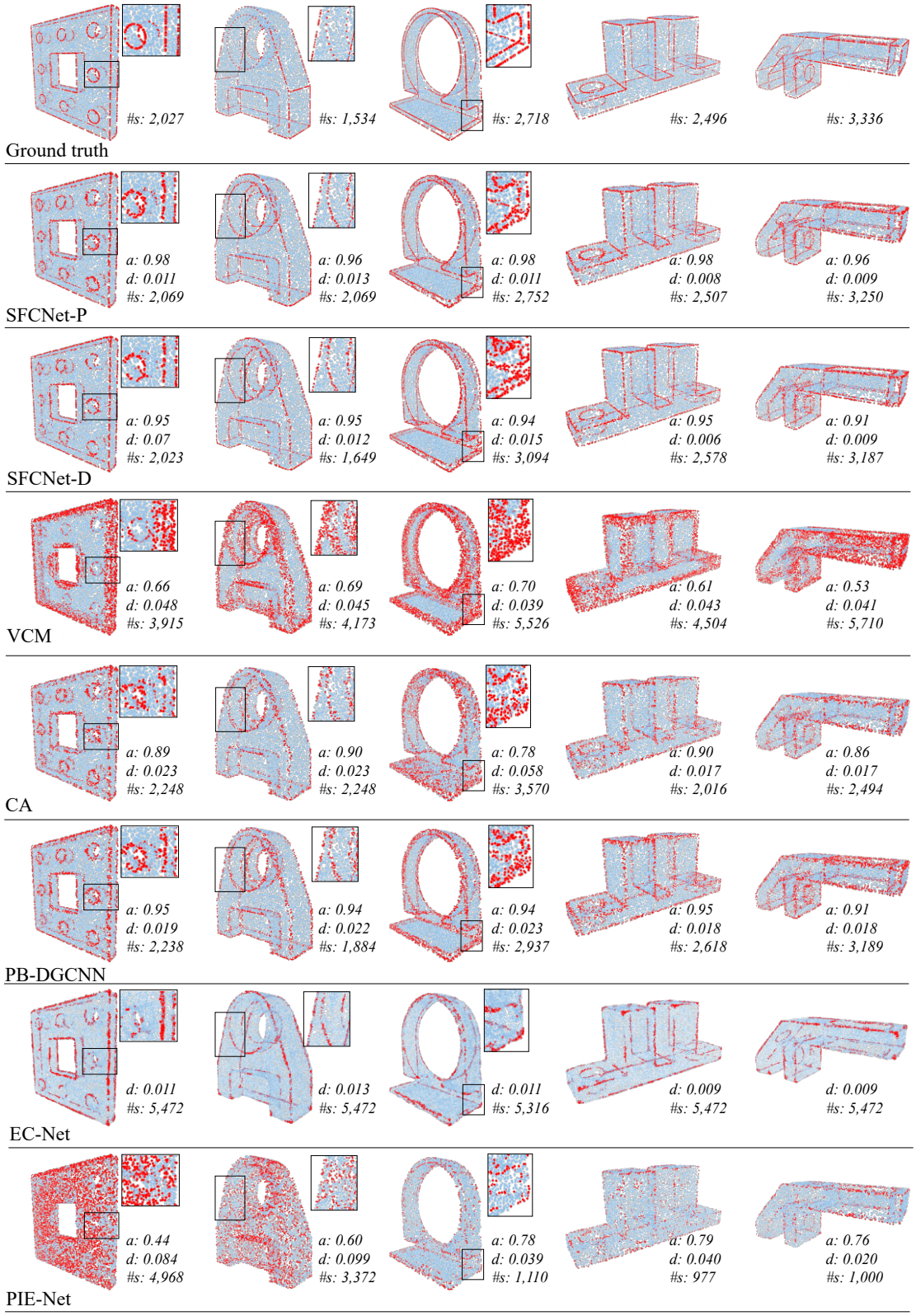


Figure 4.9 – *Visual comparisons with supervised and unsupervised methods on selected models from the test set. According to the closeups, our methods can detect more meaningful sharp feature points that are closed to the sharp features.*

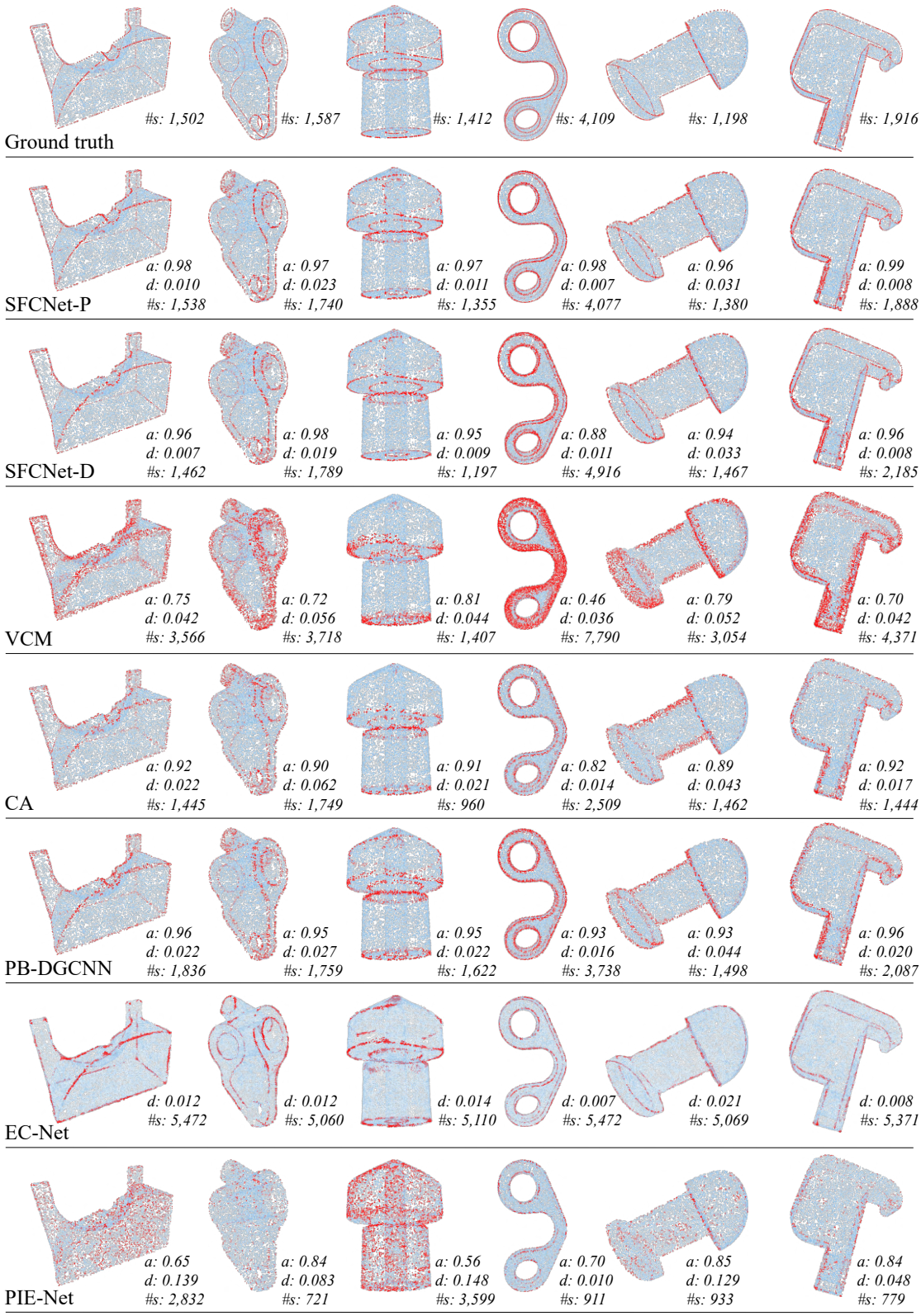


Figure 4.10 – *Visual comparisons with supervised and unsupervised methods on selected models from the test set. According to the closeups, our methods can detect more meaningful sharp feature points that are closed to the sharp features.*

	Cla(%)					Dis($\times 10^{-2}$)	
	Rec	Pre	IoU	F1	Acc	Cha	*PtF*
CA	67.18	63.06	43.50	57.58	78.98	11.80	8.23
VCM	81.29	38.10	34.09	48.99	72.29	12.01	7.91
PB-DGCNN	85.81	77.90	68.29	79.55	92.20	6.17	3.81
EC-Net	-	-	-	-	-	6.90	4.25
PIE-Net	48.87	45.11	20.32	32.47	62.86	16.65	14.07
SFCNet-D	79.65	82.68	67.65	78.98	92.65	4.55	2.65
SFCNet-P	**88.47**	**88.58**	**79.86**	**87.38**	**95.74**	**4.31**	**2.63**

TABLE 4.2 – *Quantitative comparison with unsupervised and deep learning sharp feature points detection methods on the ABC dataset. Note that we cannot compute classification metrics of EC-Net since the produced point clouds are upsampled. CA : [BCRH15], VCM : [MOG10], PB-DGCNN : [LAK20], EC-Net : [YLF+18a], PIE-Net : [WXX+20].*

Mesh-based approaches Feature-preserving mesh denoising approaches can be utilized for sharp feature detection and consolidation from point clouds, combining with an extra surface reconstruction step. We compared our approach with Bilateral Normal Filtering [ZFAT10], Mesh Denoising via L0 Optimization [HS13], Guided Mesh Normal Filtering [ZDZ+15], Edge Sharpener [AFRS03] and Non-iterative Feature-preserving Mesh Denoising [JDD03].

Typical mesh-based approaches work on orientable meshes with noisy vertex positions, as shown in the *Fandisk* model in Figure 4.11. We sampled the noisy mesh to obtain a noisy point cloud. As a consequence, the resulting point cloud contains structural piecewise linear noises which are rarely present in raw point clouds and our approach cannot achieve a good performance on such inputs.

Conversely, mesh-based approaches are not dealing well with noise in raw point clouds and errors originating from the reconstruction step. The *Mechanical part* model belongs to our test set. The raw point cloud contains random noise. To reconstruct an oriented mesh, we estimate and orient normals, then reconstruct the surface using the popular Screened Poisson Reconstruction [KH13]. Since smoothness priors are introduced during reconstruction, most mesh-based approaches fail to detect connected and clean sharp features, and outliers are inevitable. In addition, not all point clouds are orientable, as shown in the *Art piece* model. We utilize a recent learning-based surface reconstruction approach, referred to as DSE meshing [RGA+21]. DSE combines 3D Delaunay triangulations with learned local parametrizations to yield quality meshes, even if it may generate non-manifold edges. The advantage of this approach is that it does not require any normal information. Most mesh-based approaches fail on this model. The Edge Sharpener contributed by [AFRS03] successfully orients the noisy mesh while it introduces artifacts around the

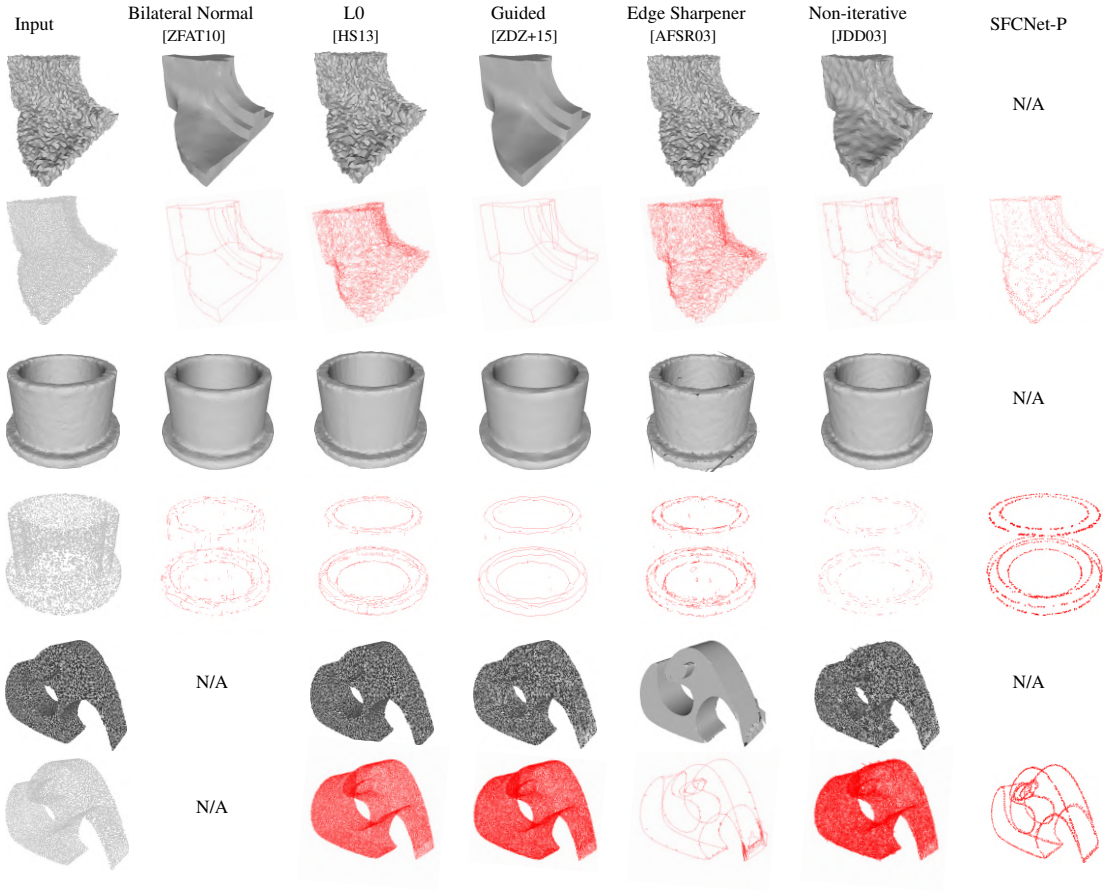


Figure 4.11 – *Qualitative comparison with mesh-based approaches. The first two rows : Fandisk model. The third and fourth rows : Mechanical part model from ABC Dataset. The last two rows : Art piece model from [XWD+22] (Bilateral Normal fails on this model). Input : point cloud for SFCNet-P (odd rows), mesh for the other approaches (even rows). The sharp features are extracted from mesh by comparing the maximum angle between the normal vectors of adjacent triangles using [The21].*

sharp feature in the intersection of two nearby surface patches with opposite orientation. Since our approach is normal-free, it yields satisfactory outputs on the whole model.

4.3.4 Cascaded models

We test our cascaded model on the ABC dataset. A pre-trained SFCNet-P network is used to iteratively detect and displace the sharp feature points. The enriched point clouds are used to train another SFCNet-P network, as SFCNet-Refiner in Figure 4.4. Figure 4.12 depicts two models together with the identified sharp features points by a single SFCNet model and our cascaded model. Compared to the single model, the cascaded model consolidates sharp features with more points without loss of accuracy.

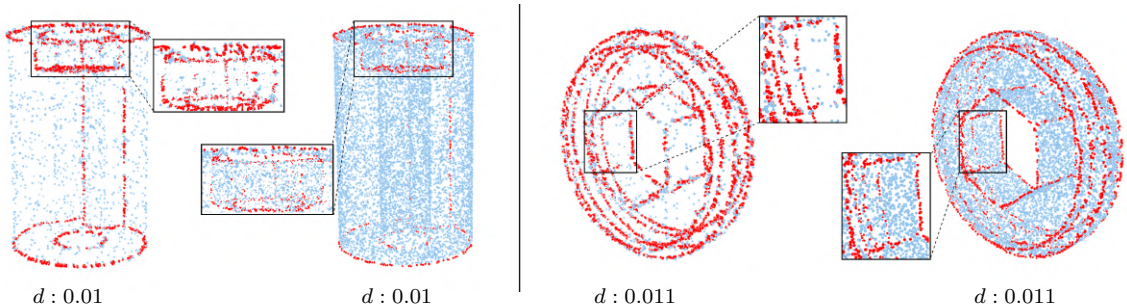


Figure 4.12 – *Visual comparison on two point clouds with cascaded model and single model. c denotes the Chamfer distance. The left is produced by the cascaded model and the right produced by the single model.*

4.3.5 Results on real scans

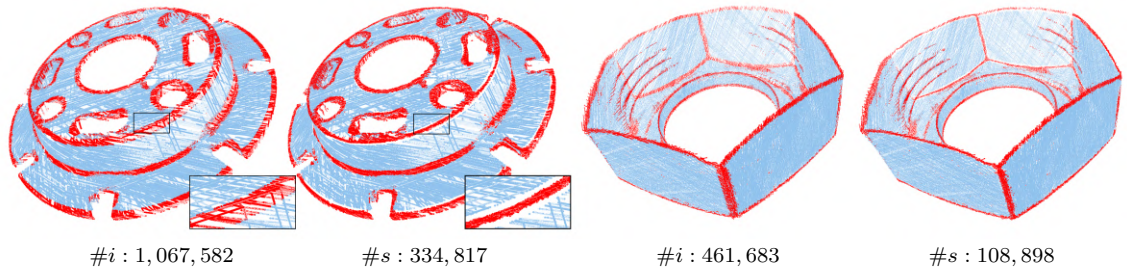


Figure 4.13 – *Scanned point clouds of Visionair repository. The first and the third image show the detected sharp feature points without displacement. The second and the forth ones show the predicted sharp feature points with displacement. $\#i$ and $\#s$ denote the number of input points and the number of detected sharp feature points, respectively.*

We evaluate our SFCNet-P model on real scanned 3D point clouds taken from the online Visionair repository [Vis]. These point clouds have never been seen during the training phase. Compared to the training dataset, we highlight three major differences in real scanned point clouds : (1) The cardinality of the point clouds (from 460k to 1 millon) is substantially larger than that of our training dataset (10k); (2) The intrinsic noise in scanned point clouds originates from the laser, instead of being simulated randomly; (3) The models scanned by real-world scanners are often open, while all models of our training dataset are closed. Despite the above differences, our model predicts accurate sharp feature points from the real scanned point clouds with displacement learning, see Figure 4.13.

Due to the lack of ground truth labels, it is hard to obtain quantitative results on real-world data. We compared our proposed approach with other approaches qualitatively on a scanned point cloud.

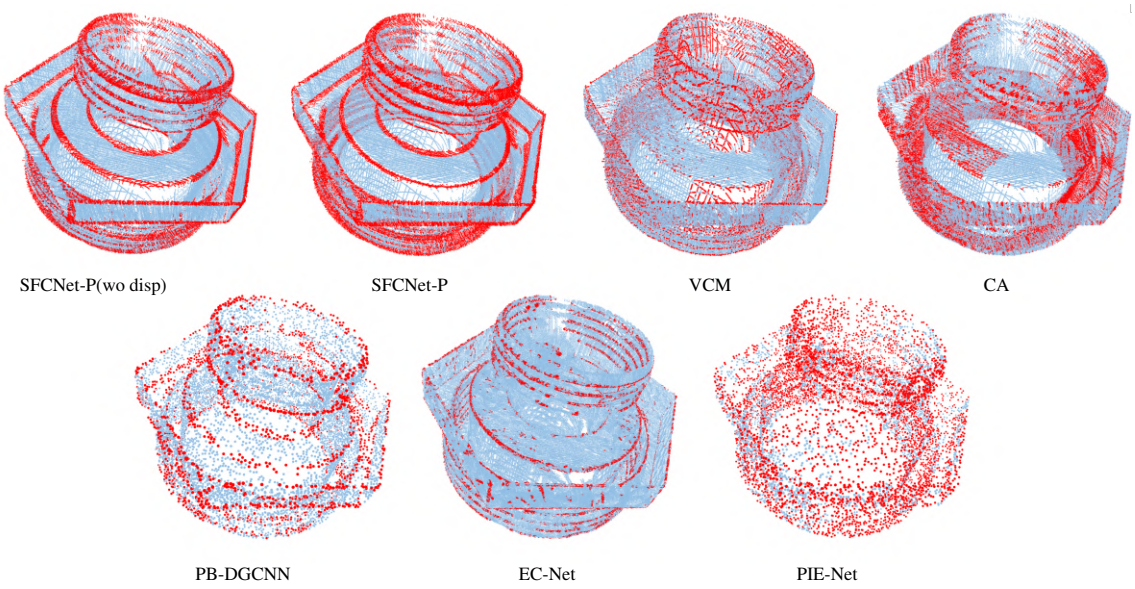


Figure 4.14 – *Comparisons on a scanned point cloud (718k points). SFCNet-P(wo disp) denotes our SFCNet-P model without adding displacement vectors. The output of VCM is sensitive to the quality of estimated normals, since the ground truth normals are not available for this point cloud. Inputs for PB-DGCNN and PIE-Net are subsampled due to the restriction for their input.*

4.3.6 Limitations

Our current approach presents several limitations. First, the detected sharp feature points may not distribute uniformly over the sharp features. We intend to devise a loss function that favors the uniform distribution of inliers along sharp creases. However, the difficulty lies in the fact that we have no knowledge of continuous sharp feature graphs. Second, we classify points into two classes : sharp or smooth, instead of classifying them into instances. In future work, we wish to extend our framework to perform instance segmentation of sharp features, in order to cluster sharp feature points into several creases, possibly meeting at corners, darts or cusps. To achieve this goal, the network needs to predict the number of sharp features, which is substantially more difficult than binary classification.

4.4 Applications

We show two potential applications of our approach : 3D feature line extraction and 3D surface reconstruction. Nevertheless, our approach is not limited to these two applications. It can be further adapted to other applications such as instance segmentation or feature graph extraction.

4.4.1 Feature line extraction

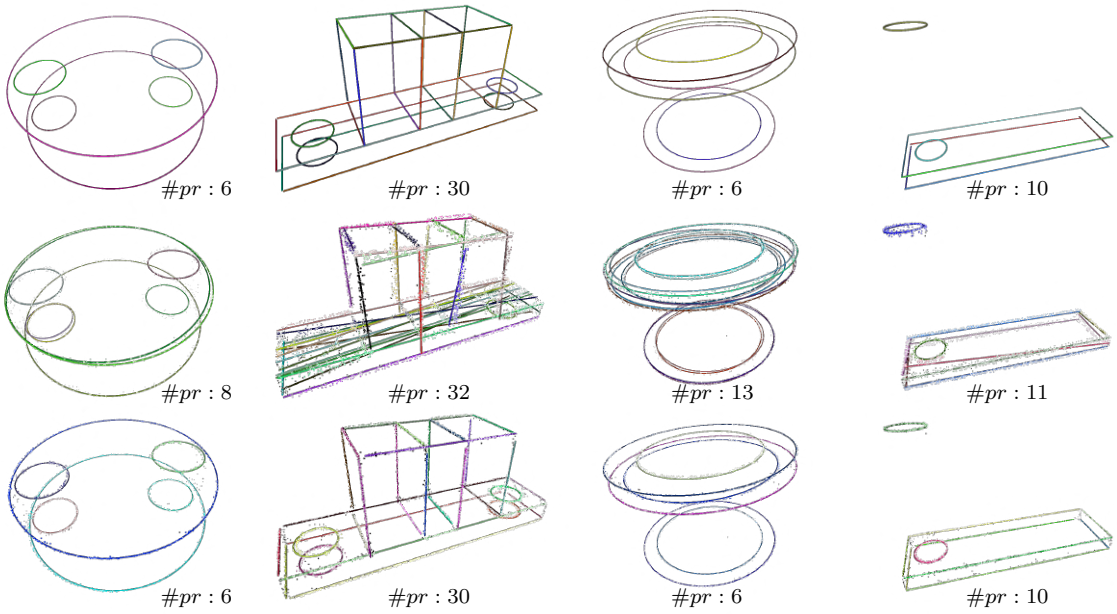


Figure 4.15 – *Feature line extraction results using RANSAC. From top to bottom : results of dense, uniformly sampled points on ground truth feature creases, results of the detected sharp feature points without displacement, results of the consolidated sharp feature points by our SFCNet-P model. #pr denotes the number of detected primitives. The ones without displacement detect many wrong primitives and there are more outliers (black points) left after the detection.*

After consolidating the sharp feature points, we can apply primitive detection methods to extract parametric representations of sharp feature creases. We adapt RANSAC [FB81] algorithm to extract feature lines and the results are shown in Figure 4.15. The parameters are carefully tuned for all experiments. We observe that the extractions of consolidated sharp feature points are more robust and have a better quality than the one of the detected sharp feature points without displacement.

4.4.2 Surface reconstruction

Some surface reconstruction methods can benefit from consolidated point clouds. We utilize a recent learning-based surface reconstruction approach, referred to as DSE meshing [RGA+21].

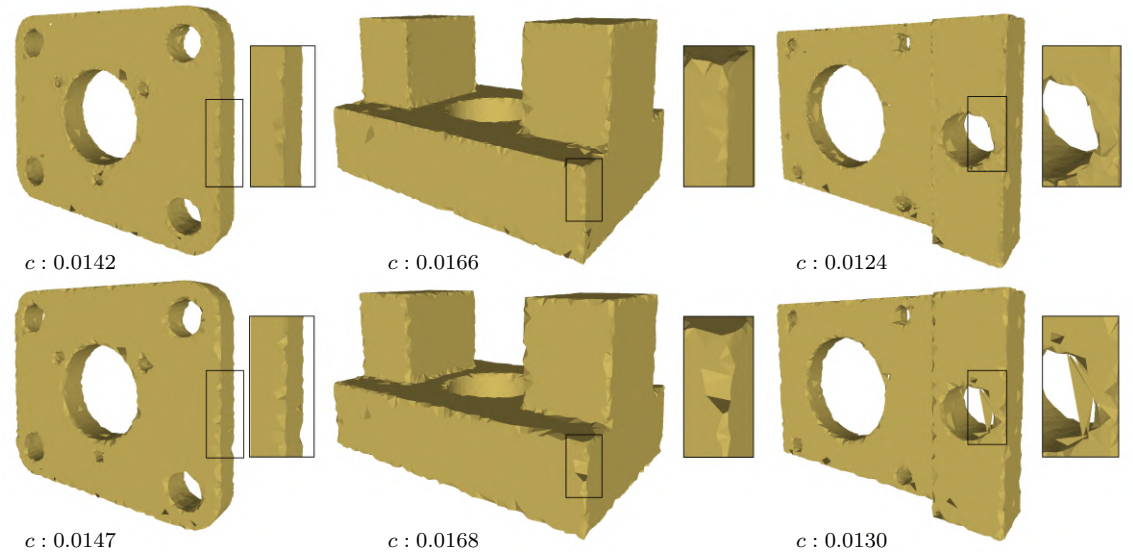


Figure 4.16 – *Reconstructed meshes using DSE meshing. First row : reconstruction from the consolidated 3D point clouds. Second row : reconstruction from the 3D input point clouds. c denotes the Chamfer distance between the reconstructed mesh and the ground truth mesh in the ABC dataset. DSE meshing is able to capture more details on sharp creases with consolidated point clouds and the Chamfer distance shows that the meshes are closer to ground truth compared to the original point clouds.*

DSE combines 3D Delaunay triangulations with learned local parametrizations to yield quality meshes, even if it may generate non-manifold edges. The advantage of this approach is that it does not require any normal information. We show the quality improvement of our approach by comparing the surface reconstruction results from the input 3D point clouds with the consolidated 3D point clouds. For generating a consolidated 3D point cloud, we first combine the input point cloud with the displaced sharp feature points, then 10k points are sampled using farthest point sampling to meet the input size requirement of the DSE.

Figure 4.16 depicts the reconstruction results before and after consolidation using our SFCNet-P model, in which we selected three 3D point clouds without noise from the test set in order to better compare the reconstruction quality around sharp creases.

4.5 Discussion

We contributed a novel deep learning based-framework devised to detect and consolidate sharp feature points from raw 3D point clouds. Compatible with existing backbones devised to extract features from 3D point clouds, our framework comprises two learnable modules : the first module learns to predict binary smooth/sharp labels for all points, and the second module learns to

regress the displacement vectors used for relocating sharp feature points onto sharp features. Our framework is capable of identifying and consolidating sharp features, altogether. Our approach is robust to intrinsic noise, thanks to the point-to-feature oracle, which performs data augmentation during the training phase, and displacement learning, which relocates sharp feature points onto their nearest sharp features. Our experiments demonstrate that our framework outperforms the state-of-the-art in terms of detection accuracy and distance criteria.

CHAPITRE 5

Variational Shape Reconstruction via Quadric Error Metrics

Inspired by the strengths of quadric error metrics initially designed for mesh decimation, we propose a concise mesh reconstruction approach for 3D point clouds. Our approach proceeds by clustering the input points enriched with quadric error metrics, where the generator of each cluster is the optimal 3D point for the sum of its quadric error metrics. This approach favors the placement of generators on sharp features, and tends to equidistribute the error among clusters. We reconstruct the output surface mesh from the adjacency between clusters and a constrained binary solver. We combine our clustering process with an adaptive refinement driven by the error. Compared to prior art, our method avoids dense reconstruction prior to simplification and produces immediately an optimized mesh.

5.1 Introduction

Mesh reconstruction consists in finding a mesh that piecewise approximates a point sampling of a 3D surface well. In addition to the inherent ill-posed nature of the reconstruction problem, several dilemmas witness the difficulty of this problem, such as interpolation vs. approximation, or greedy vs. variational approaches.

In this section, we focus on concise mesh reconstruction from 3D point clouds that are unstructured and come without oriented per-sample normal vector. In addition, we aim to produce concise surface triangle meshes through a low-memory footprint process, see Figure 5.1. One possible solution to the problem is to perform dense mesh reconstruction followed by mesh decimation but the transient memory consumption is large. If one seeks additional features such as noise resilience or hole filling, one may resort to implicit reconstruction followed by isosurfacing and mesh decimation. However, this sequence adds another inconsistency since the bias induced by implicit reconstruction is unknown by the decimation step, resulting in a suboptimal complexity-distortion tradeoff. Proceeding coarse-to-fine has in general a lower memory footprint, but approaches such as greedy Delaunay refinement result in suboptimal approximations. This calls for a coarse-to-fine variational approach.

Another challenge comes from sharp features. Nothing is really sharp in the physical world when looking at fine scales. Nevertheless, a physical fillet (rounded edge) becomes sharp when zooming out at coarse scales. In a surface triangle mesh, any edge or vertex that is not flat is a sharp feature, but edges and vertices are also used to approximate smooth parts. In other words, the sharpness of a feature depends on the chosen scale or approximation tolerance error. As we wish to proceed coarse-to-fine, this rules out approaches based on early sharp feature detection. Instead, we favor a variational approach where sharp features emerge from an optimization process.

5.2 Related Work

We first review the shape reconstruction problem, with a focus on generating concise meshes as output. We then review the pioneering work on quadric error metrics and its variants, and a few variational approaches.

5.2.1 Shape reconstruction

We focus next on key aspects sought after in our problem statement : conciseness and recovery of sharp features. We refer to books or surveys for a more complete review on shape reconstruction [Dey06, BTS+17].

Conciseness. Hoppe et al. [HDD+94] pioneered a piecewise smooth reconstruction approach that takes as input a dense reconstructed mesh, and interleaves decimation and optimization to

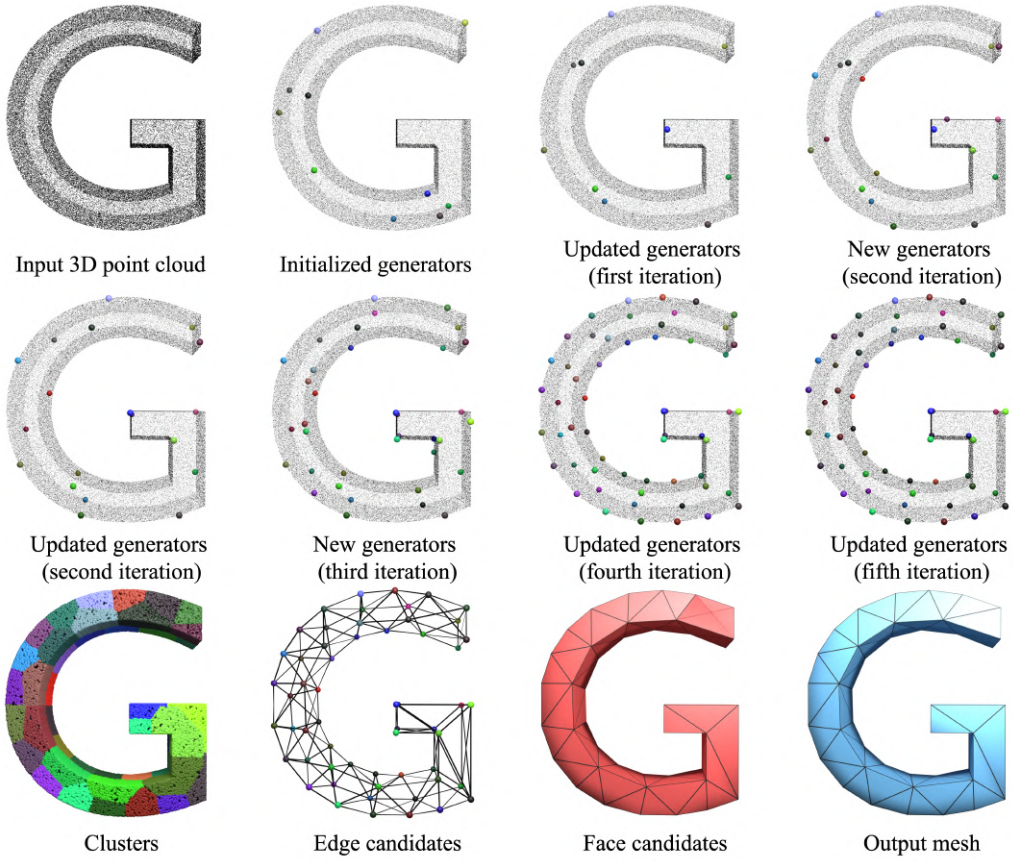


Figure 5.1 – *Variational shape reconstruction. Clustering of the points is randomly initialized, then alternates partitioning and generator updating. Some generators relocate to sharp features after one iteration. New generators are then added. Clustering converges after six iterations. A set of candidate edges is derived from the adjacency between clusters, and candidate facets (red) are generated. The output 2-manifold mesh is reconstructed via a constrained binary solver that selects a subset of the red facets.*

fit a subdivision surface. Digne et al. [DCSA+14] formulate shape reconstruction as an optimal transport problem between a discrete measure (points seen as Dirac masses) and a dense simplicial complex seen as the support of a piecewise-constant measure, which is then decimated. In addition to being compute-intensive, these fine-to-coarse approaches contradict our desire to proceed coarse-to-fine with low memory consumption. The literature is sparser on coarse-to-fine mesh-based approaches. Sahillioğlu and Yemez [SY10] start from the visual hull mesh of an object, and perform smoothing and restructuring operations. Delaunay refinement is another instance of the coarse-to-fine paradigm [CGAY13], but for the two above approaches the meshes are too isotropic to offer the level of conciseness sought after.

Sharp features. The literature is abounded by methods trying to detect sharp features, but we seek for an approach that *recovers* sharp features by optimizing vertex locations such that sharp features are formed when they yield the best approximation. Several non-linear regression approaches have been proposed to find such optimal locations : e.g., robust implicit moving least-squares surfaces (MLS) [ÖGG09], or robust MLS [FCOS05]. Another approach consists of consolidating the point clouds before reconstruction, by removing outliers, reducing noise and relocating some points onto sharp features. Avron et al. [ASGCO10] first solves a global sparse minimization problem to consolidate normal orientations, then uses them to consolidate point locations and reconstructs the mesh via ball pivoting [BMR+99]. Xu et al. [XWD+22] recently pioneered a multistage approach that removes noise, identifies feature lines via discrete optimal transport and interpolates the consolidated point cloud. Consolidating the mesh after reconstruction is also an option. Attene et al. [AFRS03] proposed a so-called edge sharpener. Wang et al. [WYZC13] first remove outliers and noise, before inputting the consolidated 3D point cloud to Poisson surface reconstruction [KBH06]. Sharp features are then recovered by applying bilateral filtering to the reconstructed surface mesh. Yadav et al. [YRP18] proposed another two-stage mesh denoising algorithm. All these methods are powerful but operate on dense meshes. Several learning-based methods have been recently proposed for extracting sharp feature graphs from raw point clouds. Liu et al. [LDSW21] proposed a deep network structure predicting first the sharp corners then selecting edges. Only segments can be dealt with in this method. Matveev et al. [MRA+22] learns a distance-to-feature field before corner prediction and spline fitting

5.2.2 Quadric Error Metrics

Garland and Heckbert [GH97] pioneered a mesh decimation approach that applies a sequence of edge collapse operators sorted by so-called quadric error metrics (QEM). QEM capture weighted sums of squared errors to the supporting planes of triangle facets (see Section 5.3). This powerful idea has become one of the most popular approaches for mesh decimation. Hoppe [Hop99] extended this approach to deal with appearance attributes such as colors or texture coordinates. Deng et al. [DLM11] utilized quadric error metrics to build cages for shape editing. Thiery et al. [TGB13] defined a spherical quadric error metric in 4D for extreme shape approximation. Salinas et al. [SLA15] contributed a structure-aware variant that adds to the quadrics a set of plane quadrics detected from the input mesh. Legrand et al. [LTB19] proposed a mesh smoothing and clustering approach via filtering quadrics : a field of filtered quadrics is defined on the input mesh by diffusing quadrics via a bilateral filter, a spatial kernel and a QEM-based range kernel. The mesh vertices are connected to form a tree structure according to a relevance score estimated from the filtered quadric field, and the final clusters are generated by pruning the branches. Recently, Trettner and Kobbelt [TK20] extended the QEM approach by defining probabilistic quadrics in

closed form, in order to deal with uncertainty in the input meshes. Such an approach improves the resilience to noise and quality of the mesh elements.

5.2.3 Variational approaches

Cohen-Steiner et al. [CSAD04] introduced the variational shape approximation (VSA) approach, which formulates the task as a discrete, variational partitioning problem with planar proxies. VSA optimizes clusters of connected mesh triangles by minimizing a one-sided error metric (L^2 or $L^{1,2}$) formulated between triangles and planar proxies. The number of clusters is adjusted via four operators (*merge*, *split*, *add*, *teleport*). The final output is an anisotropic polygonal mesh derived from the clustering partition. Wu and Kobbelt [WK05] extended VSA to deal with higher order proxies such as spheres and cylinders. Skrodzki et al. [SZP20] adapted VSA to 3D point sets, and added a *switch* operator to guarantee convergence. Yu and Lafarge [YL22] devised a pliant approach for partitioning a 3D point cloud into planar parts. From an initial partition, a multi-objective function is optimized through five types of operators (*insert*, *exclude*, *transfer*, *merge*, *split*) to obtain a balance between fidelity, complexity and coverage. The optimized partition is then inputted to a piecewise-planar reconstruction approach [BL20]. The above clustering approaches are powerful, but partitioning solves only half of the problem, as it remains to mesh the optimized partition which is in general composed of polygonal elements. Intersecting planes is an option, but it is numerically unstable, as already observed by Zimmer et al. [ZCHK12] for planar panelization of freeform surfaces. Meshing is also hampered by concave elements of the partition. Resorting to a volumetric partition through a kinetic-based approach is a reliable alternative, but is compute-intensive [BL20]. Departing from partitioning with planar elements, we adopt a dual approach that partitions with "conical" elements, i.e. vertices with their adjacent planar elements where the vertex locations are computed through integrating quadrics over clusters.

5.2.4 Positioning and Contributions

The original QEM approach, designed for greedy mesh decimation, is fast and very effective. We observe that it may be seen as a clustering algorithm over the input mesh vertices, where the objective is to minimize the maximum (over the clusters) of the sum of squared distances from the optimal vertices to the set of planes of the clusters. This maximum is minimized in a greedy hierarchical way using a priority queue for decimation operators, and the output mesh is the dual (i.e., nerve) of the clustering : one vertex per cluster, and one edge per pair of neighboring clusters.

We utilize the strengths of QEM to design a novel coarse-to-fine variational reconstruction algorithm. Our starting point is also QEM, but we replace the greedy fine-to-coarse clustering by a variational expectation-minimization (EM) approach. The objective function is no longer the maximum cluster error but rather the sum of errors over the clusters. Our method may also be seen as a form of dual to the VSA approach initially designed for mesh partitioning ; instead

of considering squared distances from data points to planar cluster proxies, we consider squared distances from tangent planes at data points to clusters estimated cone points. Our experiments show that, especially for extreme approximations, our approach outperforms the state-of-the-art.

In addition, our method takes point clouds as input, allowing to address concise mesh reconstruction. For this harder problem, taking the nerve of the clustering as output mesh may not suffice anymore, and we output instead a mesh computed by a specific binary solver.

Our main technical contributions are :

- A novel variational partitioning method designed to cluster the input points based on quadric error metrics. The generators are optimal points minimizing the sum of quadric error metrics for the clusters.
- The resulting partitioning tends to equidistribute the errors among clusters, yielding clusters with anisotropy in accordance to the local geometry.

5.3 Background

We briefly review quadric error metrics (QEM) pioneered by Garland and Heckbert [GH97] for mesh decimation, and some other variants such as the probabilistic QEM recently introduced by Trettner and Kobbelt [TK20].

Given a triangle t, we consider its supporting plane π and denote by n a unit normal vector of π and by p an arbitrary point on π. The matrix of the *plane quadric associated to* t is the 4×4 symmetric matrix

$$Q_t = \begin{bmatrix} nn^T & -p^T nn^T \\ -nn^T p & (p^T n)^2 \end{bmatrix} = \begin{bmatrix} A & -b \\ -b^T & c \end{bmatrix}. \tag{5.1}$$

Given a query point $q \in \mathbb{R}^3$, the corresponding error function is defined as $f(Q_t, q) = \tilde{q}^T Q_t \tilde{q}$, where $\tilde{q} = (q^T, 1)^T$ are the homogeneous coordinates of q. This function captures the squared distance from point q to plane π. The *probabilistic QEM* incorporates Gaussian noise for both the unit normal vector n and the position p on the plane π. In this case, A, b and c are replaced by their corresponding expectations $\mathbb{E}(A)$, $\mathbb{E}(b)$ and $\mathbb{E}(c)$.

A diffused quadric is assigned to each vertex of the input surface triangle mesh. Given a vertex v on a 2-manifold, its *diffused quadric* Q_v is defined as the sum of the weighted quadrics of its 1-ring neighboring faces, i.e.

$$Q_v = \sum_{t_i \in \text{ neighbors(v)}} \frac{1}{3} a_{t_i} Q_{t_i}, \tag{5.2}$$

where a_{t_i} denotes the area of triangle t_i. Each plane quadric is thus evenly distributed on the three vertices of its triangle. From a geometric viewpoint, summing quadrics is equivalent to taking the union of the planes. Such an initialization guarantees that the vertices are the optimal minimizers of their diffused quadrics, while it is in general not the case during decimation. Equation (5.2) is

extended to deal with boundaries by adding quadrics corresponding to planes that are orthogonal to boundary edges.

When an edge e connecting v_1 and v_2 is collapsed, the quadric of the new vertex v_e is computed as :

$$Q_{v_e} = Q_{v_1} + Q_{v_2}, \tag{5.3}$$

meaning that the new collapsed vertex receives the contribution from all weighted planes of the input surface mesh assigned to v_1 and v_2. Finding the optimal location of v_e is achieved by solving for point p_e^* that minimizes QEM $f(Q_{v_e}, p_e)$, i.e. the point which realizes the smallest sum of squared distances to all assigned planes. If all planes are similar then all points on the plane minimize $f(Q_{v_e}, p_e)$; if all planes intersect in a line, all points on this line minimize $f(Q_{v_e}, p_e)$; if all planes intersect at one point, the intersection point is the only minimizer of $f(Q_{v_e}, p_e)$. In other words, QEM has the virtue of placing points onto sharp creases or corners, while an infinite number of optimal point locations (lines or planes) exist for linear sharp creases or corners. Such an ill-posed problem is solved via singular value decomposition (SVD) [Lin00], which computes the closest point in the optimal set from a given point - commonly the collapsed edge midpoint.

We note that there is an intrinsic relation between incremental decimation via QEM and fine-to-coarse vertex clustering. After initialization, each vertex is considered as an individual cluster. An edge collapse operation merges two clusters into a new cluster associated with the new optimal vertex. During decimation, the evolution of clusters can be tracked with a tree structure.

5.4 Approach

Figure 5.2 provides an overview of our approach. The input is a 3D point cloud sampled on a closed surface, and the output is a surface triangle mesh. A quadric error metric (QEM) is first estimated for each point. Clustering of the input 3D points is then initialized with random points chosen as initial generators. Clustering is performed by alternating partitioning, via region growing with updating the generators, until a maximum error tolerance is met for each cluster. Refinement is performed by adding batches of new generators where the error is large. The output mesh is extracted by solving a constrained binary integer problem.

5.4.1 Quadric estimation

The original QEM approach designed for mesh decimation utilizes the mesh triangles to initialize a quadric per vertex, by summing plane quadrics on the 1-ring adjacent facets of each vertex. We first estimate a planar quadric per input point, based on a local normal and area estimation, and then compute a diffused quadric per point in order to capture its local geometry.

Denote by $\mathcal{P}$ the input 3D point cloud. Denote by p_i a point of $\mathcal{P} \in \mathbb{R}^3$ with normal $n_i \in \mathbb{R}^{3\times 1}$. We assume that normal n_i is either read from the input data, or estimated using a local normal

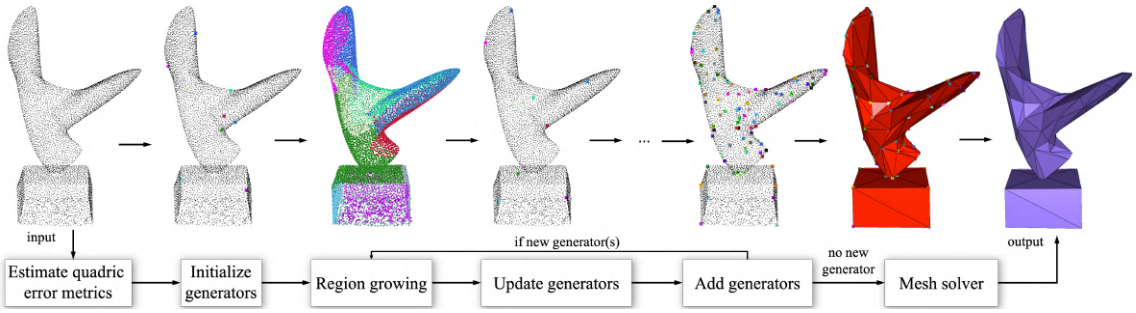


Figure 5.2 – *Overview : reconstruction via quadric error metrics.*

estimation method. Normal orientation is not required. We compute the plane quadric of p_i as :

$$Q_{p_i} = (n_i^x, n_i^y, n_i^z, -n_i \cdot p_i^T)^T \cdot (n_i^x, n_i^y, n_i^z, -n_i \cdot p_i^T). \tag{5.4}$$

To compute the diffused quadrics, we first need to define a neighborhood and support area for each point. We choose by default the k-Nearest Neighbor (KNN) graph for both computations. The support area a_i for point p_i is estimated as :

$$a_{p_i} = \frac{1}{2k^2} \cdot \Big(\sum_{p_j | (p_j, p_i) \in \mathrm{KNN}(\mathcal{P})} \|p_i - p_j\| \Big)^2. \tag{5.5}$$

The diffused quadrics for a point p_i is then computed as :

$$Q_{v_i} = \sum_{p_j | (p_j, p_i) \in \mathrm{KNN}(\mathcal{P})} a_{p_j} \cdot Q_{p_j}. \tag{5.6}$$

Such a quadric reflects an approximation of the local geometry and point density. Figure 5.3 depicts the estimated diffused quadrics centered at each vertex of two point clouds, using quadric ellipsoids. Each ellipsoid represents a quadric error isosurface, that is the locus of points with equal errors. The center of the quadric realizes the error minimum. Intuitively, we can classify ellipsoids into three main types : pancakes, cigars and balls, corresponding to smooth areas, creases and corners.

The above estimation assumes that the points are sampled on a smooth surface. Our initialization step is flexible and can also take into account additional geometric information that would be available as input, e.g. information on sharp features via multiple local plane quadrics. However, the strength of our method is to avoid the ill-posed detection step, and yet to recover sharp creases even when using diffused quadrics that mollify the sharp features existing in the sampled physical object.

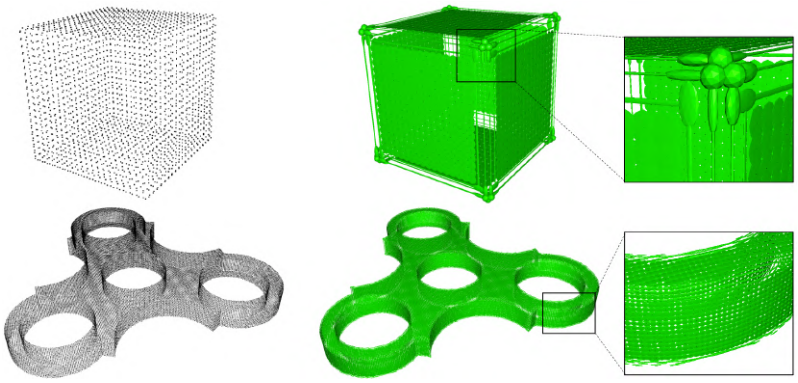


Figure 5.3 – *Quadric ellipsoids on two input point clouds.*

5.4.2 Initialization

Clustering of the input points is initialized by randomly selecting a handful of input points as initial cluster generators $\{c_i | 1 \leq i \leq m\}$. The generators coincide with the input points after initialization, but this is no longer the case later as they will be updated as optimal QEM 3D points.

5.4.3 Clustering

The clustering step operates by alternating partitioning with the generator updating until the clusters stabilize or a maximum number of iterations is reached.

Partitioning It proceeds by region growing. More specifically, growing progresses on the KNN graph greedily one point at a time, with a priority queue that minimizes a cost E combining QEM and Euclidean distance. The cost of adding a point p_i to the cluster l_j is :

$$E(p_i, l_j) = [c_j, 1]^T \cdot Q_{v_i} \cdot [c_j, 1] + \lambda \cdot \|p_i - c_j\|^2, \tag{5.7}$$

where c_j denotes the current generator of cluster l_j and λ denotes a coefficient for regularizing region growing on planar areas. λ is set by default to a small value such that the coarse-to-fine refinement focuses mainly on minimizing the QEM error. There are three reasons for using such a Euclidean distance term : 1) to regularize partitioning of flat or straight areas by competing growing fronts, so that we obtain a Voronoi-like partition instead of noisy adjacency frontiers, 2) to obtain evenly distributed vertices on flat areas or along straight creases, when these areas are densely populated by generators, and 3) to provide enough edge candidates for meshing concave planar clusters. To make the two metrics compatible, we set λ to be k times the squared average spacing of the input point cloud, because Q_{v_i} is computed as the sum of plane quadrics weighted by support areas. See Figure 5.4.

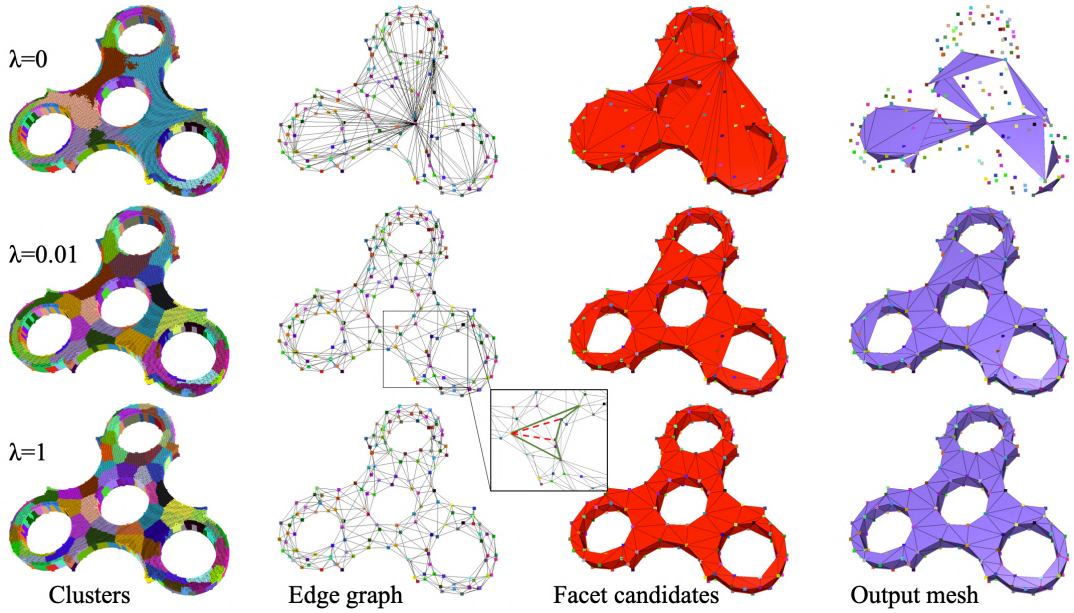


Figure 5.4 – *Partitioning and λ coefficient. On flat areas the QEM errors equate zero. This yields a clustering with noisy adjacency frontiers, and an insufficient adjacency between clusters for the meshing step. Increasing λ improves the configuration, including for concave clusters.*

Once a point is labeled as l_j, all its unlabeled neighbors in the KNN graph are pushed to the priority queue with label j as new candidates. Such a partitioning proceeds until the priority queue is empty.

Note that in Equation (5.7), $Q_{v_i} \cdot [c_j, 1]$ measures an error that is oriented, i.e. realized by the generator point c_j of cluster l_j for the quadric of the candidate point to growing. Consider a 3D point set sampled on a perfect cube, with little noise. If a cluster has a generator point on a corner C of the said cube, then the error would be very small for the (almost flat pancake) quadrics of the candidate points sampled on the three faces adjacent to C, as well as for the (thin cigar) quadrics of the candidate points on the three creases adjacent to C (see Figure 5.5, pink area). A similar behavior is observed when a cluster has a generator point on a crease point of the cube, but with two adjacent faces and creases (see Figure 5.5, green area). Such a behavior is also similar for the tip of a cone, hence our reference to conical elements in Section 5.2.

Generator updating In this step, the optimal generator of each cluster is recomputed. For cluster l_j, the optimal point c_j^* is computed as :

$$c_j^* = \underset{p \in \mathbb{R}^3}{\arg\min} [p, 1]^T \cdot Q_{c_j} \cdot [p, 1]. \tag{5.8}$$

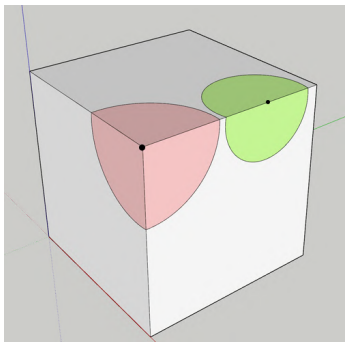


Figure 5.5 – *Illustration of generators on a perfect cube.*

More specifically, c_j^* is obtained by solving the following linear system :

$$Ac^* = \begin{pmatrix} Q^{11} & Q^{12} & Q^{13} & Q^{14} \\ Q^{12} & Q^{22} & Q^{23} & Q^{24} \\ Q^{13} & Q^{23} & Q^{33} & Q^{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} c^* = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.9}$$

When A is not invertible, c^* is computed by a SVD solver.

Note that the next clustering iteration, and in particular its partitioning step that proceeds by region growing over the KNN graph, requires each updated generator c_j^* to be associated with an input point in $\mathcal{P}$. Each generator c_j^* is associated its nearest input point in cluster l_j. Only the optimal points c^* are later used for constructing the output surface triangle mesh.

Clustering terminates when there is no change between the old and new set of generators, or when a user-selected maximum number of iterations is reached.

5.4.4 Batch splitting

The maximum QEM error is monitored for all points of a cluster, once clustering terminates. We adopt a farthest point approach to design a splitting operator that inserts a new generator in the cluster realizing the maximum error. More specifically, for each cluster l_j, we find the point labeled j that maximizes Q_{c_j}, i.e.

$$p_{\max}(l_j) = \underset{p_i \in l_j}{\arg\max} [p_i, 1]^T \cdot Q_{c_j} \cdot [p_i, 1]. \tag{5.10}$$

If the corresponding QEM exceeds a maximum user-defined tolerance, then $p_{\max}(l_j)$ is added as a new candidate for cluster splitting. We then greedily compute an independent set of clusters to be split, enforcing that two adjacent clusters cannot be split in the same splitting batch to avoid over-refining. As it is not intuitive to define a meaningful QEM tolerance, we take a Euclidean

distance tolerance parameter and convert it to a QEM scale according to the relationship between these two metrics, see Section 5.3.

Alternating between partitioning and batch splitting stops when all clusters satisfy the user-defined tolerance error or when a maximum number of iterations is reached.

5.4.5 Meshing

The current partition of clusters yields a set of vertices that are optimal in the QEM sense. It remains to connect these vertices by finding a set of triangle facets. We first derive a graph of edges from the adjacency between clusters, from which we compute facet candidates and then solve a constrained binary program (CBP) to extract the output surface triangle mesh.

Edge graph. After partitioning, a graph of edges is derived from the adjacency between clusters in the KNN graph of the input point cloud.

Facet candidates. Given the above edge graph, we search for 3-cycles in the graph by selecting all triplets of vertices that are mutually connected by an edge. This yields a set of triangle facet candidates, possibly overlapping in some areas. It remains to select a subset of these facets.

Mesh extraction. From the set of facet candidates, we wish to keep the ones that fit well, cover well the input 3D point cloud and favor a 2-manifold mesh. Building upon the PolyFit approach [NW17], we minimize an objective function rewarding a data fitting term F_f and a data coverage term F_c, under a 2-manifold constraint. More specifically, we assign a binary label b_{f_i} for each facet candidate f_i and a binary label b_{e_i} for each edge e_i. Label one indicates that the facet or edge is kept for the final mesh, and label zero indicates that it is discarded. We then minimize :

$$\max_{\{b_{f_1},\dots,b_{f_n}\}} \sum_{i=1}^{n} b_{f_i} \cdot \Big(F_f(f_i) + F_c(f_i)\Big) \tag{5.11}$$

$$\text{s.t.} \quad 2b_{e_i} - \sum_{f_j \text{ around } e_i} b_{f_j} = 0, \quad \forall e_i. \tag{5.12}$$

Figure 5.2 (right) illustrates facet filtering : some quads overly covered by four red facets are later triangulated with two triangles in the output mesh. When the 2-manifold property is not sought after, we can deactivate the constraints and search instead for a balance between high fitting, high coverage and low complexity of output. We can then reconstruct non-manifold meshes with boundaries by simply selecting facets such that $F_f(f_i) + F_c(f_i) - 1 > 0$.

Fitting term. For each facet candidate, we find all input points whose distance to the triangle facet is smaller than a tolerance error ϵ and compute the fitting term as :

$$F_f(f_i) = \sum_{p_j | d(p_j, f_i) < \epsilon} \Big(1 - \frac{d(p_i, f_j)}{\epsilon}\Big). \tag{5.13}$$

Coverage term. Coverage prevents large triangles from covering empty areas. For each facet candidate, the ϵ-selected input points are projected onto its supporting plane. Then, we construct a 2D alpha shape with alpha set to s times the average spacing of $\mathcal{P}$ and compute the ratio between the total area of the triangles of the alpha shape, and the area of the facet. s is by default set to 5.

$$F_c(f_i) = \min\Big(1, \frac{\text{area}\big(\alpha(\{p_j | d(p_j, f_i) < \epsilon\})\big)}{\text{area}(f_i)}\Big) \tag{5.14}$$

Manifold constraint. To favor 2-manifold output meshes, a constraint is added for each edge e_i such that if it is kept by the solver ($e_i = 1$), then exactly two triangles adjacent to e_i must be kept, and zero adjacent triangles otherwise (Equation (5.12)).

5.5 Experiments

We implemented our approach in C++, using several libraries : CGAL [The21] for geometric computations, Eigen [GJ+10] for linear algebra and SCIP [BB+21] for solving binary linear problems. All experiments are conducted on a MacBook Pro with a 2.9GHz Quad-Core Intel i7 CPU and 16GB memory. Our implementation is running on a single core.

5.5.1 Qualitative results

Figure 5.6 – *Reconstructing a capsule. Left : input 3D point cloud. Middle : Poisson+QEM with 83 vertices. Right : our reconstruction with 83 vertices.*

Figure 5.6 validates our concise reconstruction approach on a 3D point set sampled on a capsule made up of an open cylinder and two half-spheres. The clustering algorithm generates elon-

gated clusters on parabolic areas and isotropic clusters on the spherical areas, that translate into skinny and isotropic triangles, respectively.

Figure 5.1 illustrates our approach on a point cloud sampled on a "G" letter, containing curved sharp creases. The variational clustering approach tends to equidistribute the errors among clusters where possible, translating into evenly placed vertices along the curved creases. Figure 5.7 shows three concise reconstructions from a point cloud sampled on a mechanical part.

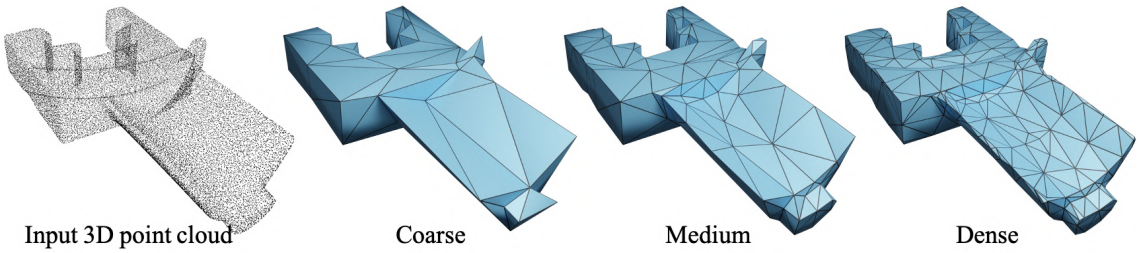


Figure 5.7 – *Reconstructing a blade. Left : input 3D point cloud. Right : reconstructions with increasing mesh complexity.*

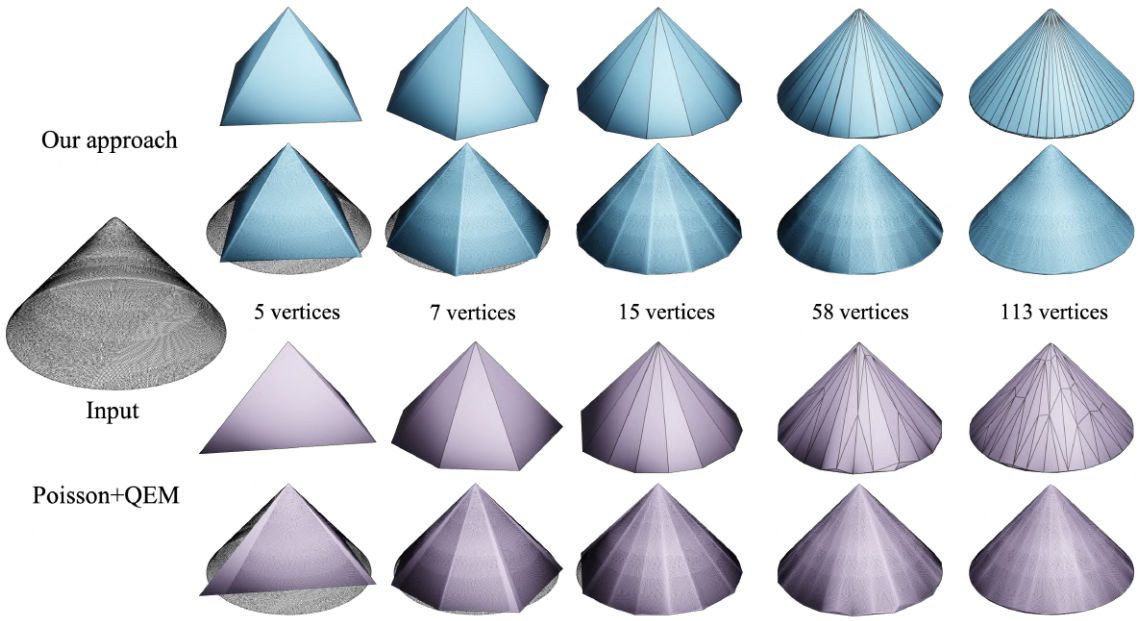


Figure 5.8 – *Reconstructing a smoothed circular cone. Left : input 3D points. Top : our reconstruction with increasing resolution. Meshes are shown with or without the input points. Bottom : Poisson reconstruction followed by QEM-based mesh decimation.*

Figure 5.8 highlights the difference between variational partitioning and dense Poisson reconstruction followed by greedy QEM decimation [KH13, GH97], when dealing with smooth creases. Creases should be reconstructed as sharp edges at coarse scales, and as smooth edges at finer scales. The input point set is sampled on a circular cone without a boundary, with a smoothed tip

and crease at the bottom. As well as being low-memory, our reconstruction is more symmetric than the greedy approach, reflecting the error equidistribution among clusters. The tip remains as a unique isolated corner for 15 vertices, then only the smoothed tip and crease start being refined.

Figure 5.9 shows our method at work on more diverse sharp features, including sharp creases subtending small angles. The cluster anisotropy reflects the local geometry and the generators are located on sharp corners, creases and curved areas. Other reconstructions with two resolutions are shown Figure 5.10.

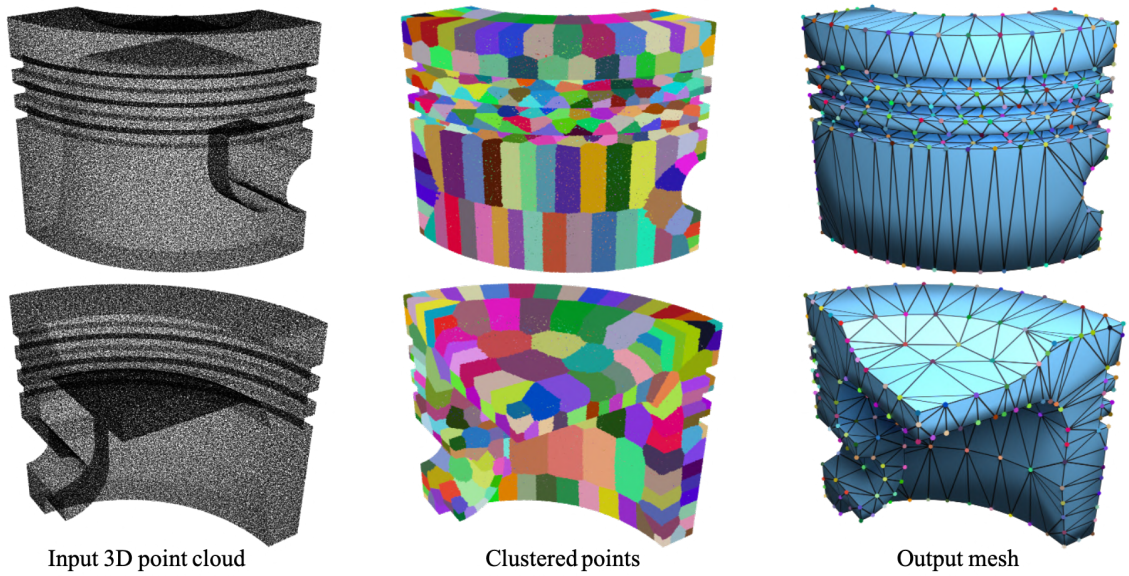


Figure 5.9 – *Reconstructing a piston. Left : input 3D point cloud. Middle : input points after clustering. Right : output mesh, and vertices colored by their corresponding clusters. The vertices are preferably located on sharp corners and creases.*

5.5.2 Quantitative results

Table 5.1 reports maximum Euclidean distance errors from the 3D point cloud to the output mesh, for comparing with Poisson followed by QEM-based decimation. A blue cell indicates the smallest values.

Model	**Capsule**	**Cone (#v :5)**	**Cone (#v :7)**	**Cone (#v :15)**	**Cone (#v :58)**	**Cone (#v :113)**
Max (P+Q)	0.07489	0.3836	0.1078	0.06092	0.01455	0.008757
Max (Ours)	0.03256	0.2561	0.08905	0.01721	0.007488	0.004118
Model	**Hilbert**	**Hilbert (0.5%)**	**Hilbert (1%)**	**Bunny**	**Part**	**Hand**
Max (P+Q)	1.6725	1.7233	1.5873	0.00328	2.145	0.01345
Max (Ours)	0.1729	1.0565	2.0487	0.002055	2.1063	0.008587
Model	**Elephant (Middle)**	**Elephant (Right)**	**Mother (Middle)**	**Mother (Right)**	**Rocker arm (Middle)**	**Rocker arm (Right)**
Max (P+Q)	0.04299	0.01088	4.8591	1.3958	0.002321	0.001147
Max (Ours)	0.0232	0.007387	4.5793	1.1293	0.001209	0.0005845

TABLE 5.1 – *Maximum distances from 3D point clouds to output meshes.*

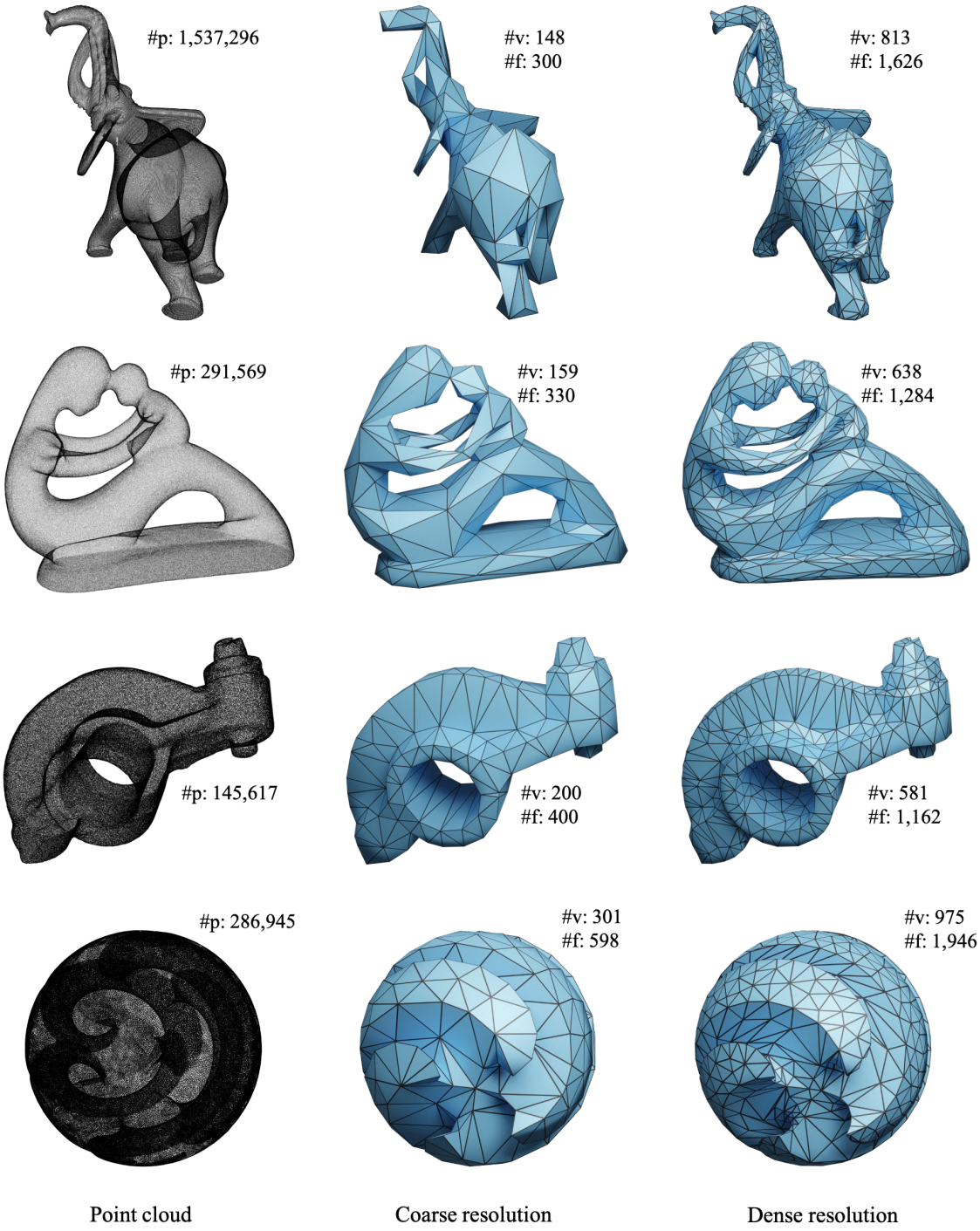


Figure 5.10 – *Meshes reconstructed with two resolutions. #p : number of points. #f : number of faces. #v : number of vertices. From top to bottom : Elephant, Mother, Rocker arm, Sharp sphere.*

5.5.3 Timing and peak memory

Figure 5.11 plots timings and peak memory consumption against number of input points or complexity of the output mesh. The input point cloud is uniformly sampled on a sphere. The memory consumption is only relative, as our current implementation is interactive and the rendering part consumes a large memory fraction of the whole program.

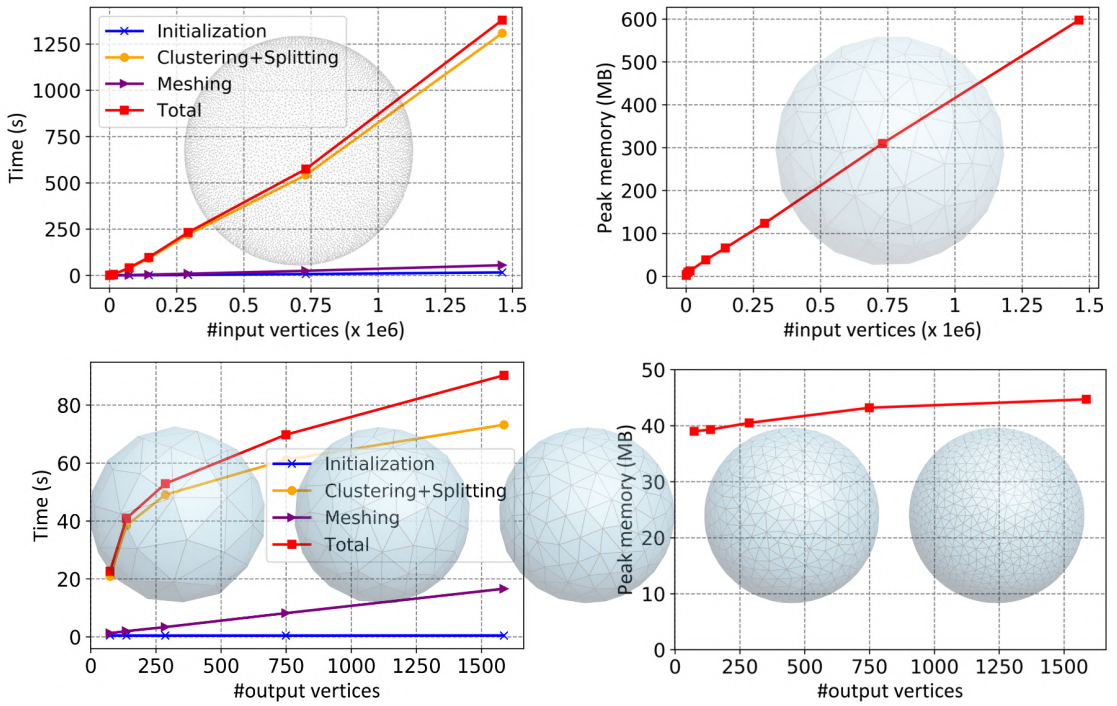


Figure 5.11 – *Timing and peak memory. Top : the number of input points increases from 732 to 1M, while the complexity of the output mesh is constant (150 vertices). Bottom : the number of input points is constant (73k points) while the complexity of the reconstructed mesh increases.*

5.5.4 Comparison

Figure 5.12 and 5.13 show visual comparisons. We compare our method with Poisson reconstruction, Poisson followed by QEM-based decimation and VSA, kinetic shape reconstruction [BL20], partitioning followed by kinetic shape reconstruction [YL22], an interpolant Delaunay-based approach [RGA+21], a recent feature-aware shape reconstruction RFEPS [XWD+22], and RFEPS followed by QEM-based decimation and VSA. Note that the last two approaches and our approach make no assumption about normal orientation, while the others require oriented normals. We can match the exact same mesh complexity only for Poisson followed by QEM decimation, as the other methods deal with planar shapes before meshing. For the Stanford bunny, the differences between the two meshes with the same complexity are subtle

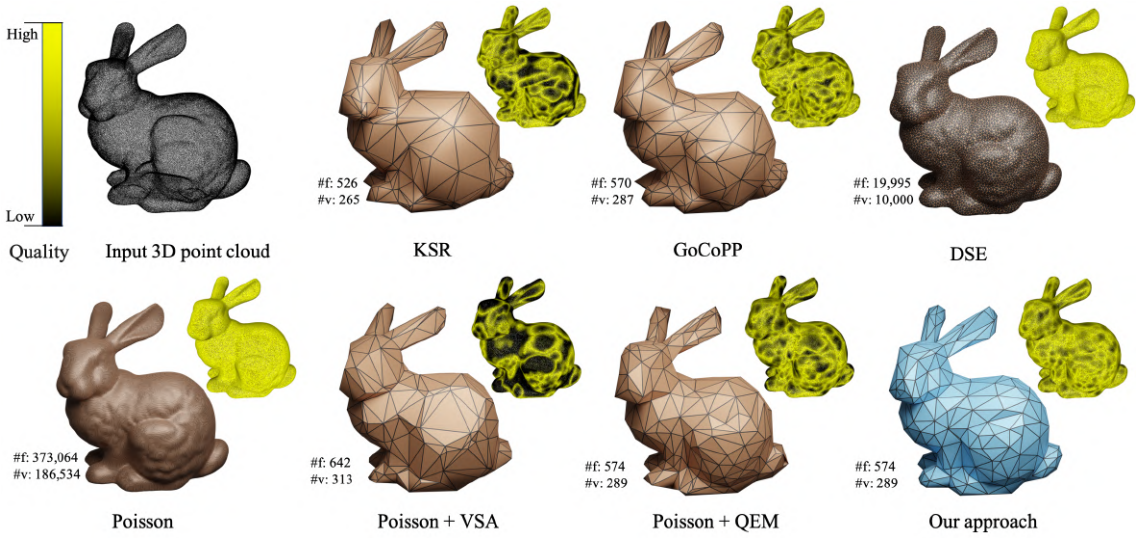


Figure 5.12 – *Comparisons on the bunny. KSR : kinetic shape reconstruction [BL20]. DSE : Delaunay surface elements [RGA+21]. Partitioning : [YL22].*

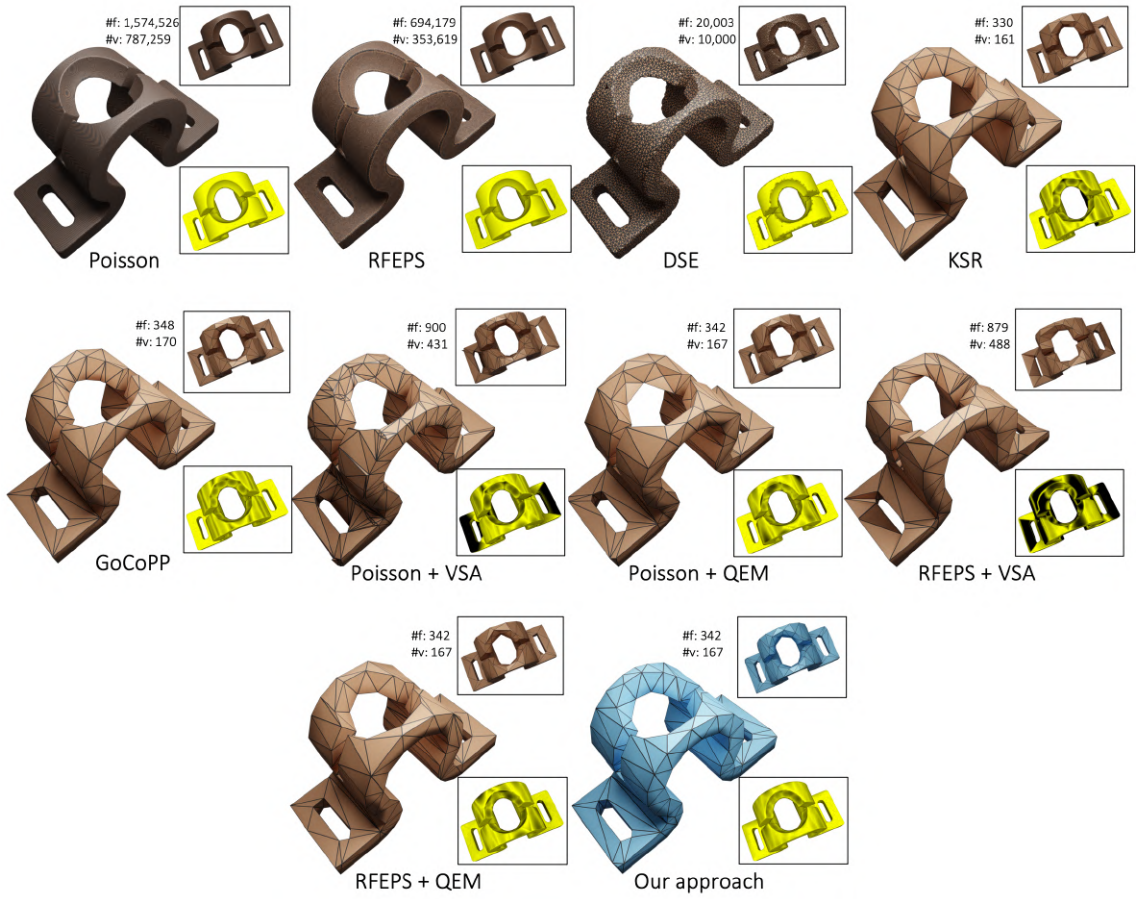


Figure 5.13 – *Comparisons on a mechanical part (point cloud taken from RFEPS [XWD+22]). KSR denotes kinetic shape reconstruction. DSE denotes Delaunay surface elements.*

but noticeable by comparing the shading of the triangles with the one of the dense Poisson reconstruction (e.g., nose, crease of the neck and tail). The ear is discretized differently, with a vertex located on a saddle point.

5.5.5 Robustness

Figure 5.14 illustrates the stability of the clustering step and approximation errors with respect to four different random initializations. Some salient features such as the saddle in the palm are approximated with some stability.

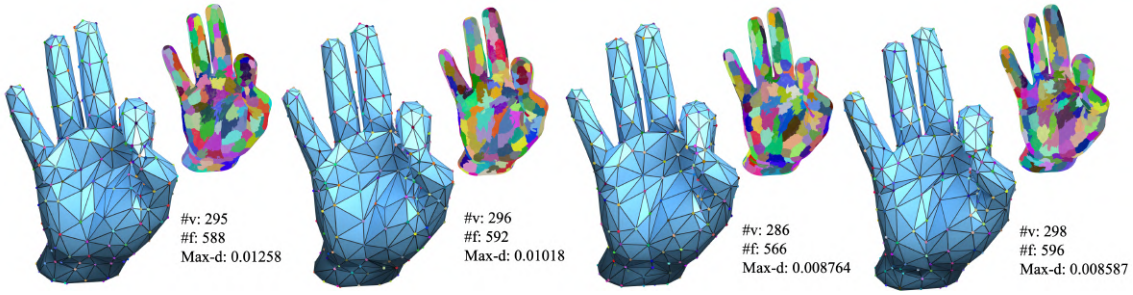


Figure 5.14 – *Meshes reconstructed from four different initializations.*

Figure 5.15 evaluates robustness to noise on dense point clouds sampled on a CAD model, altered with random uniform noise with different magnitude–up to 3% of the bounding box diagonal. We measure : (1) the average distance from a densely sampled reconstructed mesh to the ground truth, (2) the average distance from densely sampled ground truth sharp creases to the reconstructed mesh and (3) the manifoldness of the output mesh.

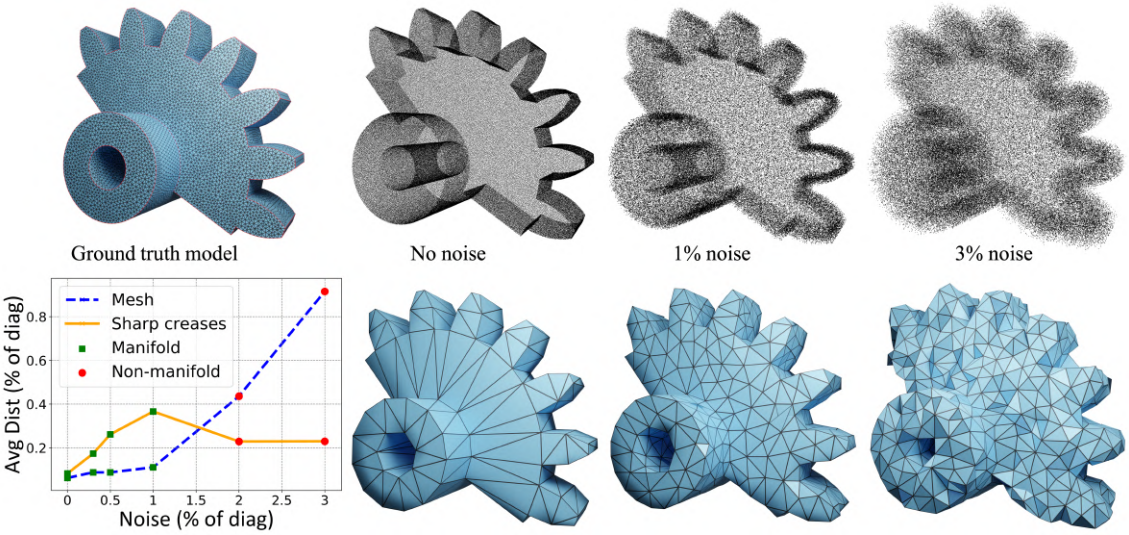


Figure 5.15 – *Robustness to noise.*

Figure 5.16 illustrates robustness to noise on a genus 120 model : reconstruction degrades gracefully when noise increases.

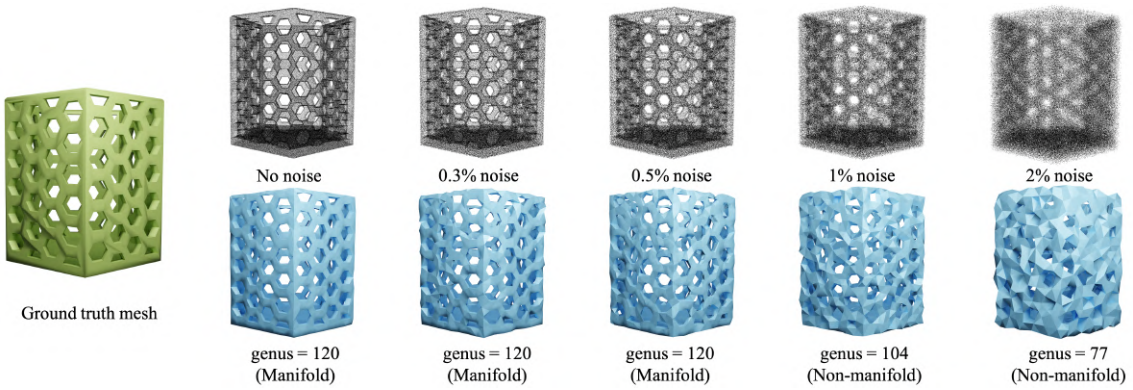


Figure 5.16 – *Robustness to noise on a genus 120 hex pen pot (taken from Thingi10K [ZJ16]).*

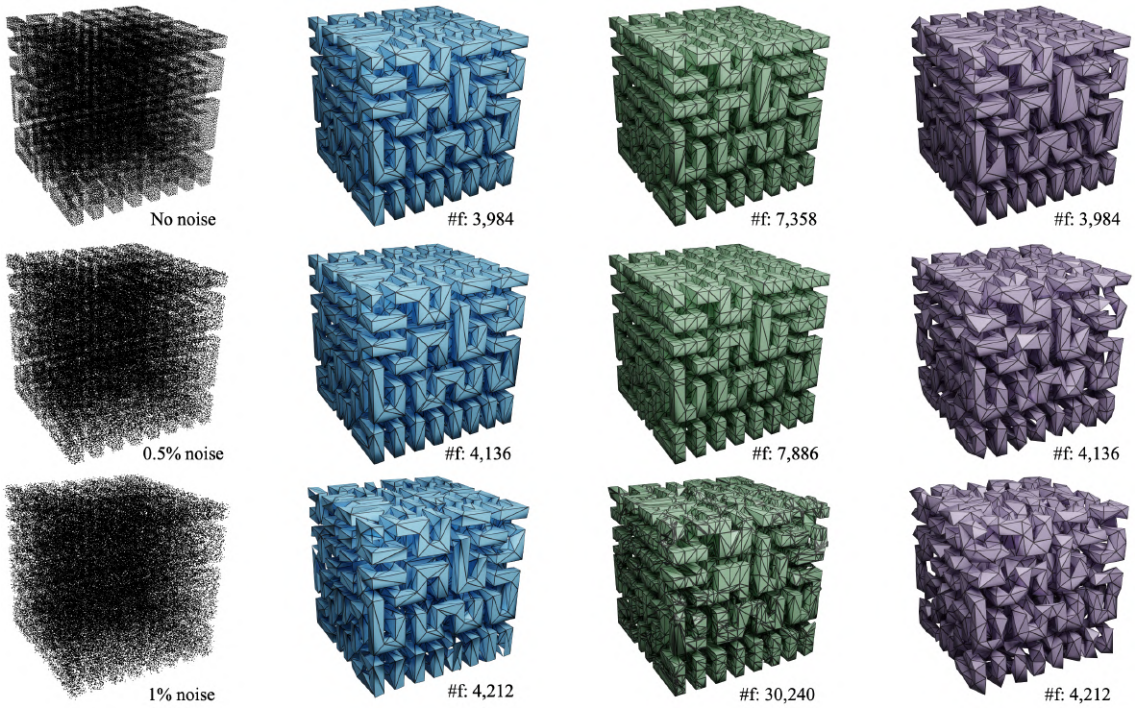


Figure 5.17 – *Robustness to noise on the Hilbert cube. First column : noisy point cloud. Second column : our method. Third column : output of Kinetic reconstruction. Fourth column : output of Poisson reconstruction followed by QEM-based mesh decimation. #f : number of faces.*

Figure 5.17 compares our method with kinetic shape reconstruction [BL20], and with Poisson reconstruction followed by QEM decimation, on a piecewise-planar Hilbert cube with increasing random uniform noise.

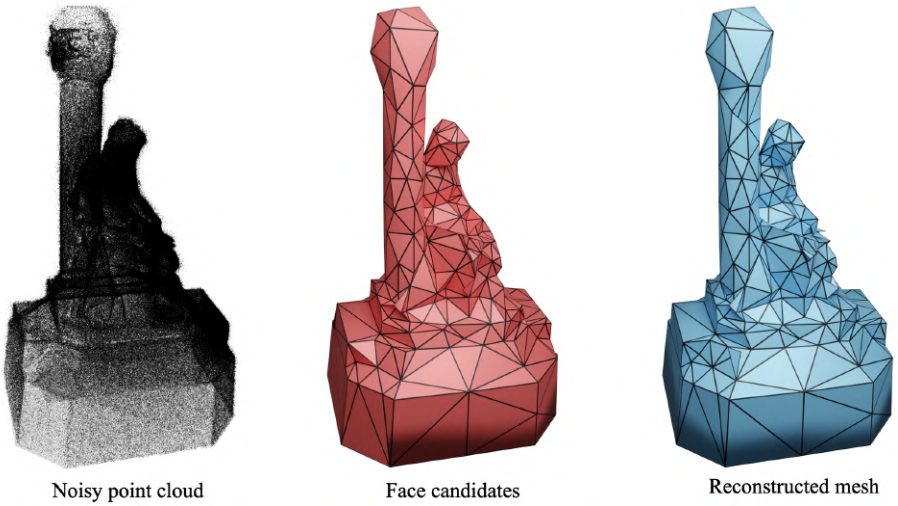


Figure 5.18 – *Left : point cloud generated by photogrammetry, with noise and outliers. We estimated point normals on the 100 nearest neighbors in order to smooth the normal field. Middle : facet candidates deduced from clustering. Right : reconstructed mesh.*

Figure 5.18 shows results on a point cloud generated by photogrammetry, with a large amount of noise and outliers.

5.5.6 Limitations

Over-refinement. Batch splitting steps may over-refine some areas, despite the independent set, so that some straight creases may be overly discretized. A richer set of operators (e.g., join or switch) may help but can lead to looping between refinement and coarsening of the partitioning. We leave this for future work.

Fold-overs. The constraints of the binary solver guarantee that each edge is adjacent to either 0 or 2 facets, making the output mesh orientable and free of non-manifold edges. Nevertheless, it does not prevent from having two selected faces around an edge nearly co-planar with opposite orientations (see inset figure - the output mesh contains a fold-over). A larger value for λ parameter (distance for region growing) helps to reduce the problem but does not solve it entirely. In addition, we cannot get rid of non-manifold vertices.

Boundaries. Clustering does not place generators on boundary curves, unless boundary points are first detected and orthogonal quadrics are added there. This contradicts our initial desire to avoid resorting to a detection step. Our approach is designed to deal only with point clouds sampled on closed objects. The solver can fill some holes via the 2-manifold constraints, but only when

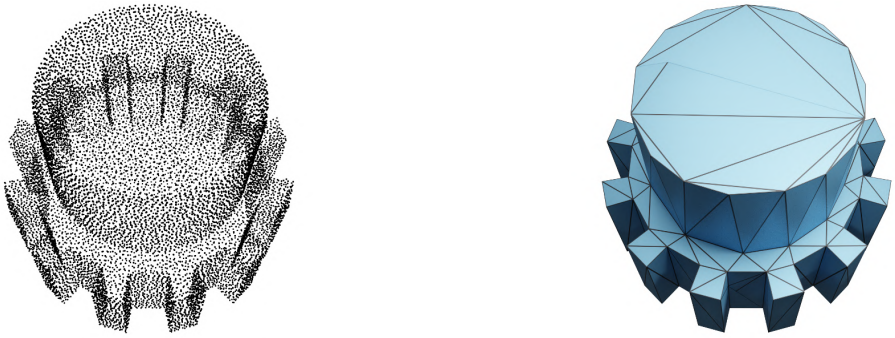


Figure 5.19 – *Reconstruction result contains a fold-over face.*

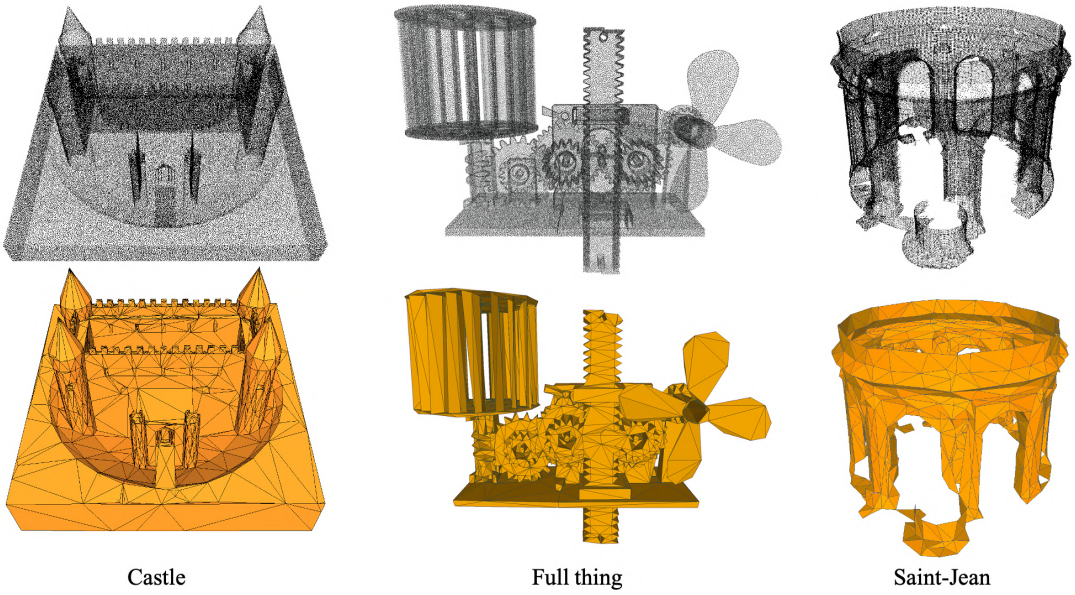


Figure 5.20 – *Non-manifold mesh reconstruction. By replacing the hard manifold constraint in the mesh solver by an objective function which rewards only low complexity of the output mesh, we can reconstruct meshes with boundaries and non-manifold meshes. The output is a triangle soup.*

there are enough candidate facets filling these holes. Deactivating the constraints is feasible (see Figure 5.20) but the 2-manifold property is no longer favored.

5.6 Discussion

We proposed a variational shape reconstruction method leveraging the strengths of two concepts : quadric error metrics (QEM) and variational partitioning. It can deal with unoriented

3D point clouds as input and is designed for concise triangle mesh reconstruction. Most previous variational approaches optimize planar elements before determining vertices and then a connectivity to form triangles. As the optimized errors often refer to infinite planes and the elements may be concave, determining vertices and meshing is difficult. We adopt a dual approach and optimize the placement of optimal cone points of clusters, that are later connected to form the output triangle mesh. Intuitively, this brings us one step closer to the final discretization, with improved consistency that translates into lower approximation errors and better recovery of sharp features and symmetries.

In future work we plan to extend our approach so that it can discover open boundaries during clustering, while keeping the 2-manifold property. We will also explore alternative solvers for the final mesh extraction steps, so that the user can better trade complexity for distortion, or balance with other objectives such as shape of the mesh elements. Finally, we will extend this approach in order to reconstruct piecewise-smooth surfaces with curved Bézier triangles.

CHAPITRE 6

Conclusion

6.1 Summary and conclusion

With the advancement of 3D data acquisition devices and the growing diversity of 3D applications, surface reconstruction is facing more and more challenges. In this thesis, we proposed three surface reconstruction approaches for reconstructing triangular surface mesh from raw 3D point clouds with laden defects. We stick to the following three principles throughout the whole design process :

- Our approaches reconstruct progressively the surface mesh in order that it is both input-aware and output-aware. The progressivity is controlled both by carefully chosen or user-defined criteria and by surface mesh qualities at the same time. We take care of multi-scale features through progressive reconstruction and attempt to achieve a good trade-off between the complexity and the accuracy of the surface mesh.
- We solve the reconstruction problem in a global manner. Global regularity helps to eliminate noises and fill holes. What's more, the surface mesh is guaranteed to be manifold without post-processing, which encourages wider usage in further applications.
- We combine multiple priors such that the ill-posed nature of the reconstruction problem is well-regularized around the whole surface. For instance, primitive priors are suitable for treating canonical parts, smoothness priors can be applied for free-form parts and data-driven priors are well-fitted to deal with sharp features.

In our first contribution (Chapter 2), we devise a progressively discretized domain for solving popular global implicit reconstruction approaches. Given an initial 3D Delaunay triangulation of the domain and an implicit solver, our approach iterates over three main steps (solve, optimization, adaptive refinement), all steps being designed to cooperate with each other and improve the conditioning of the solver, and the quality and complexity-distortion tradeoff the output isosurface mesh. In such a progressive approach, the implicit solver is no longer used once, but iteratively as a means to discover more and more details for the isosurface. The benefit is to reconstruct and generate altogether output meshes with well-shaped triangles and adapted to the intrinsic geometric complexity of the reconstructed surface, instead of to the input point set density, as in previous work.

Building upon the above domain discretization approach, we explore a progressive implicit reconstruction approach guided by both smoothness priors and primitive priors in Chapter 3. With the detected planar primitive set as preliminary information, the triangulation is split into primitive zones, free-form zones, and transition zones. By replacing the implicit function values of primitive zones with their signed distance values and eliminating them from the whole system, the other zones are resolved by a constrained global solver. Both the optimizer and the refiner are primitive-aware in order to reduce the information loss for the next iteration.

To generalize this approach to all types of primitives, we need to conceive an intelligent way to compute the intersections among primitives. Chapter 4 gives a possible solution to deal with this issue. We present a deep learning-based approach that detects and consolidates sharp feature points on raw 3D point clouds. Based on a high dimensional pointwise feature, a binary classifier selects a set of sharp feature points and a displacement regressor predicts a vector to consolidate them.

In Chapter 5, we propose a progressive surface reconstruction approach from unoriented point clouds guided by quadric error metrics. We start by selecting random seeds as an initial guess and then extend a region growing step to find the best clustering result which minimizes the QEM energy. More degrees of freedom are added as needed progressively according to the QEM criteria. The surface mesh is extracted from a 3D triangulation composed of all optimal cluster centers via a global solver.

6.2 Outlook

In this thesis, we focused on the surface reconstruction of raw 3D point clouds from a global and progressive perspective. However, there still remain several open problems in this domain. We detail three of them as follows.

Perspective 1 : How to treat open boundaries in global implicit reconstruction methods ?

Common global implicit reconstruction methods are designed on the assumption that the approximate surface is bounded and closed, which makes them inapplicable to large-scale scene reconstruction. These methods usually assemble global systems on an enlarged discrete domain bounded by the input point cloud and solve them with the Dirichlet boundary condition or the Neumann boundary condition. Without prior boundary information, the boundary condition cannot be properly involved, while acquiring the information is as hard as solving the reconstruction problem. In future work, we plan to devise learning-based methods to estimate boundary prior and then discover the surface mesh boundary in a progressive manner.

Perspective 2 : Can we leverage multi-modal data for regularizing the ill-posed nature of reconstruction problems ?

Surface reconstruction from raw point clouds faces great difficulty when dealing with ambiguous regions. A typical example is that we cannot distinguish with further information if two nearby groups of points belong to two surface patches or if it is due to a misalignment between devices. Another example is that we cannot determine if a region without input points is a hole or if it is due to missing data. In such situations, introducing multi-modal data (e.g. 2D images, RGB-D images, videos, etc.) can be a powerful regularization tool. We will explore this direction by conceiving reconstruction approaches combining more priors.

Perspective 3 : What is a good representation for deep learning-based reconstruction methods ?

In our thesis, we devised a learning-based approach for detecting sharp feature points. In our future work, we will attempt to conceive deep learning-based methods for reconstructing surfaces. A key breakthrough is to find a proper representation for 3D surfaces in the neural network. Current approaches rely on grids, graphs or implicit functions. None of them can describe completely the geometric and topologic properties of a surface mesh (e.g. gradient, curvature, manifoldness, genus). We intend to design a differentiable mesh that allows formulating of the above properties as loss functions. What's more, it will facilitate the integration of multi-modal data into the neural networks.

Bibliographie

[AA09] Marc Alexa and Anders Adamson. Interpolatory point set surfaces - convexity and Hermite data. *ACM Transactions on Graphics*, 28(2) :20, 2009.

[AB94] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6) :641–647, 1994.

[ABCO+01] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Point set surfaces. In *Proceedings of the Conference on Visualization*, pages 21–28. IEEE Computer Society, 2001.

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer graphics and Interactive Techniques*, pages 415–421, 1998.

[ACTD07] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In Alexander G. Belyaev and Michael Garland, editors, *Proceedings of the 5th Eurographics Symposium on Geometry Processing*, volume 257 of *ACM International Conference Proceeding Series*, pages 39–48. Eurographics Association, 2007.

[AFRS03] Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. Edge-Sharpener : Recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 62–69, Goslar, DEU, 2003. Eurographics Association.

[AHKSH20] Marc Alexa, Philipp Herholz, Maximilian Kohlbrenner, and Olga Sorkine-Hornung. Properties of Laplace operators for tetrahedral meshes. *Computer Graphics Forum*, 39(5) :55–68, 2020.

[ASGCO10] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. l1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics*, 29(5) :1–12, 2010.

[BB+21] Ksenia Bestuzheva, Mathieu Besançon, et al. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.

[BCRH15] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *International Conference on Digital Image Computing : Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2015.

[BK19] Onur Rauf Bingol and Adarsh Krishnamurthy. NURBS-Python : An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9 :85–94, 2019.

[BKK20] Jan Brandts, Sergey Korotov, and Michal Křížek. *Refinement Techniques*, pages 87–114. Springer International Publishing, Cham, 2020.

[BL20] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *ACM Transactions on Graphics*, 39(5) :1–14, 2020.

[BM22] Alexandre Boulch and Renaud Marlet. POCO : Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6302–6314, 2022.

[BMR+99] 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*, 5(4) :349–359, 1999.

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Cláudio T. Silva. State of the art in surface reconstruction from point clouds. In Sylvain Lefebvre and Michela Spagnuolo, editors, *35th Annual Conference of the European Association for Computer Graphics, Eurographics 2014 - State of the Art Reports*, pages 161–185. Eurographics Association, 2014.

[BTS+17] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Cláudio T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1) :301–329, 2017.

[CBC+01] 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. ACM, 2001.

[CC08] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2) :223–236, 2008.

[CDS13] Siu-Wing Cheng, Tamal K. Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. Chapman and Hall / CRC computer and information science series. CRC Press, 2013.

[CFG+15] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet : An information-rich 3d model repository. *ArXiv Preprint arXiv :1512.03012*, 2015.

[CGAY13] Ricard Campos, Rafael Garcia, Pierre Alliez, and Mariette Yvinec. Splat-based surface reconstruction from defect-laden point sets. *Graphical Models*, 75(6) :346–361, nov 2013.

[CGS19] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal ConvNets : Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[CLP10] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1261–1268. IEEE, 2010.

[CP05] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2) :121–146, 2005.

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. METRO : Measuring error on simplified surfaces. *Computer Graphics Forum*, 17 :167–174, 06 1998.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3) :905–914, aug 2004.

[CSHD21] Jiong Chen, Florian Schäfer, Jin Huang, and Mathieu Desbrun. Multiscale cholesky preconditioning for ill-conditioned problems. *ACM Transactions on Graphics*, 40(4) :1–13, 2021.

[CT11] Fatih Calakli and Gabriel Taubin. SSD : smooth signed distance surface reconstruction. *Computer Graphics Forum*, 30(7) :1993–2002, 2011.

[CTFZ22] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 41(4), 2022.

[CZ21] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 40(6), 2021.

[CZZ+17] Jianjun Chen, Jianjing Zheng, Yao Zheng, Zhoufang Xiao, Hang Si, and Yufeng Yao. Tetrahedral mesh improvement by shell transformation. *Engineering with Computers*, 33(3) :393–414, 2017.

[DCSA+14] Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando De Goes, and Mathieu Desbrun. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of Mathematical Imaging and Vision*, 48(2) :369–382, 2014.

[DDSD03] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics*, 22(3) :689–696, 2003.

[Dey06] Tamal K. Dey. *Curve and Surface Reconstruction : Algorithms with Mathematical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2006.

[DHLM05] Mathieu Desbrun, Anil N Hirani, Melvin Leok, and Jerrold E Marsden. Discrete exterior calculus. *ArXiv Preprint math/0508341*, 2005.

[DHOS07] Joel II Daniels, Linh K Ha, Tilo Ochotta, and Claudio T Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications*, pages 123–136. IEEE, 2007.

[DK91] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Transactions on Information and Systems*, 74(1) :214–224, 1991.

[DLM11] Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology*, 26 :538–547, 2011.

[DM98] Leonardo Dagum and Ramesh Menon. OpenMP : an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1) :46–55, 1998.

[FB81] 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*, 24(6) :381–395, 1981.

[FCOS05] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3) :544–552, 2005.

[FDC03] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics*, 22(3) :950–953, 2003.

[FG14] Simon Fuhrmann and Michael Goesele. Floating scale surface reconstruction. *ACM Transactions on Graphics*, 33(4) :46 :1–46 :11, 2014.

[FH+15] Yasutaka Furukawa, Carlos Hernández, et al. Multi-view stereo : A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2) :1–148, 2015.

[GBA+17] Paul Louis George, H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, D. Marcum, and L. Maréchal. *Mesh Generation and Mesh Adaptivity : Theory and Techniques*, pages 1–51. American Cancer Society, 2017.

[GG07] Gaël Guennebaud and Markus H. Gross. Algebraic point set surfaces. *ACM Transactions on Graphics*, 26(3) :23, 2007.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In G. Scott Owen, Turner Whitted, and Barbara Mones-Hattal, editors, *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209–216. ACM, 1997.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010.

[GKOM18] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. PCPNet : Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2) :75–85, 2018.

[GSH+07] Ran Gal, Ariel Shamir, Tal Hassner, Mark Pauly, and Daniel Cohen-Or. Surface reconstruction using local shape priors. In *Eurographics Symposium on Geometry Processing*, pages 253–262, 2007.

[GXT+21] Jingyu Gong, Jiachen Xu, Xin Tan, Jie Zhou, Yanyun Qu, Yuan Xie, and Lizhuang Ma. Boundary-aware geometric encoding for semantic segmentation of point clouds. 35(2) :1424–1432, 2021.

[GXW18] Xiaoyuan Guo, Jun Xiao, and Ying Wang. A survey on algorithms of hole filling in 3d surface reconstruction. *The Visual Computer*, 34(1) :93–103, 2018.

[HCJ19] Zhiyang Huang, Nathan Carr, and Tao Ju. Variational implicit point set surfaces. *ACM Transactions on Graphics*, 38(4) :124 :1–124 :13, 2019.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 295–302, Orlando, 1994. ACM.

[HLP+21] Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. PCEDNet : A neural network for fast and efficient edge detection in 3d point clouds. *ACM Transactions on Graphics*, 41(1), nov 2021.

[HMGC20] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2Mesh : A self-prior for deformable meshes. *ACM Transactions on Graphics*, 39(4) :126, 2020.

[Hop99] Hugues Hoppe. New quadric metric for simplifiying meshes with appearance attributes. In *Proceedings of the Conference on Visualization*, VIS '99, pages 59–66, Washington, DC, USA, 1999. IEEE Computer Society Press.

[HS13] Lei He and Scott Schaefer. Mesh denoising via L_0 minimization. *ACM Transactions on Graphics*, 32(4) :64 :1–64 :8, 2013.

[HT20] Joel Hass and Maria Trnkova. Approximating isosurfaces by guaranteed-quality triangular meshes. *Computer Graphics Forum*, 39(5) :29–40, 2020.

[HWG+13] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics*, 32(1) :1–12, 2013.

[HWW+22a] Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. Iterative Poisson surface reconstruction for unoriented points. *ACM Transactions on Graphics*, 41(4) :1–13, 2022.

[HWW+22b] Zhangjin Huang, Yuxin Wen, Zihao Wang, Jinjuan Ren, and Kui Jia. Surface reconstruction from point clouds : A survey and a benchmark. *ArXiv Preprint arXiv :2205.02413*, 2022.

[JDD03] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3) :943–949, 2003.

[JJC20] Yiwei Jin, Diqiong Jiang, and Ming Cai. 3D reconstruction using deep learning : a survey. *Communications in Information and Systems*, 20(4) :389–413, 2020.

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 339–346, 2002.

[JWZ11] Xiangmin Jiao, Duo Wang, and Hongyuan Zha. Simple and effective variational optimization of surface and volume triangulations. *Engineering with Computers*, 27(1) :81–94, 2011.

[Kaz05] Michael Kazhdan. Reconstruction of solid models from oriented point sets. In *Proceedings of the 3rd Eurographics Symposium on Geometry Processing*, page 73. Eurographics Association, 2005.

[KBH06] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, volume 256 of *ACM International Conference Proceeding Series*, pages 61–70. Eurographics Association, 2006.

[KCRH20] Misha Kazhdan, Ming Chuang, Szymon Rusinkiewicz, and Hugues Hoppe. Poisson surface reconstruction with envelope constraints. *Computer Graphics Forum*, 39(5) :173–182, 2020.

[KFS13] Dilip Krishnan, Raanan Fattal, and Richard Szeliski. Efficient preconditioning of Laplacian matrices for computer graphics. *ACM Transactions on Graphics*, 32(4) :1–15, 2013.

[KH13] Michael M. Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3) :29 :1–29 :13, 2013.

[KH19] Misha Kazhdan and Hugues Hoppe. An adaptive multi-grid solver for applications in computer graphics. *Computer Graphics Forum*, 38(1) :138–150, 2019.

[Kli08] Bryan Matthew Klingner. *Improving Tetrahedral Meshes*. PhD thesis, EECS Department, University of California, Berkeley, Nov 2008.

[KMJ+19] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

[KZB19] Adrien Kaiser, José Alonso Ybáñez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Computer Graphics Forum*, 38(1) :167–196, 2019.

[LA13] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. *Computer Graphics Forum*, 32(2pt2) :225–234, 2013.

[LAK20] Marios Loizou, Melinos Averkiou, and Evangelos Kalogerakis. Learning part boundaries from 3d point clouds. *Computer Graphics Forum*, 39(5) :183–195, 2020.

[LB16] Kai Wah Lee and Pengbo Bo. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering*, 3(2) :102–111, 2016.

[LC87] William E Lorensen and Harvey E Cline. Marching cubes : A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4) :163–169, 1987.

[LCL18] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net : Self-organizing network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018.

[LCS09] Jianfei Liu, YQ Chen, and SL Sun. Small polyhedron reconnection for mesh improvement and its implementation based on advancing front technique. *International Journal for Numerical Methods in Engineering*, 79(8) :1004–1018, 2009.

[LDSW21] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. PC2WF : 3d wireframe reconstruction from raw point clouds. 2021.

[Lev04] David Levin. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, pages 37–49. Springer, 2004.

[LGP+21] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1788–1797, 2021.

[Lin00] Peter Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 259–262. ACM, 2000.

[LSD+19] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019.

[LTB19] Hélène Legrand, Jean-Marc Thiery, and Tamy Boubekeur. Filtered quadrics for high-speed geometry smoothing and clustering. *Computer Graphics Forum*, 38(1) :663–677, 2019.

[LTLH19] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[MdGD+10] Patrick Mullen, Fernando de Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Signing the Unsigned : Robust surface reconstruction from raw pointsets. *Computer Graphics Forum*, 29(5) :1733–1741, 2010.

[MLR+22] Corentin Mercier, Thibault Lescoat, Pierre Roussillon, Tamy Boubekeur, and Jean-Marc Thiery. Moving level-of-detail surfaces. *ACM Transactions on Graphics*, 41(4) :1–10, 2022.

[MOG10] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6) :743–756, 2010.

[MON+19] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks : Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470. Computer Vision Foundation / IEEE, 2019.

[MRA+22] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. DEF : Deep estimation of sharp geometric features in 3d shapes. *ACM Transactions on Graphics*, 41(4) :1–22, 2022.

[Nie04] Gregory M Nielson. Dual marching cubes. In *IEEE Visualization*, pages 489–496. IEEE, 2004.

[NLNZ16] 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*, 8(9) :710, 2016.

[NNZ21] Alexander Naitsat, Gregory Naitzat, and Yehoshua Y Zeevi. On inversion-free mapping and distortion minimization. *Journal of Mathematical Imaging and Vision*, 63(8) :974–1009, 2021.

[NW17] Liangliang Nan and Peter Wonka. PolyFit : Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2353–2361. IEEE, 2017.

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3) :463–470, jul 2003.

[ÖGG09] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer graphics forum*, 28(2) :493–501, 2009.

[OVJ+22] Sven Oesau, Yannick Verdie, Clément Jamin, Pierre Alliez, Florent Lafarge, Simon Giraudot, Thien Hoang, and Dmitry Anisimov. Shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.1 edition, 2022.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. DeepSDF : Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174. Computer Vision Foundation / IEEE, 2019.

[PFVM20] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *European Conference on Computer Vision*, pages 103–118. Springer, 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch : An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[PJL+21] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points : A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34 :13032–13044, 2021.

[QHL+21] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet, and Bernard Ghanem. ASSANet : An anisotropic separable set abstraction for efficient point cloud representation learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[Qiu21] Yixuan Qiu. Spectra, 2021.

[QSMG17] 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/CVF Conference on Computer Vision and Pattern Recognition, pages 652–660, 2017.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ : Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[RGA+21] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22–31, 2021.

[RLBG+20] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet : Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum*, 39(1) :185–203, 2020.

[RUG17] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet : Learning deep 3d representations at high resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6620–6629. IEEE Computer Society, 2017.

[RVDHV06] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5) :248–253, 2006.

[SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In Alexander G. Belyaev and Michael Garland, editors, *Proceedings of the 5th Eurographics Symposium on Geometry Processing*, volume 257 of *ACM International Conference Proceeding Series*, pages 109–116. Eurographics Association, 2007.

[SDK09] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum*, 28(2) :503–512, 2009.

[SGWJ18] Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. Natural boundary conditions for smoothing in geometry processing. *ACM Transactions on Graphics*, 37(2) :23 :1–23 :13, 2018.

[She98] Jonathan Richard Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In Ravi Janardan, editor, *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 86–95. ACM, 1998.

[She02] Jonathan Shewchuk. What is a good linear finite element ? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*, 73 :137, 2002.

[Si15] Hang Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 41(2) :11 :1–11 :36, 2015.

[SJS+18] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet : Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[SLA15] David Salinas, Florent Lafarge, and Pierre Alliez. Structure-aware mesh decimation. *Computer Graphics Forum*, 34(6) :211–227, 2015.

[SLS+06] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse-to-fine surface reconstruction. *Computer Graphics Forum*, 25(3) :389–398, 2006.

[SOS05] Chen Shen, James F O'Brien, and Jonathan R Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2005 Courses*, page 204. ACM, 2005.

[SW04] Scott Schaefer and Joe Warren. Dual marching cubes : Primal contouring of dual grids. In *12th Pacific Conference on Computer Graphics and Applications*, pages 70–76. IEEE, 2004.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2) :214–226, 2007.

[SY10] Y. Sahillioğlu and Y. Yemez. Coarse-to-fine surface reconstruction from silhouettes and range data using mesh deformation. *Computer Vision and Image Understanding*, 114(3) :334–348, 2010.

[SYM10] Nader Salman, Mariette Yvinec, and Quentin Mérigot. Feature preserving mesh generation from 3d point clouds. *Computer Graphics Forum*, 29(5) :1623–1632, 2010.

[SZP20] Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Variational shape approximation of point set surfaces. *Computer Aided Geometric Design*, 80 :101875, 2020.

[SZT19] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. MVX-Net : Multimodal voxelnet for 3d object detection. In *International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[TDB17] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks : Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[TGB13] Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. Sphere-Meshes : Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics*, 32(6), nov 2013.

[The21] The CGAL Project. CGAL user and reference manual, 2021.

[TK20] Philip Trettner and Leif Kobbelt. Fast and robust QEF minimization using probabilistic quadrics. *Computer Graphics Forum*, 39(2) :325–334, 2020.

[Vis] VISIONAIR advanced infrastructure for research.

[VWP13] Dimitris Vartziotis, Joachim Wipper, and Manolis Papadrakakis. Improving mesh quality and finite element solution accuracy by GETMe smoothing in solving the Poisson equation. *Finite Elements in Analysis and Design*, 66 :36–52, 2013.

[Wan06a] Charlie C. L. Wang. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(4) :629–639, 2006.

[Wan06b] Charlie C. L. Wang. Incremental reconstruction of sharp edges on mesh surfaces. *Computer-Aided Design*, 38(6) :689–702, 2006.

[WHH10] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Sharp feature detection in point clouds. In *Shape Modeling International Conference*, pages 175–186. IEEE, 2010.

[WK05] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3) :277–284, 2005.

[WLG+17] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN : Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics*, 36(4) :1–11, 2017.

[WSL+19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5) :1–12, 2019.

[WSLT18] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN : A patch-based deep representation of 3d shapes. *ACM Transactions on Graphics*, 37(6) :1–11, 2018.

[WXX+20] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. PIE-Net : Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems*, 33 :20167–20178, 2020.

[WYL+14] Ruimin Wang, Zhouwang Yang, Ligang Liu, Jiansong Deng, and Falai Chen. Decoupling noise and features via weighted l1-analysis compressed sensing. *ACM Transactions on Graphics*, 33(2) :1–12, 2014.

[WYZC13] Jun Wang, Z Yu, W Zhu, and J Cao. Feature-preserving surface reconstruction from unoriented, noisy point data. *Computer Graphics Forum*, 32(1) :164–176, 2013.

[XWD+22] Rui Xu, Zixiong Wang, Zhiyang Dou, Chen Zong, Shiqing Xin, Mingyan Jiang, Tao Ju, and Changhe Tu. RFEPS : Reconstructing feature-line equipped polygonal surface. *ACM Transactions on Graphics*, 41(6), nov 2022.

[YL22] Mulin Yu and Florent Lafarge. Finding good configurations of planar primitives in unorganized point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6367–6376, June 2022.

[YLF+18a] 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 386–402, 2018.

[YLF+18b] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net : Point cloud upsampling network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018.

[YRP18] Sunil Kumar Yadav, Ulrich Reitebuch, and Konrad Polthier. Robust and high fidelity mesh denoising. *IEEE Transactions on Visualization and Computer Graphics*, 25(6) :2304–2310, 2018.

[ZAB+21] Tong Zhao, Pierre Alliez, Tamy Boubekeur, Laurent Busé, and Jean-Marc Thiery. Progressive discrete domains for implicit surface reconstruction. *Computer Graphics Forum*, 40(5) :143–156, 2021.

[ZBCS+23] Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, and Pierre Alliez. Variational shape reconstruction via quadric error metrics. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–8, 2023.

[ZCHK12] Henrik Zimmer, Marcel Campen, Ralf Herkrath, and Leif Kobbelt. Variational tangent plane intersection for planar polygonal meshing. In *Advances in Architectural Geometry*, Paris, 2012. Springer.

[ZDZ+15] Wangyu Zhang, Bailin Deng, Juyong Zhang, Sofien Bouaziz, and Ligang Liu. Guided mesh normal filtering. *Computer Graphics Forum*, 34(7) :23–34, 2015.

[ZFAT10] Youyi Zheng, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. Bilateral normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics*, 17(10) :1521–1530, 2010.

[ZJ16] Qingnan Zhou and Alec Jacobson. Thingi10K : A dataset of 10,000 3d-printing models. *arXiv preprint arXiv :1605.04797*, 2016.

[ZLZ+19] Wenbo Zhao, Xianming Liu, Yongsen Zhao, Xiaopeng Fan, and Debin Zhao. NormalNet : Learning based guided normal filtering for mesh denoising. *CoRR*, abs/1903.04015, 2019.

[ZT18] Yin Zhou and Oncel Tuzel. VoxelNet : End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[ZYAL23] Tong Zhao, Mulin Yu, Pierre Alliez, and Florent Lafarge. Sharp feature consolidation from raw 3d point clouds via displacement learning. *Computer Aided Geometric Design*, 103 :102204, 2023.

Liste des figures

Liste des tableaux