Algoritmos Para el Pre Procesamiento de Nubes de Puntos Mediante Representaciones Dispersas

Esmeide Alberto Leal Narváez

Supervised by PhD. John William Branch Bedoya

Co-supervised by PhD. German Sánchez Torres

Ingeniería de Sistemas e Informática

Facultad de Minas

Universidad Nacional de Colombia - Sede Medellín

April, 2020

A dissertation submitted in partial fulfilment of the requirements for the degree of Ph.D. Doctor en Ingeniería - Sistemas e Informática.

$Mk^{-3}\ k_i$

Hoy en día, los escáneres 3D se han convertido en una fuente estándar que proporciona millones de puntos como entrada para un número creciente de áreas de aplicaciones como: industria, entretenimiento, medicina, fotogrametría, visión por computadora etc. La gran cantidad de puntos que se generan a partir del proceso de escaneo se denomina nube de puntos y, a menudo, se vuelve complicado de manejar debido a múltiples problemas, como el ruido producido por el proceso de escaneo, la falta de información (agujeros) y el exceso de información (puntos), también es importante en algunas aplicaciones detecta características sobresalientes, como bordes y valles. Todos estos problemas se abordan en la etapa de la reconstrucción de superficies llamada preprocesamiento de la nube de puntos.

El objetivo de esta tesis es el uso de representaciones para desarrollar algoritmos computacionales robustos para resolver los problemas presentes en el preprocesamiento de nubes de puntos.Las representaciones dispersas son métodos inspirados in el sistema de visión humano, los cuales pueden ser adaptados a las características de los problemas encontrados en el pre procesamiento de nubes de puntos.

Presentamos contribuciones sobre algunos temas fundamentales como, la eliminación de ruido, la extracción de características finas, la simplificación de puntos y la detección de huecos. Al usar directamente los puntos 3D, evitamos la necesidad de métodos de reconstrucción de superficie, que son complejos y requieren mucho tiempo de procesamiento.

En esta tesis, se introduce un método de suavizado el cual es efectivo para eliminar el ruido y preservar las bordes y esquinas en nubes de puntos. La capacidad de preservación de características proviene de la combinación de la mediana L1 y la norma L1 para estimar las normales y la actualización de las posiciones de los puntos.

Para reducir la complejidad del muestreo de la nube, presentamos un método de simplificación basado en saliencia. Para ello, se lleva a cabo un proceso de aprendizaje de diccionario y codificación dispersa sobre las normales y las curvaturas para encontrar las saliencias. A continuación, se realiza una selección de los vectores dispersos que representan las características con mas saliencia, llevando a cabo de esta manera la simplificación.

Introducimos un método para detectar bordes y huecos en nubes de puntos. Primero, construimos una matriz de covarianza a partir de la información geométrica en un vecindario alrededor de cada punto en la nube. Luego se estiman los valores propios de la matriz de covarianza, y se combinan para formar vectores de características, que se utilizan como señales para llevar a cabo un aprendizaje de diccionario seguido de

un proceso de codificación disperso. Finalmente, al imponer un umbral sobre los coeficientes dispersos, detectamos las características (bordes, esquinas, valles) y agujeros. Demostramos la utilidad de todos nuestros algoritmos en una amplia variedad de modelos geométricos escaneados de diferentes tamaños, complejidad y detalles.

Palabras Claves— Nubes de puntos, escaneo 3D, Suavizado, Ruido, Detección de huecos, Simplificación de puntos, Representaciones Dispersas, Aprendizaje en diccionarios, Codificación dispersa.

Sparse Representation-Based Algorithms for PreProcessing Point Clouds

Esmeide Alberto Leal Narváez

Supervised by PhD. John William Branch Bedoya

Co-supervised by PhD. German Sánchez Torres

Ingeniería de Sistemas e Informática

Facultad de Minas

Universidad Nacional de Colombia - Sede Medellín

April, 2020

A dissertation submitted in partial fulfillment of the requirements for the degree of Ph.D. Doctor en Ingeniería - Sistemas e Informática.

WWW.UNAL.EDU.CO

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN

Declaration by Postgraduate Students

Authenticity of Dissertation

I hereby declare that I am the legitimate author of this Dissertation and that it is my original work.
No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.
I hold the National University harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

Faculty/Institute/Centre/School	Facultad de Minas
Degree	Ph.D. Doctor en Ingeniería - Sistemas e Informática
Title	Sparse Representation-Based Algorithms for PreProcessing Point Clouds
Candidate	Esmeide Alberto Leal Narváez

Signature of Student	_____
Date	August 22, 2020

To My wife Nohemi, My children Matias and Maria, My parents, My brothers

Acknowledgements

I immensely thank God and Jesus for giving me the wisdom and the fortress to complete this thesis.

I would like to thank my advisor, Professor John William Branch, for this opportunity to complete my PhD studies. For his advice, support, enthusiasm, friendship, and patience in all these years. Many thanks also to my co-advisor, Professor German Sanchez, for discussions, recommendations, and interest in this work.

My special thanks go to Professor Francisco Jose Abad Cerdá, for accepting me as an intern in the DSIC at Universidad Politécnica de Valencia España, very grateful for his help, support, and discussion in part of this investigation.

I also want to thank the Administrative Department of Science, Technology, and Innovation of Colombia—COLCIENCIAS, for supporting me under the program of doctoral scholarships 727-2015.

My sincere thanks go to my brother Nallig, for so many discussion on several issues in this thesis, without your help, it would have been hard to understand.

To my mother and father for their guidance and love, and for encouraging me to pursue my goals.To my brothers, I thank you for your trust and support during the tour of this hard but wonderful experience.

Finally, this thesis would not be possible without the patience and support of my beloved wife and children who accompanied me during these four years.

Abstract

Nowadays, 3D scanners have become a standard source that provides millions of points as input for a growing number of applications areas such as industry, entertainment, medicine, computer vision, photogrammetry, etc. The large number of points generated by the scanning device is called point clouds. Point clouds often become complicated to handle due to multiple problems, such as the noise produced in the scanning process, lack of information (holes), or excess of information (points). Also, it is crucial in some applications to detect sharp features, like edges and valleys. We addressed all these problems in a stage of surface reconstruction called point cloud pre-processing.

The focus of this thesis is the use of sparse representations for developing robust computational algorithms to solve the problems included in the pre-processing stage. Sparse representations are methods inspired in the human visual system, which can be adapted to the characteristics of problems found in the pre-processing of point clouds.

We present contributions on some fundamental topics as denoising, sharp features extraction, hole detection, and simplification. With the direct use of the 3D points, we avoid the need for surface reconstruction methods, which are computationally complex and time-consuming.

In this thesis, we introduce a smoothing method, which is effective in removing noise and preserving the sharp features and corners. The features preserving capability comes from the combining L1 median and L1 norm to estimate the normals and the point positions update.

To reduce the sampling complexity of the cloud, we present a simplification method based on saliency. A Dictionary learning and sparse coding process are carried out over the normals and curvatures to find the saliencies. Next, it makes a selection of the sparse coefficients that represent the most salient features, carrying out in this way the simplification.

We introduce a method for detecting features and holes in point clouds. First, we build a covariance matrix from the geometric information in a neighborhood around each point in the cloud. Then we estimate the eigenvalues of the covariance matrix, and combining them, we build feature vectors. The feature vectors are the signals to carry out a dictionary learning followed by a sparse coding process. At last, Imposing a threshold over the sparse coefficients, we detect features (edges, corners, valleys) and holes. We show the effectivity of our algorithms in a wide range of scanned geometric models of varying sizes, complexity, and details.

Keywords— Point Clouds, 3D scanning, Smoothing, Noise, Holes detection, Points Simplification, Sharp Features, Sparse Representations, Dictionary Learning, Sparse Coding.

Contents

List of Figures

List of Tables

1

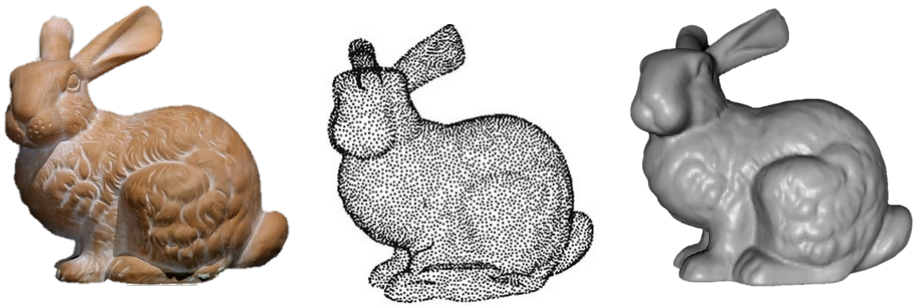
Introduction

With the growing and development in 3D acquisition devices technology, i.e., scanners, tomography, RMI, ultrasound, and LiDAR scanning, the digitization of real objects has become a robust and efficient task that allows the recovering of the detailed geometry of complex objects, getting precise computational representations. Many of the acquisition process applications are considered a Reverse Engineering process. It has been successfully applied in different scientific fields and industries with different purposes, such as medicine, entertainment, visual inspection, cultural heritage, scientific visualization, etc. The output of the scanning process is a dense point cloud. Some 3D devices can generate up to over 40.000 points per second (BSKH09). In this thesis, we considered the problem of processing datasets, which result from the acquisition through 3D scanning devices. We focus on datasets, which are represented as unstructured point clouds. Unstructured point clouds mean that no connectivity information can be assumed from the underlying topology between the points samples. In this thesis, we use synthetic datasets for a quantitative comparison of the proposed algorithms compared with the state-of-the-art. Figure 1.1 shows the points generated by a 3D scanning process from a 3D real object and its corresponding mesh triangulation from the point cloud.

1.1 | Surface Reconstruction and Reverse Engineering

The aim of surface reconstruction is determining a mathematical surface S', that represent an unknown real surface S, so that $S' \approx S$, with $S \in \mathbb{R}^3$, through a set $P = \{\mathbf{p}_i \mid i \in [1, n]\}$, with $\mathbf{p}_i \in \mathbb{R}^3$, where P is the observation set, i.e., the point clouds (BTS+14).

Reverse engineering is the process of converting a 3D point cloud into a model bounded by a surface, often represented in a CAD model (BMV01). The aim of reverse



(a) (b) (c)

Figure 1.1: Stanford Bunny, (a) 3D Real object, (b) Point Cloud, (c) Mesh renderization

engineering is to obtain a model from a physical object, which can be manipulated and modified. That is why the surface reconstruction is a reverse engineering process, which comprises the following stages:

- Data acquisition: using a 3D capture device.
- Registration: fusion of different views from an object into the same coordinate system.
- Pre-processing: noise removal, filtering, boundary extraction, hole filling, simplification.
- Segmentation: divide the surface into segments for a criterion.
- Surface fitting: triangulation, Radial Basis Functions, NURBS surfaces.

Each of these stages is a research field on its own. The data acquisition is carried out through the scanning of a physical object. The scanner must be calibrated first and if it is necessary to prepare the object, i.e., if the object surface is shiny, it must be painted with a matt color to decrease the reflectance distortion effect. Because the scanner cannot scan the entire object surface in a single pass, different object views must be captured. All the partial views are joined in the same reference system, to get the point cloud; this stage is named Registration. The pre-processing step begins with the filtering and noise reduction of the point cloud, and if it is necessary, edges and boundaries extraction are performed. If the point cloud contains holes, this stage fills them. Depending on the application and if the point cloud is very dense, simplification is necessary. The segmentation stage tries to find regions of the point cloud, which can be adjusted with a single patch. This stage is applied when the cloud belongs to a CAD model or a regular

geometrical model. In the surface fitting stage, a continuous surface is fitted to the entire cloud. Different surface approximations methods as Triangular Meshes, Non-Uniform Rational B-Spline (NURBS), Radial Basis Function (RBF), etc., are used to reconstruct the surface. Figure 1.2 shows the pipeline of surface reconstruction.

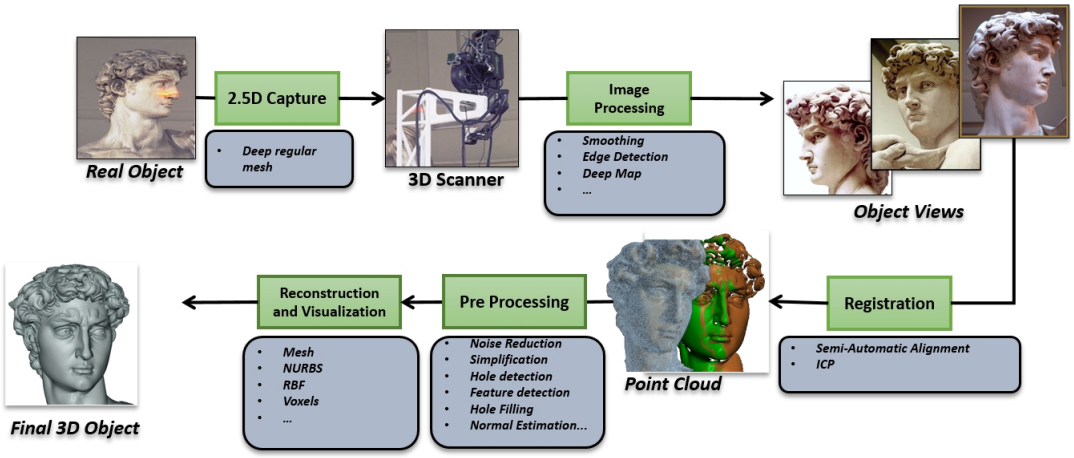


Figure 1.2: Surface Reconstruction Pipeline

The algorithms presented in this thesis addresses some problems of the preprocessing stage. In the next section, we will describe these problems.

1.2 | Research Problem

In the 3D scanning process, there are shortcomings; some of them inherently related to the acquisition devices while others relates to human operator handling. Some typical device-dependent problems include the noise added to the point cloud, the dependency on the lighting condition, (for light-based scanners), holes in the dataset (because of the occlusion of the line-of-vision), inaccessibility of the surface in complex shapes and spot shining found in specular surfaces.

For a wide range of applications, the large number of points generated by the 3D scanners can be a problem, e.g., if there are limitations in memory (mobile devices), bandwidth transmission, and storage access. Also, in the 3D surface reconstruction, it is crucial to detect sharp features (edges, corners, valleys) for segmenting and approximating 3D shapes. But identifying sharp features in point clouds is a challenging task,

even more in noisy data sets, since it is necessary to distinguish between features and noise in the data.

Point sampling is a common issue related to the 3D scanning device handling. Because of the continuous distance fluctuations between the scanner and the scanned surface, the point sampling spacing can be variable, producing point cloud with irregular density and different average sampling spacings, making it difficult for the dataset handling. The above problems are tackled by different techniques included in the point clouds pre-processing stage, and its aim is improving the quality of the raw data within the surface reconstruction process.

In summary, some of the problems related to the point cloud pre-processing are noise in data, holes in data, data simplification, and sharp features detection. The mathematical and statistical tools involved in the point cloud pre-processing belong to the discipline called Digital Geometry Processing, which uses mathematical models and algorithms for analyzing and manipulating geometric data.

Considering the above, we can conclude that point cloud pre-processing is a decisive step (BBK$^+$09) if we want to get a high-quality digital representation of the scanned physical objects. The high number of researches on this topic, confirm the pre-processing stage relevance for the 3D surface reconstruction process and its applications.

1.3 | Problem Description

In this section, we describe the problems related with point cloud pre-processing stage.

1.3.1 | Noise Reduction

Point cloud filtering is a fundamental step in the surface reconstruction process; the goal of the filtering is to remove or reduce the effect of noise while preserving the most relevant features of the model. Formally we can express the measurement model for the noise reduction problem as:

$$\mathbf{p'}_i = \mathbf{p}_i + \varepsilon(\mathbf{p}_i), \quad for \;\; i = 1, 2, .., n.$$

Where $\mathbf{p}_i$, is the underlying but unknown point of interest at the position $\mathbf{p}_i = \{x_i, y_i, z_i\}$, $\mathbf{p'}_i$ is the observation or noisy measure, and $\varepsilon(\mathbf{p}_i)$ is the noise distribution with zero mean and variance σ^2. Usually, the normal distribution is assumed for noise.

The problem of interest is to retrieve the complete set of samples $\mathbf{p}_i$, denoted by $P = \{i \in [1, n]\}$, from the corresponding set of observations $\mathbf{p'}_i$, distinguishing between noise and fine structures that are part of the original 3D object.

The point cloud filtering is regarded as a similar problem to image filtering since techniques applied in the filtering of point clouds have inspiration in the image processing field, therefore the most recent researches in this field, are also valid in point cloud filtering (WYP+15) (LDC16) (LCS17). It has been proven that the effective restoration of signals (images, point clouds) (DTB06)(PBL10)(RG14), will require methods that model the signal using prior knowledge or learn the underlying characteristics of the signal from given data. The above involves the use of learning methods, non-parametric methods, or Bayesian methods for effective signal restoration.

Some of the most remarkable methods for signal filtering are those based on patches (WRK12)(RDK13). This new generation of algorithms exploits both local and non-local redundancy or self-similarity in the images or point cloud being treated. The Bilateral Filter (TM98) was developed with the same idea in mind, as were other filters similar such as the Susan filter, the normalized convolution, and Yaroslavsky filters (YBS06). The common goal of these algorithms is to measure and use the affinities between patches or regions of interest. Within patch-based methods, we can also include sparse representations, as dictionary learning (XZZ+14).

Recent developments in point cloud filtering come from different areas, resulting in a great diversity of algorithms for multidimensional data processing. The methods developed include: Moving Least Squares (MLS) (ABCO+03)(from computer graphics) Bilateral Filter (BF) (FDCO03)(JDD03) and Anisotropic diffusion (CDR00) (from computer vision), core-based methods and spectral methods (from machine learning) (PG01) (SSC+18), sparse representations (OF96) (VG00) (XZZ+14)(from neurosciences and machine learning), Non-Local Mean (NLM) (DG10) (GAB12) and its variants (from signal processing), Bregman Iterations (from applied mathematics) (SKTJ14), regression kernel and Iterative Scaling (from statistics) (SBS05) (ZWN17). While these approaches have different origins, they are deeply connected.

The state-of-the-art on noise filtering shows different methods, among which are the Isotropic. The isotropic methods, filter the noise independent of the surface geometry, within this category are the Laplacian smoothing methods and its variants (Tau95) (VMM99) (HS13). These methods do not preserve the sharp features of the point cloud, also produce data shrinkage, since they use averages and are not robust to outliers.

There are also Anisotropic methods; these methods consider the surface geometry. Examples of anisotropic methods are the adaptations of Bilateral Filter (ZFAT11) (WYP+15) (ZDZ+15). These methods preserve sharp features such as corners and edges,

but if the point cloud is very noisy, these methods can fail because there is ambiguity treating the noise as sharp features.

Interpolation-based methods such as MLS and RBF are effective smoothing and eliminating high levels of noise, but they are not robust to outliers and fail when point sampling density is low. Sparse-based methods are robust to noise and outliers, since these approaches use the L_0 or L_1 norm, but can produce (WYL+14) (MC17) false features and flatten the geometric textures.

Probabilistic methods, which are supported by Bayesian theory (DTB06) (JWB+06), are also robust to outliers, but because of the use of the L_2 norm, they can not preserve the sharp features and always assume Gaussian noise.

Data-Driven Methods (WLT16) are robust to noise and outliers. They have proved to be a suitable alternative in the state-of-the-art. Still, their drawback is, the extensive amount of data necessary for training, since they use Neural Networks. Other disadvantages are the training time, which consumes computational resources, and the adaptation to point clouds that do not have a regular distribution as an image that has its data arranged in a grid.

The methods based on sparse coding and dictionary learning (XZZ+14) are robust to outliers because they are based on the L_0 and L_1 norm, but because of the dictionary learning process, the problem of flattening surfaces and fine features could be overcome. They have the disadvantage that must be trained, but unlike the neural networks, the adaptation to point clouds is easier, and the training is less heavy.

1.3.2 | Simplification

The problem of approximate a point cloud with a less dense but geometrically faithful representation is fundamental in areas such as computer graphics and geometric processing. Given the visual complexity required to create realistic scenes, simplification efforts can be essential for efficient and accurate representation. The considerable demand (LWC+02) to develop practical algorithms for simplifying point clouds has become necessary. A significant number of algorithms to simplify point clouds have been proposed, and we can say that they differ in approach, efficiency, quality, and generality. Some techniques offer efficient processing but produce a simplified point cloud that is visually undesirable. Others create more nice approaches, but they are slow and challenging to implement. Some algorithms try to preserve the original topology of the point cloud while others alter it arbitrarily.

Point clouds can contain millions of points, such as David's model, in the Miguel Angel Digital project, in which the model already generated as a triangular mesh contains

2000 million triangles (LPC+00). These highly complex mesh models are a challenge in storage, processing, transmission, and rendering in real-time. However, the complexity of these models is not always necessary since the computational cost of using a model is directly related to its complexity. It is useful to have simplified versions of such complex models. Therefore, it is necessary to simplify the models maintaining the final appearance of the object as much as possible (LWC+02).

Regarding the simplification methods, in (PGK02) (SFC10), several techniques for the simplification of the point clouds are presented. The methods used are based on hierarchical and incremental grouping, particle simulation, interactive simplification, and bilateral filtering. These methods use the points without meshing and use the local surface variation and normals to guide the simplification process. In (XHC+18) (HHSH15), they first detect edges to guide the simplification process and in this way preserve the sharp features in the resulting point cloud; this is a good strategy, but it does not guarantee that the original surface structure remains in the simplified cloud. In (LLB09), it uses a genetic algorithm for simplification, producing a proper simplification result. Still, the genetic algorithm technique is time-consuming, and it can lose some fine details.

1.3.3 | Features and Hole Detection

A significant number of methods have been developed for features and holes detection (BPB06) (SB10) (WHH10) (NTT15) (BSK06). The proposed methods detect feature points that belong to edges and corners of the 3D models, using the local information of the surface.

Feature detection methods applied to point clouds are less common compared to triangular meshes detection methods. The drawback of these point-based methods is the lack of connectivity information between the points. Feature detection methods applied to point clouds are more challenging tasks than mesh-based methods.

Different approaches have been proposed for feature extraction at 3D points. Between the approaches are the graph-based methods (GWM01) (PKG03); the connectivity information for the point cloud is created using graphs, to a local approximation of the surface. These methods can distinguish between corners, edges, and line-type features. The graph-based methods use Principal Components Analysis (PCA) to find sharp features, but it is known that the classical PCA is not robust to noise. A multi-scale scheme must be applied to the neighborhood, doing it more robust to noise, but with a high time-consuming computation.

Another approach is based in region growing (DVV+06), different regions are intersected in sharp features; the region growing process is guided by the normal variation.

The problem facing this method is that an accurate normal estimate has to be made to separate regions, in the presence of noise.

With concern to hole detection, some methods use the angle criterion. The angle criterion considers a k-*nearest* neighborhood for each point $\mathbf{p}_i$ in the cloud, and then the neighbors are projected into the tangent plane of $\mathbf{p}_i$. Next, the projected samples are sorted radially, and the angles between consecutive samples are computed. The maximum of these gaps is used to determine the probability of $\mathbf{p}_i$ being a point on a boundary. These methods can fail if the point cloud is noisy due to the imprecision in the plane estimation.

The methods for hole detection based on the computation of convex hull have been applied to detect the boundary of a surface of 3D points set (SS07) (XXFJ08). These methods have a high computational cost because they must estimate the convex hull for each point in the point cloud.

1.4 | The Thesis Objectives

1.4.1 | General

To propose a methodology for pre-processing 3D point clouds based on sparse representations to address the problems of noise reduction, holes and sharp features detection, and simplification.

1.4.2 | Specific

1. To propose a method for eliminating noise through adaptive representations obtained from sparse models while maintaining the local and geometric features of the point cloud.

2. To propose a method for detecting holes and sharp features (edges, valleys) in point clouds, based on sparse models, using the geometric and spatial information given by the points.

3. To propose a point cloud simplification method based on sparse models to reduce the information according to the local surface geometry.

4. To analyze the precision of the proposed methods based on what is reported in the state-of-the-art.

1.5 | Goal And Overview Of This Thesis

The overall goal of this thesis is to improve some tasks in the point clouds pre-processing stage, by combining sparse mathematical models, concepts of visual perception and numerical optimization, to detect and preserve the most relevant features of the point cloud in each one of tackled tasks, and in this way improve the visual representation of the objects. In this thesis, we address the problems of noise reduction while preserving sharp features, point cloud simplification, keeping the most relevant features of the original, and finally detecting sharp features and holes in the point cloud even in the presence of noise. We establish a methodology oriented to identify the sparse features in the point cloud, applying sparse representation mathematical models and analyzing the problems involved in this research from the same point of view, unifying them in a single framework, which is the sparsity modeling.

We are not the first to explore the idea of applying sparse modeling in geometric processing, which has already yielded several successful methods in recent years (see, e.g., (XWZ+15)). Still, there is a long way to go, and many unsolved problems in 3D geometry processing that can benefit from this point of view. Throughout this thesis, we are moving in that direction, presenting solutions to the issues involved in the point cloud pre-processing stage.

The remainder of this thesis is structured as follows:

Chapter 2 gives an overview of concepts related to the research fields. It has focused on point clouds, k-nearest and r-neighborhoods, Principal Component Analysis, surface variation, and sampling conditions.

Chapter 3 we present the mathematical and statistical tools using in this research. It has focused on inverse problems from the Bayesian point of view and the relation with the sparse representations and Minimum Description Length principle; we introduce the L1-median filter and the L1 norm as regularization factor and the optimization algorithm for L1 norm.

Chapter 4 a feature-preserving and robust point cloud denoising algorithm is presented. We combine the L1-median filter as the fitting term and the L1 norm as the regularization term. The surface denoising algorithm is applied to every point in the input model, resulting in a simple and effective algorithm for point cloud denoising.

Chapter 5 deals with point clouds saliency detection, presenting an approach to saliency detection on point clouds from a local perspective. Using the sparse coding and the

Minimum Description Length principle; we relate the Bayesian theory with information theory, to state a criterion for finding saliencies in point clouds.

Chapter 6 we address the simplification problem, presenting a global approach for saliency detection in point clouds. We use dictionary learning and sparse coding techniques to detect the saliencies. Next, the saliency is used as a starting point for the simplification process. In this way, the saliencies guide the simplification step preserving the most representative features in the reduced point cloud.

Chapter 7 is devoted to sharp features extraction and hole detection in the presence of noise, presenting a multi-scale dictionary learning algorithm. We propose a global multi-scale edge analysis identifying which features are more relevant in the point cloud to improve the edge and hole detection.

Chapter 8 summarizes the thesis and concludes with some directions for future research.

1.6 | Contributions And Academic Products

This thesis presents contributions to the point cloud pre-processing stage focused on denoising, simplification, and feature extraction. These goals have been achieved by the combination of the biologically inspired tools as sparse representations and visual perception and mathematical optimizations algorithms.

- We propose the combination of the L1-median filter with L1-norm to reach robust normal estimation that helps to keep sharp features in the object geometry. Use the L1-median filter made the algorithm robust to outliers and high levels of noise.
- We propose an adaptive weight strategy, which gives preference to points located on the same side of a surface when they are near a discontinuity, helping in a more precise estimation of the tangent plane and the normal vector near the sharp features.
- We related the concept of the Minimum Description Length (MDL) principle with the Bayesian formulation to establish the level of saliency in a point cloud.
- We use the level of sparsity of each input signal as clusterization radius for point cloud simplification.

- We use a dictionary learning algorithm to extract the most relevant sparse features from a point cloud and, in this way, detect holes and sharp features.

Parts of this thesis are based in the following publications:

- Leal, E.; Sanchez-Torres, G.; Branch, J.W. “Sharp Feature Detection on Point Sets via Dictionary Learning and Sparse Coding”, Conference: the 15th International Joint Conference on Computer Graphics Theory and Applications. GRAPP, Prague, Czech Republic, February 25-27, 2019.
- Leal, E.; Sanchez-Torres, G.; Branch, J.W. “Point Cloud Saliency Detection Via Local Sparse Coding”. Journal Dyna rev. fac. nac. minas vol.86 no.209 Medellín Apr./June 2019. DOI: 10.15446/dyna.v86n209.75958.
- Leal, E.; Sanchez-Torres, G.; Branch, J.W. “Sparse Regularization-Based approach for Point Cloud Denoising and Sharp Feature Enhance”. Sensors 2020, Volume 20 Issue 11 , 3206.DOI:10.3390/s20113206.
- Leal, E.; Sanchez-Torres, G.; Branch, J.W.; Abad, F.; Leal, N. “A Saliency Based Sparse Representation Method for Point Cloud Simplification”. Submitted IEEE Access.

During the course of this thesis, I have contributed to the following work, which is not included in this thesis:

- Leal, N.; Zurek, E.; Leal, E. "Non-Local SVD Denoising of MRI Based on Sparse Representations". Sensors 2020, Volume 20 Issue 5, 1536.DOI:10.3390/s20051536.

2

Point Cloud Fundamentals

Point clouds play the central role in this research since they are the primary data input for the algorithms presented in this thesis. In this chapter, we give some fundamentals concepts and algorithms for point clouds characterization used by the methods presented in the subsequence chapters of this thesis. We present definitions for local surface estimation and its properties like normal vectors, surface variation, tangent planes, eigenvalues, and eigenvectors.

2.1 | Point Clouds

A point cloud is an unordered collection of n points $P = \{\mathbf{p_i} \in [1, n]\}$, with $\mathbf{p}_i \in \mathbb{R}^3$, defined by a coordinate system $\mathbf{p}_i = \{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$, that represent the geometry of an object that has been scanned.

Typically, the point clouds are generated by a 3D scanner device but can be created too by software, ultrasound, stereoscopy, LiDAR, etc. (ZRP02) (GW11) (KDK$^+$14). Triangular meshes are constructed from point clouds, with meshes being the most common presentations used in various fields of application (CAK12) (ACK13).

2.2 | Local Neighborhoods

In a discrete term, a local neighborhood is defined through spatial relations between the sampled points. Given a point $\mathbf{p} \in P$, a neighborhood is defined as a set of indexes $N_g(\mathbf{p})$, such that each $\mathbf{p}_i$, $i \in N_g(\mathbf{p})$ satisfice specific neighborhood metric. This condition should be set in such a way that the points of $N_g(\mathbf{p})$ appropriately represent a small, local surface patch around the point $\mathbf{p}$. In this research, we use two approaches

for defining local neighborhoods of points. The first approach is based on the definition of k-*nearest* neighbors. The second approach is based on *r-neighborhood*.

2.3 | K-nearest Neighbors (K-NN)

By *k-nearest* neighbors $N_g^k(\mathbf{p})$, we consider the ordered set, which is sorted by increasing distance, of all points in P according to their distance to the point $\mathbf{p}$. As follow $\{|\mathbf{p}-\mathbf{p}_1| < |\mathbf{p}-\mathbf{p}_2| < \cdots < |\mathbf{p}-\mathbf{p}_k|\}$. From this set, the first k neighbor points are selected as $N_g^k(\mathbf{p})$. The set of k-*nearest* neighbors is then:

$$N_g^k(\mathbf{p}) = \{k - nearest\ neighbors\ of\ \mathbf{p}\ in\ the\ set\ data\ P\} \tag{2.1}$$

2.4 | r- Neighborhood

The r-*neighborhood* is the set $N_g^r(\mathbf{p})$ that contain all points, within a sphere centered at $\mathbf{p}$ with radius r, the number of points contained in $N_g^r(\mathbf{p})$ are variable.

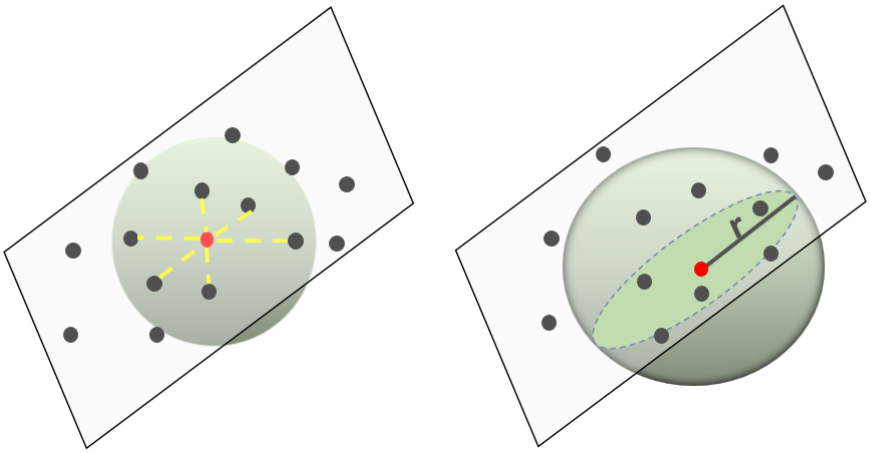
$$N_g^r(\mathbf{p}) = \{\mathbf{p}_i \mid \|\mathbf{p}-\mathbf{p}_i\| \leqslant r\} \tag{2.2}$$

where $\|\cdot\|$, denotes the Euclidean norm.

Usually, the two definitions produce the same result when the data point are evenly distributed. Still, the r-*neighborhood* strategy perform better than k-NN, since the r-*neighborhood* approach has a better performance when the points or samples are not uniformly distributed. Because the r-neighborhood depends only on the radius of the disk irrespective of point cloud resolution unlike the k-neighborhood. Figure 2.1 shows two types of neighborhoods.

2.5 | Principal Component Analysis and Normal Estimation

Based on the above definition of neighborhood relationships, the local surface properties at a point $\mathbf{p} \in P$ can be estimated using eigenanalysis over the neighborhood of $\mathbf{p}$. The statistical tool to carry out the eigenanalysis is the Principal Component Analysis (PCA). The PCA is a powerful method to extract principal directions of data and commonly used for dimensionality reduction (Jol86). Eigenanalysis over the neighborhood using the PCA produces a covariance matrix, which is decomposed on its eigenvectors



(a) (b)

Figure 2.1: Neighborhood on point clouds, (a) K-nearest neighbors, (b) r-neighbors

and eigenvalues. Eigenvector and eigenvalues are used to estimating normal vectors, tangent planes, surface variation, and other properties in the neighborhood.

To estimate the PCA, one computes the centroid $\bar{\mathbf{p}} \in \mathbb{R}^3$ of the $N_g(\mathbf{p})$:

$$\bar{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i, \;\; with \;\; n = \left|N_g(\mathbf{p})\right| \tag{2.3}$$

The 3x3 Covariance Matrix Cm of $N_g(\mathbf{p})$ is estimated as:

$$Cm = \frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{p}_i - \bar{\mathbf{p}}\right)\left(\mathbf{p}_i - \bar{\mathbf{p}}\right)^T, \;\; \mathbf{p}_i \in N_g(\mathbf{p}) \tag{2.4}$$

Since Cm is symmetric and positive semi-definite, all the eigenvalues λ_i are real-valued. The eigenvectors v_i form an orthonormal base, which shows the variation of the greatest and least variance along the direction of the corresponding eigenvectors.

The eigenvalues satisfy the following relationship $\lambda_0 \leqslant \lambda_1 \leqslant \lambda_2$. The eigenvectors, $\mathbf{v}_1$ and $\mathbf{v}_2$ expand a tangent plane at the point $\mathbf{p}_i$, and the eigenvector $\mathbf{v}_0$ approximates the normal $\mathbf{n}_p$ in $\mathbf{p}_i$. The tangent plane is the set of points $\mathbf{p}_i \in \mathbb{R}^3$ such that $\mathbf{n}_p^T\left(\mathbf{p}_i - \bar{\mathbf{p}}\right) = 0$. The normal $\mathbf{n}_p$ is unoriented because it only shows the direction of the smallest variance of points in the neighborhood. In consequence, the normal can point outwards or inwards. Compute a consistently oriented normal field on a point cloud is mandatory for surface reconstruction. Figure 2.2 illustrates the 2D geometric interpretation of the covariance matrix Cm.

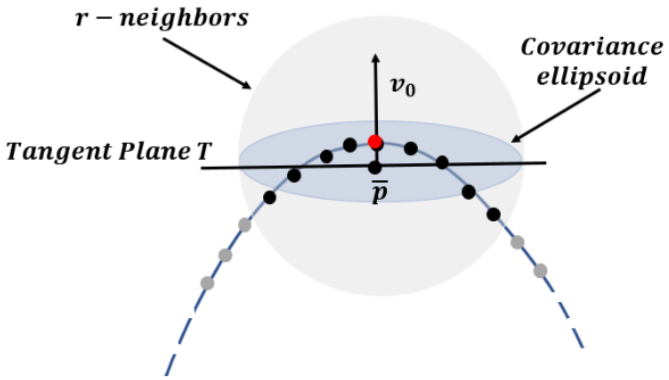


Figure 2.2: Normal and tangent plane estimation using PCA, adapted from Pauly (PGK02).

2.6 | Normal Orientation

A consistent orientation of the normal vectors is required for some applications. If we know the scanner position, then the normal can be oriented regarding its position. We compute a consistent normal field orientation via Minimum Spanning Tree (MST) of the point cloud, as described in (HDD+92). The edge weights in the MST are then computed from the distances between adjacent points and the deviations of their corresponding normal directions. The normal vector of each adjacent point in the MST can then be oriented based on the assumption that the angle of the normal vectors of adjacent points is almost equal (Figure 2.3).

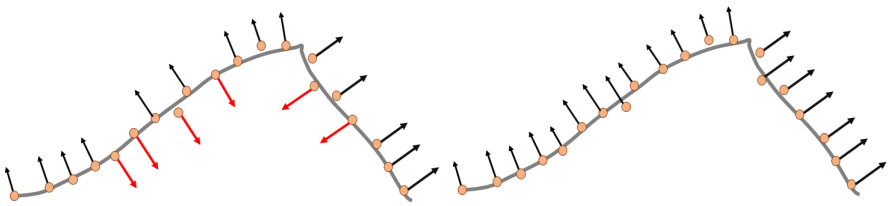


Figure 2.3: Normal orientation using MST, (a) Unoriented normals, (b) the normal has been flipped base on the assumption that the angle between normal vectors is almost equal.

2.7 | Surface Variation

The surface variation at point $\mathbf{p}$ in a neighborhood is defined as

$$\sigma(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{2.5}$$

Where λ_0, is a measure of the variation along the normal direction, i.e., describe how far the points deviate from the tangent plane. If the surface is locally flat, then $\sigma(\mathbf{p}) = 0$, but if the surface point is isotropic the variation $\sigma(\mathbf{p}) = 1/3$. Surface variation is closely related to the mean curvature (PGK02).

2.8 | Edge Points and Corner Points

When a set of points are on a surface discontinuity, it can be observed that λ_0 will be large compared to λ_2. The quotient between the eigenvalues λ_0 and λ_2, can be used as a measure of the probability that the point $\mathbf{p}$ will fall near a discontinuity on the surface S. That is

$$\frac{\lambda_0}{\lambda_2} > \varepsilon \tag{2.6}$$

where ε is a threshold. We say that $\mathbf{p}$ is an edge point. On the other hand, if the quotient

$$\frac{\lambda_2 - \lambda_0}{\lambda_2} > \delta \tag{2.7}$$

where δ is a threshold, we say that $\mathbf{p}$ is a corner point.

In a border point, the eigenvalues λ_1 and λ_2 obey $2\lambda_1 \approx \lambda_2$ and we can establish the following criterion, to identify the point $\mathbf{p}_i$ as border.

$$bp(\mathbf{p}_i) = |\lambda_2 - 2\lambda_1| / \lambda_2 \tag{2.8}$$

2.9 | Sampling Requirements

All the methods presented above, are based on the supposition that the k-nearest neighborhood of point $\mathbf{p}$, represent a small patch of the underlying surface S around $\mathbf{p}$. The intersection of the ball that encloses the k-*nearest* neighborhood with the surface S should be a disk. The intersection must be sufficiently flat to ensure the accurate normal estimation (see Figure 2.2), because we are approximating the surface S using a regression

plane to the k-*nearest* neighborhood of **p**. It means that the enclosing ball should not be too large in surface regions with high curvature (PGK02).

3

Mathematical Background

This chapter presents the fundamental theories and concepts in which are based the algorithms in this thesis. We describe mathematical and statistical tools used for processing the point clouds and the optimization techniques to minimize the proposed algorithms. We present the concepts of inverse problems and the Bayesian theory, the sparsity visual theory, and its mathematical representation as sparse vectors. The above tools are the core of the methods presented in the rest of the chapters.

3.1 | Inverse Problems

Every technological and information system often works with input signals to understand or process relevant information. During capturing data through a sensor, there is some degradation of the signal being measured. This degradation is because of technological limitations, electronic noise, defective sensors, and optical distortions; these are all typical factors that reduce the signal quality. Therefore, any measurement or capture of information that is done, such as recording audio, taking photographs, medical examinations (ultrasound, Computerized Axial Tomography, Magnetic Resonance Imaging), 3D scanning of objects, among others, often needs to be restored and improved. Recover the original data from its degraded measurement is named an inverse problem and is the aim of the signal processing field. Many real-world signals bring with them some inherent structure or other characteristics that make these problems more tractable. Using this prior knowledge is what allows researches to develop successful methods for signal processing applications (Tur15).

3.2 | Pre-Processing as Inverse Problems

Since the point cloud results from a measurement process, we have a degraded signal that needs to be reconstructed. Then we can conclude that the problems tackled by this research have in common that they are ill-posed problems that may have many solutions and, therefore can be treated as an inverse problem. Inverse problems can be solved using the Bayesian theory; this theory contributes to an approach to the solution sought since it allows us to build models to deal with this kind of problem.

As already mentioned, in this thesis, we tackled the following three problems included in the pre-processing stage: noise reduction, feature detection, hole detection, and point simplification. In general, we can describe the three problems as inverse problems, which are modeled as:

$$y = x + n \tag{3.1}$$

Where x is the latent or unknown surface, y is the observed information (points with noise, sharp features, holes boundary, different levels of simplification), and n is additive random noise.

3.3 | Bayesian Inference

Bayesian inference is used in areas such as digital image processing, machine learning, decision theory, etc. Recently it has also been incorporated into the field of surface reconstruction (Han93) (DTB06) (JWB$^+$06) (HBS$^+$18). If the information observed is y, and the latent or unknown surface is x, the goal is to compute the optimal posterior probability using the Bayes rule.

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \propto p(y \mid x)p(x) \tag{3.2}$$

where $p(y \mid x)$ is the likelihood (data, fitting error, fitting data) term and fallows a Gaussian distribution, and $p(x)$ denotes the a priori model in our case the knowledge that is had or assumed of the surface. According to how the a priori model is chosen on the surface, the problem to be solved will take different solutions. i.e., a solution method will be more effective than another since the prior information represents the model of how we want our solution to have the result we are looking for.

3.4 | Inverse Problems Applied on Geometric Processing

As mentioned before, depending on how we choose, the prior model implies how the solution to our problem will be modeled. Equation 3.2 can also be taken to the form of Equation 3.3 if we modeled with Gaussian distributions, and the logarithm function applies to the equation terms

$$f(x) = \frac{1}{2} \|x - y\|_2^q + Pr(x) \tag{3.3}$$

where the term $\frac{1}{2} \|x - y\|_2^q$, is the mean square error or the fitting term, when $q = 2$, results in a convex optimization problem. The term $Pr(x)$, is known as prior or regularization term. This problem is solved by adopting the Maximum A-posteriori Probability (MAP) estimator. Over the past decades, all kinds of guess have been made about the prior information $Pr(x)$ in problems in which the MAP is used as a solution. That is why the a priori information has had different forms that range from the minimization of energy with L_2 norm, robust statistic models, the L_1 norm of the total variation, the use of sparse wavelets, Markov random fields, until the use sparse representations, with the L_0 and L_1 norms.

3.5 | Sparse Representations

Sparse representations have a biologically inspired motivation. Recent researches (OF96) (VG00) in the neuroscience field proved that cells of primary visual cortex, in the human brain, uses sparsity for visual recognition task to extract information from outside sources, because the information of interest is always sparse compared with the whole data. For example, Figure 3.1 shows different images where the human visual system fixed the view in particular objects (red car, woman and dog, white flower, red line) that are different respect to the background, the human brain is using sparsity to disregard the rest of visual information in each image.

From the linear algebra point of view, the term sparsity refers to the number of non-zero elements in a vector. One indicator to measure the sparsity of a vector $\boldsymbol{\alpha}$ is the l_0–norm or l_0–pseudonorm, which measures the dispersion, counting the number of non-zero coefficients in the vector $\boldsymbol{\alpha}$. The l_0–norm is defined as follows:

$$\|\boldsymbol{\alpha}\|_0 = |\{i : \boldsymbol{\alpha}_i \neq 0\}| \tag{3.4}$$

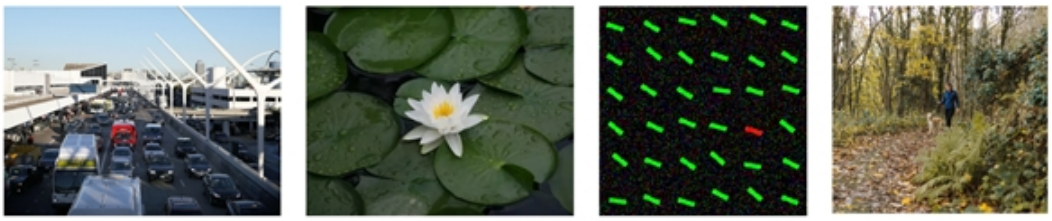


Figure 3.1: Set of images illustrating the sparsity concept.

where $|\{i : \boldsymbol{\alpha}_i \neq 0\}|$ denotes the cardinality of $\{i : \boldsymbol{\alpha}_i \neq 0\}$. Thus, a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ is sparse if $\|\boldsymbol{\alpha}\|_0 = k \ll n$. Another way to approximate sparsity is using the l_1–norm, which is the sum of the absolute value of the elements in a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$, is defined as follows:

$$\|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^{n} |\alpha_i| \tag{3.5}$$

Thus, a vector $\boldsymbol{\alpha} \in \mathrm{R}^n$ is sparse, if $\|\boldsymbol{\alpha}\|_1 = k \ll n$. If a signal can be characterized by a few significant terms in some domain, the signal is sparse.

3.6 | Sparse Coding

The purpose of sparse coding is to approximate a feature input vector as a linear combination of basis vectors, called atoms, which are selected from a dictionary learned from the data themselves. Sparse coding provides a low-dimensional approximation of a signal in a set of basis (OF97).

Formally, let $\mathbf{x}$ be a signal of dimension n, the sparse coding aims to find a dictionary $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots \mathbf{d}_\mathrm{N}\}$, such that x can be approximated by a linear combination of the atoms $\{\mathbf{d_i}\}_{\mathrm{i}=1}^{N}$ this is $x = \mathbf{D}\boldsymbol{\alpha} = \sum_{j=1}^{N} \boldsymbol{\alpha}_j \mathbf{d}_j$, (if the dictionary is overcomplete then $N > n$) where most of the coefficients α_j are zeros or close to zero (BJQS16). We have that the sparse coding problem can typically be formulated as an optimization problem as Equation (3.6) indicates:

$$\widehat{\boldsymbol{\alpha}} = arg \min_{\boldsymbol{\alpha}} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \;\; \text{s.t} \;\; \|\boldsymbol{\alpha}\|_0 \leq \mathrm{L} \tag{3.6}$$

In this formulation, the dictionary $\mathbf{D}$ is given, and L controls the sparsity of $\mathbf{x}$ in $\mathbf{D}$. The term $\|\boldsymbol{\alpha}\|_0$ measures the dispersion of the decomposition. It can be understood

as the number of non-zero coefficients in $\boldsymbol{\alpha}$, or, sparse coefficients, to approximate the signal x as sparse as possible. Alternatively, as (3.7):

$$\widehat{\boldsymbol{\alpha}} = arg \min_{\boldsymbol{\alpha}} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \tag{3.7}$$

It is an optimization problem where the norm $l_0 (\|\cdot\|_0)$ is changed, by the norm $l_1 (\|\cdot\|_1)$, being λ the regularization parameter. The solution to Equation (3.6) with l_0 norm is an NP-hard problem; fortunately, under certain conditions, it is possible to relax the problem using l_1 norm and find an approximated solution using equation (3.7).

3.7 | Numerical Methods for L1 Norm Minimization

Although the l_0-norm is the real measure of sparsity, the problems (3.6) become NP-hard, for which there is no optimal algorithm except brute force. As an alternative, the l_1-norm is used as an approximation; this is called relaxation. The algorithms presented in this research use the l_1-norm, as a regularization term. There is a wide range of numerical methods to solve problems that involve the l_1-norm some of them are

- Basis Pursuit Denoising Lasso type algorithms: Least Angle Regression (LARS).
- Constraint Optimization: Gradient Projection Sparse Reconstruction (GPRS), Interior Point Method Based Sparse Representation, Alternated Direction Method (ADM) Sparse Representation.
- Proximity Algorithm: Proximal Gradient Soft Thresholding or Shrinkage Operator, Iterative Shrinkage Thresholding Algorithm (ISTA), Fast Iterative Shrinkage Thresholding Algorithm (FISTA), Augmented Lagrange Multiplier (ALM), Split Bregman Method.

In this thesis, we use the LARS algorithm strategy for Sparse Coding and the Proximal Gradient strategy for smoothing.

3.7.1 | Least Angle Regression for Lasso - LARS Algorithm

LARS proposed by (EHJT04) is a model selection method for linear regression in high dimensional data, with fast convergence. At each step, it finds the predictor most correlated with the response. When multiple predictors have equal correlation, instead of continuing along with the same predictor, it proceeds in a direction equiangular between the predictors. LARS is summarized in Algorithm 3.1.

Algorithm 3.1: The Least Angle Regression (LARS) Algorithm

Input: $m \times n$ matrix $\mathbf{A}, \mathbf{y}$

1: **Initialize:**
$\mathbf{x} = 0, S = supp(x) = \boldsymbol{\phi}, \mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x} = \mathbf{y}$

2: **Select first variable:**
find the predictor (column in $\mathbf{A}$) most correlated with residual $\mathbf{r}$:
$i = \arg\max_i \mathbf{a}_i^T \mathbf{r}$
$\hat{x}_i = \max_i \mathbf{a}_i^T \mathbf{r}$
$S \leftarrow S \cup \{i\}$ //update the support

3: Move the coefficient x_i from 0 towards its least-squares coefficient $\hat{x}_{ik}$, updating the residual $\mathbf{r}$ along the way, until some other predictor $\mathbf{a}_j$ has as much correlation with the current residual as does $\mathbf{a}_j$; then add it to the support: $S \leftarrow S \cup \{j\}$

4: Move x_i and x_j in the direction defined by their joint least-squares coefficient:

$$\delta_k = (\mathbf{A}_{S^k}^T \mathbf{A}_{S^k})^{-1} \mathbf{A}_{S^k}^T \mathbf{r}$$

of the current residual on the current support set S, until some other predictor $\mathbf{a}_k$ has as much correlation with the current residual; then add it to the support: $S \leftarrow S \cup \{k\}$.

5: Continue adding predictors for $\min(m-1, n)$ steps, until full OLS solution is obtained. if $n < m$, all predictors are now in the model.

3.7.2 | Proximal Gradient

Proximal gradient algorithms (PB14) are useful when the function to minimize is divided into two parts, one of which is smooth, and the other part convex non-smooth, and there must exist a proximal operator. i.e., It is a way of solving non-smoothing convex problems with a composite structure.

$$\min_{x \in \mathbb{R}^n} F(x) = f(x) + h(x) \tag{3.8}$$

Where
$f(x)$ is convex and differentiable (example $f(x) = \frac{1}{2} \|Ax - b\|_2^2$),
$h(x)$ is convex, non-smooth and with an inexpensive proximal operator (example $h(x) = \gamma \|x\|_1$)

The proximal operator associated with the convex function $h(x)$ is

$$\mathbf{prox}_{h,\gamma}(n) = arg\min_{z}\left(h(z) + \frac{1}{2\gamma}\|z-n\|_{\frac{2}{2}}\right) \tag{3.9}$$

The proximal gradient descendent has an iteration with form

$$\mathbf{x^{k+1}} = \mathbf{prox}_{h,\gamma}\left(\mathbf{x^k} - \gamma\nabla f(\mathbf{x^k})\right) \tag{3.10}$$

Where $\gamma > 0$, is a scalar named step size that can be fixed or determined by line search 3.10 is computed iteratively until convergence.

The proximal operator corresponding to the function $h(x)$ and γ is the shrinkage or soft thresholding function.

$$\mathbf{prox}_{h,\gamma}(x_i) = \begin{cases} x_i - \gamma\lambda\,, & \text{if } x_i > \gamma\lambda \\ 0\,, & \text{if } |x_i| \leq \gamma\lambda \\ x_i + \gamma\lambda\,, & \text{if } x_i < -\gamma\lambda \end{cases} \tag{3.11}$$

Or equivalently $\mathbf{prox}_{h,\gamma}(x_i) = \mathbf{soft}(x,\gamma)$, where

$$\mathbf{soft}(x,\gamma) = \mathbf{sig}(x) * max(|x| - \gamma\lambda, 0) \tag{3.12}$$

The proximal gradient is summarized in Algorithm 3.2.

Algorithm 3.2: Proximal Gradient

1. **Initialize:**
   choose $\mathbf{x_0} \in R^n$
   $L \leftarrow L(f)$, a Lipshitz constant of ∇f

2. **Iteration step** k**:**
   $G_f(\mathbf{x}_{k-1}) \leftarrow \mathbf{x}_{k-1} - \frac{1}{L}\nabla f(\mathbf{x}_{k-1})$
   $\mathbf{x}_{k-1} \leftarrow Prox_{\mu g}(G_f(\mathbf{x}_{k-1}))$
   $k \leftarrow k+1$

3. **If** a convergence is satisfied, **then** exit, **otherwise** go to step 2.

3.8 | Dictionary Learning

Dictionary learning is the method of learning to find a set of basis vectors **D** represented in a matrix called dictionary, such that we can write a signal as a linear combination of a few columns from the matrix as possible, instead of using a pre-determined dictionary **D**, for instance, a Wavelet basis, contourlets, curvelets, etc. which are restricted to signal of a specific type and have a limited proficiency to make sparse the signals, the dictionary learning, learn an effective dictionary from an overcomplete basis features. A dictionary learning algorithm iteratively builds a dictionary from a training database of signal instances. Some research (OF96) has shown that learn the dictionary directly from training data rather than a pre-defined dictionary generally leads better representations, and it is reflecting in the improvement of results in different applications.

Given a dataset of n training signals $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$, the dictionary learning problem can be formulated as:

$$\widehat{\mathbf{D}}, \widehat{\boldsymbol{\alpha}} = arg \min_{\mathbf{D}, \boldsymbol{\alpha}} \sum_{i=1}^{n} \frac{1}{2} \|\mathbf{x_i} - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1 \tag{3.13}$$

Where $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots \boldsymbol{\alpha}_n\}$, are the sparse representation of the input signals **X**. Since both **D** and $\boldsymbol{\alpha}$ are unknown, this minimization is non-convex. An optimal local solution is obtained using an alternating-minimization strategy; this iterate until convergence between two optimization steps: (1) optimization respect to $\boldsymbol{\alpha}$, given a fixed **D**, and (2) optimization respect to **D**, given a fixed $\boldsymbol{\alpha}$.

The step (1) was already explained in section 3.6, with its respective optimization algorithm LARS for LASSO. Step (2) can be optimized using the Method of Optimal Directions (MOD) (EAHH99) or the K-SVD algorithm. In this research, we use the K-SVD.

3.8.1 | The K-SVD Algorithm

The K-SVD algorithm is a generalization of K-Means clustering process presented in (AEB06). It Is used for dictionary learning to build a dictionary for sparse representations. K-SVD is an iterative method that alternates between sparse coding step and updating the dictionary atoms to fit the data better. The K-SVD is described in Algorithm 3.3.

Algorithm 3.3: The K-SVD dictionary learning algorithm

Requires: Set of training signals Y
Ensure: Trained dictionary D

Initialize Dictionary: Build $D_{(0)} \in \mathbb{R}^{nxm}$
while $\|\mathbf{Y} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 > \epsilon$ **do**
  Sparse Coding Stage: Use a pursuit algorithm to approximate the solution of
  $$\alpha_i' = \min_\alpha \|\mathbf{y}_i - \mathbf{D}_{k-1}\boldsymbol{\alpha}\|_2^2 \; s.t. \; \|\boldsymbol{\alpha}\|_0 \leq k_0$$
  obtaining sparse representations α_i' for $1 \leq i \leq M$. These form the matrix $\mathbf{X}_{(4)}$
  K-SVD Dictionary - Update Stage:
  for $j_0 = 1, 2, \ldots, m$ **do**
    Update the columns of the dictionary and obtain $D_{(k)}$
    Define the group of examples that use the atom α_{j0},
    $$\omega_{j0} = \{i \mid 1 \leq i \leq M, \mathbf{X}_{(k)}[j_0, i] \neq 0\}$$
    Compute the residual matrix
    $$E_{j0} = \mathbf{Y} - \sum_{j \neq j_0} \mathbf{d}_i \boldsymbol{\alpha}_j^T$$
    where x_j are the $j'th$ rows in the matrix $\mathbf{X}_{(k)}$
    Restrict E_{j0} by choosing only the columns corresponding to ω_{j0}, and obtain E_{j0}^R
    Apply SVD decomposition $E_{j0}^R = U\Delta V^T$
    Update the dictionary atom $d_{j0} = u_1$ where u_1 is the higher value of matrix U, obtained from SVD decomposition.
  end for
end while
return D

3.9 | L1 Median

The L_1-Median is a robust estimator related with the multivariate median, it is defined to be the point x, which minimizes the sum of Euclidean distances to all points in the data set $\{p_j\}_{j \in J}$

$$arg \min_x \sum_{j \in J} \|p_j - \mathbf{x}\| \tag{3.14}$$

L_1-Median is insensitive to outliers and noise, compared with the mean (LCOLTE07). We use the L_1-median as the data-fidelity term, due to these properties.

3.10 | Sparse Regularization and Fitting

Sparse regularization and fitting can be classified into two types (XWZ+15):

- Imposing sparsity norm on the regularization term. This type of sparse regularization is used to denoise and preserve sharp features. Second term (3.3)
- Imposing sparsity-based measurement in the fitting term. This type of fitting term is used because it works very well for a dataset with noise and outliers. First-term (3.3)

In chapter 4, we use both sparsity types for point cloud denoising, using $L_1 - median$ as the fitting term and $L_1 - norm$ as the regularization term.

3.11 | Minimum Description Length

The MDL principle (RS12) (SCMP14) (Ris78), states that given a model class $\mathfrak{M}$ of candidate models, its parameter M, and its data sample x, the MDL provides a generic solution to the model selection that minimally represent the data x. Formally, given a set of candidate models $\mathfrak{M}$, and a data vector x, MDL search for the best model $\widehat{M} \in \mathfrak{M}$, that can be used to describe x completely with the shortest length.

$$\widehat{M} = arg \min_{M \in \mathfrak{M}} L(x, M) \tag{3.15}$$

Where, $L(x, M)$ is a coding assignment function, which gives the code length required to describe (x, M) uniquely. In this research, we use the MDL concept to establish a relationship between the length of the sparsity vector (prior term) and fitting term (likelihood) with saliency detection on point clouds, as is described in chapter 5.

4

Point Cloud Denoising

During the 3D scanning process it is inevitable to get a point cloud with noise and outliers. Denoising the point cloud is fundamental for reconstructing high quality surfaces with details, in order to eliminate noise and outliers in the 3D scanning process. The challenges for a denoising algorithm are noise reduction and sharp features preservation. In this chapter, we present a new sparse regularization model for reconstructing and smoothing point clouds. The proposed approach combines the L1-median filter with the sparse L1-norm for both denoising the normal vectors and updating the position of the points to preserve sharp features in the point cloud. The L1-median filter is a robust estimator insensible to outliers and noise compared to the mean, and its inclusion in the proposed method is to find a local reference plane to estimate the normal vector. L1 norm is a way to measure the sparsity of a solution, and applying an L1 optimization to the point cloud, can measure the sparsity of sharp features, producing clean point set surfaces with sharp features. We optimize the model recurring an alternating optimization strategy, which involves a proximal gradient descent algorithm to solve the L1 minimization problem. Experimental results show that our approach has advantages respect to the state-of-art methods since filters out 3D models with a high level of noise keeping their geometric features. Both visual and quantitative comparison shows that our approach, sometimes, outperforms the competing methods.

4.1 | Introduction

With the rapid expansion of 3D scanning devices, the process of capturing and scanning real objects has become a common task in many areas, ranging from medicine and entertainment to industry and 3D printing. Despite significant development in the precision of 3D scanning technology, the raw data produced by the scanning devices inevitably

contains noise and outliers, caused by the inherent measurement error of 3D devices and the digitization process. Hence the importance of denoising the point cloud in a pre-processing step before proceeding with surface reconstruction or shape analysis. The goal of the denoising algorithms is to suppress noise and outliers while preserving the sharp features such as edges and corners. Unlike denoising methods focused on triangular meshes, methods focused on point clouds do not have connectivity information, introducing an additional challenge. Denoising point clouds with sharp features is a complex problem as the features and noise are high frequency and therefore difficult to distinguish.

Many point set surfaces are piecewise smooth almost everywhere except for a few features such as corners and edges (SSW15) (ASGCO10). This means that these features are sparse, allowing a sparsity analysis in the point clouds to be conducted to estimate them. We can measure the sparsity of a solution using either the L_0 norm or the L_1 norm. The L_0 norm counts the number of non-zero elements in a vector, directly measuring the sparsity, but it is challenging to optimize due to its non-convexity; meanwhile, the L_1 norm can approximate the L_0 norm. The L_1 norm is convex, and under certain conditions, produces sparse solutions. Some works exploit the sparsity theory on point clouds (SSW15) (ASGCO10) (MC17) (SBV$^+$18), and applying this theory is motivated by the field of sparse signal reconstruction and compressed sensing (Don06) (CRT06). These works have attempted to overcome the problems related to noisy point clouds; the algorithms proposed in these studies perform well for feature preservation with a certain level of noise. Still, when the noise scale is larger, or impulsive noise is present, they usually do not do well. Although the L_0 norm produces sparser solution than the L_1 norm; the application of the L_0 minimization can over-flatten and over-sharpen effects for small geometric features.

In this thesis, we propose a robust method that focuses on removing noise and outliers while preserving the sharp features in a point cloud. Our approach comprises two iterative steps: (1) the normal estimation, finding a regression plane equidistant to all heights in a local neighborhood to then calculate the normal at the plane; and (2) based on the estimation of the normals, the position of the points is updated, using the orthogonal distance of the noisy point to the local regression plane, shifting the point along the normal direction projecting it onto the plane. This two-step procedure is repeated until a minimum error threshold is reached.

Our work is motivated by three observations: (1) L1-median filtering data-fidelity term encourages us to find a local regression plane to approximate the input points while discarding the noise and outliers; (2) points that belong to the same local smooth region will have similar normals and the differences between them should be sparse,

while large differences in values would reveal sharp features; and (3) points in a local neighborhood must comply with a local planarity criterion, except in the sharp features.

4.1.1 | Contributions

The major contributions of this chapter are:

1. We present a two-step method that combines the L_1-median filtering as a data-fidelity term with L_1 minimization strategies as a regularization term. For both normal estimation and point position update. The method measures the sparsity of sharp features and discriminates between noise and features.

2. An adaptive weighted strategy is employed to improve the normal estimation on sharp features, giving more importance to neighboring points in the same surface and discriminating neighboring points that cross sharp features, these points are identified as outliers.

3. The method can handle point clouds with significant scale noise, outliers, impulsive noise, and is easy to implement.

4. The performance of our method is shown in a variety of models scanned by 3D devices and synthetic point cloud as CAD models, outperforming some point clouds denoising methods of the state-of-the-art.

4.2 | Related work

Point cloud denoising algorithms can be roughly divided into six categories: moving least squares (MLS)-based methods, locally optimal projection (LOP)-based methods, sparsity-based methods, non-local similarity-based method, graph-based methods, and normal smoothing based methods. In this chapter, we are interested in point cloud denoising coupled with features preservation.

4.2.1 | MLS-Based Methods

The MLS (ABCO$^+$03) methods approximate a noisy input point cloud with a smooth surface by projecting the noisy points onto the MLS surface. Three steps are required to project each point: (1) finding a local reference domain to each point, (2) defining a function above the reference domain by fitting a bivariate polynomial using its neighboring points, (3) compute the projection by evaluating the polynomial at the origin.

MLS methods have some drawbacks, because they are not robust to outliers. The projection procedure can be unstable in high curvature and low sampling rate and can over-smooth the surface. Several variants to this method have been proposed to correct the cited problems and for handling sharp features; e.g., algebraic point set surfaces (APSS) (GG07), and robust implicit MLS (RIMLS) (OGG09).

4.2.2 | LOP-Based Methods

Without the use of normal information, LOP-based methods (LCOLTE07) generate a set of points called particles and using the L_1 median and a regularization term, carry out a iterative process to reach a clean surface. This method projects the points to an underlying surface while enforcing a uniform distribution of points. A variation to this method is the Weighted LOP (WLOP) (HLZ$^+$09). This method produces more evenly distributed points on the surface but can not handle well the sharp features. Edge Aware Resampling (EAR)(HWG$^+$13) or Anisotropic LOP (AWLOP), improve sharp features by modifying LOP to use normal information as a weight function. However, LOP-based methods can produce over-smoothing because of the use of local operators.

4.2.3 | Non-Local Similarity -Based Methods

These methods are inspired by the image processing techniques Non-Local Mean (NLM) (BCM05) and the BM3D (DFKE07) algorithms, and they exploit the concepts of self-similarity between small surface patches in the point cloud. The direct application of this concept to point cloud is not straightforward, because the point clouds structure does not exhibit the regular disposition such as the pixels in an image in their structure. The methods based on MLN or BM3D better preserve structural features under a high level of noise. One of the first works extending the NLM algorithm to operate in point clouds is (DG10). This method uses the MLS surface and its polynomial coefficients as neighborhood descriptors to find similar patches or neighborhoods. In (RDK13), [16], the authors generalized the BM3D, searching for similar neighborhoods globally in the point cloud using the Iterative Closest Point (ICP) algorithm. Low-rank matrix representation is used in (Sarkar et al., 2018), wherein the authors use dictionary representation from the noisy patches to smooth 3D patches. The drawback of this method is its computational complexity given the global point clouds search.

4.2.4 | Graph-Based Methods

Graph-based methods treat the point cloud as a signal in a graph, and the smooth surface is chosen by using graph filters. In (SPV15), the authors use the graph of the k-nearest neighborhood to represent a point cloud as a signal, and carry out the smoothing via convex optimization. The method in (DCBY18), used weighted graph Laplacian over the normals and total variation L1-norm as the regularized term to model two kinds of additive noise. Using a bipartite graph, establish a linear relationship between the points and normals to proceed with the optimization (ZCN$^+$20), , the authors proposed the use of a graph Laplacian regularization and low manifold model to find self-similarity between patches to the smooth point cloud, avoiding the direct smoothing of point coordinates or point normals. In (DCK18), the authors compute local tangent planes based on a graph and then reconstruct the point cloud by the weighted average of its projections in the tangent planes.

4.2.5 | Normal Smoothing-Based Methods

Normal smoothing methods are focused on estimating noise-free normals, followed by an updating of the position of the points based on the clean normals. In Leal (LL06), use a robust version of Principal Component Analysis (PCA), to estimate the normal vector, the authors propose weights factors that are inversely proportional to the sum of the distance to the mean. The weights defined in this way make the method robust to outliers and noise. They propose a simple solution to avoid data shrinkage using bootstrap bias correction. The method iteratively smooths the surface and preserves sharp features.

Zheng (ZLW$^+$17), use the multi-normal concept to correctly estimate the normals in edges and corners, based on the observation that points on the same side of the edges have the same normal orientation. The first step is to detect the edges and then refine them using the l_1-skeleton algorithm (HWCO$^+$13), , followed by an estimation of the multi-normals, and finally, an updating of the position by optimizing a height-based function. (ZLX$^+$18) use the same multi-normal concept for denoising. The first step here is the same as for (ZLW$^+$17), but for the point position update, they introduce an energy term to avoid point cloud deformation close to the edges. The method is robust to the noise and preserves edges and corners. Yadav (YRS$^+$18), use normal voting tensor analysis and binary optimization to estimate noise-free normals. Next, to update the position of points, the method classifies each point on the cloud as edge, corner, or planar point, and based on this classification, they propose three optimization proce-

dures for each type of point. The method is effective for denoising the point cloud and preserves the sharp features.

4.2.6 | Sparsity-Based Methods

These methods are based on the theory of sparse representation of certain geometric features of the point cloud. The Sparsity-based methods assume local planarity for the optimization model. The local planarity comprises the following assumption, if two points belong to the same smooth region, its normal vectors will be similar, therefor the gradient should be sparse. Recently attention is being devoted to sparsity-based methods in geometry processing (XWZ+15). These methods comprise two sparse modeling steps. The first carries out a sparse reconstruction of the normals by solving a global minimization problem with sparse prior regularization. The prior model can be the L_0-norm or L_1-norm. Based on the smoothed normals, the point positions are updated following a second sparse global model optimization. The methods (ASGCO10) and (SSW15) follow this strategy. Mattei (MC17), propose a method called Moving Robust Principal Component Analysis (MRPCA), using weighted minimization of the point deviations from a local reference plane to preserve sharp features. However, when the noise level is high, over-smoothing or over-sharpening occurs. Our approximation belongs to sparsity-based methods and is in the same spirit of MRPCA, but the difference lies in that we use sparsity in both data fitting and the prior term. Our method uses the L_1-median for the fitting term and the L_1-norm for the regularization term. For the proposed method, we apply a local sparse optimization strategy based on the proximal gradient.

4.3 | Preliminaries

In this section, we present the main concepts related to point cloud smoothing in the proposed algorithm.

4.3.1 | Noisy Point Clouds

Starting from a noisy and possible nonuniform input point cloud, $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots \mathbf{p}_n\} \subset \mathbb{R}^3$, sampling from a 2D surface in a 3D space, with $n \in \mathbb{N}$, denoting the number of points. The goal of point cloud denoising is to recover the original point cloud $\mathbf{P}'$ from

its noisy measurement $\mathbf{P}$. The point cloud denoising problem can be modeled as

$$\mathbf{P} = \mathbf{P}' + \varepsilon \tag{4.1}$$

where ε is zero-mean i.i.d noise. This is an ill-posed problem since $\mathbf{P}'$ and ε are unknown. To recovery $\mathbf{P}'$, we use the L_1-median, L_1-norm strategy.

To every point $\mathbf{p}_i \in \mathbf{P}$, a neighborhood $N_g(\mathbf{p}_i)$ with radius $r > 0$, is defined as follow: $N_g(\mathbf{p}_i) = \{\mathbf{p}_j \mid \|\mathbf{p}_i - \mathbf{p}_j\| \leqslant r\}$, are all the points $\mathbf{p}_j$ that have a distance to $\mathbf{p}_i$ less or equal than r. The neighborhood $N_g(\mathbf{p}_i)$, is used to estimate the surface local properties.

4.3.2 | Surface Normal

We begin estimating the normal vector $\mathbf{n}_i$ for each point $\mathbf{p}_i \in \mathbf{P}$, getting the set of normals $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots \mathbf{n}_n\}$, with $\mathbf{n}_i \in \mathbb{R}^3$, we define $N_g(\mathbf{p}_i) \subset \mathbf{N}$, as a neighborhood of normals around $\mathbf{n}_i$. There are several methods for normal estimation in the literature (LSK+10) (MWP18) (YWG+19), the most popular is the PCA method (HDD+92). Almost all of these methods adjust a regression plane to the surface in a point $\mathbf{p}_i$, using its k-nearest neighboring points, and take a vector perpendicular to the plane. Our method follows a similar approach minimizing the height of points over a regression plane with respect to the neighborhood.

4.3.2.1 | L1-Median Filter

As we mentioned in section 3.9, the L_1-Median is a robust estimator related with the multivariate median, and is defined to be the point $\mathbf{p}$, which minimizes the sum of Euclidean distances to all points in the data set $\mathbf{I}$ i.e. $\{\mathbf{p}_i\}_{i \in I}$. We use the L_1-Median filter to apply sparsity in the fitting data term (section 3.10), being the proposed method robust to outliers, and to high level of noise.

4.3.3 | L1-Sparse Regularization

The L_1 regularization has been applied for feature selection (Ng04), sparse signal reconstruction (LW16), signal processing as image decomposition (ESQD05), and basis pursuit (SD94). Although L_0 regularization produces the sparsest solution, under certain conditions, L_1 regularization produces a sparse solution (DET06). In image processing, L_1 norm has been successfully applied to preserve fine details and edges through the minimization of gradient (ROF92). The above is known as total-variation regularization or TV and is used to measure the sparsity of the gradient. The proposed method uses

TV for estimating the normal preserving the sharp features. Our approach uses the local planarity criterion to measure the deviation of each point $\mathbf{p}_i \in \mathbf{P}$ from the regression plane defined over its neighborhood $N_g(\mathbf{p}_i)$ and its corresponding normal vector $\mathbf{n}_i$.

4.4 | Robust Point Cloud Denoising

Starting from a noisy point cloud $\mathbf{P}$ near a unknow surface S, the goal of the proposed algorithm is found a noise-free point set $\mathbf{P}'$, that conserve the features of the original point cloud. We use local neighborhoods to each point $\mathbf{p}_i \in \mathbf{P}$ to approximate the surface S. The main idea is to define a local reference tangent plane $\mathfrak{T}_p$ to every $\mathbf{p}_i$ in the point cloud and determine its normal vector $\mathbf{n}_i$, then shift the point $\mathbf{p}_i$ in the normal direction to a distance $\tau_i \in \mathbb{R}$, obtaining a new position $\mathbf{p'}_i = \mathbf{p} + \tau_i \mathbf{n}_i$, $\mathbf{p'}_i \in \mathbf{P}'$. The new position $\mathbf{p'}_i$ being the projection of the point $\mathbf{p}_i$ onto the tangent plane $\mathfrak{T}_p$, which is the linear approximation of the surface S at the point $\mathbf{p}_i$. See Figure 4.1

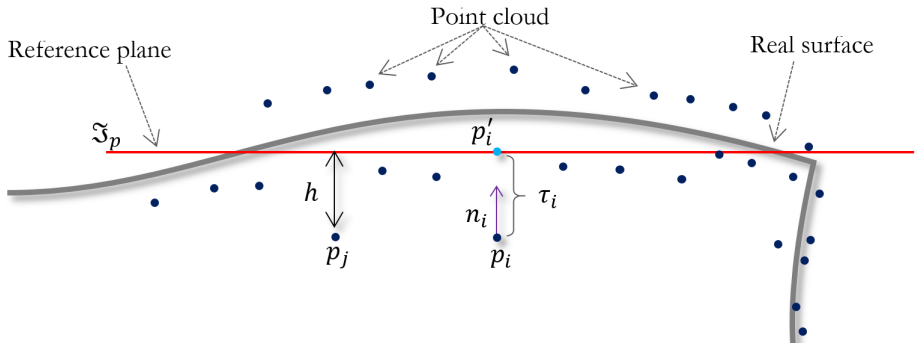


Figure 4.1: The point $\mathbf{p}_i$ is projected onto the reference plane $\mathfrak{T}_p$. The tangent plane $\mathfrak{T}_p$ is a linear approximation to the surface S, around the point $\mathbf{p}_i$

The normal $\mathbf{n}_i$ and the displacement τ_i, are computed iteratively by adjusting the tangent plane $\mathfrak{T}_p$ to the neighborhood $N_g(\mathbf{p}_i)$. To estimate the tangent plane $\mathfrak{T}_p$, we are looking for an equidistance height to all heights h of the points $\mathbf{p}_j \in N_g(\mathbf{p}_i)$ over $\mathfrak{T}_p$ (Figure 4.1). To estimate the local reference tangent plane $\mathfrak{T}_p$, we minimize the cost function (4.2) respect to τ and $\mathbf{n}$, subject to the constraint $\|\mathbf{n}\| = 1$.

4.4.1 | Cost Function

To denoise the noisy point cloud, we integrate L_1-median height filter and L_1-norm of gradient or total variation. The normal is obtained, minimizing the following energy

functional.

$$\min_{\tau,n} E_f + \lambda E_{reg} \tag{4.2}$$

Where E_f is the fidelity term, E_{reg} is the regularization term, λ is the regularization parameter and $\|\mathbf{n}\| = 1$. The detailed description of the three terms is provided below.

4.4.1.1 | L_1-median height Fidelity Term E_f:

The fidelity term E_f is used to fit a robust hyperplane in the neighborhood of the sampled point $\mathbf{p}_i$, and to estimate the normal vector with respect to the hyperplane. The estimation of the hyperplane is robust to large deviations of points $\mathbf{p}_j \in N_g(\mathbf{p}_i)$. The outliers are identified by the L_1-median height filter, which penalizes points $\mathbf{p}_j$ with large orthogonal projections or heights h_j respect to the hyperplane, see Figure 4.1. We define our fidelity term as

$$E_f = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \|h_j - \tau_i\| \, \psi\left(h_j, \tau_i\right) \theta\left(\|\mathbf{p}_i - \mathbf{p}_j\|\right) \tag{4.3}$$

Where $h_j = \mathbf{n}_i^t(\mathbf{p}_i - \mathbf{p}_j)$, $\psi\left(h_j, \tau_i\right) = e^{-(h_j-\tau_i)^2/\sigma_h^2}$, $\theta\left(\|\mathbf{p}_i - \mathbf{p}_j\|\right) = e^{-d^2/\sigma_d^2}$

We minimize the orthogonal projections (height) of points $\mathbf{p}_j \in N_g(\mathbf{p}_i)$, to the hyperplane with a localized version of Equation (3.14). τ_i is the height to be found, which minimizes the orthogonal projections of each point $\mathbf{p}_j \in N_g(\mathbf{p}_i)$ to the hyperplane, Figure 4.1

Consequently, points $\mathbf{p}_j \in N_g(\mathbf{p}_i)$ with considerable heights h_j are probably located passing through the sharp features; these points are possible outliers. As such, we propose an adaptive weighting strategy, which adaptively assigns the weight of each point as a function of the height. Thus, the weighting term $\psi(\cdot)$ in (4.3) adaptively encourages the reduction of the influence of points $\mathbf{p}_j \in N_g(\mathbf{p}_i)$, with large height h_j values, and σ_h is the height parameter which controls sensitivity to outliers. Thus, the term $\psi(\cdot)$, only considers points located in the same smooth region to estimate the normal vector, the results of including this weight function combined with the L_1 total variation can be appreciated in Figure 4.2. The weighting term $\theta(\|\cdot\|))$ in (4.3), is the distance weight function and σ_d is the position parameter used to control the action range.

4.4.1.2 | L_1 norm Regularization Term E_{reg}:

The regularization term E_{reg} is introduced as a measure of sparsity to preserve the sharp features and to smooth the underlying surface. If a point cloud is piecewise smooth,

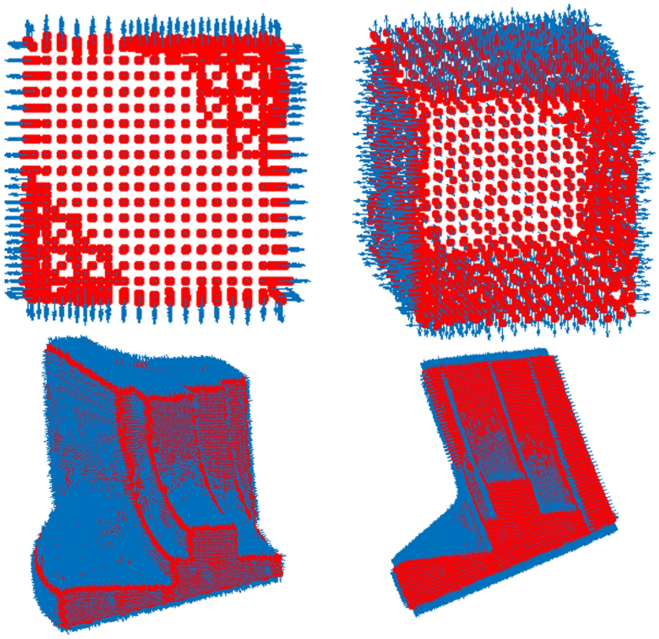


Figure 4.2: Normal estimation by the proposed method. Note how the method estimate in the right way the normals in the of sharp features. First row, the Cube model with irregular sampling and contaminated with Gaussian noise ($\sigma = 0.3h$). Second row, the Fandisk model with high irregular sampling and contaminated with Gaussian noise ($\sigma = 0.28h$). Where h, is the average distance between the points.

many of the gradients in the normals field $\mathbf{N}$ (consistently oriented) tend to be zero; in contrast, the large values of the gradient only indicate sharp features. This means that normals $\mathbf{n}_j \in N_g(\mathbf{n}_i)$ in a neighborhood must be similar. Then the regularization term is formulated as a total variation.

$$E_{reg} = \sum_{\mathbf{n}_j \in N_g(\mathbf{n}_i)} w_{i,j} \left\| \mathbf{n}_i - \mathbf{n}_j \right\|_1 \tag{4.4}$$

Where $w_{i,j} = e^{-\left(\frac{1-\mathbf{n}_i^t \mathbf{n}_j}{1-cos(\sigma_n)}\right)^2}$, is the normal weight function. σ_n is the angle parameter that measures the similarity between normals $\mathbf{n}_j \in N_g(\mathbf{n}_i)$ and $\mathbf{n}_i$, is typically set $\sigma_n = 15°$.

4.4.2 | Model Optimization

Once the terms of (4.2) are defined, we continue with the model optimization for $\mathbf{n}_i$ and τ_i subject to $\|\mathbf{n}\| = 1$. Our energy minimization problem is formulated as

$$\min_{\mathbf{n}_i, \tau_i} \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \|h_j - \tau_i\| \, \psi\left(h_j, \tau_i\right) \theta\left(\|\mathbf{p}_i - \mathbf{p}_j\|\right) + \lambda \sum_{\mathbf{n}_j \in N_g(\mathbf{n}_i)} w_{i,j} \left\|\mathbf{n}_i - \mathbf{n}_j\right\|_1 \tag{4.5}$$

We found the optimal values of $\mathbf{n}_i$ and τ_i by an alternating optimization strategy. This procedure is shown in Algorithm 4.1.

Algorithm 4.1: Alternating Optimization Strategy

Initialization: $j \leftarrow 0$
repeat
  $\tau^0 \leftarrow 0$
  repeat
    fix τ^k, solve for $\mathbf{n}^k$ as minimum of Eq. 4.5
    fix $\mathbf{n}^k$, solve for τ^k as minimum of Eq. 4.5
    $\mathbf{p}^{k+1} = \mathbf{p}^k + \tau^k \mathbf{n}^k$
  until $\left\|\mathbf{p}^{k+1} - \mathbf{p}^k\right\|_2^2 < \varepsilon$
  edge points correction
  $j \leftarrow j + 1$
until $j > j_{max}$

4.4.2.1 | $\mathbf{n}_0$ initialization

First, we estimate an initial normal set $\mathbf{N}$ as input in the first iteration ($j = 1$) of the algorithm 4.1. The set $\mathbf{N}$ is an initial solution of the equation (4.5) using only the equation corresponding to the fidelity term, with $\tau = 0$. Similar to (MVF03), we use the constraint $\|\mathbf{n}\| = 1$, and compose the Lagrange form of (4.3) to compute the derivative with respect to $\mathbf{n}$, obtaining:

$$L(\mathbf{n}, \lambda) = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \|h_j\| \, \psi\left(h_j\right) \theta\left(\|\mathbf{p}_i - \mathbf{p}_j\|\right) + \frac{\lambda}{2} \left(1 - \|\mathbf{n}_i\|\right)^2 \tag{4.6}$$

$$L_n(\mathbf{n}, \lambda) = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \omega_i (\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^t \mathbf{n}_i - \lambda \mathbf{n}_i = 0 \tag{4.7}$$

with weight $\omega_i = \psi(h_j)\theta\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|\right) / \left\|h_j\right\|$, as before $h_j = \mathbf{n}_i^t(\mathbf{p}_i - \mathbf{p}_j)$. The details of the derivation is given in Appendix A.1
We can see that the term $\omega_i(\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^t$ on (4.7) is a symmetric and definite positive matrix (weighted covariance matrix) and we can rewrite (4.7) depending on $\mathbf{n}$ as:

$$Cm(\mathbf{n})\mathbf{n} = \lambda\mathbf{n} \tag{4.8}$$

where $Cm(\mathbf{n}) = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \omega_i(\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^t$

Equation (4.8) is an eigensystem and can be iteratively solved as follows:

$$Cm(\mathbf{n}^k)\mathbf{n}^{k+1} = \lambda^{k+1}\mathbf{n}^{k+1} \tag{4.9}$$

Where λ^{k+1}, is the smallest eigenvalue of $Cm(\mathbf{n}^k)$ and $\mathbf{n}^{k+1}$ is an orthonormal eigenvector. We start the initialization with $\mathbf{n}^0 = 0$, i.e., $Cm(0)\mathbf{n}^1 = \lambda^1\mathbf{n}^1$, is the first iteration. A few iterations are necessary for obtaining good results since Equation 4.9 always converges to the solution. Once we get the initial estimation of the set $\mathbf{N}$, we proceed with the optimization of (4.5).

We solve the energy minimization problem (4.5) for $\mathbf{n}$, having fixed τ. Since the minimization problem (4.5) is non-differentiable due to the regularization term E_{reg} we use the proximal gradient descendent method (PB14) as the optimization strategy.

4.4.2.2 | $\mathbf{n}$ minimization

keeping τ and $\psi(\cdot)$ fixed to solve (4.5), $\psi(\cdot)$ is treated as a constant in the optimization, because the problem has a complex form and fixing it, is a practical way to make it computationally tractable. Thus, the fidelity term E_f has gradient ∇E_f:

$$\nabla E_f = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \eta_i(h_j - \tau_i)(\mathbf{p}_i - \mathbf{p}_j)^t \tag{4.10}$$

With weight $\eta_i = \psi(h_j, \tau_i)\theta\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|\right) / \left\|h_j - \tau_i\right\|$, η_i is undefined when $h_j = \tau_i$. Therefore, when $\left\|h_j - \tau_i\right\| < 10^{-3}$, we set $\eta_i = \theta\left(\left\|\mathbf{p}_i - \mathbf{p}_j\right\|\right)$.The details of the derivation is given in Appendix A.2

Set $d_i = n_i - n_j$, for the regularization term E_{reg}, we define the proximal mapping (or operator), associated with a convex non-differentiable function $h()$ as follows.

$$\mathbf{prox}_{h,\gamma}(d) = arg\min_z \left(h(z) + \frac{1}{2\gamma}\left\|z - d\right\|_2^2 \right)$$

The proximal gradient descendent has an iteration form

$$\mathbf{d}^{\mathbf{k+1}} = \mathbf{prox}_{h,\gamma}\left(\mathbf{d}^{\mathbf{k}} - \gamma\nabla f(\mathbf{n}^{\mathbf{k}})\right) \tag{4.11}$$

Where $\gamma > 0$, is a scalar termed step size and 4.11 is computed iteratively until convergence. The proximal operator corresponding to the L_1-norm or regularization term E_{reg} is the shrinkage or soft thresholding function:

$$\mathbf{prox}_{h,\gamma}(d_{ic}) = \begin{cases} d_{ic} - \gamma\lambda w_{ij}\,, & \text{if } \; d_{ic} > \; \gamma\lambda w_{ij} \\ 0\,, & \text{if } \; |d_{ic}| \leq \gamma\lambda w_{ij} \\ d_{ic} + \gamma\lambda w_{ij}\,, & \text{if } \; d_{ic} < -\gamma\lambda w_{ij} \end{cases} \tag{4.12}$$

Where d_{ic}, is each component of the normal vector $\mathbf{d}_i$.

4.4.2.3 | τ minimization

With n fixed, we solve the equation (4.5) for τ, omitting the regularization term E_{reg}, since it does not involve τ. The fidelity term E_f has gradient ∇E_f:

$$\nabla E_f = \sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \eta_i(h_j - \tau_i) = 0 \tag{4.13}$$

By solving $\nabla E_f(\tau)$, we obtain an iterative solution, which yields the following local update equation.

$$\tau_i^{k+1} = \frac{\sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \eta_i h_j}{\sum_{\mathbf{p}_j \in N_g(\mathbf{p}_i)} \eta_i} \tag{4.14}$$

where $\eta_i = \frac{\psi(h_j,\tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|h_j - \tau_i\|}$. $\mathbf{n}$ and τ are iteratively optimized using (4.11) and (4.14) until convergence, η_i is undefined when $h_j = \tau_i$. Therefore, when $\|h_j - \tau_i\| < 10^{-3}$, we set $\eta_i = \theta\left(\|\mathbf{p}_i - \mathbf{p}_j\|\right)$. The details of the derivation is given in Appendix A.3

4.4.3 | Point Position Update and Point Border Correction

In the last stage of the denoising method, we follow the update vertex position with a distance-based constraint proposed by (YRS+18), where the resulted filtering point cloud $\mathbf{P}$ is bounded within a prescribed distance to the input point cloud.

4.4.3.1 | Point Position Update

(YRS+18), propose a parameter provided by the user $\varepsilon \in \mathbb{R}^+$, bounding the maximum deviation d_i between the initial noisy point cloud and its corresponding iteratively denoised version point $\mathbf{p'}_i \in \mathbf{P'}$. The update position point $\mathbf{p'}_i$, for our algorithm is determined as follow:

$$\mathbf{p'}_i = \begin{cases} \mathbf{p}_i + \tau_i \mathbf{n}_i \,, & \text{if } d_i \leqslant \varepsilon \\ \mathbf{p}_i \,, & \text{if } d_i > \varepsilon \end{cases} \tag{4.15}$$

Where d_i is computed as the difference between $\mathbf{p}_i$ and the corresponding original point in the noisy point cloud. The parameter ε is set to 4h. i.e., $\varepsilon = 4\text{h}$. Where h, is the average spacing between the points of the point cloud.

To make our algorithm more robust against edge artifacts and blurring, we detect the edge and corner points, and it corrects its position to obtain cleaner and more defined borders, as is shown in Figure 4.3.

4.4.3.2 | Edge Points Detection

To detect sharp features in the point cloud, we recur to the method proposed in (ZLW+17), who use the normals associated to each point in $\mathbf{P}$, and measure the normal variability into the neighborhood. If the variability is lower than a predefined threshold, th, is the point labeled an edge. The similarity between normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ is defined as follow:

$$w_n(\mathbf{n}_i, \mathbf{n}_j) = exp\left(\frac{\|\mathbf{n}_i - \mathbf{n}_j\|^2}{2\sigma_n^2}\right) \tag{4.16}$$

Where σ_n is an angle threshold, using (4.16) they defined the normal variation in $N_g(\mathbf{n}_i)$ as follow:

$$V_n(i) = \frac{1}{|N_g(\mathbf{n}_i)|} \sum_{\mathbf{n}_j \in N_g(\mathbf{n}_i)} w_n(\mathbf{n}_i, \mathbf{n}_j) \tag{4.17}$$

All points that satisfy $V_n(i) < th$, are labeled as edge points. The threshold values used in our experiments are $th = \{0.2, 0.3, 0.5\}$.

4.4.3.3 | Edge point Correction

After edge points are detected and taking advantage of the fact that the estimated normals near the edges and corners belong to surfaces on one side or another of the sharp

features, we propose a scheme to correct the position of points that belong to edges or corners, which present a deviation from the corner or borderline. As shown in Figure 4.3, we find the closest point $\mathbf{p}_j$ with normal vector $\mathbf{v}_j$ on an opposite surface to the edge point $\mathbf{p}_i$ and its normal $\mathbf{n}_i$. Next, we project the point $\mathbf{p}_i$ onto the plane that contains the point $\mathbf{p}_j$ and its normal $\mathbf{n}_j$. The new position is computed as:

$$d_{proj} = \mathbf{n}_j^t(\mathbf{p}_i - \mathbf{p}_j) \tag{4.18}$$

We only correct the point positions that meet the following criteria:

$$\mathbf{p}_{corr} = \begin{cases} \mathbf{p}_i - d_{proj} \cdot \mathbf{n}_j^t\,, & \text{if } \delta < |d_{proj}| < \rho h \\ \mathbf{p}_i\,, & other\ case \end{cases} \tag{4.19}$$

Where h, is the average spacing between the points of the point cloud, ρ is a fixed value that represents the percentage of maximum shift of the point $\mathbf{p}_i$ towards the edge line, and δ is a fixed value close to zero. The bound displacement ρ is introduced, because when the edge points are detected, also many false positives are detected. In order not to move these points towards the position of the real edges, we established the displacement parameter, which tells us that, if a point detected as an edge needs to move more than x of the average distance between points h, then it is probably that it is a point that does not belong to the edge, which indicates that is a false positive.

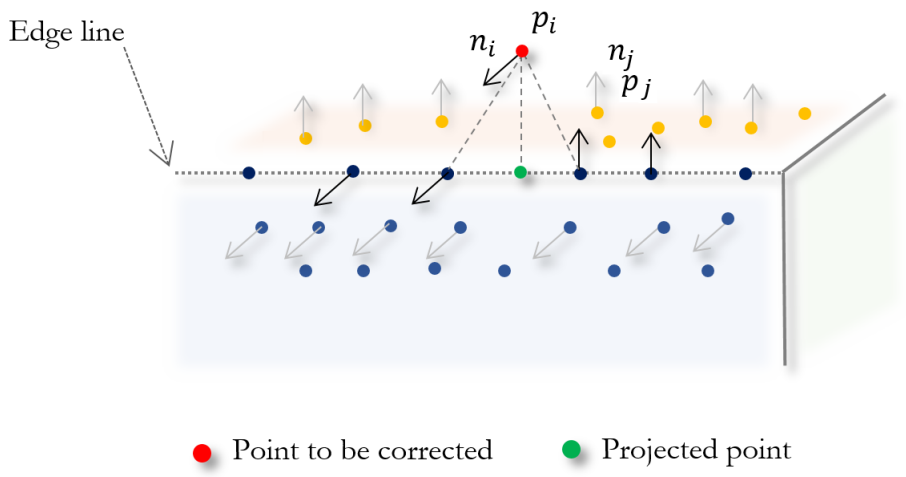
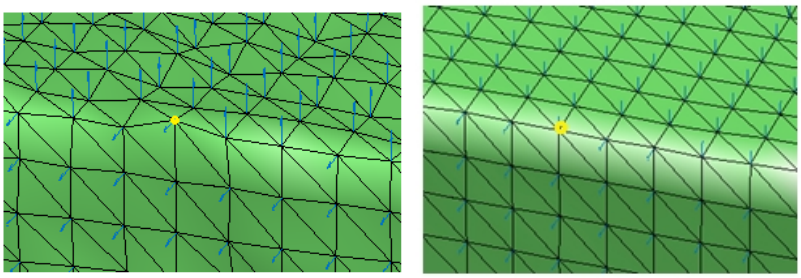


Figure 4.3: Edge points correction

Figure 4.4 shows how the proposed method corrects the point position (yellow point) according to 4.19, projecting it into the edge line.



(a) (b)

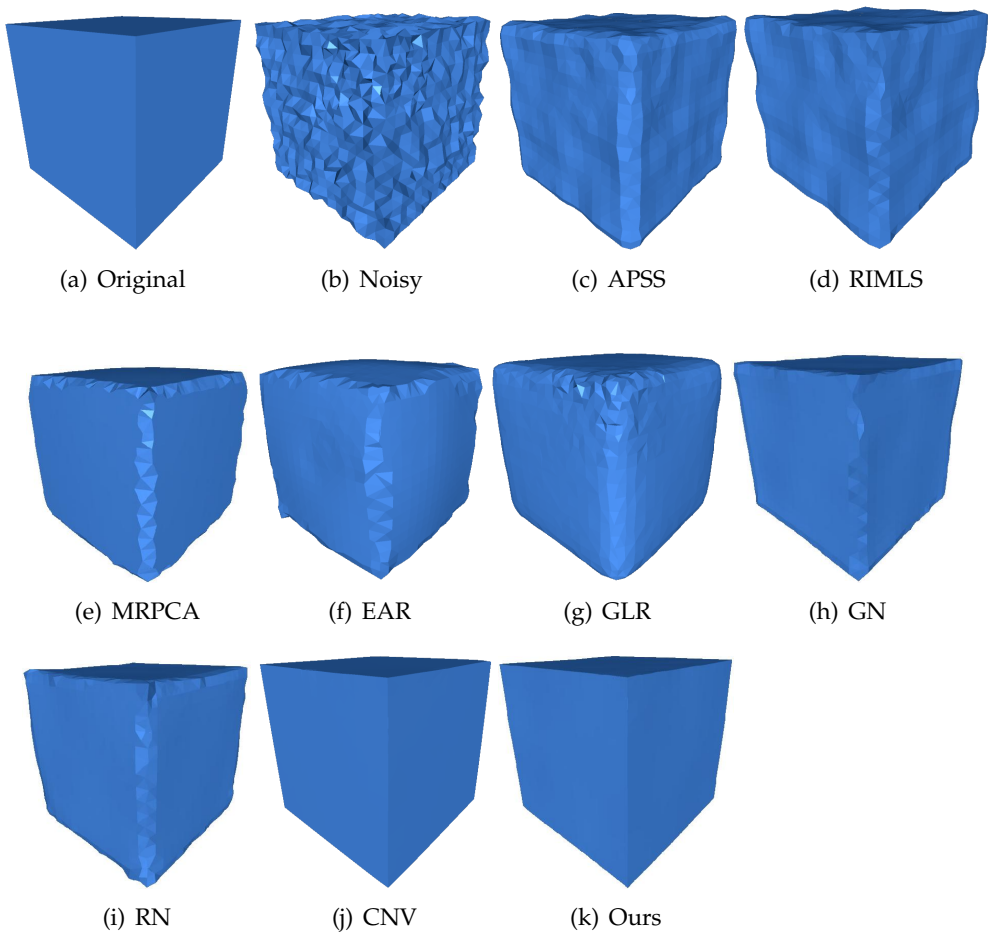
Figure 4.4: Edge artifacts, (a) the yellow point is not aligned with the edge line (b) the yellow point is corrected using the Equation (4.15)

4.5 | Experimental Results and Discussion

The proposed method was implemented in MATLAB and run on a laptop with Intel Core i7-2670QM CPU, 2.20 GHz processor, and 8GB RAM. We test the method using several point clouds with sharp features and smooth surfaces including irregular sampling. Also, synthetic and real scanned noisy point clouds are used to validate our method. The synthetic models are contaminated with Gaussian noise and impulsive noise along the normal directions or random directions. Different levels of Gaussian noise with zero mean and standard deviation σ is applied to the models; the standard deviation is proportional to the average distance between the points of the ground-truth point clouds. The noise of raw scanned data is natural. We compare our method with eight state-of-the-art denoising approaches including two MLS-based methods, APSS (GG07) and RIMLS (OGG09); one LOP based method EAR (HWG$^+$13); one sparsity-based method MRPCA (MC17); one graph-based method GLR (ZCN$^+$20); and three normal smoothing-based methods, CNV (YRS$^+$18), RN (ZLX$^+$18) and GN (ZLW$^+$17). Methods APSS and RIMLS are implemented in MeshLab software. The GLR code and EAR software are provided by the authors, as are the results of the MRPCA, CNV, RN and GN methods.

4.5.1 | Parameters Selection and Tuning

Like other previous point cloud denoising methods, we need to set the parameters correctly to produce the best quality results. Our method presents seven parameters: the sparsity parameter λ, the height sensitivity σ_h, the distance action range σ_d, the bound displacement ρ , the low bound δ, the radius of neighborhood r and the total number of iterations k.



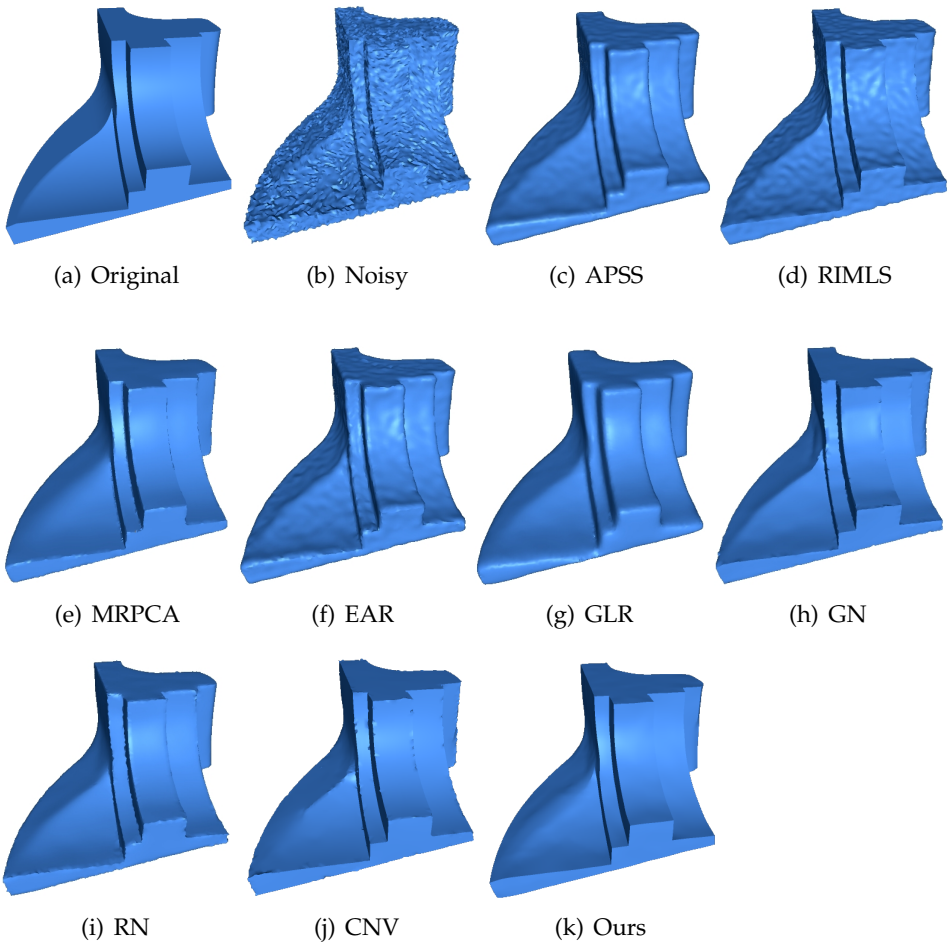
(a) Original (b) Noisy (c) APSS (d) RIMLS

(e) MRPCA (f) EAR (g) GLR (h) GN

(i) RN (j) CNV (k) Ours

Figure 4.5: The Cube model with non-uniform distribution of points, corrupted by Gaussian noise ($\sigma = 0.3$h) along all directions, where h is the average distance between the points of the point cloud. It can be seen that the proposed method can preserve sharp features effectively compare to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

The sparsity regularization parameter λ, depends on the desired gradient sparsity level and will affect the reconstruction of sharp features and the smoothness of the point cloud. A larger λ would yield a smoother result. To determine the optimal λ, we run the proposed method varying the λ values into the range [0.05, 0.5] and calculate the MSE error for each output, for several models and we found that values around 0.2 produce the minimum error. We set $\lambda = 0.2$ for all the testing point set used in the experiments. The displacement ρ is fixed throughout all experiments with $\rho = 0.7$.
The parameter δ is fixed throughout all experiments with $\delta = 10^{-4}$.



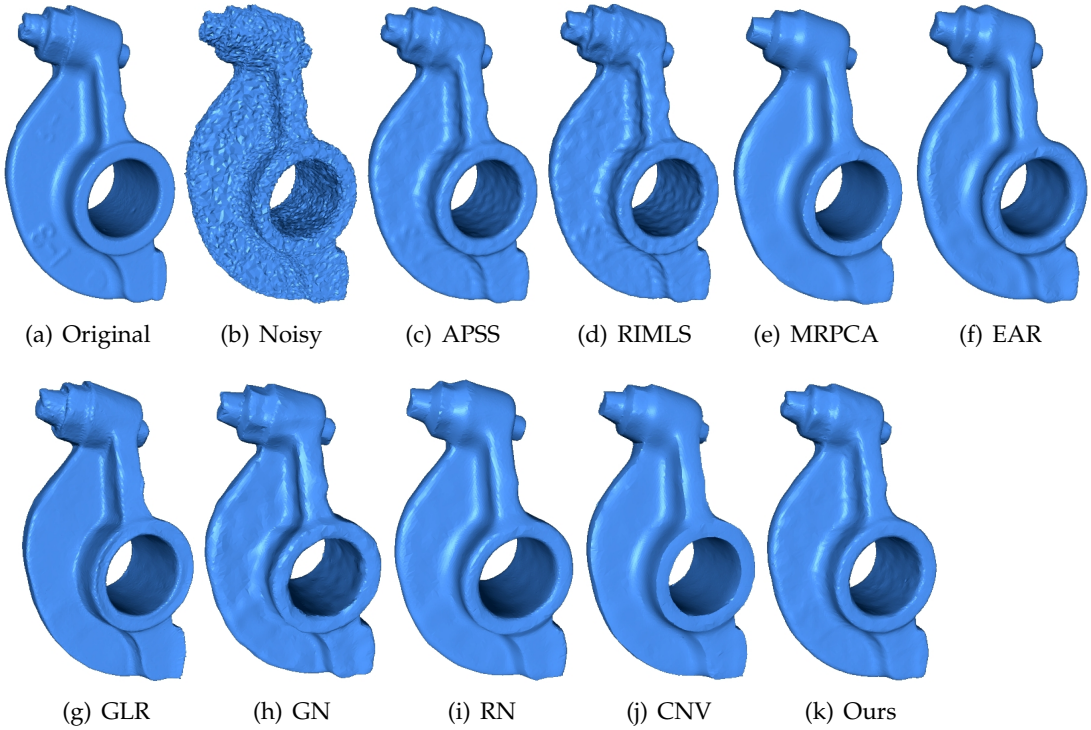
(a) Original (b) Noisy (c) APSS (d) RIMLS

(e) MRPCA (f) EAR (g) GLR (h) GN

(i) RN (j) CNV (k) Ours

Figure 4.6: The Fandisk model with non-uniform distribution of points, corrupted by Gaussian noise ($\sigma = 0.28$h). We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

The parameter h is the average spacing between the points. We compute the value of h, taking the six nearest neighbors to each point. The distance action range σ_d, and the height sensitivity σ_h are user-defined values times the parameter h. In all the experiments, the radius r of the neighborhood was set to σ_d. i.e. $r = \sigma_d$; a small value of σ_d leads to faster computation because of the neighborhood $N_g(\mathbf{p}_i)$ is small, and large values may cross sharp features and over smooth the results. Alternatively, $r = \sigma_d$ can be chosen as a function of the local point density. In the results of the experiments, we chose the values of this parameter constant, tuned to achieve visually appealing results. The height sensitivity σ_h, control the outliers in the point cloud; for small values



(a) Original (b) Noisy (c) APSS (d) RIMLS (e) MRPCA (f) EAR

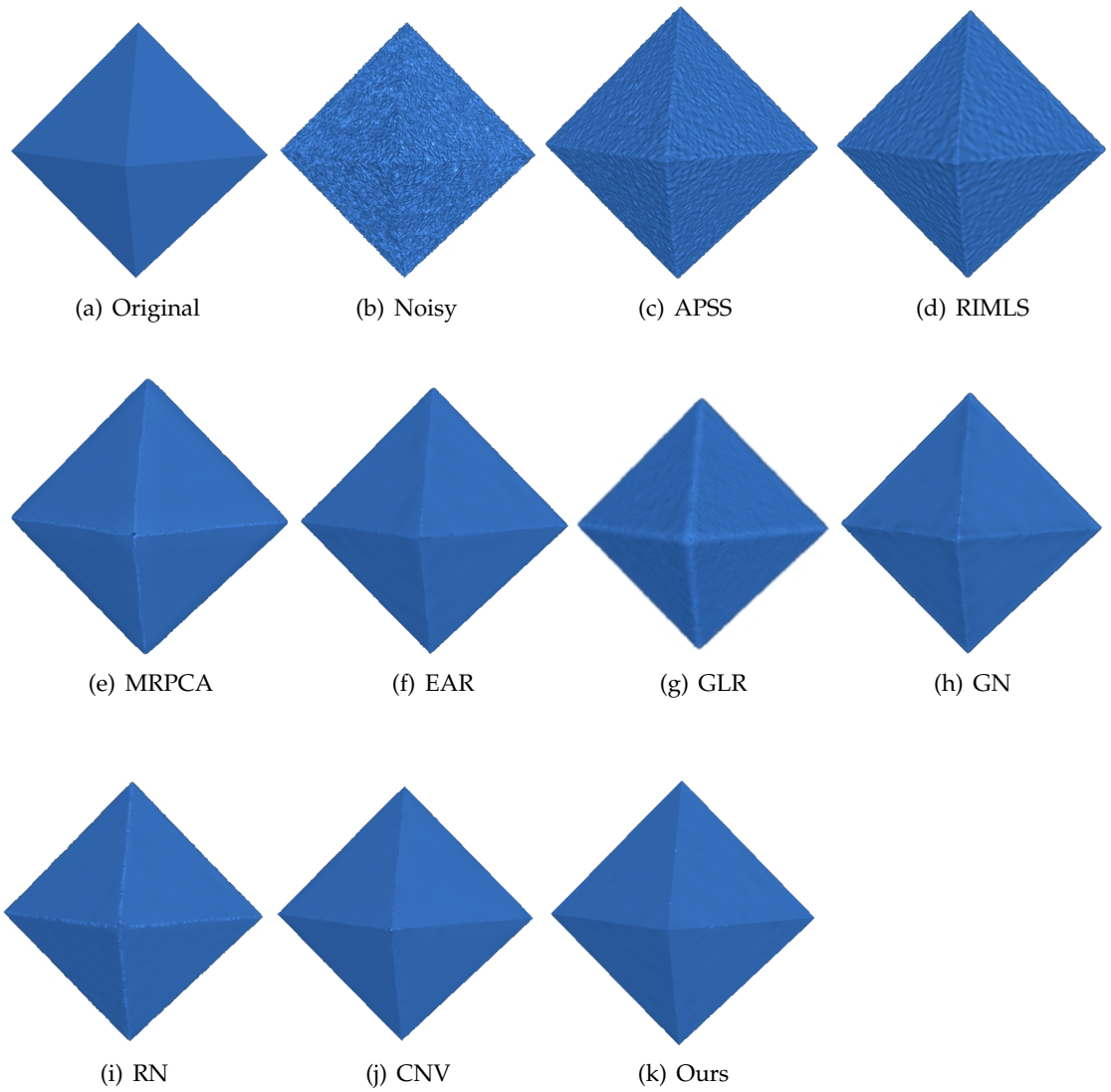
(g) GLR (h) GN (i) RN (j) CNV (k) Ours

Figure 4.7: The Rocker arm model corrupted by Gaussian noise ($\sigma = 0.3$h), in the normal direction. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

of σ_h features of the models are preserved, and for large values, only salient features are preserved.

The values of σ_h and σ_d depend on the level of noise. The bigger the noise level, the larger the value of these parameters should be chosen. We use $\sigma_d \in \{1.5h, 2h, 3h, 4h\}$ and $\sigma_h \in \{0.5h, 0.7h, 0.9h\}$ for synthetic data and $\sigma_d \in \{1.5h, 2h\}$ and $\sigma_h \in \{0.1h, 0.2h, 0.3h\}$ for real scanned point clouds. The difference of parameter values between synthetic and real models is because the level of noise in real models is lower than synthetic models. The number of iterations k for the best results was set $k \in \{10, 16, 20, 50\}$. At last, there are only three parameters for our algorithm to tune the results (σ_h, σ_d, k).

Regarding the parameters (σ_h, σ_d, k), we established a range of values, which we found as follow: we ran the proposed method with several models, and leave fixed two of the parameters and the third is free to vary, then the same procedure was done with the other parameters and in this way, we found the best values for each model.



(a) Original (b) Noisy (c) APSS (d) RIMLS

(e) MRPCA (f) EAR (g) GLR (h) GN

(i) RN (j) CNV (k) Ours

Figure 4.8: The Octahedron model corrupted by Gaussian noise ($\sigma = 0.3$h), in the normal direction. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

In our comparison experiments, we use the following parameter set for the eight selected state-of-the-art methods. For the methods (MC17), (ZLW$^+$17) and (ZLX$^+$18), we mentioned "Default" in the parameter Table 4.1, because the corresponding smooth models are provided by the authors; (GG07) = (sale, # of iterations, α); (OGG09)=(σ_r, σ_n); (HWG$^+$13)=(Default values); (YRS$^+$18)=(τ, ρ, p) and (ZCN$^+$20) the parameters settings in their paper. Our method = (σ_h, σ_d, k).

Table 4.1: Parameter setting of comparative methods for different models

Methods	Cube	Fandisk	Rocker Arm	Octhaedron
EAR	Default	Default	Default	Default
APSS	(2, 45, 0.5)	(4, 15, 0)	(4, 15, 0.5)	(2, 45, 0.5)
RIMLS	(4, 0.75)	(4, 1)	(4, 1)	(4, 0.75)
MRPCA	Default	Default	Default	Default
GLR	Paper	Paper	Paper	Paper
GN	Default	Default	Default	Default
RN	Default	Default	Default	Default
CNV	(0.3, 0.95, 150)	(0.3, 0.9, 150)	(0.25, 0.9, 80)	(0.25, 0.9, 80)
Ours	(0.98h, 4h, 30)	(0.7h, 3h, 16)	(0.7h, 3h, 14)	(0.7h, 2h, 20)

4.5.2 | Quantitative Analysis

In this section, we show the behavior of the proposed method against different levels of noise and density point variability. We compared our method to other approaches using three quantitative metrics, shown in Table 4.2, Table 4.3. When the number is highlighted in bold means the best performance. To continue, we define the three metrics used in our quantitative analysis. To quantify feature preservation, we measure the orientation error between the smoothed point cloud and the ground truth. Mean angular deviation (MAD) is defined to measure the orientation error:

$$\mathbf{MAD} = \frac{1}{n}\sum_{i=0}^{n-1} < (\overline{\mathbf{n}}_i, \widehat{\mathbf{n}}_i) \tag{4.20}$$

Where $\overline{\mathbf{n}}_i$ and $\widehat{\mathbf{n}}_i$ are the point normals corresponding to the ground truth and the smoothed point cloud, respectively.

To quantify the closeness between the ground truth model and the smoothing model, we use the mean-squared-error (MSE) metric, which measures the average of the squared Euclidean distances between the ground truth points and their closest denoised points, and vice versa between the denoised points and their closest ground truth points. Finally the average between two measures give the MSE.

If the ground truth model and the smoothed model are $\mathbf{P_1} = \{\mathbf{p}_i\}_{\mathrm{i}=1,..,\mathrm{n}_1}$ and $\mathbf{P_2} = \{\mathbf{q}_j\}_{\mathrm{j}=1,..,\mathrm{n}_2}$, the point clouds can be of different sizes, i.e., $n_1 \neq n_2$. The MSE is defined

as follows.

$$\mathbf{MSE} = \frac{1}{2n_1} \sum_{\mathbf{p}_i \in \mathbf{P_1}} \min_{\mathbf{q}_j \in \mathbf{P_2}} \left\| \mathbf{p}_i - \mathbf{q}_j \right\|_2^2 + \frac{1}{2n_2} \sum_{\mathbf{q}_i \in \mathbf{P_2}} \min_{\mathbf{p}_j \in \mathbf{P_1}} \left\| \mathbf{q}_j - \mathbf{p}_i \right\|_2^2 \tag{4.21}$$

The signal-to-noise ratio (SNR) is measure in dB and is defined as follows.

$$\mathbf{SNR} = 10\, log \frac{1/n_2 \sum_{\mathbf{q}_i \in \mathbf{P_2}} \left\| \mathbf{q}_j \right\|_2^2}{\mathbf{MSE}} \tag{4.22}$$

Table 4.2 shows the comparison between our method and the eight competing state-of-the-art methods. We can observe for the Cube model, how our method can preserve the sharp features (see Figure 4.5), it is confirmed by the low MAD value. In Table 4.2, we can also see that the MSE and SNR metrics are the lowest values compared to all state-of-the-art methods. For the Fandisk model, the proposed method reaches the second position in all three metrics, only outperformed by the GN method in the MAD metric and by the RN method in MSE and SNR metric. However, in Figure 4.6, we observe how the proposed method can reconstruct better sharp features compared to the competing methods. For the Rocker Arm, our method outperforms the state-of-the-art methods in the MAD metric, but with MSE and SNR, our method only is outperformed by the APSS method. However, in Figure 4.7, it can be seen that the proposed method keeps details that have been lost in the other methods. For the Octahedron model, our method outperforms in all metrics the state-of-the-art methods.

Table 4.2: Comparison of the different methods evaluated with three error metrics for each one of 3D objects (Cube, FanDisk, Rocker Arm, Octahedron). Bold highlights indicate the best results.

		Methods								
		EAR	**APSS**	**RIMLS**	**MRPCA**	**GLR**	**GN**	**RN**	**CNV**	**Ours**
Cube	**MAD**	4.3634	5.5588	4.4552	4.4341	6.2429	3.4878	4.4531	2.8668	**2.6985**
	MSE	0.0183	0.0125	0.0117	0.0273	0.0336	0.0330	0.0352	0.0064	**0.0046**
	SNR(dB)	39.438	42.613	43.272	35.749	33.936	33.920	33.305	48.394	**51.418**
Fandisk	**MAD**	4.4038	5.0465	5.6874	3.7932	7.7937	**2.9186**	3.1585	3.5273	2.9691
	MSE	0.0073	0.0057	0.0060	0.0067	0.0257	0.0060	**0.0038**	0.0108	0.0039
	SNR(dB)	45.105	46.168	45.965	45.525	39.653	45.963	**48.006**	43.410	47.833
Rocker	**MAD**	5.9647	4.8825	4.9493	6.0163	7.1012	7.7694	5.7894	7.1894	**4.2611**
	MSE	0.1392	**0.0468**	0.0717	0.1345	0.2554	0.6141	0.5873	0.1651	0.0665
	SNR(dB)	36.116	**40.830**	38.988	36.234	33.450	29.717	29.885	35.340	39.300
Octha	**MAD**	1.8779	3.9838	4.8495	4.9541	5.2574	1.3606	1.3776	1.0415	**1.0196**
	MSE	9.5E-4	0.0014	0.0011	0.0014	0.0016	0.0074	0.0074	7.0E-4	**5.6E-4**
	SNR(dB)	54.384	51.007	52.846	51.006	50.008	55.731	55.631	57.057	**58.931**

Table 4.3, We are comparing with four state-of-the-art methods, with three different levels of noise, i.e., $\sigma = 0.1$h, $\sigma = 0.2$h, and $\sigma = 0.3$h. We can see that the MAD grows as the noise increases. Thus, if the noise level is high, the orientation error will be larger compared to the lower noise level. Table 4.3 shows that the proposed method achieves the best results for the MAD metric, with all levels of noise, but for the level of noise $\sigma = 0.1$h, RIMLS reaches the best MSE and SNR. However, for the level of noise $\sigma = 0.2$h and $\sigma = 0.3$h, our method outperforms the competing methods. In all the experiments, the proposed method achieves the best results on average in all three metrics and all noise levels.

4.5.3 | Visual comparison

For visual comparison, we use the ball pivoting algorithm (BPA) (BMR$^+$99) to reconstruct the mesh from the smoothed point cloud. The point clouds are contaminated with Gaussian noise along random directions and normal directions, and impulsive noise in a random direction.

Table 4.3: The results of the error metrics of different compared methods for Block and Trim-star objects varying the noise levels.

		Methods	EAR	APSS	RIMLS	GLR	Ours
$\sigma = 0.1$**h**	Block	**MAD**	3.8083	4.2386	3.1723	2.9909	**2.9232**
		MSE	0.0693	0.0641	**0.0466**	0.0469	0.0477
		SNR(dB)	34.518	34.842	**36.238**	36.199	36.124
	Trim-star	**MAD**	5.0802	4.7813	4.1111	7.0203	**3.6042**
		MSE	0.0634	**0.0324**	0.0408	0.1105	0.0370
		SNR(dB)	29.572	**32.459**	31.472	27.120	31.855
$\sigma = 0.2$**h**	Block	**MAD**	6.2682	8.2802	4.1979	4.9876	**3.5737**
		MSE	0.1339	0.1191	0.0688	0.0911	**0.0551**
		SNR(dB)	31.676	32.149	34.552	33.310	**35.492**
	Trim-star	**MAD**	6.9177	7.0610	5.3676	8.3487	**4.8816**
		MSE	0.1068	0.0525	0.0573	0.1456	**0.0522**
		SNR(dB)	27.341	30.353	30.011	25.914	**30.363**
$\sigma = 0.3$**h**	Block	**MAD**	7.6707	11.629	4.6145	9.2070	**4.4352**
		MSE	0.1573	0.1640	0.1029	0.2735	**0.0784**
		SNR(dB)	30.980	30.753	32.816	28.488	**33.955**
	Trim-star	**MAD**	8.1821	10.929	6.7664	10.495	**5.8392**
		MSE	0.0695	0.0995	0.0822	0.1828	**0.0632**
		SNR(dB)	29.158	27.576	28.465	24.915	**29.608**

4.5.3.1 | Irregular Point Clouds

The Cube (Figure 4.5), the Fandisk (Figure 4.6), and the Rocker Arm (Figure 4.7) models, have non-uniform density points corrupted by Gaussian noise in a random direction ($\sigma = 0.28$h, $\sigma = 0.3$h, and $\sigma = 0.3$h, respectively). Figure 4.5 shows that our method can preserve sharp features and in the flat areas do not produce bumps features like APSS (GG07), RIMLS (OGG09), and EAR (HWG$^+$13) methods. GLR (ZCN$^+$20), MRPCA (MC17), RN (ZLX$^+$18) and GN (ZLW$^+$17), clean the noise effectively over flat regions, but there is over smoothing in the corners and edges. CNV (YRS$^+$18), properly reconstructs the sharp features and cleans the flat areas but small artifacts do appear in some corners. In Figure 4.6, our method can reconstruct sharp features and shallow features. APSS smooth around the sharp features and does not remove the noise correctly. RIMLS and EAR, preserve sharp features but produce some bump features in the resulting models. MRPCA remove the noise and preserve some sharp features, but smooth shallow areas around flat regions. GLR removes noise effectively but over smoothes the sharp features and shallow areas. GN, RN, and CVN produce a similar output to our method, but there are some artifacts on the borders and in corners. In Figure 4.7, the

Rocker Arm model shows how our method smooths the noise and at the same time can preserve sharp features and little details, that are lost in the competing algorithms.

4.5.3.2 | Dense Regular Point Clouds

In the octahedron model (Figure 4.8), our method reduces the noise while preserving the sharp features compare to APSS, RIMLS, GLR, and EAR models. GN, RN, and CNV have similar behavior that our method, but our method produces better quantitative metrics.

4.5.3.3 | Different Levels of Noise

Figure 4.9 and 4.10 shows a comparison with APSS, RIMLS, EAR, and GLR, with different levels of noise. Our method can remove the noise from the input point cloud effectively while preserving sharp features and smooth the surfaces. The block model is planar everywhere except at sharp features, and the trim-star model is smooth everywhere except at sharp features.

4.5.3.4 | Impulsive Noise

Figure 4.11 shows the results of handling a corrupted point cloud adding an impulsive noise of $\sigma = 0.5$h along the normal direction. The Twelve model has been smoothed by the proposed method and its edges have been preserved. RIMLS, APSS, and EAR methods are not able to smooth the noise properly and reconstruct the edges.

4.5.3.5 | Real 3D Scanned Data

We also compared these approaches on real scanned data. Figures 4.12, 4.13 and 4.14, show the results of different methods applied to raw data scan, In Figure 4.12 from the Rabbit model, it may seem like our method effectively removes the noise while preserving features, when compared to APSS, RIMLS, and EAR. GLR and RN preserve features, but some fine details as the eye and grooves in the ear have been lost. GN and CNV preserve more detail than any of the other methods but they lost details in the eye and nose. MRPCA and the proposed method produce very similar results. In Figure 4.13, shows the Ball Joint medical data. We can see that our method removes the noise, while details and sharp features are preserved and the spherical shape is effectively smoothed. The APSS and RIMLS methods are not able to smooth the noise properly, and the resulting surfaces present bumps. MRPCA, GN, RN, and GLR effectively remove the noise component but smooth the sharp features. The EAR and CVN methods produce similar

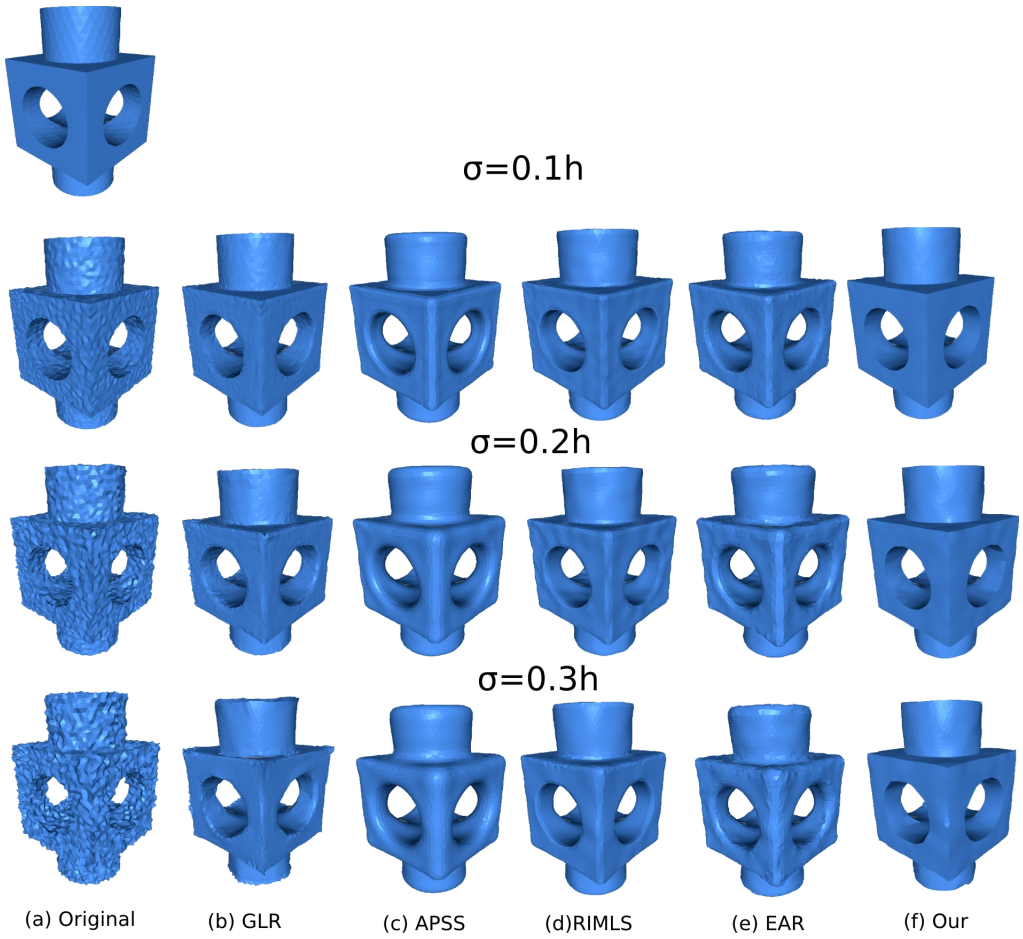


Figure 4.9: The Block model corrupted by Gaussian noise (σ = 0.1h, 0.2h, 0.3h), in the normal direction. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

results preserving the sharp features and smoothing the surfaces.

Irregular Surface Sampling

Figure 4.14 The Gargoyle model is a point cloud with irregular sampling points. The model is a raw data scanning with natural noise. Our method removes noise and keep

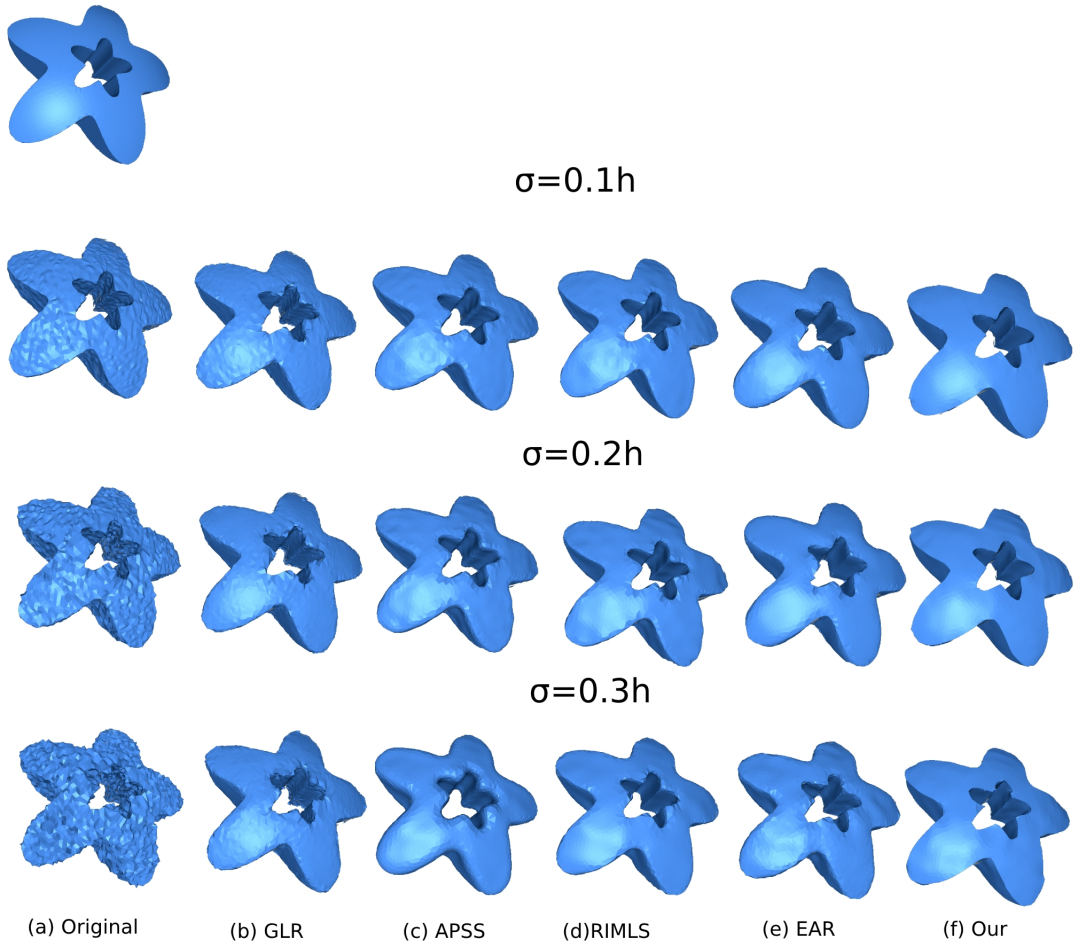
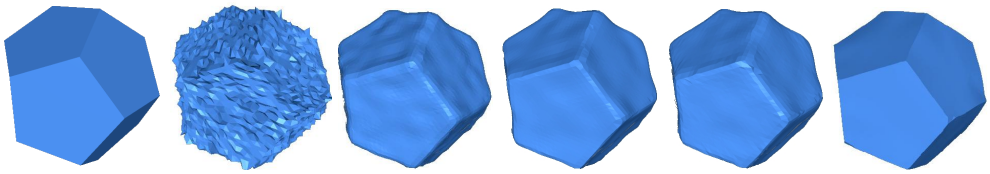


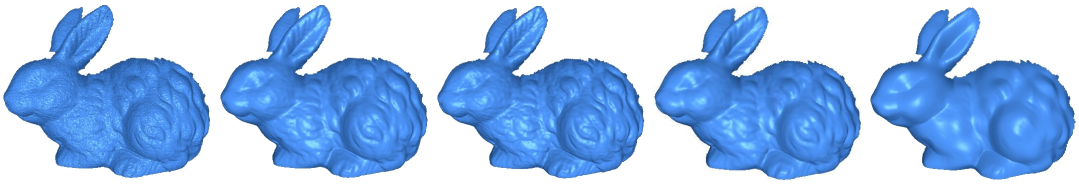
Figure 4.10: The Trim-Star model corrupted by Gaussian noise (σ = 0.1h, 0.2h, 0.3h), in the normal direction. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

fine details and sharp features; compared to the competing methods, APSS, RIMLS, GLR, EAR, and CNV.

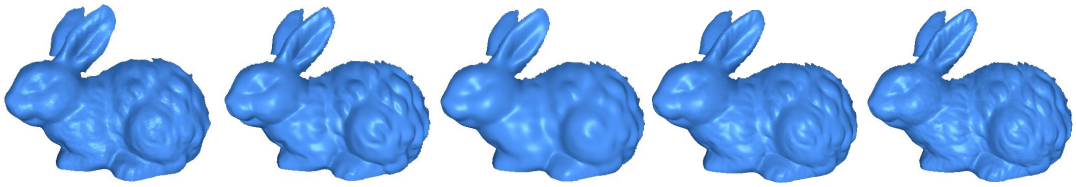


(a) Original (b) Noisy (c) APSS (d) RIMLS (e) EAR (f) Our

Figure 4.11: The Twelve model corrupted by Impulsive noise ($\sigma = 0.5$h), in the normal direction. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.



(a) Original (b) APSS (c) RIMLS (d) MRPCA (e) EAR

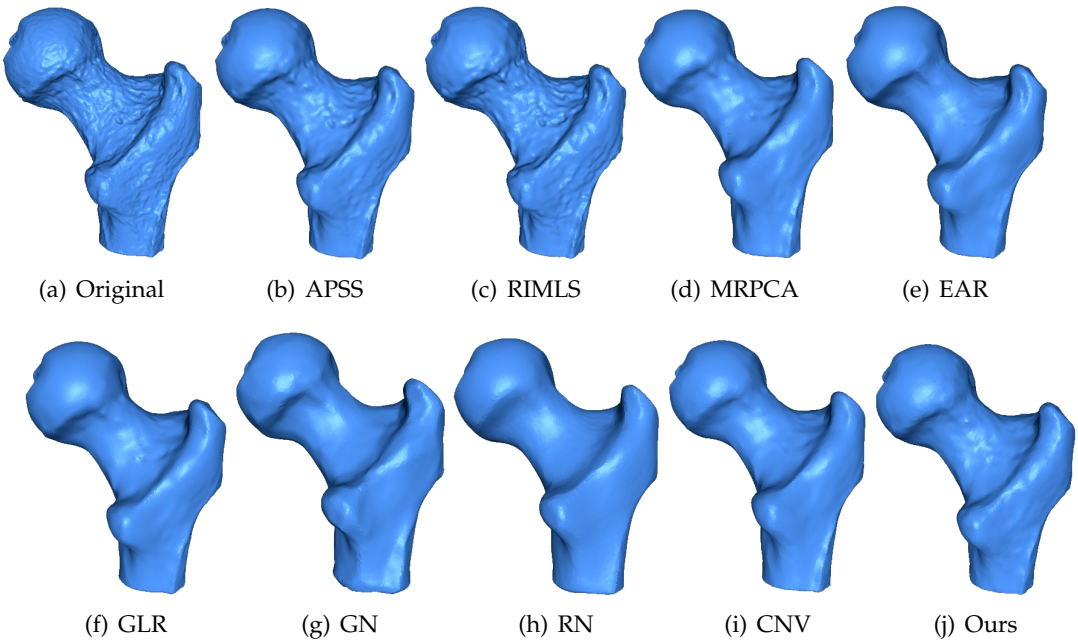


(f) GLR (g) GN (h) RN (i) CNV (j) Ours

Figure 4.12: The Rabbit model, with natural noise. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.

4.5.3.6 | Convergence

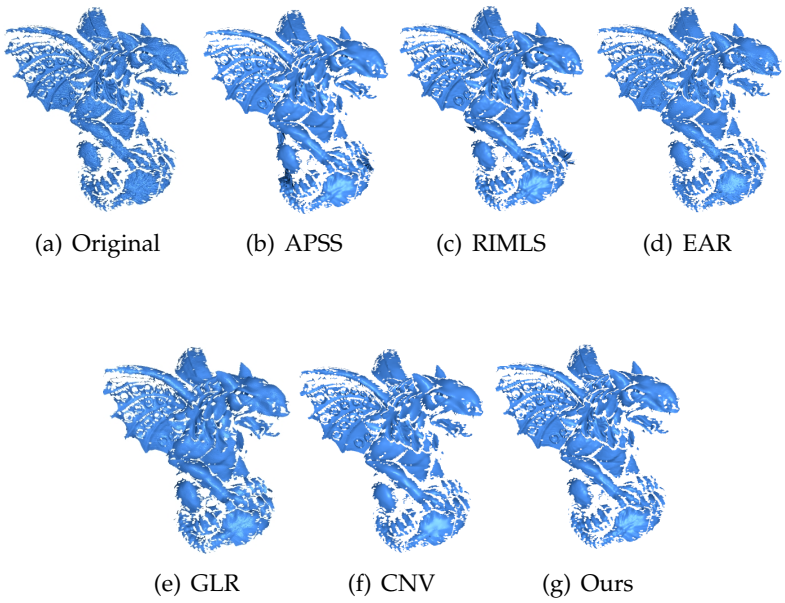
Figure 4.15 shows the convergence rate of the proposed method. We can appreciate how our approach has a better convergence rate compared with the GLR (ZCN$^+$20) method. We observe that after eight iterations, the proposed method is almost stable. Combining the L_1-median and L_1-norm in the optimization process not only helps to preserve features but also improves the convergence rate of the method.



(a) Original (b) APSS (c) RIMLS (d) MRPCA (e) EAR

(f) GLR (g) GN (h) RN (i) CNV (j) Ours

Figure 4.13: The Ball Joint model, with with natural noise. We can see that the proposed method is able to preserve sharp features effectively when compared to the state-of-the-art methods. The surface is reconstructed using the ball pivoting algorithm.



(a) Original (b) APSS (c) RIMLS (d) EAR

(e) GLR (f) CNV (g) Ours

Figure 4.14: The Gargoyle model, with natural noise. The surface is reconstructed using the ball pivoting algorithm.

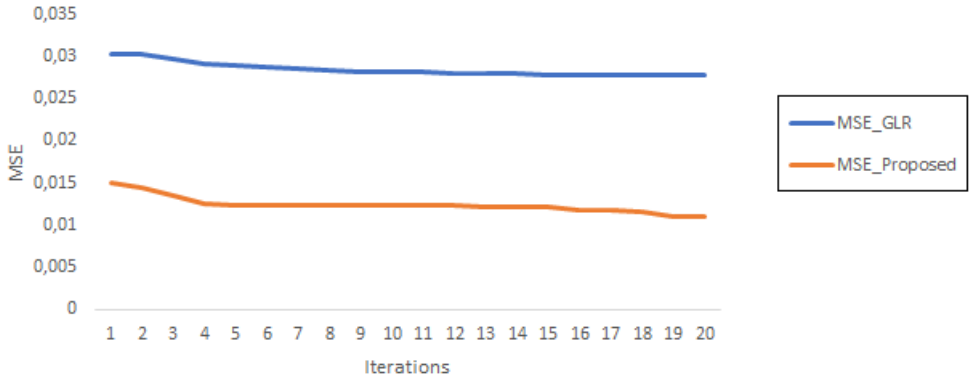
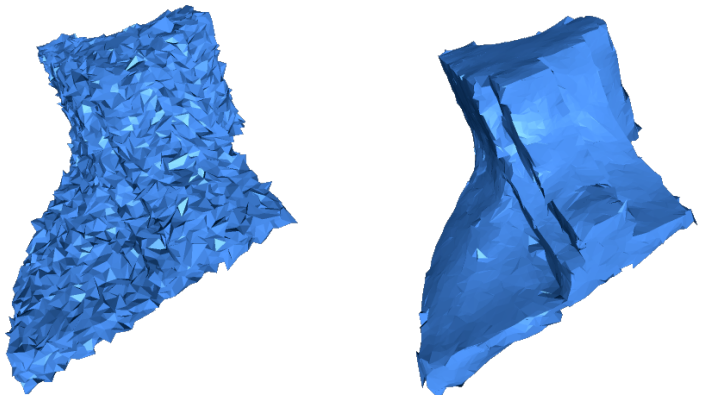


Figure 4.15: Convergence plot comparison between the proposed method and the GLR (ZCN$^+$20) method. The MSE error metric is computed for the Fandisk model. The convergence rate of the proposed method is better than GLR method.



(a) Original (b) Our

Figure 4.16: The Fandisk model. Contaminated with Gaussian noise ($\sigma = 0.7$h) It can be seen that the proposed method cannot preserve sharp features in point clouds with a high level of noise.

4.6 | Conclusion

In this chapter, we propose combining the L_1-median filter and the L_1-norm regularization for a point cloud denoising algorithm that preserves the sharp features. The algorithm uses double sparsity modeling both in the fitting term and in the regularization term. The L_1-median is insensitive to outliers and noise, while the L_1-norm, preserves the sharp features and smooths the surface. The combined L_1-median and L_1-norm cost function were optimized with an alternating minimization strategy using proximal gradient and a descendent iterative schema, allowing the implementation of a

simple algorithm. The proposed method can handle models contaminated with Gaussian and impulse noise. High noise level could produce erroneous results as they affect the normals estimation (see Figure 4.16). Another issue is the concern with irregular point sampling models. While the irregular sampling remains low, the output of our algorithm produces good results; but when the point cloud density is highly irregular, the output quality decreases. To recover the sharp features, we introduce a border correction procedure that helps to correct edges and corners, preserving the models' original sharp features.

Experimental results reveal that our proposal can preserve sharp features when compared to previous point cloud denoising methods, and the algorithm is robust in denoising both synthetic and raw point scans.

5

Saliency Detection

The human visual system (HVS) can process large quantities of visual information instantly. Visual saliency perception is the process of locating and identifying regions with a high degree of saliency from a visual standpoint. Mesh saliency detection has been studied extensively in recent years, but few studies have focused on 3D point cloud saliency detection. The estimation of visual saliency is important for computer graphics tasks such as simplification, segmentation, shape matching, and resizing. In this chapter, we present a local method for the direct detection of saliency on unorganized point clouds. First, our method computes a set of overlapping neighborhoods and estimates a descriptor vector for each point inside it. Then, the descriptor vectors are used as a natural dictionary to apply a sparse coding process. Finally, we estimate a saliency map of the point neighborhoods based on the Minimum Description Length (MDL) principle. Experiment results show that the proposed method achieves similar results to those from the literature review and, in some cases, even improves on them. It captures the geometry of the point clouds without using any topological information and achieves acceptable performance. The effectiveness and robustness of our approach are shown by comparing it to previous studies in the literature review.

5.1 | Introduction

The human visual system (HVS) can process large amounts of visual information instantly and can locate objects of interest and distinguish them from complex background scenes. Research shows that HVS pays more attention to infrequent features and suppresses repetitive ones (JZLZ14) (Wol94). Visual saliency plays a vital role in the process in which HSV identifies scenes and detects objects. It is also concerned with the way the biological system perceives the environment. For example, every time we look at

a specific place, we pay more attention to regions distinct from the surrounding area. Visual saliency is an active research field in areas such as psychology (Wol94), neuroscience (KP99), computer vision (SCMP14) (KAD16) and computer graphics (KVJG10) (LDS+16). There are many computational methods for simulating the HVS and visual saliency, but it nonetheless remains an unexplored field because of the difficulty of designing algorithms to simulate this process (Kim et al., 2010). As for computer graphics, while the concept of visual saliency has been explored for mesh saliency (KVJG10), (JS17; KVJG10; LKF16; LML16; LTC+16; NCL15; SLME16; SLMR14; TCL+15; WLL+15; WSZL13; ZLW+16), few studies have explored visual saliency in point clouds (GWX17) (TKD15) (SLT13) (AJ10). Visual saliency is an important topic for 3D surface study and has important applications in 3D geometry processing such as resizing (JZLZ14), simplification (YWC+12) (AWK16), smoothing (DBBB14), segmentation (JWQ18), shape matching and retrieval (TKD16b) (GCO06), 3D printing (WCT+15), and so forth. Because of the develop of 3D data scanning technology, current scanners generate thousands of points for every scanned object. Therefore, for rendering, point clouds have become an alternative to triangular meshes. A common way to process a point cloud is to reconstruct the surface using methods such as triangular mesh, NURBS representation, and Radial Basis Functions. However, because of many points, different sampling densities, and the inherent noise produced by the scanning process, reconstruction is an expensive and challenging computational task. For these reasons, it is necessary to develop geometry processing algorithms that operate directly on the point sets. Applying existing saliency detection techniques to point clouds is not a trivial task; this is due to the absence of topological information, which is not a problem for mesh-based methods. The method we propose here is inspired by Li et al. (LZX+09), and we made it fit for general use supported by the Minimum Description Length (MDL) principle. In this way, we use MDL as a criterion for distinguishing regions in the point clouds.

5.1.1 | Contribution

The main contributions of this chapter are:

1. A method for saliency detection without topological information using only the raw point sets.

2. The use of the MDL principle for defining saliency measurements, extended to sparse coding representation to get saliency maps of point clouds.

Experimental results show that the proposed algorithm improves in the capture of the geometry of the point clouds using no topological information and achieves an acceptable performance when compared to previous approaches.

5.2 | Related work

Visual saliency detection has its origin in computer vision, specifically in 2D images. Inspired by this research, visual saliency detection has been applied successfully to 3D meshes and point clouds. In recent years, much research has tried to develop methods for visual saliency on 3D surfaces (LTC$^+$16) (TCL$^+$15) (Tao et al., 2015) (SLMR14) (JS17) (GGCJ17) (TKD15) (SLT13) (LVJ05).

Early advances in mesh saliency used 2D projections for 3D applications. While such works often ignored (or could not exploit) the importance of depth in the human perception, they set a basis for further research by highlighting the importance of human perception in image analysis (Song et al., 2016). Some of the first researchers to exploit saliency for 3D mesh processing were Lee et al. (LVJ05). The authors introduced the concept of mesh saliency and computed it using a Gaussian-weighted center-surround mechanism. Results were used to implement algorithms for mesh simplification and point-of-view selection.

Wu et al. (WSZL13) presented an approach based on the principles of local contrast and global rarity. This method has applications for mesh smoothing, simplification, and sampling. Leifman et al. (LST12) propose a method for detecting a region of interest on surfaces where they capture the distinctness of the vertex descriptor, characterizing the local geometry around it. The method is applied to viewpoint selection and shape similarity. Tao et al. (TCL$^+$15) put forward a mesh saliency detection approach that reached state-of-the-art performance even when handling noisy models.

The manifold ranking was employed in a descriptor space to imitate human attention. Then descriptor vectors were built for each over-segmented patch on the mesh, using Zernike coefficients and center-surround operators. Afterward, background patches were selected as queries to transfer saliency, helping the method to handle noise better. An approach for mesh saliency detection based on a Markov Chain is proposed by Liu (LTC$^+$16); the input mesh was partitioned into segments using Neuts algorithms and then over-segmented into patches using Zernike coefficients. Instead of employing center-surround operators, background patches were selected by determining feature variance to separate the insignificant regions. Limper et al. (LKF16) applied the concept of Shannon entropy, which is defined as the expected information value within a

message, for 3D mesh processing. Here, the curvature is established to be the primary source of information in the mesh.

Song et al. (SLME16) argued for the benefits of using not only local but also global criteria for mesh saliency detection. Their method comprised computing local saliency features while considering the later computation of global saliency using a statistic Laplacian-based algorithm, which captures salient features at multiple scales. Recently, point saliency detection-based methods have been introduced. Shtrom et al. (SLT13) propose a multi-level approach to find distinct points, basing their approach on the context-aware method for image processing. Tasse et al. (TKD15) propose a cluster-based approach; the method decomposed the point cloud into the small clusters using an adaptive fuzzy clustering algorithm and is applied in the detection of key-points. Guo et al. (GGCJ17) propose a saliency detection method based on a covariance descriptor to capture the local geometry information. They use a sigma set descriptor to transform the covariance descriptor from a Riemannian space to a Euclidian space to facilitate apply Principal Component Analysis for the inner structure analysis to whether or not a point is salient.

5.3 | Point Cloud Saliency Detection

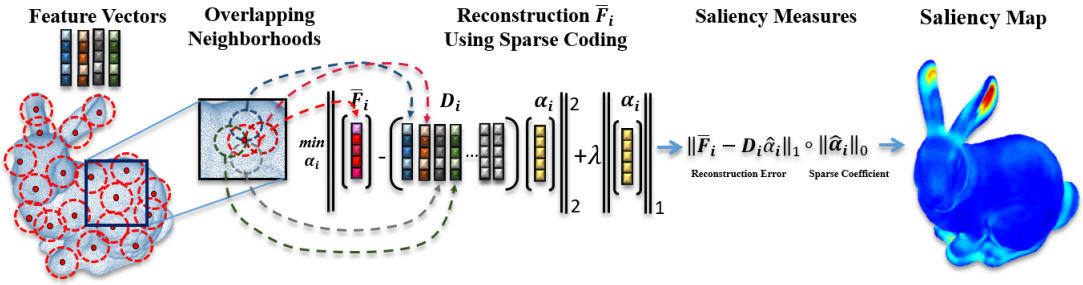


Figure 5.1: Processing pipeline of our sparse coding-based saliency detection method.

In Figure 5.1 we show the saliency detection framework based on sparse coding. The input point cloud is represented in sparse form, and the saliency is estimated by a center-surround hypothesis (IKN98) (IK01), which states that a region is salient if it is distinct from its surrounding regions. Taking a point cloud as input, we estimate a neighborhood with radius r for each point in the cloud, then select some of its boundary points. We estimate a neighborhood with radius r around each selected point, allowing overlapping between neighborhoods to capture the structure of the cloud. Then, for

each point in the neighborhood, we compute a set of basic features to build a descriptor vector and finally; we estimate the mean of the feature vectors to build only one descriptor vector per neighborhood. We do the same with the neighborhoods surrounding the central one. Afterward, we construct a dictionary $\mathbf{D}$, using the feature vectors from the surrounding neighborhoods, and then a sparse coding is carried out using the feature vector of the central neighborhood and by searching for a sparse representation for it with the basis in the dictionary $\mathbf{D}$. Finally, we compute the neighborhood saliency using the MDL principle based on the sparse representation measure of vector x, and the residual of the sparse reconstruction measure. The final saliency map is computed fusing both measures.

5.3.1 | Minimum Description Length principle (MDL)

As we stated in chapter 3, the Minimum Description Length principle searches for the best model $\widehat{\mathbf{M}} \in \mathfrak{M}$ that can be used to describe x in its entirety with the shortest length. In (BT05), a method based on information theory was introduced for image saliency detection. This method measures the saliency concerning the likelihood of a patch given the patches that surround it. The method measures the self-information using the negative log-likelihood. Defining x as an image and $\mathbf{p}(\mathbf{x})$ as the probability of occurrence of the patch x, given its surrounding neighborhood patches, the saliency measure of the patch is defined as $-\mathbf{logp}(\mathbf{x})$, that is, the self-information characterizes the raw likelihood of the n-dimensional vector values given by x.

Based on (RS12), (SCMP14) (BT05), we use the MDL principle to propose a method for salience detection in point clouds based on sparse coding. The coding assignment function $\mathbf{L}(\mathbf{x}, \mathbf{M})$ can be defined in terms of probability assignment $\mathbf{P}(\mathbf{x}, \mathbf{M})$, based on the Ideal Shannon Codelength Assignment (RS12), that is, $\mathbf{L}(\mathbf{x}, \mathbf{M}) = -\mathbf{logP}(\mathbf{x}, \mathbf{M})$. Using Bayes theorem, we can establish $\mathbf{P}(\mathbf{x}, \mathbf{M})$ as $\mathbf{P}(\mathbf{x}, \mathbf{M}) = \mathbf{P}(\mathbf{x} \mid \mathbf{M})\mathbf{P}(\mathbf{M})$ and then by applying maximum a posteriori (MAP), the penalized likelihood form of the coding model is formulated thus:

$$\widehat{\mathbf{M}} = arg\min_{M \in \mathfrak{M}} -\mathbf{logP}(\mathbf{x} \mid \mathbf{M}) - \mathbf{logP}(\mathbf{M}) \tag{5.1}$$

where $-\mathbf{logP}(\mathbf{x} \mid \mathbf{M})$ is the term that describes how well the model adjusts the data x and $-\mathbf{logP}(\mathbf{M})$ defines the model complexity, model cost, or prior term.

Once $\widehat{\mathbf{M}}$ is estimated, the terms in Equation 5.1 can be interpreted as follows:

- $\mathbf{P}(\widehat{\mathbf{M}})$ is the description length or code length of the model; and

- $\mathbf{P}(\mathbf{x} \mid \widehat{\mathbf{M}})$ is the description length or code length of the data encoded using the model.

The standard Gaussian is assumed to be coding assignment function L, and assuming that x can be represented sparsely given a basis dictionary $\mathbf{D}$ and a sparse vector $\boldsymbol{\alpha}$, the probability distribution of the reconstruction error follows a Gaussian distribution $\mathbf{P}\left(\boldsymbol{x} \mid \mathbf{D}\boldsymbol{\alpha}\right) \sim \mathbf{N}\left(\mathbf{D}\boldsymbol{\alpha}, \sigma^2\right)$, with known variance σ^2. The term $-\mathbf{logP}(\mathbf{x} \mid \mathbf{M}) - \mathbf{logP}(\mathbf{M})$ in 5.1 becomes $-\mathbf{logP}\left(\boldsymbol{x} \mid \mathbf{D}\boldsymbol{\alpha}\right) = \frac{1}{2\sigma^2}\left\|\boldsymbol{x} - \boldsymbol{D\alpha}\right\|_2^2$, and the term $-\mathbf{logP}(\mathbf{M})$ in 5.1, assuming sparsity constraint, becomes $-\mathbf{logP}(\mathbf{M}) = \|\boldsymbol{\alpha}\|_0$.

The sparsity condition and the sparse reconstruction error conform to the MDL principle if we set the estimation parameter $\widehat{\mathbf{M}} = \widehat{\alpha}$ and evaluate it with the prior term $\mathbf{P}(\widehat{\alpha})$, and with the likelihood term $\mathbf{P}(\mathbf{x} \mid \widehat{\alpha})$, and so $\|\boldsymbol{\alpha}\|_0$ and $\|\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2$ are obtained respectively. We conclude that the sparsity of the vector $\widehat{\alpha}$ is the codelength of the model, and the residual of the sparse reconstruction error is the codelength of the data given the model.

The MDL principle selects the best model that produces the shortest description of the data. The more regularity that is presented in the data, the shorter the description the model will produce. If a neighborhood is equal or slightly different to its surroundings in terms of information, it means these signals (neighborhoods) are redundant and can be represented sparsely with a suitable basis dictionary, meaning that its description length (the sparsity of a vector $\widehat{\alpha}$ i.e., $\|\boldsymbol{\alpha}\|_0$) will be short, and we can conclude that this neighborhood is not salient; on the other hand, if the neighborhood is very different from its surroundings, its description length will be longer. In other words, it cannot be represented sparsely by the basis dictionary, and we can conclude that the neighborhood is salient.

If the sparse reconstruction error (i.e., $\|\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2$) produces a high residual, it means that its description length will be longer, and this implies that the neighborhood is dissimilar from its surroundings and, therefore, more salient. On the other hand, if the sparse reconstruction error produces a low residual, its description length will be short, and this implies that the neighborhood is similar to its surroundings and, consequently, less salient. Based on these saliency measurements, we describe our point cloud saliency detection method below.

5.3.2 | Sparse Coding Saliency Detection

Unlike some saliency detection methods for point clouds compiled in the literature review, our method does not estimate saliency using individual points directly; instead, a

neighborhood is used to determine its saliency.

5.3.2.1 | Feature Vector

Given the point set $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}, i = 1, 2, \ldots,$N., and a neighborhood $\mathbf{N_g}(\mathbf{p_i})$. (See Figure 5.1) To differentiate when a neighborhood is different from its surroundings; first, it is essential to characterize it by a descriptor, of which there are several which describe low-level features for each $\mathbf{p}_i$, for example, normal, curvatures, shape index, etc. In this method, the normal and Gaussian curvatures are selected; these features are rotationally invariant. A third feature, $\mathbf{d}_i$, is also selected. With the features defined, a five-dimensional feature vector $\mathbf{F}_i$ is formed for every point $\mathbf{p}_i$ of $\mathbf{N_g}(\mathbf{p_i})$, i.e., $\mathbf{F}_i = (nx_i, ny_i, nz_i, \mathbf{k}_i, \mathbf{d}_i)$, where nx_i, ny_i, nz_i are the three components of the normal vector $\mathbf{n}_i$; $\mathbf{k}_i$ is the Gaussian curvature and $\mathbf{d}_i$ is the fifth component coordinate of $\mathbf{F}_i$, which will be defined below. It is necessary to have a single descriptor vector for each neighborhood, so the mean of the characteristic vectors belonging to the neighborhood is estimated as follows:

$$\overline{\mathbf{F}}_i = \frac{1}{|N_g(\mathbf{p}_i)|} \sum_{j=1}^{k_i} \mathbf{F}_j \tag{5.2}$$

where $k_i = |N_g(\mathbf{p}_i)|$ is the cardinality of $N_g(\mathbf{p}_i)$.

$\mathbf{d}_i$ is a global measure, which establishes the difference between the feature vector of each neighborhood $\mathbf{F'}_i$ and the global mean $\overline{\mathbf{F'}}_i$ (5.3) of all the feature vectors of the point cloud, that is:

$$\overline{\mathbf{F}}_g = \frac{1}{N} \sum_{j=1}^{N} \mathbf{F}_j \tag{5.3}$$

$$\mathbf{d}_i = \left\| \overline{\mathbf{F}}_i - \overline{\mathbf{F}}_g \right\| \tag{5.4}$$

5.3.2.2 | Surrounding Neighborhoods

Once the neighborhood descriptor is established, the next step is to find the surrounding neighborhoods to each $N_g(\mathbf{p}_i)$. To find these neighborhoods, first, the 3x3 covariance matrix of $N_g(\mathbf{p}_i)$ is computed as:

$$Cm_i = \frac{1}{k_i - 1} \sum_{j=1}^{k_i} (\mathbf{p}_i - \bar{\mathbf{p}}) (\mathbf{p}_i - \bar{\mathbf{p}})^T \tag{5.5}$$

where $\bar{\mathbf{p}}$ is the mean of $N_g(\mathbf{p}_i)$ and $k_i = \left|N_g(\mathbf{p})\right|$.

After the covariance matrix Cm_i has been estimated, we will use its two largest eigenvalues with its corresponding eigenvectors that follow the x and y axes, which expand the tangent plane $\mathbf{T}_i$ to $N_g(\mathbf{p}_i)$ at the point $\mathbf{p}_i$. Next, the points within $N_g(\mathbf{p}_i)$ are projected onto the 2D plane $\mathbf{T}_i$ as shown in Figure 5.2(a). Before projecting the points within $N_g(\mathbf{p}_i)$, we establish the area of the surrounding neighborhoods as the n-ring, as can be seen in Figure 5.2(b), that is, the 1-ring corresponds to radius r, the 2-ring corresponds to $2r$, and so on. In these experiments, we used the 1-ring. Then, we proceed to trace a set of radii, separated by an θ angle, from the center of the projected neighborhood in Figure 5.2(b); the length of radii is n-ring. In each of the radii, we mark points (green dots) depending on if we have used the 1-ring, 2-ring, or n-ring, as shown in Figure 5.2(b). Then, we find the nearest point to each marked point within the projected neighborhood and find its corresponding point within the 3D neighborhood; then, we estimate a neighborhood for each of the 3D points with radius r, as seen in Figure 5.2(c). Finally, a feature vector is estimated for each surrounding neighborhood, in the same way as in the previous section.

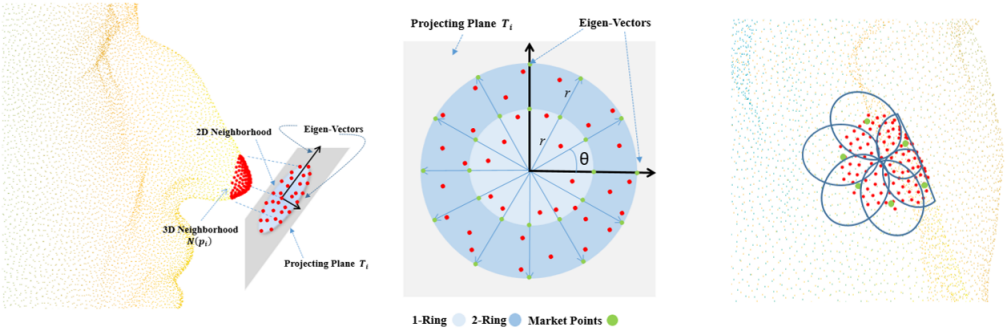


Figure 5.2: Surroundings neighborhood selection. (a) Projecting the 3d Points to 2D, (b) mark points and n-ring area, (c) selecting surrounding neighborhoods.

5.3.2.3 | Dictionary Construction and Sparse Coding Model

It can be observed that the surrounding feature vectors $\overline{\mathbf{F}}_j$ are a natural over complete basis dictionary $\mathbf{D}$, i.e., $\mathbf{D} = \left\{\overline{\mathbf{F}}_1, \overline{\mathbf{F}}_2, \ldots, \overline{\mathbf{F}}_N\right\}$ and the central feature vector $\overline{\mathbf{F}}_i$ acts as the sparse linear combination of these basis or atoms, i.e., $\overline{\mathbf{F}}_i = \mathbf{D}\boldsymbol{\alpha} = \sum_{j=1}^{N} \alpha_j \overline{\mathbf{F}}_j$. Now we can write the sparse model as:

$$\hat{\boldsymbol{\alpha}} = arg\min_{\boldsymbol{\alpha}} \left\|\overline{\mathbf{F}} - \mathbf{D}\boldsymbol{\alpha}\right\|_2^2 + \lambda \left\|\boldsymbol{\alpha}\right\|_1 \tag{5.6}$$

Equation 5.6 is a linear regression problem optimization for estimating $\hat{\boldsymbol{\alpha}}$, known as Lasso. The LARS algorithm gives the solution.

5.3.2.4 | Saliency Detection

Previously, in the Feature Vector Section, $\mathbf{d}_i$ was mentioned as being the fifth component of the feature vectors $\mathbf{F}_i$, and it was estimated using (5.4). This component was added because often, a local neighborhood has similar surroundings, but the local and surroundings are globally distinct over the entire point cloud. Using only normal vectors and Gaussian curvature can produce areas in the cloud that have saliency, but the center of these areas can be empty. Adding d_i solves this difficulty. When a sparse solution to (5.6) is achieved, the code length for the neighborhoods is established, as was laid out in Section 5.2.2. Our model is based on the MDL principle, therefore, the saliency of each neighborhood $N_g(\mathbf{p}_i)$ is proportional to $\|\boldsymbol{\alpha}\|_0$ and $\left\|\overline{\mathbf{F}} - \mathbf{D}\boldsymbol{\alpha}\right\|_1$. We replaced the L_2-norm with the L_1-norm in the residual error because it is more discriminative and robust to outliers. Next, following the instructions in Borji and Itti (BI12), we rewrote the equations $\|\boldsymbol{\alpha}\|_0$ and $\left\|\overline{\mathbf{F}} - \mathbf{D}\boldsymbol{\alpha}\right\|_1 = S_l(N_g(p_i))$ and $\|\boldsymbol{\alpha}\|_0 = S_M(N_g(p_i))$, both saliency measurements are then normalized and combined.

$$S_T(N_g(p_i)) = N(S_M) \circ N(S_l) \tag{5.7}$$

where $S_M(\cdot)$ is the saliency produced by the sparse reconstruction error, $S_l(\cdot)$ is the saliency produced by the sparse coefficient and S_T is the combination of both saliency measurements; $N(\cdot)$ normalizes the saliency measurements for better fusion. The symbol $\circ$ in 5.7, is an integration scheme $\{*, +, max, min\}$ (LZX+09); see Figure 5.3. The best results in all experiments were produced using the $*$ operator. Since overlapping is allowed between neighborhoods, the total saliency map is got by accumulating the saliency by neighborhood. Our method differs from that proposed in (LZX+09), since it only uses the dispersion vector (i.e., $\|\hat{\boldsymbol{\alpha}}\|_0$) as a measure of saliency, while our method also takes into account the reconstruction error or residual (i.e., $\|\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}\|_1$) as an additional measure of saliency; finally when these measurements are combined, we obtain a better saliency map. Our method can be seen as a generalization of the framework presented in Li et al. (LZX+09), improving the final result of the saliency map. Figure 5.4 shows the result of residuals performing as a weighting factor to measure the rarity given by the dispersion vector.

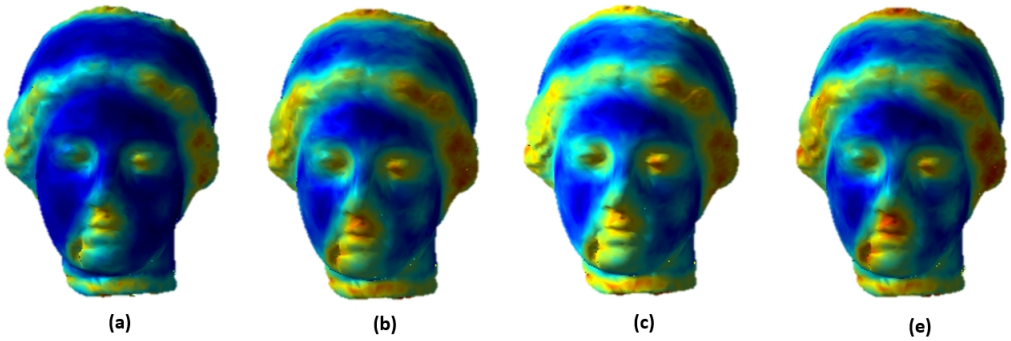


Figure 5.3: Saliency maps generated by different operators function $\circ$, according to equation (11). (a) Using $*$ operator, (b) Using + operator, (c) Using min operator and (d) Using max operator.

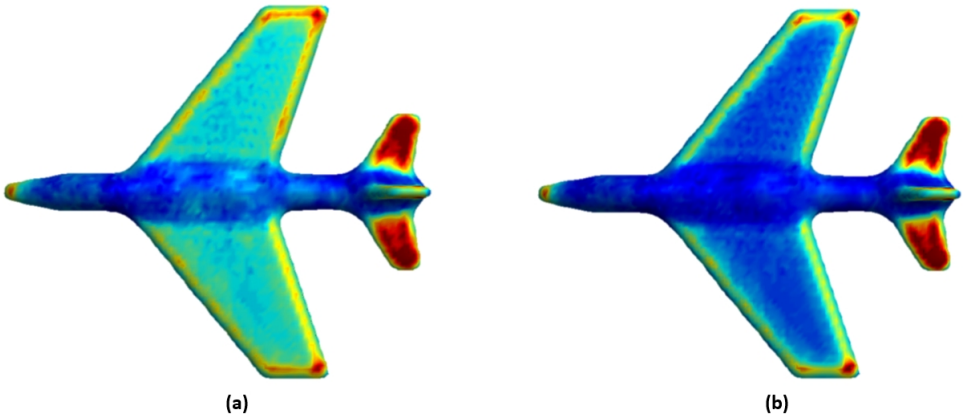


Figure 5.4: Saliency maps results, (a) Using only the dispersion vector $\|\hat{\boldsymbol{\alpha}}\|_0$ as saliency measurement (b) Using the dispersion vector and residual error $\|\boldsymbol{x} - \mathbf{D}\boldsymbol{\alpha}\|_1 * \|\hat{\boldsymbol{\alpha}}\|_0$ as saliency measurement (b).

5.4 | Experimental results and discussion

In this section, we test the proposed method on a set of objects and compare it against 3D model saliency detection approaches outlined in the literature review, considering methods based on point clouds and meshes. The object shapes were obtained from the Watertight Models of SHREC 2007 and the Stanford 3D Scanning Repository. Our method is compared to three point cloud-based methods from the literature review: Tasse et al. (TKD15), Shtrom et al. (SLT13) and Guo et al. (GGCJ17), and six mesh-based methods from the literature review: Lee et al. (LVJ05), Wu et al. (WSZL13), Leifman et al. (LST12), Tao et al. (TCL$^+$15), Song et al. (SLMR14), Liu et al. (LTC$^+$16)

and Jeon et al.(JS17). It is also compared to the pseudo-ground truth (CSPF12). All the experiments were run on a PC with an Intel Core i7-2670QM CPU@2.20 GHz and 8GB RAM. As an example, for the model of a girl in Figure 5.8, with 15.5 K vertices, the saliency computation cost of our method was 81.7s, being slower than Tasse (TKD15) and Guo (GWX17), but if a language like C++ is used, performance will improve.

5.4.1 | Parameter selection

The only parameter in our method is λ, the regularization parameter. It produces smoother results as value increases in the range $(0, 1)$; its visual effect can be seen in Figure 5.5. We found that, on average, a value of λ near 0.9 generates the best qualitative and quantitative results, therefore in all our experiments, we fix $\lambda = 0.9$. We calculate the size of the neighborhood according to the local characteristics (density, curvature) of points. To estimate the size of the neighborhood, we use the method proposed in (MN03). This method estimates the size of the neighborhood, taking into account the local characteristics named.

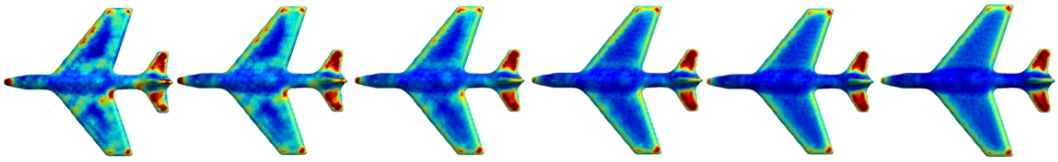


Figure 5.5: Saliency map produced by our method with different values of λ. From left to right $\lambda = 0.1, 0.2, 0.4, 0.6, 0.8, 0.9$

5.4.2 | Qualitative Evaluation

We compare our method with the state-of-the-art, both mesh-based and point cloud-based. In Figure 5.6, we compare the point-based methods, including Shtrom et al. (SLT13), Tasse et al. (TKD15), and Guo et al. (GWX17). The method in Shtrom et al. can get a reasonable saliency result and highlights a relevant saliency region on the Max Planck model, but there are also extensive areas with noise around the principal saliency features. In the method proposed by Tasse et al., less noise is perceived, but there are large regions around the features with higher saliency. In the result of the Guo et al. method, we observe a clean saliency map concentrated on the most representative saliency features, however, our method, besides achieving the same, highlights areas such as eyes, the lower part of the nose, ears, and lips with greater saliency. Regarding

the dragon model, there was a similar result as with the Max Planck model. The proposed method produces a clean saliency map compared to Shtrom et al. and Tasse et al. The saliency map produced by Guo et al. is very similar to ours, which only highlights some of the finer details such as eyes and ridges.

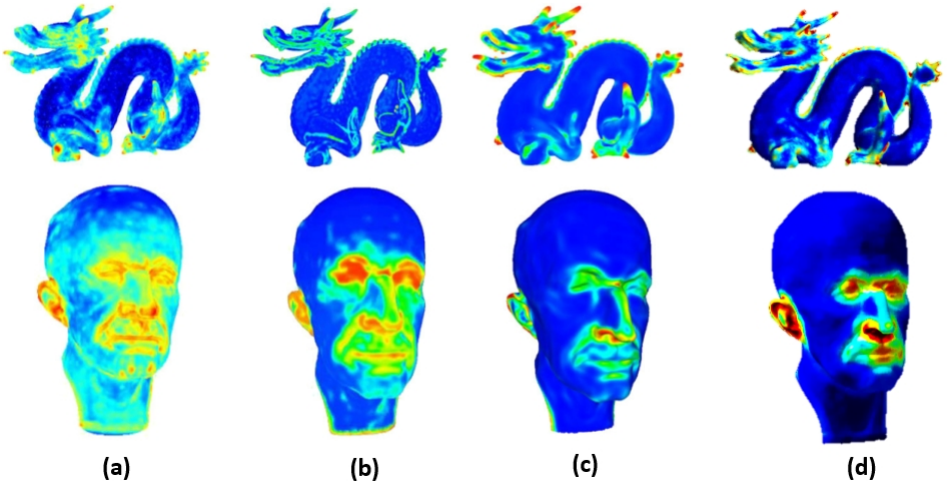


Figure 5.6: Point-based methods saliency comparison. (a) Shtrom et al. (SLT13). (b) Tasse et al. (TKD15). (c) Guo et al. (GGCJ17), and (d) proposed

We also compare our results with mesh-based methods, shown in Figure 5.7, with those of Lee et al. (LVJ05), Wu et al. (WSZL13), Leifman et al. (LST12), and Tao et al. (TCL$^+$15). It can be observed that the local changes in the curvature had less influence when using the proposed method, as seen in Lee et al. (LVJ05). In the bunny and dragon models, the saliency map is not correct because of the variation of the level of saliency in different areas. The dragon, for example, has lost some fine details like crests and ridges.

The method of Wu et al. produces a better saliency map than that of Lee et al., but some salient areas are missing, as in the case of the dragon. The method proposed by Leifman et al. produces a reliable saliency map for the dragon model, but in the case of the bunny, the feet are missing in the final saliency map. For the Tao et al. method, in the dragon model, some areas are shown to be salient where they are not. In the visual comparisons, our method achieves better results than the other four methods.

Figure 5.8 compares our results with (TCL$^+$15), (LTC$^+$16), (SLME16) and shows that how our method detects saliency more coherently, as it identifies small salient regions, such as the ears and the little tie, and the facial regions like the eyes and mouth of the bust of the girl. The hair (bun and braids) show up better with the pseudo-ground truth

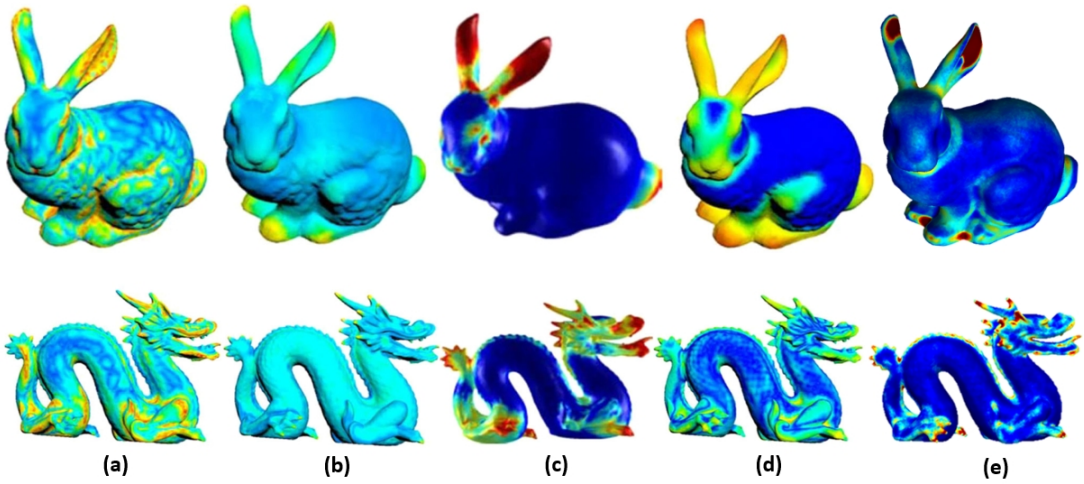


Figure 5.7: Mesh-based methods saliency comparison. (a) Lee et al. (LVJ05). (b) Wu et al. (WSZL13). (c) Leifman et al. (LST12), Tao et al. (TCL$^+$15) (d). and (e) proposed.

(CSPF12), compared to the other four methods. We observe, about the bird model, how our method detects the saliency points in the wings, tail, beak, and finally, the light area of saliency between the wings compared to the pseudo-ground truth bird model.

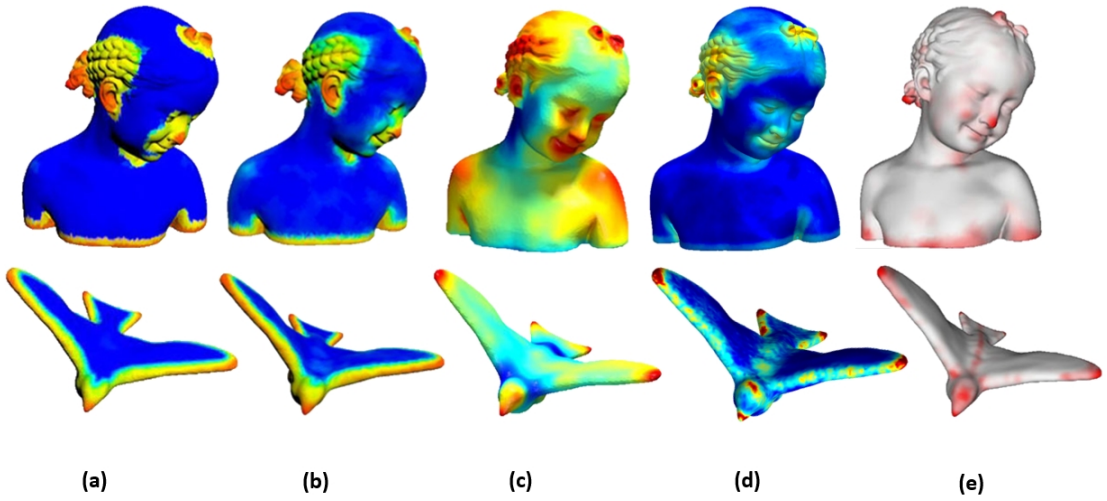


Figure 5.8: Mesh saliency results of Tao et al. (TCL$^+$15) (a), Liu et al. (LTC$^+$16) (b), Song et al. (SLME16) (c), our method (d), and the pseudo-ground truth. (CSPF12) (e).

Figure 5.9 compares our results with (JS17) and (LTC$^+$16). Note that the proposed method captures the hair, nose, eyes, and mouth of the Venus closer to pseudo-ground truth that competing methods. In the girl bust model, our method produces a clean

saliency map; it observes that the method of Jeon et al. produces a saliency map with highlighting no saliency region. The method of Liu et al. (LTC$^+$16) produces a better saliency map, but compared with the pseudo-ground truth; our method is closer.

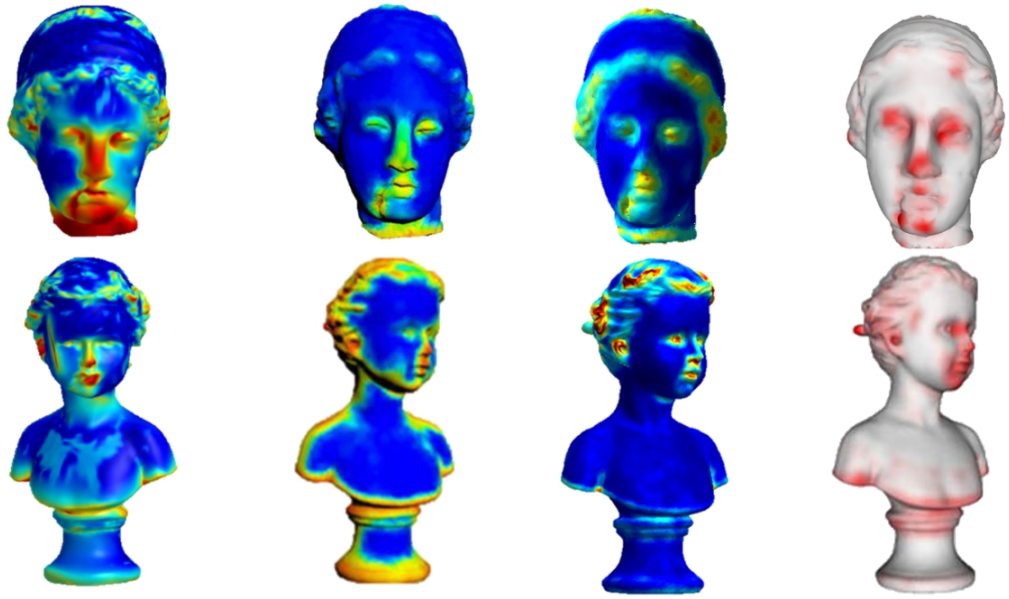


Figure 5.9: Saliency results by our method (third column) and the competing mesh-based methods including, Jeon et al. (JS17) (first row), Liu et al. (LTC$^+$16) (second row), and the pseudo-ground truth. (fourth column) (CSPF12).

Figure 5.10 compares our results with (LTC$^+$16), (TCL$^+$15), (SLMR14), (WSZL13), (LVJ05), (LST12). Our method identifies the facial features of the Max Planck, and the antennae and legs of the ant, compare to pseudo-ground truth (CSPF12), while they are dimly captured by (LTC$^+$16), (TCL$^+$15), (LVJ05) and not captured by (WSZL13). In general, our method achieves better results than the other five methods compared with (CSPF12).

Figure 5.11, shows the comparison of our method with the pseudo-ground truth Chen et al. (CSPF12). Our results are more consistent, but we can appreciate that our method cannot detect the saliency in the wings of the duck model. These are misses in our saliency map because of low density in this area of the point cloud. Low density implies less information; therefore, there are not enough neighborhoods around to estimate a correct saliency map.

Figure 5.12, shows the results when the input is a noisy point cloud. It can be seen that our method is more robust against noise compared to with three mesh-based methods (TCL$^+$15), (WSZL13) and (LVJ05). We add 30% (relative to the average of the nearest

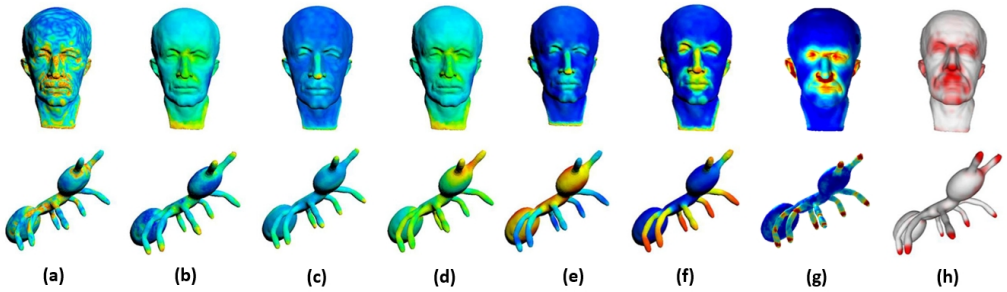


Figure 5.10: Mesh saliency results of Lee et al. (LVJ05) (a), Leifman et al. (LST12) (b), Wu et al. (WSZL13) (c), Song et al.(SLMR14) (d), Tao et al. (TCL$^+$15) (e), Liu et al. (LTC$^+$16) (f), proposed (g), and the pseudo-ground truth (CSPF12) (h).

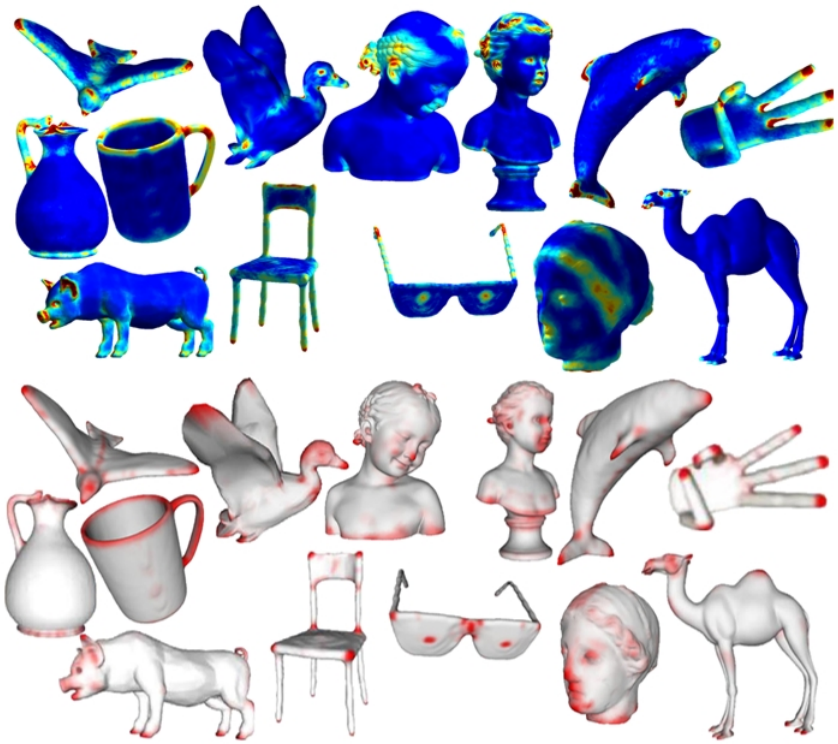


Figure 5.11: Results of point-based saliency. On the top part is shown the saliency produced by our method, and the bottom part is shown the corresponding pseudo-ground truth Chen et al. (CSPF12). The models were taken from the Watertight Models of SHREC 2007.

distance of each point) random noise to perturb points coordinates. We observe that the method proposed by Lee et al. is not robust because the noise affects the curvature. To

solve this problem, it must use a robust curvature algorithm in the presence of noise. In the wings of the gargoyle, we can observe that our method is more consistent for detecting salient features, both in the clean and noise model, in comparison with Lee et al. (LVJ05), Wu et al.(WSZL13) and Tao et al.(TCL$^+$15).

5.4.3 | Quantitative Evaluation

We carried out a quantitative evaluation of the 400 watertight models of SHREC 2007, using the distribution of Schelling points provided by Chen et al. (CSPF12) as the ground truth. To test the performance, we used the metrics proposed by Tasse et al. (TKD16a); they propose three metrics that adapt 2D image saliency metrics to 3D saliency. These metrics are: Area Under the ROC curve (AUC), Normalized Scanpath Saliency (NSS) and Linear Correlation Coefficient (LCC). In the AUC metric, the ideal saliency model has a score of 1.0, AUC disregards regions with no saliency, and focuses on the ordering of saliency values. The NSS metric measures the saliency values selecting the users as fixation points. The LCC metric has values of between -1 and 1, with values closer to 0 implying weak correlation. This metric compares the saliency map under consideration with the ground-truth saliency map (Schelling distributions). For comparison, we selected three methods from the literature review, two based on point clouds and one based on meshes, these methods are: the clustering method (CS) (TKD15), the point-wise method (PW) (GWX17) and the spectral analysis method (SS) (SLMR14). Also, as a reference score, we incorporated the human performance score (HS), provided by Tasse et al. (TKD16a).

Table 5.1 shows the AUC score values of the selected methods; our method competes with the CS and PW methods, as the final average shows. The SS method performs poorly compared to our method and the CS and PW methods. The same is true regarding the HS method, which outperforms the SS method. Table 5.1 shows that for the classes glasses, ant, octopus, bird, and bearing, our method obtained the best results, while for the classes hand, fish, spring, mechanic, airplane, and vase, our method equals the PW and CS methods. The above shows that the proposed method achieves the same results in comparison with the state-of-the-art techniques and, in some cases, even improves them.

Figure 5.13 shows NSS and LCC metrics and confirms that the performance of our method is similar to CS and PW, and outperforms SS, but HS outperforms all methods presented.

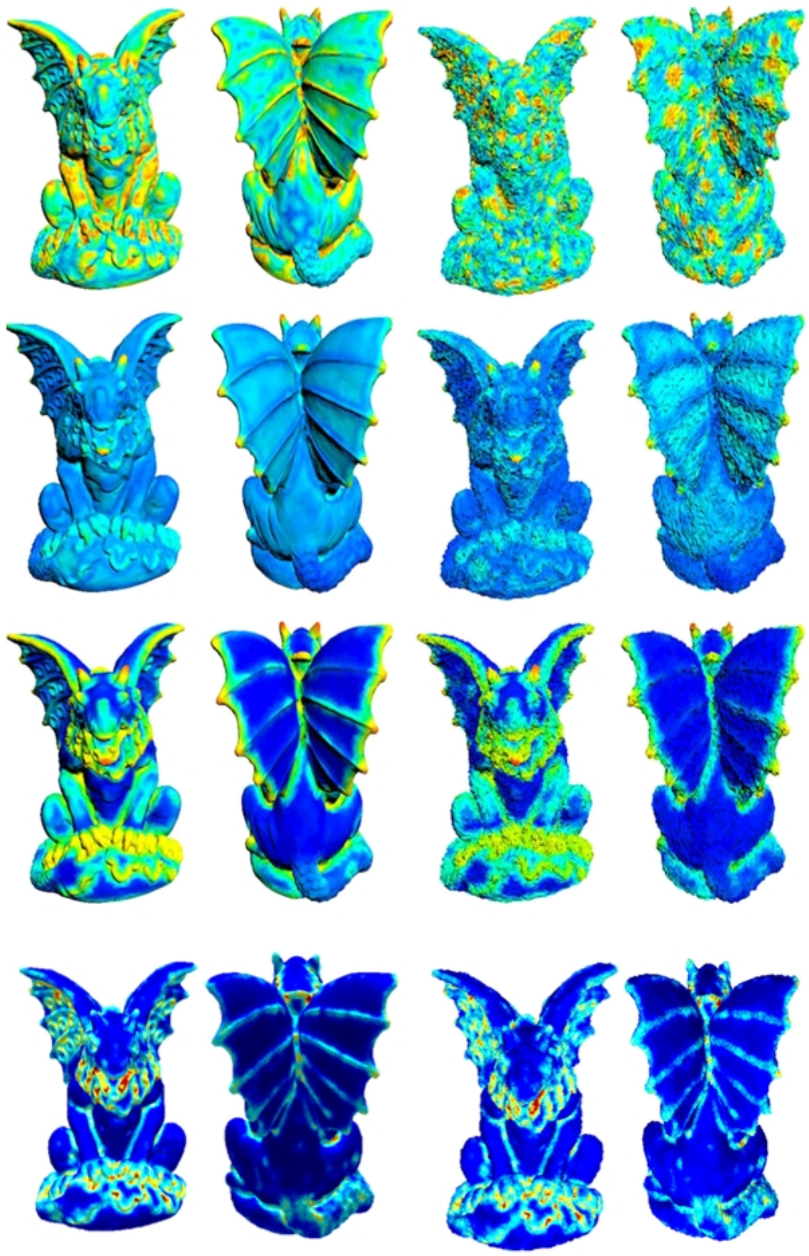


Figure 5.12: Saliency results on Gargoyle model. Our method (bottom row), Lee et al. (LVJ05) (first row), Wu et al. (WSZL13) (second row) and Tao et al. (TCL$^+$15) (third row). The first two columns are front and back view of saliency results from Gargoyle model. The second two columns are the same view results but with 30% random noise relative to the average of the nearest distance of each point.

Table 5.1: AUC performance per shape class in SHREC 2007.

Class	SS	CS	PW	Ours	HS (1 v. all)
human	0.57	0.57	0.59	0.59	0.58
cup	0.59	0.60	0.64	0.62	0.57
glasses	0.53	0.52	0.53	**0.55**	0.56
airplane	0.62	0.62	0.60	0.62	0.60
a*ant*	0.57	0.60	0.59	**0.61**	0.60
chair	0.55	0.57	0.59	0.58	0.62
octopus	0.53	0.54	0.56	**0.59**	0.60
table	0.58	0.63	0.65	0.60	0.61
teddy	0.55	0.56	0.57	0.56	0.60
hand	0.58	0.61	0.61	0.61	0.58
plier	0.56	0.59	0.59	0.58	0.56
fish	0.63	0.66	0.66	0.66	0.61
bird	0.56	0.60	0.59	**0.61**	0.57
spring	0.55	0.55	0.55	0.55	0.54
armadillo	0.60	0.65	0.66	0.65	0.62
bust	0.56	0.62	0.63	0.62	0.59
mechanic	0.53	0.68	0.70	0.70	0.64
bearing	0.50	0.60	0.62	**0.63**	0.55
vase	0.61	0.62	0.63	0.63	0.58
four-legs	0.60	0.60	0.61	0.60	0.59
average	0.57	0.60	0.61	0.61	0.59

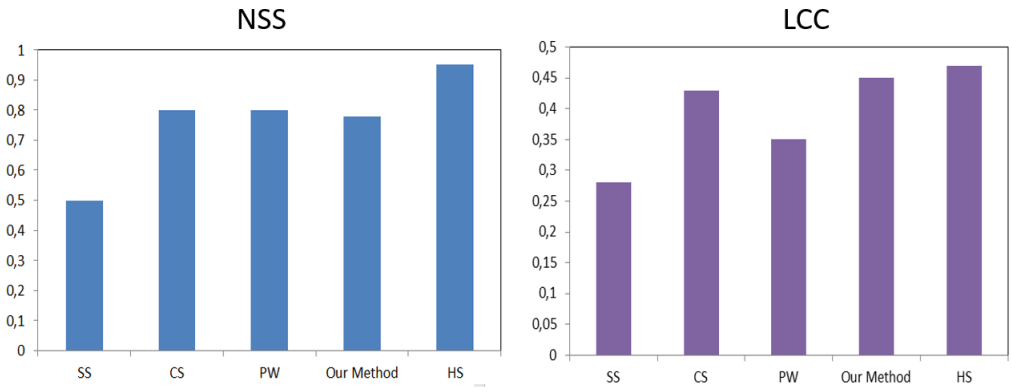


Figure 5.13: Saliency performance evaluation under the metrics NSS (left) and LCC (Right).

5.5 | Conclusion

This chapter presents a novel and simple method for point cloud saliency detection via sparse coding. Based on the MDL principle, the proposed method uses a sparse coding representation to find the minimum code length to establish when a neighborhood is salient or not regarding its surrounding neighborhoods. It is robust against noise since

it takes the mean of the feature vectors of the neighborhood as a unique feature vector. Our approach produces feasible and even faithful results on a variety of models, giving convincing results. We have compared our results to the most recent approaches found in the literature review, and we found that the proposed method competes with and, in several cases, significantly outperforms these approaches, using the pseudo-ground truth provided by Chen et al. (CSPF12) as a reference.

6

Surface Simplification

High-resolution 3D scanning devices produce high-density point clouds, which require a large capacity of storage and time consuming post-processing algorithms. As a pre-processing stage, it is common to apply surface simplification algorithms for simplifying the subsequent geometric processing. The goal of point cloud simplification algorithms is to reduce the volume of data while preserving the most relevant features of the original point cloud. In this chapter, we present a new point cloud feature-preserving simplification algorithm. We use a global approach to detect saliencies on a given point cloud. Our method estimates a feature vector for each point in the cloud. The components of the feature vector are the normal vector coordinates, the point coordinates, and the surface curvature at each point. Feature vectors are used as basis signals to carry out a dictionary learning process, producing a trained dictionary. We perform the corresponding sparse coding process in order to produce a sparse matrix. To detect the saliencies, the proposed method uses two measures, the first of which takes into account the quantity of nonzero elements in each column vector of the sparse matrix and the second the reconstruction error of each signal. These measures are then combined to produce the final saliency value for each point in the cloud. Next, we proceed with the simplification of the point cloud, guided by the detected saliency and using the saliency values of each point as a dynamic clusterization radius. We validate the proposed method by comparing it with a set of state-of-the-art methods, demonstrating the effectiveness of the simplification method.

6.1 | Introduction

Point clouds have become a standard data input tool for many fields, including scientific visualization, photogrammetry, and medical applications. For data acquisition of

3D shapes, modern 3D scanning devices can produce a vast amount of data, reaching millions of points (LPC$^+$00). This amount of data creates challenges in several fronts, like large storage requirements and increased data transmission and rendering times. In order to reduce the complexity of such point clouds and make the subsequent geometric processing algorithms more efficient, it is common to simplify the point cloud. The main requirement for point cloud simplification algorithms is that they should maintain the global shape, the sharp features and the curvatures of the original cloud. For the last of these, transitions between planar and curved areas should be preserved (CY16). It is important to preserve the representative points and the sampling density in order to approximate faithfully the original point cloud both geometrically and topologically. The simplified point cloud must be dense around the sharp features (corners, edges, and curvatures) to preserve the global topology, and sparse in flatten regions (low or zero curvature). The original point cloud data only contains the coordinates of the points with no topological information. In order to extract the implicit geometric information (normal vectors, surface variation, curvatures), the point-based simplification algorithms use the local information around each point in the cloud. Usually, the k-nearest neighbor algorithm is used to estimate such geometric information. For each point in the cloud, the proposed method uses the coordinates of the normal vector, the coordinates of the point and the curvature as a feature vector to identify potential saliency points. The feature vectors of each data point are the training signals for a dictionary learning process. With the dictionary trained, a sparse coding process is carried out to identify the most salient regions in the point cloud. Finally, the proposed method simplifies the point cloud by using the sparse vectors as a clusterization radius.

We can define the point cloud simplification problem in two ways. Problem 1: how to distribute the points on the simplified surface, and Problem 2: geometrically speaking, in the case of an error bound, how far is the simplified surface from the original one?.
Formally, we define the problem of point cloud simplification as follows: Given a surface $\boldsymbol{S}$ defined by a point cloud $\boldsymbol{P}$ and a target sampling rate $N < |P|$ the goal is to find a point cloud $\boldsymbol{P'}$ with $|\boldsymbol{P'}| = N$ such that the distance ε of the surface $\boldsymbol{S'}$ to the original surface $\boldsymbol{S}$ is minimal (PGK02). A related problem is finding a point cloud with a minimum sampling rate given a maximum distance ε. Symbolically we write the above as:

$$P \rightarrow P', \quad where$$
$$1.\ |P'| = N < |P| \ \ and \ \ \left\|P - P'\right\| \quad is\ \ minimum$$
$$2.\ \left\|P - P'\right\| < \epsilon \ \ and \ \ |P'| \ \ is\ \ minimum$$

Where $|\cdot|$, is the point cloud cardinality, $\|\cdot\|$ is the Euclidean distance. Problem 1 seeks to reduce the error between the original point cloud P and the simplified P' when a size restriction in the simplified point cloud is given. Problem 2 uses an error limit ϵ to minimize the size of the point cloud P such that no point in the simplified cloud P' is further than the distance ϵ of the original model.

6.1.1 | Contribution

The contributions of this chapter are:

- A point cloud simplification method based on dictionary learning and sparse coding that maintains a balance between sharp features and the density of points distribution.
- The proposed method downsizes the point cloud very efficiently due to its inherent perceptual nature, which selects important points based on their saliency.
- The saliency-based simplification provides an important criterion to preserve the most important geometric features.
- Our method is simple and easy to implement, is time-efficient, and does not need complicated data structures to achieve simplification.
- Experimental results confirm the effectiveness of our algorithm, making it comparable to the state-of-the-art alternatives.

6.2 | Related Work

In recent decades,a considerable amount of research has been conducted on point cloud simplification. Point cloud simplification algorithms can be roughly divided into four categories: particle simulation-based methods, Iteration-based methods, Formulation-based methods, and clustering-based methods.

6.2.1 | Particle simulation-based methods

Pauly et al. (PGK02) present a particle simulation method. The proposed algorithm distributes a set of points called particles evenly onto a surface, producing point clouds with low approximation error to the original point cloud. Collections of particle simulation-based methods are called LOP-based methods (Local Optimal Projection) (LCOLTE07).

These methods project a set of points over an underlying surface using a localized version of the L1 median filter regularized by a repulsion potential. Huang et al. (HLZ+09), propose a correction over the original LOP algorithm, comprise distributing the point evenly over the underlying surface. Huang et al. (HWG+13) and Liao et al. (LXJF13), aim to integrate the vector normal to each projected point, in order to preserve sharp features on the point cloud. These methods produce good results for surface simplification but are expensive in computational time. Futhermore, the original points are replaced by the particles, losing the original data sets.

6.2.2 | Clustering-based methods

These methods divide the point cloud into clusters, applying some criteria and then replacing the cluster points by the centroid. Pauly et al. (PGK02), present two algorithms, uniform incremental clustering, and hierarchical clustering. These methods are memory and computational time-efficient but produce high average approximation errors with respect to the original surface. Shi et al. (SLL11) present an adaptive method for simplifying point clouds. They apply a recursive subdivision scheme in which the algorithm selects representative points and removes redundant ones. They use k-means clustering to group similar spatial points and apply the maximum normal vector deviation measure to subdivide the clusters. The algorithm can handle boundaries and produce uniform density in flat regions and high density in curved regions. In Mahdaoui et al. (MBMHS17), present a comparison between two simplification algorithms using k-means and fuzzy c-means algorithms. The method proposes using a metric based on entropy estimation for clustering the point cloud. Liu et al. (LLR+20) present an edge-sensitive feature detail preserving algorithm; they use two clustering schemas to split the point cloud in the geometric and spatial domains. These methods can preserve global structures of the point clouds, and some of them preserve sharp features, but because of the clustering process they are computational time-consuming.

6.2.3 | Formulation-based methods

These methods are based on mathematically modeled optimality. Leal et al. (LLG17), propose a three-step method. In the first step, they apply a clusterization algorithm. The second step involves the identification of points with high curvature to be preserved. The last step, uses a linear programming model to simplify the point cloud, maintaining a density equivalent to the original point cloud. Chen at al. (CTF+18) employ a resampling strategy based on a graph that selects representative points while

preserving features. The minimization of the point cloud is carried out by a proposed reconstruction error based on a feature extraction operator. In (QHG19) Qi et al. propose an optimization strategy for maintaining the balance between finding the sharp features and preserving density in the point cloud, The optimization is represented using a graph filter. The proposed method are superior to some methods state-of-the-art.

6.2.4 | Iteration-based methods

Pauly et al. (PGK02), propose an iterative simplification method using quadric error metrics, this algorithm produces point clouds with low approximation errors, but they are expensive in computational time. Alexa et al. (ABCO$^+$03) propose a decimation process based on the Moving Least Square (MLS) method. The proposed method removes redundant information using a surface error metric. The global result of the algorithm is good, but it can produce uneven sampling distributions because the subsampling unnecessarily restricts the potential sampling position. Zang et al. (ZYLX18), present a method based on a multi-level strategy for point cloud simplification, which adaptively determines the optimal level of points. For each level, the method extracts the points based on a measure of importance given by a 3D Gaussian method. Zhu et al. (ZKV$^+$19), propose a multi-view method for point cloud simplification. They project the points onto the three orthographic planes getting the object edges. The edges are merged to produce 3D edges object, and the points with less importance are separated from the point cloud. Shoaib et al. (SCKC19) propose a method called fractal bubble to simplify point clouds, selecting important data points through the expansion of a recursive generation of self-similar 2D bubbles until contact is made with a point. Ji et.al. (JLFL19), present a detailed feature points simplified algorithm (DFPSA). They propose estimating the importance of each point using a four-characteristic operator, which involves estimating normal curvature distance between the points and projection distance, to each point on the point cloud. Finally, a threshold is used to decide whether each point may be classified as a feature point or not. The non-feature points are simplified using an innovative step through an octree structure to avoid creating regions with holes. Zhang et al. (ZQW$^+$19) presents a feature preserved point cloud simplification (FPPS); for the simplification, an entropy measure is defined, which quantifies the geometric features hidden in the point cloud. The key points are selected, based on the entropy.

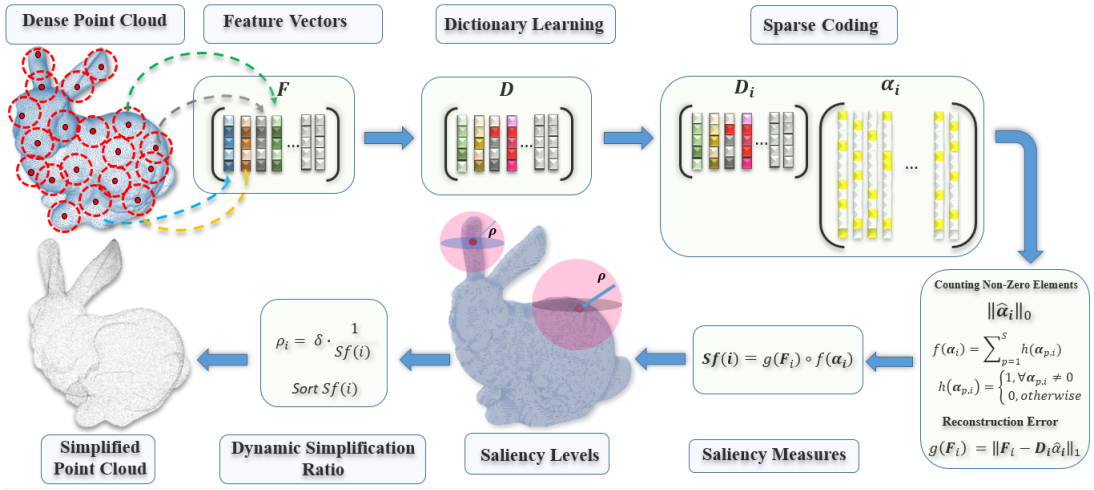


Figure 6.1: Steps involved in the proposed method to simplify a point cloud.

6.3 | Point Cloud Simplification

Our proposed method is based on dictionary learning and sparse coding. The input point set is analyzed using the covariance matrix to extract the local features; then, using the dictionary and the sparse representation matrix, the point set is analyzed globally to identify the sharp features. Finally we used the saliencies to sample the point cloud, keeping the most representative points. Figure 6.1, shows the pipeline of the proposed method.

6.3.1 | Low-level Features Estimation

To characterize the point set, we estimate as the descriptor for each point $\mathbf{p}_i$: the normal vector, the total variation of surface (curvature), and the point coordinate. The normal vector in the surface is related to the tangent plane; if two points are on the same tangent plane, the normal vectors are the same. On the other hand, if two points are on different planes (non-parallel), their respective normal vectors are different. Therefore, a large difference between the normals means that the surface at the point is more curved, i.e., is likely to be a feature point. The surface curvature at a point shows the surface variation at that point. High curvatures reflect large variations of the surface at the point, showing that is part of a feature. The point coordinates reflect the spatial relationship between the points. In a point cloud (without noise), when the spatial distance of a point and its neighboring points is large is possible that the point is in sharp feature, or the sampling density is nonuniform near the point. In the surface reconstruction process, holes can

appear in low-density areas, which means the point should be preserved.

A common way to estimate low-level features in a point set is to apply the PCA method locally to each neighborhood $N_g(\mathbf{p}_i)$. Specifically, we use a weighted version of PCA, with a covariance matrix Cm_i, established in Equation (6.1).

$$Cm_i = \frac{1}{k_i - 1} \sum_{i=1}^{k_i} w_i \left(\mathbf{p}_i - \bar{\mathbf{p}}\right) \left(\mathbf{p}_i - \bar{\mathbf{p}}\right)^T \tag{6.1}$$

where $\bar{\mathbf{p}} = \frac{1}{|N_g(\mathbf{p})|} \sum_{i=1}^{k_i} \mathbf{p}_j$, is the centroid of $N_g(\mathbf{p}_i)$ and $k_i = |N_g(\mathbf{p}_i)|$, is the cardinality of $N_g(\mathbf{p}_i)$. w_i is a weight estimated by $w_i = exp(-\frac{d^2}{k_i^2})$, $d = \|\mathbf{p}_i - \bar{\mathbf{p}}\|$, is the Euclidean distance. Next, we analyze the eigenvalues $\lambda_0 \leq \lambda_1 \leq \lambda_2$ and eigenvectors $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ of the covariance matrix Cm_i.

The eigenvector $\mathbf{v}_0$ corresponding to the smallest eigenvalue λ_0 is the normal vector $\mathbf{n}_i$ at point $\mathbf{p}_i$. Pauly et al. (PGK02) (PKG03) proved that the total variation is equivalent to the surface curvature. Hence, it is defined in Equation (6.2).

$$\sigma(\mathbf{p}_i) = \lambda_0 / (\lambda_0 + \lambda_1 + \lambda_2) \tag{6.2}$$

$$\mathbf{n}_i = (n_x, n_y, n_z) \tag{6.3}$$

$$\mathbf{p}_i = (p_x, p_y, p_z) \tag{6.4}$$

Once the low-level features are defined, we build a seven-dimensional feature vector $\mathbf{F}_i$, for each point $\mathbf{p}_i \in \mathbf{P}$ where $\mathbf{F}_i = (n_x, n_y, n_z, \sigma, p_x, p_y, p_z)$.

6.3.2 | Dictionary Construction and Sparse Coding Model

Using the feature vectors defined in section 6.3.1, as data vectors $\mathbf{F}_i \in \mathbb{R}^{n \times 1}$, with $n = 7$ (number of low-level features). We construct the data matrix $\boldsymbol{F} = \{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_K\} \in \mathbb{R}^{n \times k}$, where $K = |P|$ is the number of feature vectors. A sparse coding matrix $\alpha \in \mathbb{R}^{S \times K}$ and a dictionary $\mathbf{D} \in \mathbb{R}^{n \times S}$ are defined using sparse coding theory. S is the number of atoms of the dictionary (in our experiment, we set $S = 200$, for all the models). The Equation (3.13) is solved using the K-SVD algorithm (AEB06) obtaining the estimation of $\boldsymbol{\alpha}$ and $\mathbf{D}$. Now $\mathbf{F}$ can be reconstructed as $\mathbf{F} = \mathbf{D}\alpha$, obtaining the sparse representation of data matrix $\mathbf{F}$ in the dictionary $\mathbf{D}$. Now we are able to find the saliency points by analyzing the sparse matrix $\boldsymbol{\alpha}$.

6.3.3 | Relating the MDL Principle to Sparse Coding

Following the idea presented in section 5.2.1 and 5.2.2; a similar approach is followed in this chapter for the global saliency model. Starting from the Bayes theorem and applying MAP. We establish: $-\mathbf{logP}(\mathbf{x} \mid \mathbf{M})$, is the fitting data and $-\mathbf{logP}(\mathbf{M})$, is the prior

- $\mathbf{P}(\widehat{\mathbf{M}})$ is the description length or code length of the model; and
- $\mathbf{P}(\mathbf{x} \mid \widehat{\mathbf{M}})$ is the description length or code length of the data encoded using the model.

if we set the estimation parameter $\widehat{\mathbf{M}} = \widehat{\alpha}$ and evaluate it on the prior term $P(\widehat{\alpha})$, and on the likelihood term $P(x \mid \widehat{\alpha})$, we have $P(\widehat{M}) = \|\widehat{\boldsymbol{\alpha}}\|_1$ and $P(x \mid \widehat{\alpha}) = \|\mathbf{x} - \mathbf{D}\widehat{\boldsymbol{\alpha}}\|_2$, We conclude that the sparsity of the vector $\widehat{\alpha}$ is the codelength of the model, and the residual of the sparse reconstruction error is the codelength of the data given the model.

The MDL principle selects the best model that produces the shortest description of the data. The more regularity that is presented in the data, the shorter the description the model will produce.

6.3.4 | Detecting Saliency Points

Once the sparse coding matrix $\boldsymbol{\alpha}$, has been obtained, we analyze what vectors correspond to the saliencies. Let $\boldsymbol{\alpha}_j$, $\mathbf{F}_j$, be two column vectors of the matrices $\boldsymbol{\alpha}$ and $\mathbf{F}$, respectively. A feature vector of a point is considered salient if its sparse representation $\|\boldsymbol{\alpha}_j\|_1$ has many nonzero elements, implying that linear combination of many atoms is required to represent the point correctly; and if its sparse reconstruction error $\|\mathbf{F}_j - \mathbf{D}\boldsymbol{\alpha}_j\|_2$ produces a high residual. On the other hand, a feature vector of a point is not considered salient, if its sparse representation $\|\boldsymbol{\alpha}_j\|_1$ has few nonzero elements. That is, if it can be represented by the linear combination of only a few atoms and its sparse reconstruction error $\|\mathbf{F}_j - \mathbf{D}\boldsymbol{\alpha}_j\|_2$ produces a low residual.

On this basis, we sum the nonzero elements of each column of the matrix $\boldsymbol{\alpha}$, Equation (6.5). And a score vector with these sums is built.

$$f(\boldsymbol{\alpha}_j) = \sum_{p=1}^{s} h(\boldsymbol{\alpha}_{p,j}) \;\; \forall_j = 1, 2, \ldots, S \tag{6.5}$$

$$h(\boldsymbol{\alpha}_{p,j}) = \begin{cases} 1\,, & \forall \boldsymbol{\alpha}_{p,j} \neq 0 \\ 0\,, & otherwise \end{cases} \tag{6.6}$$

And regarding the sparse reconstruction error, we sum the residuals resulting from the difference between each signal $\mathbf{F}_j$ and its respective reconstruction $\mathbf{D}\boldsymbol{\alpha}_j$. i.e. $r_j = \left\|\mathbf{F}_j - \mathbf{D}\boldsymbol{\alpha}_j\right\|_2$. Equation (6.7). A score vector with these sums is built.

$$g(\mathbf{F}_j) = r_j \;\; \forall_j = 1, 2, \ldots, S \tag{6.7}$$

Now we normalize the score vectors $f(\boldsymbol{\alpha}_j)$ and $g(\mathbf{F}_j)$, divide each vector component by the highest value.

$$f(\boldsymbol{\alpha}_j) = f(\boldsymbol{\alpha}_j)/max(f(\boldsymbol{\alpha}_j)) \;\; \forall_j = 1, 2, \ldots, S \tag{6.8}$$

$$g(\mathbf{F}_j) = g(\mathbf{F}_j)/max(g(\mathbf{F}_j)) \;\; \forall_j = 1, 2, \ldots, S \tag{6.9}$$

Next, both score vectors are integrated into a unique score vector as follow:

$$Sf(i) = f(\boldsymbol{\alpha}_i) * g(\mathbf{F}_i) \;\; \forall_i = 1, 2, \ldots, S \tag{6.10}$$

We use the vector score Sf, as a metric for the simplification process. Figure 6.2 shows the saliency levels founded into the vector Sf, when different threshold values T are applied.

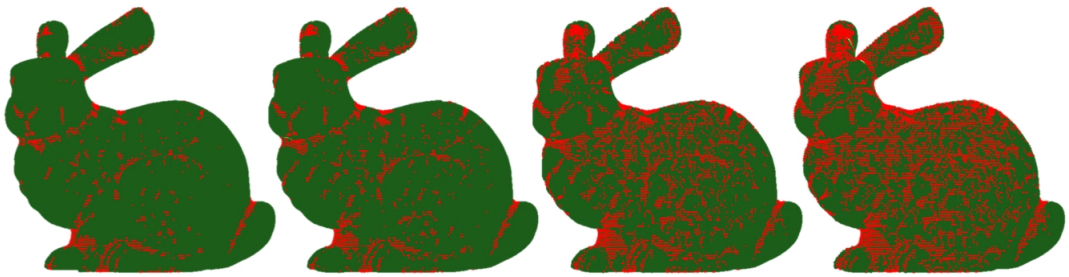


Figure 6.2: The bunny model, different levels of saliency produced by the thresholding of the vector Sf, with different values (left to right) T = 0.9, T = 0.8, T = 0.7 and T = 0.6. Red points are high saliency, green points are low saliency.

6.3.5 | Simplification based on saliency

The saliency points characterize the most relevant features in the point cloud. These points must be retained in the simplification process. On the other hand, points with low saliency are redundant and have less importance for representing the surface to be simplified. Using the vector score defined by (Equation 6.10), we establish a dynamic

ratio of influence that depends on the importance of the saliency of each point in the entire cloud. If point $\mathbf{p}_i$ is salient, the radius of influence will be small and few points will be removed. If, however, it is not salient, the ratio of influence will be large, and more points will be removed (see Figure 6.3).

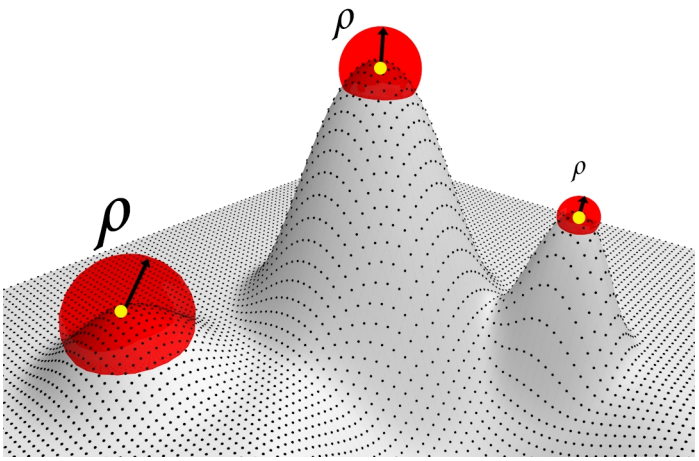


Figure 6.3: Dynamic ratio. The simplification radius is large in regions with low saliency (flatten regions) and small in regions with high saliency (sharp features).

To proceed with the simplification, as a first step, the vector score $Sf(i)$ is sorted by the absolute value of its components. In the second step, we calculate the ratio of influence as follows.

$$\rho_i = \delta \cdot \frac{1}{Sf(i)} \tag{6.11}$$

According to (6.11), the dynamic ratio ρ_i is determined by $\frac{1}{Sf(i)}$. Therefore, in points with high saliency the ratio is small, while in points with low saliency, the ratio is large as shown in Figure 6.3. Where δ, is a user defined scale parameter that controls the number of points to be simplified.

6.4 | Results and Discussion

We have evaluated the proposed method using a set of models namely, the Max Plank, data set (50,112 points, few detail features), the Fandisk data set (6,475 points, high sharp features), the Asian dragon data set (3,609,600 points, many detail features), the Bunny data set (35,947 points, few detail features), the Elephant data set (24,955 points, many detail features), the Horse data set (48,485 points, few detail features), Gargoyle data set

(25,038 points, many detail features) and the Nicolo data set (50,419 points, few detail features).

We also compare the results of our method to other approaches. For quantitative comparison, our method, which we named Saliency Dictionary Based Simplification (SDBS), is compared to three point based methods, the Curvature based method (CV), implemented by Geomagic Studio, Simplification on Graph (FPUC) (QHG19), Fast Resampling via Graphs (FRGR) (CTF$^+$18), and one mesh-based method, namely Poison sampled disk (PSD), implemented by Meshlab. For visual comparison, we replicate the same experiment carried out in (JLFL19), and we use the results to compare the proposed algorithm with six state-of-the-art simplification methods: Grid Simplification (GRID), from CGAL library, Hierarchical Clustering Simplification (HCS) (PGK02), Weighted LOP (WLOP) (HLZ$^+$09), Simplification on Graph (FPUC) (QHG19), Fast Resampling via Graphs (FRGR) (CTF$^+$18) and detail feature points simplified algorithm (DFPSA) (JLFL19).

All the experiments were run on a PC with Intel Core i7-2670QM CPU@2.20 GHz and 8GB RAM. For implementing the proposed method, we use Matlab R2016b programming environment.

Figures 6.4, 6.5, and 6.6 are an example of the effectivity of the proposed simplification method in different types of point clouds (free-form surfaces and surfaces with sharp edges and corners).). It is clear that the proposed method is capable of preserving the global structure of the clouds as the simplification rate increases in all cloud types, since the needed information is integrated into the dictionary training. In Figure 6.4, shows the Fandisk model, the edges and corners are preserved as the simplification rate increase, and in flat regions, the method tries to distribute the points evenly.

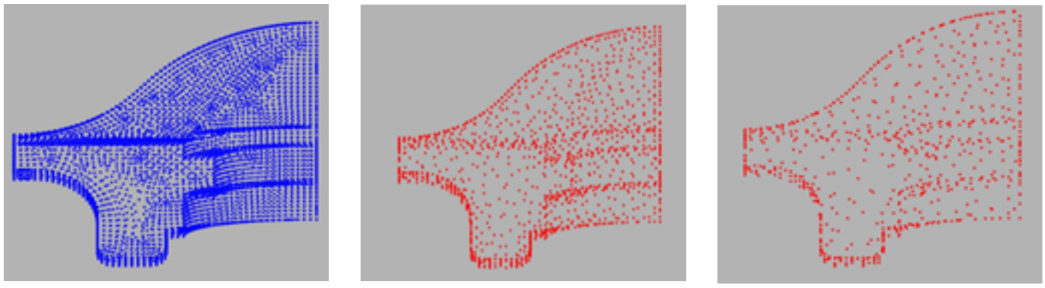


Figure 6.4: The Fandisk model (a) original 6,475 points (b) simplified to 1,465 points and (c) simplified to 738 points.

Figure 6.5 shows how the Asian dragon model is simplified from millions of points

(3,609,600) to thousands of points (1,502). The proposed method preserves the global structure and the most relevant details of the original point cloud.

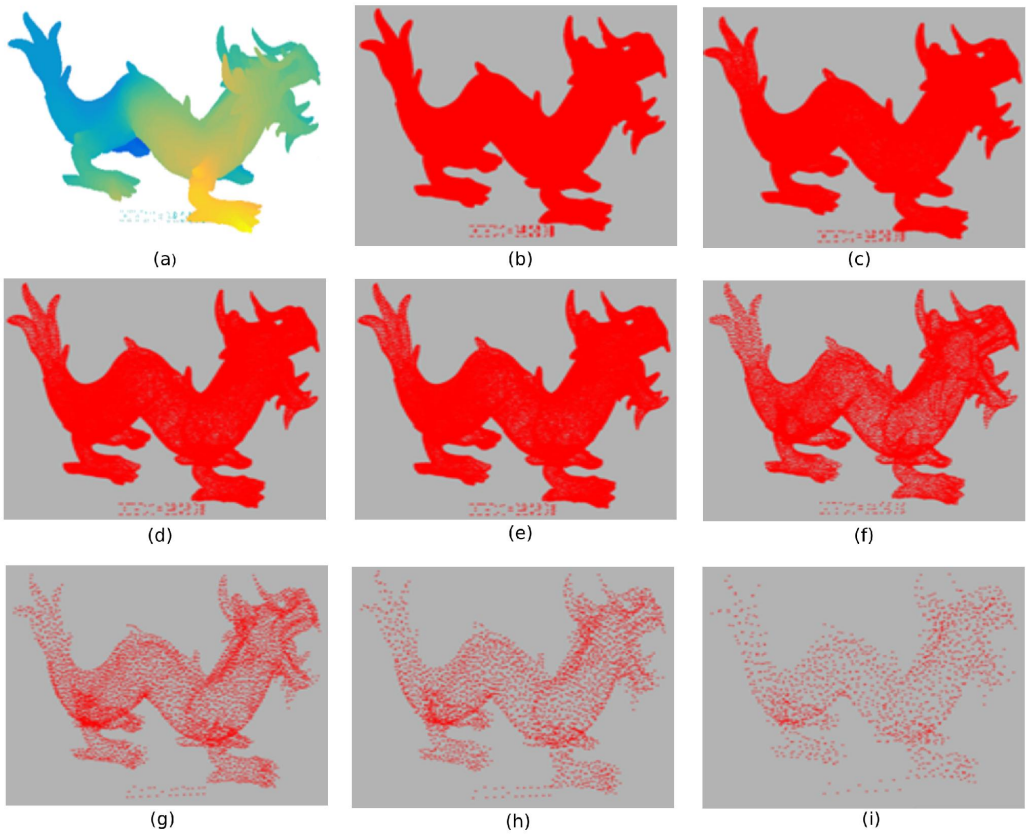


Figure 6.5: The Asian dragon model (a) original 3,609,600 points, (b) simplified to 410,208 points, (c) simplified 78,268 points, (d) simplified to 30,487 points (e) simplified to 12,621 points (f) simplified to 8,196 points, (g) simplified to 5,758 points, (h) simplified to 3,307 points, (i) simplified to 1,502 points.

In Figure 6.6 shows how the Max Plank model is simplified from 50,112 to 1,502 points. The proposed method preserves the global structure, and some of the details of the original point set. The Max Plank model; is a free form surface, showing that our method operates efficiently over these types of models.

6.4.1 | Parameters Selection

There are three parameters in our method: the regularization parameter λ Equation 3.7, dictionary size S, and the scale of points number to be simplified δ. The parameter λ is the balance between the data-fidelity and the regularization term. Small values can

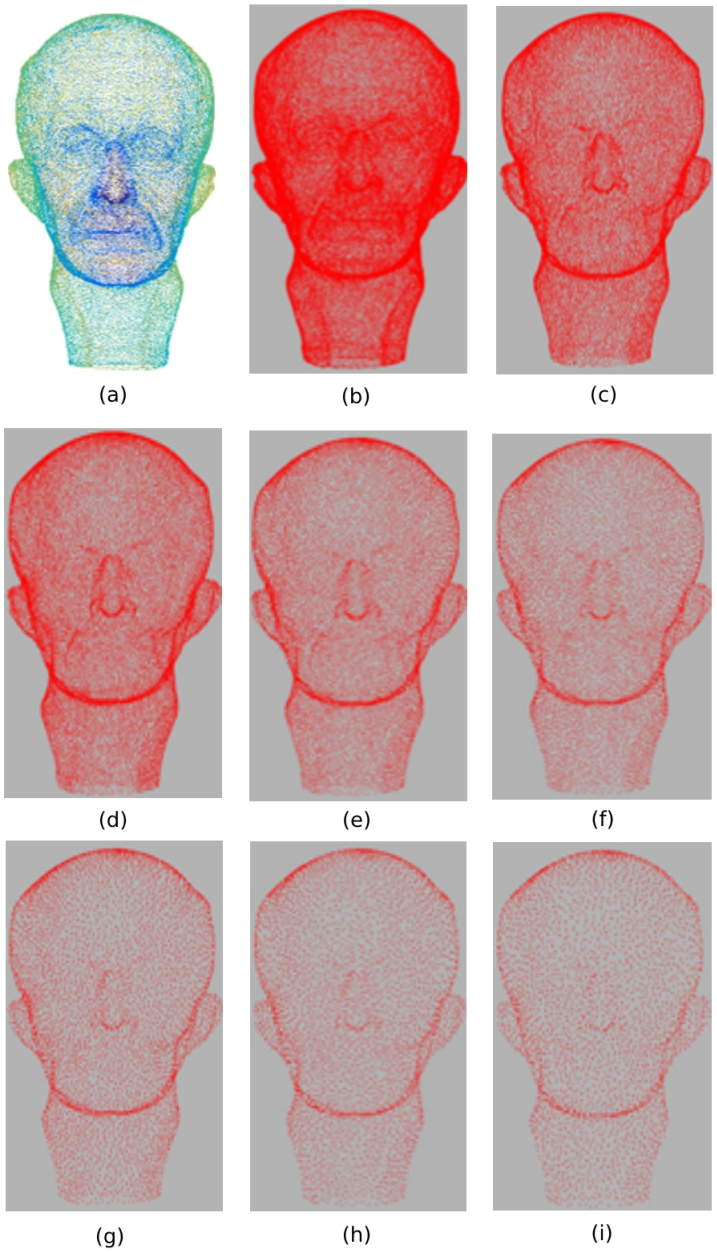


Figure 6.6: The Max Plank model (a) original 50,112 points, (b) simplified to 40,108 points, (c) simplified to 26,387 points, (d) simplified to 20,105 points (e) simplified to 12,761 points (f) simplified to 8,898 points, (g) simplified to 6,588 points, (h) simplified to 5,108 points, (i) simplified to 4,100 points.

produce a simplification with few details, points, and features, while large values can result in more details, points, and features. This may be observed in Figure 6.7. In all

our tests, we set $\lambda = 0.5$, which obtain the best results since this value maintains the balance between the number of points and the features.

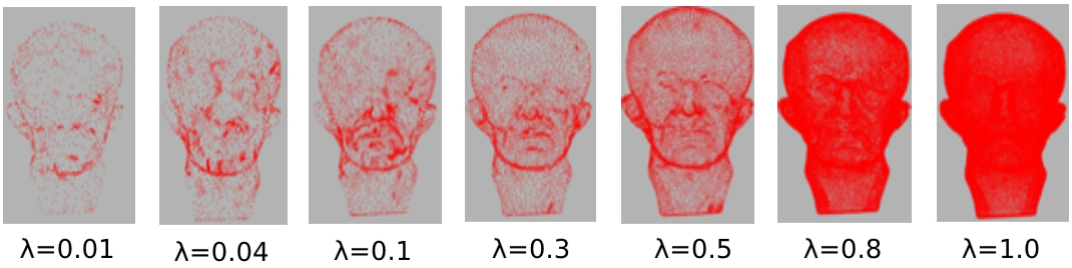


Figure 6.7: Variation of the parameter λ, with the parameter $\delta = 1.8$ fixed.

We establish the size S of the dictionary based on Figure 6.8. It shows the Mean Square Error (MSE) variation as the dictionary size increases. If the size of the dictionary increases, the MSE is low, but processing time increases. On the other hand, when the dictionary size is reduced, the MSE increases, but the processing time decreases. Our goal is to find a balance between a suitable dictionary size and low processing time.

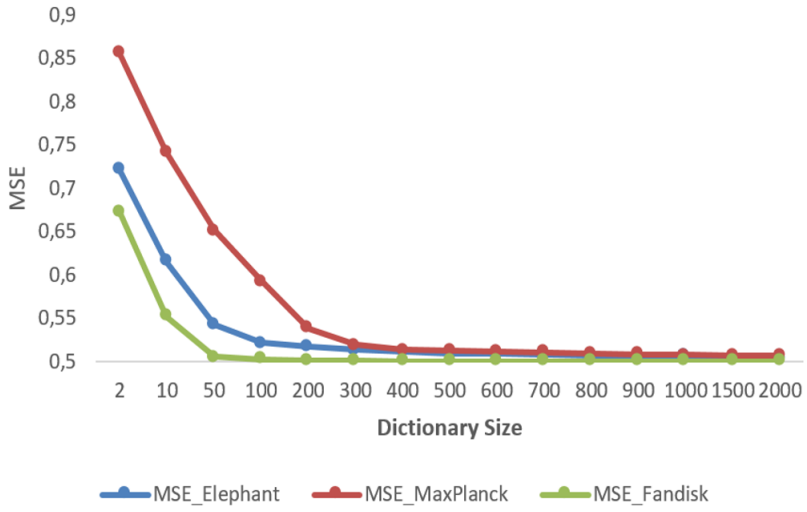


Figure 6.8: MSE variation vs. Dictionary size.

We plot the MSE of three different point clouds. Figure 6.8 shows that in the range of values between 200 and 400 the MSEs are low, and the size of the dictionary is not significant. In all the experiments, we fix the dictionary size $S = 200$, producing good results. The scale parameter δ is the only free user-defined parameter, and it is used for tuning the number of points to be removed.

6.4.2 | Quantitative Analysis

We choose the geometric error between the original and the simplified point cloud as a metric to evaluate the quality of the proposed simplification method, following Pauly et al. (PGK02). Similarly, we measure the maximum error distance and the average error distance between the original point cloud P, and the simplified point cloud, P'. We denote the surface of P as S, and the surface of P' as S'. The simplification error is estimated using the maximum error (Equation 6.12) and the average error (Equation 6.13) as follows:

$$\Delta_{\max}\left(S, S'\right) = \max_{\mathbf{p}_i \in S} |d(\mathbf{p}_i, S')| \tag{6.12}$$

$$\Delta_{avg}\left(S, S'\right) = \frac{1}{\|S\|} \sum_{\mathbf{p}_i \in S} |d(\mathbf{p}_i, S')| \tag{6.13}$$

For each point $\mathbf{p}_i \in S$, the geometric error $d(\mathbf{p}_i, S')$, is defined as the Euclidean distance between the sampled point $\mathbf{p}_i$ and its projection point $\bar{\mathbf{p}}_i$ on the simplified surface approximation S'. The metric is a variant of the Hausdorff distance. Since our method is mesh-free, we approximate the simplified surface S' using a least squares plane (LSP). To estimate the LSP, we select a set of neighboring points NH_i in P' closest to $\mathbf{p}_i$, using a Kd-tree data structure, while for points NH_i, we perform a PCA regression plane (L_{NH_i}), which represents the local approximation S'. i.e., $d\left(\mathbf{p}_i, S'\right) \cong d\left(\mathbf{p}_i, L_{NH_i}\right)$ (See Figure 6.9).

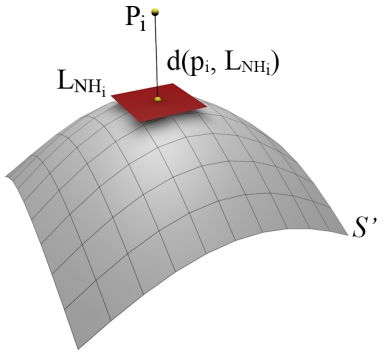


Figure 6.9: Local surface approximation and error computation as the distance from $\mathbf{p}_i$ to L_{NH_i}.

Table 6.1, shows the test models with the original numbers of points and the sampled points with different sampling rates (the value shown is the arithmetic average of the number of points resulting from the different methods for each simplification rate.). Table 6.2, shows the quantitative comparison between our method and the state-of-the-art

methods. Table 6.2 shows four simplification rates i.e. 5%, 10%, 20% and 50%. All five methods reduce the original number of points to a similar number of simplified points. Our method provides the most accurate simplification result of the five algorithms with respect to the average error metric Δ_{avg}. But taking into account the maximum error metric $\Delta_{\max}$, the Poisson disk mesh-based method is the best closely followed by our method.

Table 6.1: Test models with the original number of points and the sampling results at different simplification rates.

Models	Original Points	Sampled points 5%	Sampled points 10%	Sampled points 20%	Sampled points 50%
Bunny	35947	1797	3610	7186	17976
Elephant	24955	1246	2489	4991	12478
Gargoyle	25038	1253	2496	5008	12522
Horse	48485	2428	4872	9693	24247
MaxPlank	50112	2459	4892	9826	24569
Nicolo	50419	2519	5053	10082	25213
Fandisk	25894	1249	2480	4974	12437

As shown in Table 6.2, the CV, FRG and PSD methods produce similar results in terms of average surface error. The PSD method achieved relatively better results in terms of maximum surface error; however, a mesh structure must be used in the simplification. In some practical applications, only the 3D coordinate information is available, which limits the application of the PSD sampling method. The SGR method and our SDBS method achieved the best results in terms of average surface error, but the SDBS outperforms all other methods.

We compare the SDBS method with the other methods in accuracy and running time. Table 6.3 shows the running time and the number of preserved points of the proposed approach compared to six state-of-the-art methods. We simplify all the point clouds at similar simplification rate with all the algorithms. We run each method 10 times on each point cloud, and the average execution time is shown in Table 6.3. The programming language is also showed. It is worth noting that the simplification rate of our method is the lowest in the study (the bunny model was simplified from 35945 points to 4517 points, and the Elephant model was simplified from 24955 points to 2154). The SDBS keeps the balance between the sharp features and the point density in the data set. Although our method is implemented in MATLAB, Table 6.3 shows that it is computationally efficient compared to methods implemented in C++.

Table 6.2: Simplification results, comparison at different (S.R) sampling rates (5%, 10%, 20% and 50%) using the Δ_{max} and Δ_{avg} metrics between the proposed method and the state-of-the-art methods.

	Mesh based		**Point based**							
	PSD		**CV**		**FRG**		**SGR**		**SDBS**	
					S.R 5%					
Models	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}
Bunny	**0.005065**	0.000535	0.012529	0.000786	0.010989	0.000781	0.023727	0.001267	0.006019	**0.000517**
Elephant	**0.029524**	0.004453	0.071016	0.006473	0.079300	0.007481	0.076502	0.006723	0.032534	**0.004307**
Gargoyle	**0.520473**	0.096518	1.920498	0.129355	2.191607	0.130839	1.334782	0.222911	0.653588	**0.090725**
Horse	**0.003544**	0.000343	0.008435	0.000487	0.009263	0.000490	0.017894	0.001056	0.004340	**0.000322**
MaxPlank	**1.301519**	0.099681	3.702109	0.145190	2.802459	0.165132	4.725397	0.283958	1.618001	**0.087707**
Nicolo	**0.134143**	0.011021	0.370415	0.015898	0.331987	0.016816	0.292358	0.015014	0.149977	**0.010250**
Fandisk	**0.206912**	0.017887	1.251090	0.079158	0.441557	0.032766	0.627595	0.044314	0.215029	**0.016890**
					S.R 10%					
Models	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}
Bunny	**0.004430**	0.000308	0.008944	0.000426	0.007164	0.000416	0.012347	0.000496	0.005212	**0.000287**
Elephant	**0.022318**	0.002414	0.048641	0.003695	0.057984	0.003880	0.051137	0.003412	0.029524	**0.002312**
Gargoyle	**0.038192**	0.006216	1.283042	0.082879	2.162895	0.084856	0.968352	0.012522	0.054968	**0.005627**
Horse	**0.002506**	0.000190	0.006470	0.000261	0.005413	0.000245	0.008680	0.000369	0.003544	**0.000171**
MaxPlank	**0.961236**	0.059349	2.856882	0.084547	1.798308	0.085617	4.404936	0.135728	1.240950	**0.049349**
Nicolo	**0.106050**	0.006580	0.246437	0.009023	0.280581	0.009329	0.177455	0.008315	0.116171	**0.005710**
Fandisk	**0.149206**	0.009414	1.240782	0.050776	0.319061	0.020614	0.432045	0.016975	0.154839	**0.009010**
					S.R 20%					
Models	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}
Bunny	**0.003009**	0.000183	0.008241	0.000223	0.005630	0.000226	0.006616	0.000227	0.003475	**0.000151**
Elephant	**0.017644**	0.001367	0.039453	0.001812	0.046601	0.002064	0.035288	0.001663	0.019328	**0.001171**
Gargoyle	**0.306220**	0.040987	0.924313	0.047888	2.148394	0.050902	0.603874	0.069340	0.467759	**0.033819**
Horse	**0.002046**	0.000109	0.004798	0.000134	0.004340	0.000132	0.005011	0.000139	0.002506	**0.000087**
MaxPlank	**0.679693**	0.033580	2.000972	0.042923	1.301519	0.044349	1.359393	0.048901	0.961236	**0.024317**
Nicolo	**0.094854**	0.003902	0.217337	0.004568	0.189707	0.004759	0.157297	0.004514	**0.094854**	**0.003155**
Fandisk	**0.101366**	0.005030	1.240782	0.035560	0.242986	0.014598	0.411750	0.009057	0.130863	**0.004398**
					S.R 50%					
Models	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}	Δ_{max}	Δ_{avg}
Bunny	**0.002128**	0.000094	0.006143	0.000072	0.003885	0.000077	0.002747	0.000067	0.002747	**0.000060**
Elephant	**0.006863**	0.000717	0.028450	0.000592	0.024952	0.000624	0.017644	0.000516	0.017644	**0.000526**
Gargoyle	**0.228243**	0.024144	0.653588	0.017368	1.436299	0.017210	0.368031	0.020228	0.250028	**0.012773**
Horse	**0.002046**	0.000054	0.003544	0.000043	0.002506	0.000041	0.004340	0.000038	**0.002046**	**0.000033**
MaxPlank	**0.554970**	0.016812	1.359393	0.015499	0.877484	0.013778	0.784846	0.012912	**0.554970**	**0.009639**
Nicolo	**0.067072**	0.001986	0.142280	0.001626	0.106050	0.001638	0.067072	0.001366	**0.067072**	**0.001191**
Fandisk	**0.071676**	0.002220	1.237327	0.030615	0.155924	0.009728	0.383764	0.005845	0.092534	**0.001499**

Table 6.3: Comparison of simplification time and preserved number of points.

Method	Preserved number of points (Bunny)	Preserved number of points (Elephant)	Bunny running time (s)	Elephant running time (s)	Language
SDBS	4517	2154	21.223	15.186	MATLAB
DFPSA	4566	2872	56.156	26.220	- - -
FPUC	4644	2165	38.094	29.503	MATLAB
FRGR	4638	2164	9.5740	1.0030	MATLAB
WLOP	4572	2438	16.678	10.879	C/C++
GRID	4562	2154	0.6920	0.5170	C/C++
HCS	4644	2184	4.4590	3.1470	C/C++

6.4.3 | Visual Comparison

In order to compare visually the result of the studied algorithms, we simplify the models to approximately the same number of points with all methods. 6.10 10 shows the simplified results of the Bunny data set by different algorithms. The surface reconstruction results was computed using the Geomagic Studio software.

Figure 6.10 (b, c, e, f, g) shows how more points are retained in curved parts, while few points are kept in smooth parts. The simplification results of Figure 6.10d is uniform. All the methods present good reconstruction results but cannot reconstruct narrow features such as ears, with the exception of the DFPSA method, which shows only a small hole. The proposed method (Figure 6.10h) retained the most relevant features and details of the model, and the reconstruction does not present the problems observed with the other algorithms.

Figure 6.11 shows the simplification result for the Elephant data set with a high simplification rate. In Figure 6.11 (c, d, g) the GRID, WLOP, and DFPSA simplification methods preserve few points in smooth regions, and more points in feature regions such as legs, ears, trunk and tusks. The HCS, FRGR and FPUC simplification methods in Figure 6.11 (b, e, f), present problems to retain the global structure of its respective point clouds. Our method also preserves more points in feature areas, but it distributes the points evenly in smooth regions. Due to the high simplification rate, all the algorithm present failures, but our method is the best preserving the overall structure of the data set.

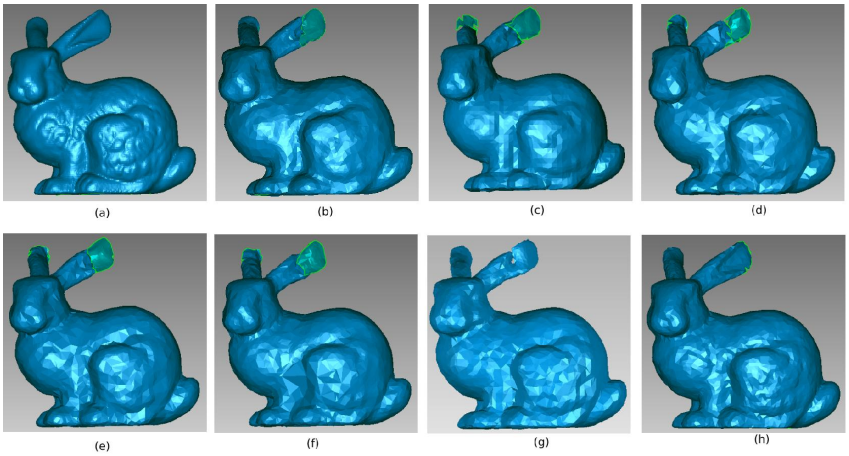


Figure 6.10: Point cloud simplification of the Bunny model. (a) The original data set, number of points = 35947, (b) HCS method, number of points = 4644, (c) GRID method, number of points = 4562, (d) WLOP method, number of points = 4572, (e) FRGR method, number of points = 4638, (f) FPUC method, number of points = 4644, (g) DFPSA method, number of points = 4566, (h) Proposed SDBS method, number of points = 4517. The image g is taken from (JLFL19).

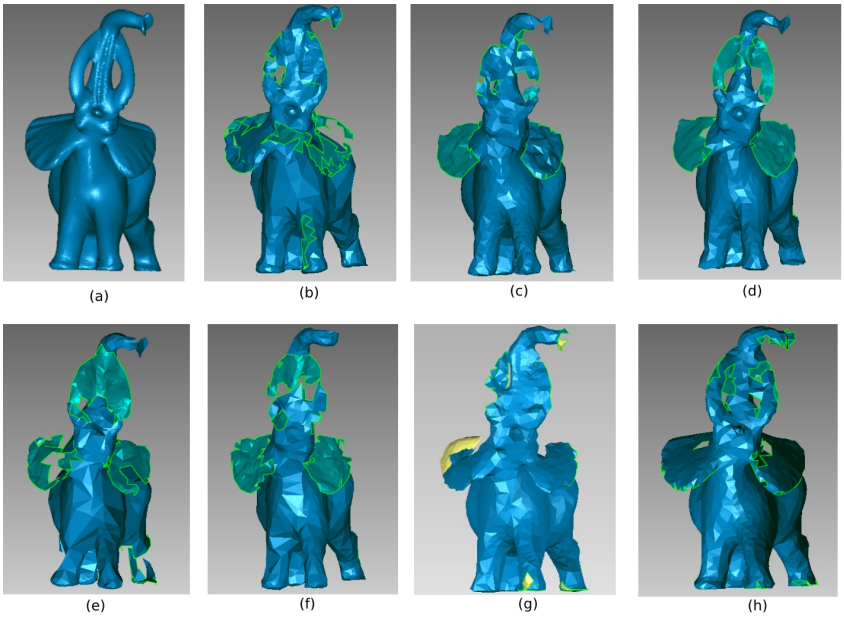


Figure 6.11: Point cloud simplification by different algorithms of elephant model. (a) The original data set, number of points = 24,955, (b) HCS method, number of points = 2184, (c) GRID method, number of points = 2154, (d) WLOP method, number of points = 2438, (e) FRGR method, number of points = 2164, (f) FPUC method, number of points = 2165, (g) DFPSA method, number of points = 2872, (h) Proposed SDBS method, number of points = 2154. The image g is taken from (JLFL19).

6.5 | Conclusion

In this chapter, we have presented a new method for point cloud simplification based on dictionary learning and sparse coding. The proposed method preserves the sharp features and produces evenly distributed points which are according to the geometry and surface appearance information. Our method uses the normal vector, curvature, and the position of the points as a component of a feature vector. The feature vectors of each point of the cloud are the input for a dictionary learning and sparse coding process for saliency detection. Compared to the method proposed in chapter 5, this method is a global and a generalized approach to the problem of saliency detection on point clouds. The proposed method uses the MDL principle to find the minimum code length to establish when a point is salient or not concerning the entire point cloud. i.e., one point is considered a salient point if its feature vector is reconstructed with many atoms from the dictionary. On the other hand, if the feature vector is reconstructed with few atoms, the point is not considered a saliency. The simplification is guided by global saliency using the sparse vectors resulting from the sparse coding process; we use its code length as an adaptive simplification ratio in different regions. The proposed method produces low simplification rates in saliency regions (borders, corners, high curvatures, valleys) and high simplification rates in relatively planar regions but keeping an appropriate density through the even distribution of points.

7

Feature Detection

In this chapter, we present a novel approach for detecting sharp features on point clouds. Detecting sharp features is an essential stage for structuring point sets as a previous step for feature lines reconstruction, surface estimation, and non-photorealistic rendering. Detecting sharp features in a point sets is not a simple task, because, without topological information that connects the points between them, only the intrinsic information brought by the raw points as discrete geometric properties is available to carry out the feature detection. The proposed algorithm estimates a neighborhood to each point in the cloud to compute a covariance matrix and uses the eigenvalues to estimate the surface variation and the sphericity. Next, using the smallest eigenvector as normal, we compute the orthogonal distance of each point to the neighborhood regression plane. Using the surface variation, the sphericity and the orthogonal distance, we construct a feature vector to every point in the cloud. We use the feature vectors as basis signals to carry out a dictionary learning process to get a trained dictionary; then, we perform the corresponding sparse coding process to get the sparse matrix. Finally, analyzing the sparse matrix, it is determined which feature vectors correspond to points that are candidates to be selected as sharp features. We show the robustness of our method on 3D objects with and without noise.

7.1 | Introduction

In recent years, 3D digital scanning devices have had significant development; these devices produce point clouds with thousands and even millions of 3D point. 3D point sets have been used for point-based graphics, surface reconstruction, geometry modeling, non-photorealistic rendering, remote sensing, photogrammetry, etc. Sharp features detection is a low-level computer vision process, which is the starting point for

help to understand and identify the primitive shapes and basic geometry from objects. Detecting sharp features is an essential step for scene understanding in robotic (LHLD18; SYC+18), surface segmentation (XZW+15), noise filtering (WCL+12), and surface reconstruction and simplification (DCSA+14). A sharp feature is the intersection of two or more surfaces, which is visualized as edge, valley, corner, or crest. In the absence of topological information that connects the points, detect sharp features in point sets is a non-trivial task; therefore, we use the intrinsic geometric properties given by the same points, to carry out the feature detection. The estimation of differential geometric properties is hard, even more, when the point set is contaminated with noise. Sharp feature detection is a significant challenge when the point set is noisy; this is because surface regions with a high level of noise can be confused with sharp features. In this chapter, we present a novel method for detecting sharp features on point sets with no topological connectivity. The pipeline of our method is shown in Figure 7.1. The proposed method is based on dictionary learning and sparse coding. First, to detect sharp features from 3D point sets, we analyze the geometrical properties of neighborhoods using the covariance matrix; we estimate the following features: normal, surface variation, and orthogonal projection to each point. Next, we associate with each point a vector with this feature. We construct a dictionary with the feature vectors and train the dictionary. Next, a sparse coding process is performed to find the most representative atoms that best reconstruct each feature vector, and analyzing the sparse matrix, we determine which feature vectors are candidates to be selected as sharp features. The proposed method has low computational complexity, is easy to implement, and is robust to noise.

7.1.1 | Contribution

We summarize the contributions of the proposed method as follows:

- The proposed method is easy to implement and fast, even with large point clouds.
- Our approach explores multiple scales to extract sharp features with certain levels of noise. In most cases, the detected features are one-point wide.
- Accurate and robust results are achieved by the method even for noisy data and objects with complex geometry.

7.2 | Related Work

In this section, we present the state-of-the-art methods to estimate features in mesh and point clouds. Feature detection is a fundamental problem that has been studied in the graphics and vision communities over the last decades and presents significant numbers of challenges to its solution. To continue, we present a brief review of feature detection methods.

7.2.1 | Mesh-based Methods

There have been many works for detecting sharp features on triangular meshes. In Xu et al. (XZW+15), surface segmentation and edge feature detection are presented. The method works on fractured fragment scanned from relics. The method extracts edges and feature lines from triangular meshes using a novel integral invariant to compute the surface roughness. Wang et al. (WCL+12) propose a robust method based on normal tensor voting to feature detection; the method can operate on noisy meshes. Wang (Wan06), aims to use bilateral filtering for recovering sharp edges. The proposed method is robust to noise, and no normal information is required, detecting the sharp edges by its dihedral angle. The method prevents shrinkage in regions without sharp features. Attene et al. (AFRS05) proposed a method called Sharpen&Bend to identify chamfer on triangular meshes to reconstruct sharp edges, by splitting the chamfer edges and then shifting the new vertices towards the intersections of planes. The algorithm does not recover sharp features in noisy models. Ohtake et al. (OBS04) use supported radial basis functions for global smoothing. They use differential geometry properties as the curvature tensor and its derivatives to characterize the sharp features. Then, projecting the mesh vertices onto the surface identifies the sharp features. The principal curvatures and directions classify the vertices of the mesh in edges and valleys.

7.2.2 | Point-based Methods

Zhang et al. (ZCL+18), use normal estimation to identify feature points, allowing to feature points to have multiple normals. This method is robust to noise and can handle point cloud with variations in the sampling density. Wang et al. (WWC+18) analyze the covariance matrix of normal vectors to find feature points. The feature points guide the segmentation of point clouds of industrial parts; they use the RANSAC algorithm to estimate tangent planes to different segmented surfaces; the method is robust to noise. Ni et al. (NLN+16), use the RANSAC algorithm, to calculate tangent planes and normal to

each point, then analyze the geometry properties of neighborhoods, to detect 3D edges and then trace feature lines. Cao et al. (CJXH17), extract sharp features from RGB-D images; the method is robust to noise and uses depth information and color to detect normal vectors and ridges/valleys sharp features. Bazazian et al. (BCRH15), analyze the eigenvalues of the covariance matrix to identify sharp edges. They use an agglomerative clustering algorithm to clustering the normals of the k nearest neighborhood and then estimate the final normal. Tran et al. (TCN+14), propose an automatic method for sharp feature detection. The algorithm can handle both meshes and point clouds; they use the projection distance of each point or vertex to the regression plane at the neighborhood to classify if a point is a sharp feature or not; The Otsu's method from image processing is applied to find an automatic threshold. Cao et al. (CWY+12), use tensor voting to smooth and oriented the normal vector, next use the orthogonal projection of difference between the central point of the neighborhood and the weighted average position of the points belonging to the neighborhood to classify when a point belongs to a sharp feature or not.

7.3 | Sharp Feature Detection

The proposed method is based on dictionary learning and sparse coding, as shown in Figure 7.1. We analyze the input point set using the covariance matrix to extract the local features; then, using the dictionary and the sparse representation matrix, the point set is analyzed globally to identify the sharp features.

7.3.1 | Low-level Features Estimation

To characterize the point set, we compute a descriptor for each point $\mathbf{p}_i$. There are several low-level features for each $\mathbf{p}_i$, for example, normals, curvatures, shape index, etc. In our method, we select the sphericity, total variation of surface, and the orthogonal distance. We choose these descriptors because they are related to the saliency features in the point cloud. The sphericity identifies corners, the total variation is related to the mean curvature and the orthogonal distance detect feature lines.

The classic way to estimate low-level features in a point set is through the local PCA, applied to each neighborhood $N_g(\mathbf{p}_i)$. We use a weighted version of PCA, with a covariance matrix Cm_i, established in Equation 7.1.

$$Cm_i = \frac{1}{k_i - 1} \sum_{i=1}^{k_i} w_i \left(\mathbf{p}_i - \bar{\mathbf{p}}\right) \left(\mathbf{p}_i - \bar{\mathbf{p}}\right)^T \tag{7.1}$$

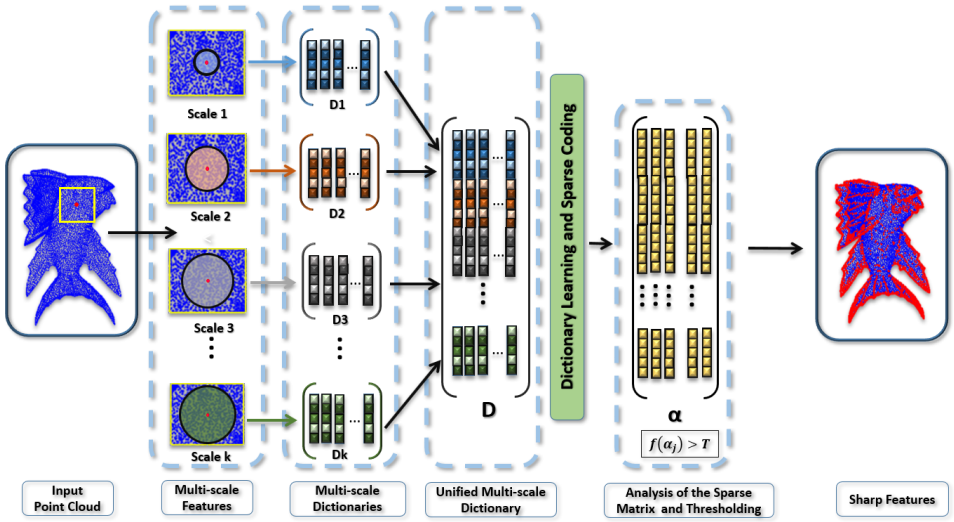


Figure 7.1: Pipeline of the proposed method.

where $\bar{\mathbf{p}} = \frac{1}{|N_g(\mathbf{p})|}\sum_{i=1}^{k_i}\mathbf{p}_j$, is the centroid of $N_g(\mathbf{p}_i)$ and $k_i = |N_g(\mathbf{p}_i)|$, is the cardinality of $N_g(\mathbf{p}_i)$. w_i is a weight estimated by $w_i = exp(-\frac{d^2}{k_i^2})$, $d = \|\mathbf{p}_i - \bar{\mathbf{p}}\|$, is the Euclidian distance. Next, we analyzed the eigenvalues $\lambda_0 \leq \lambda_1 \leq \lambda_2$ and eigenvectors $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ of the covariance matrix Cm_i.

The eigenvector $\mathbf{v}_0$, corresponding to the smallest eigenvalue λ_0, is the normal vector $\mathbf{n}_i$ at point $\mathbf{p}_i$. The total variation (PGK02), (PKG03) is defined in the Equation 7.2, the sphericity is defined in the Equation 7.3, and the projected distance (CWY+12; TCN+14) is defined in the Equation 7.4. We add the eigenvalue λ_0, to identify flat areas.

$$\sigma(\mathbf{p}_i) = \lambda_0/(\lambda_0 + \lambda_1 + \lambda_2) \tag{7.2}$$

$$Sp(\mathbf{p}_i) = \lambda_0/\lambda_2 \tag{7.3}$$

$$Pd(\mathbf{p}_i) = abs((\mathbf{p}_i - \bar{\mathbf{p}}_i)\cdot\mathbf{n}_i) \tag{7.4}$$

With these features defined, we build a four-dimensional feature vector $\mathbf{F}_i$, for every point $\mathbf{p}_i \in \mathbf{P}$ where $\mathbf{F}_i = (\lambda_0, Sp, \sigma, Pd)$.

We select the following low-level features for hole detection: the eigenvalues variation near edge points, defined in the Equation (7.5), the local density, defined in the

Equation (7.6), and the centroid difference, defined in the Equation (7.7).

$$epv(\mathbf{p}_i) = |\lambda_2 - 2\lambda_1| / \lambda_2 \tag{7.5}$$

$$Lp(\mathbf{p}_i) = k_i / (\mathbf{p}_i \cdot r^2) \tag{7.6}$$

$$Cd(\mathbf{p}_i) = \|\bar{\mathbf{p}} - \mathbf{p}_i\| \tag{7.7}$$

Where $r = \max_{\mathbf{p_j} \in N_s(\mathbf{p_i})} (\|\mathbf{p_j} - \bar{\mathbf{p}}\|)$, with these features, we build a three-dimensional feature vector $\mathbf{F}_i$, for every $\mathbf{p}_i \in \mathbf{P}$ where $\mathbf{F}_i = (epv, Lp, Cd)$. We choose these descriptors because they are related to the border points in the point cloud. In the case of a border point, the eigenvalues relation in Equation 7.5, obey $\lambda_2 \approx 2\lambda_1$; the local density (Equation 7.6) is low compare to points far away from a border and the centroid difference (Equation 7.7) has a high value compared to a point far away from a border.

7.3.2 | Dictionary Construction and Sparse Coding Model

Using the feature vectors defined in section 7.3.1, as data vectors $\mathbf{F}_i \in \mathbb{R}^{n \times 1}$, with $n = 4$ (number of low-level features for edges) and $n = 3$ (number of low-level features for holes). We construct the data matrix $\mathbf{F} = \{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_K\} \in \mathbb{R}^{n \times K}$, where $K = |P|$ is the number of feature vectors. According to sparse coding theory, a sparse coding matrix $\alpha \in \mathbb{R}^{S \times K}$ and a dictionary $\mathbf{D} \in \mathbb{R}^{n \times S}$ are defined. S is the number of atoms of the dictionary. Equation (3.13) is solved using the K-SVD algorithm (AEB06), getting the estimation of α and $\mathbf{D}$. Now $\mathbf{F}$ can be reconstructed as $\mathbf{F} = \mathbf{D}\boldsymbol{\alpha}$, obtaining the sparse representation of the data matrix $\mathbf{F}$ in the dictionary $\mathbf{D}$. Now we can find the sharp feature or hole points analyzing the sparse matrix $\boldsymbol{\alpha}$. It is noteworthy that the dictionary training process to both sharp feature detection and hole detection is carried out separately.

7.3.3 | Detecting Sharp Feature Points

Once we have the sparse coding matrix $\boldsymbol{\alpha}$, we analyze what vectors correspond to sharp features. Let $\boldsymbol{\alpha}_j$, $\mathbf{F}_j$, be two column vectors of the matrices $\boldsymbol{\alpha}$ and $\mathbf{F}$, respectively. A feature vector of a point is considered a sharp feature if its sparse representation $\|\boldsymbol{\alpha}_j\|_1$ has many non-zero elements, which implies that is necessary the linear combination of many atoms for representing this point correctly, and its sparse reconstruction error $\|\mathbf{F}_j - \boldsymbol{\alpha}_j\|_2$ produce a high residual. On the other hand, a feature vector of a point is

not considered a sharp feature, if its sparse representation $\|\boldsymbol{\alpha}_j\|_1$ has a few non-zero elements, this is, it can be represented by the linear combination of a few atoms and the sparse reconstruction error $\|\mathbf{F}_j - \boldsymbol{\alpha}_j\|_2$ produce a low residual.

With the stated previously, the first step is to sum the non-zero elements of each column of the matrix $\boldsymbol{\alpha}$, Equation (7.8). A score vector with these sums is built.

$$f(\boldsymbol{\alpha}_j) = \sum_{p=1}^{s} h(\boldsymbol{\alpha}_{p,j}) \;\; \forall_j = 1, 2, \ldots, S \tag{7.8}$$

$$h(\boldsymbol{\alpha}_{p,j}) = \begin{cases} 1\,, & \forall \boldsymbol{\alpha}_{p,j} \neq 0 \\ 0\,, & otherwise \end{cases} \tag{7.9}$$

Now it is necessary to establish a threshold value for $f(\boldsymbol{\alpha}_j)$, to decide if a point is located or not on a sharp feature. The threshold T is the maximum value in $f(\boldsymbol{\alpha}_j)$, Equation (7.10). T is the value of the feature vector with most rarity; we define a rarity as the highest number of non-zero elements in each column of the matrix $\boldsymbol{\alpha}$.

$$T = \max(f(\boldsymbol{\alpha})) * \beta \tag{7.10}$$

In the Equation (7.10), $\beta \in [0, 1]$ indicates the level of rarity established as a threshold value to classify points as belonging to a sharp feature or not. When T is estimated, we proceed to extract the sharp features Sf using the Equation 7.11.

$$Sf(i) = \{\mathbf{p}_i \,|\, f(\boldsymbol{\alpha}_j) > T, \mathbf{p}_i \in \mathbf{P}\} \tag{7.11}$$

7.3.4 | Detecting Hole Contour Points

The same analysis carried out in the previous section is carried out in this one; the only difference lies in the threshold choice, instead of choosing the maximum value in $f(\boldsymbol{\alpha})$, we select the minimum value i.e.

$$T = \min(f(\boldsymbol{\alpha})) * \beta \tag{7.12}$$

In the Equation (7.12), $\beta \in [0, 1]$ indicates the level of rarity established as a threshold value to classify points as belonging to a sharp feature or not. When T is estimated, we proceed to extract the contour points of the hole, Chp using the Equation 7.13.

$$Chp(i) = \{\mathbf{p}_i \,|\, f(\boldsymbol{\alpha}_j) < T, \mathbf{p}_i \in \mathbf{P}\} \tag{7.13}$$

Table 7.1: Detection time of the proposed method for different models.

Models	Points	Time (s)
Trim-Star	5192	2.22
Smooth-Feature	6220	2.54
Fandisk	6475	2.75
Block	8771	3.12
Sharp-Sphere	10443	3.19
Fish	14223	3.61
OctaFlower	17641	6.40
Twirl	26282	7.42
Blade	29275	7.78
Daratech	114983	89.68

7.3.5 | Multiscale Processing

In this section, we propose a multi-scale method to detect sharp features at multiple scales. In a single scale, the presence of noise in the point set can lead to identifying false features. We use multi-scale processing for reducing the number of false features, as it is shown in Figure 7.4. The feature vectors are estimated at multiple scales $(k = 1, 2, \ldots, 10)$, varying the neighborhood size (Figure 7.1). Next, we build a new feature vector with these scales. The new vector Sf_i has dimension $k * \mathbf{F}_i$, and finally, we proceed in the same way as in section 7.3.2.

7.4 | Results and Discussion

We have evaluated the proposed method on a set of models corrupted with different levels of noise. The results of our method are compared against other approaches considering methods based on point clouds and meshes. Our method is compared with point-based methods as surface variation (PKG03), mean curvature (WB01), normal vector (HG01), and projected distance (TCN$^+$14)and with mesh-based method (OBS04). All the experiments are run on a PC with Intel Core i7-2670QM CPU@2.20 GHz and 8GB RAM. For the implementation of the proposed method, we use Matlab R2016b programming environment. Table 7.1 shows the computation time of the proposed method for different models.

7.4.1 | Parameters Selection

There are three parameters in our method: the regularization parameter λ Equation (3.7), dictionary size S, and the threshold level of rarity β. The parameter λ, is the balancing between the data-fidelity and the regularization term, small values can produce more detailed edge points. In contrast, large values can cause loss of details; we can observe this in Figure 7.2. In all our tests, we set $\lambda = 0.01$.

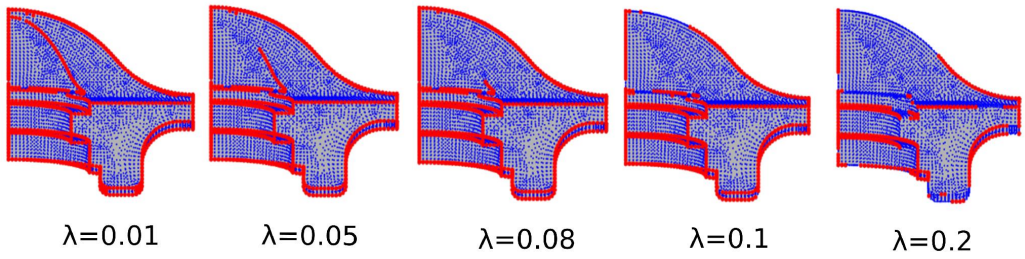


Figure 7.2: Variation of the parameter λ, with the parameter $\beta = 0.02$ fixed.

We establish the size S of the dictionary based on the analysis carried out in section 6.4.1 in chapter 6. In our tests, we found that setting the parameter $S = 100$, produced good results. The rarity parameter β, is the only free user-defined parameter that we use, but if we normalize the histogram of $f(\boldsymbol{\alpha}_j)$ vector, it is possible to obtain an automatic method described in (TCN$^+$14).

7.4.2 | Visual Comparison

The sharp features can be identified when there is a discontinuity between the normals (HG01) or the surface curvature (PKG03) (WB01). We select three methods based on this features to compare with the proposed method. The three methods are normal difference, the mean curvature variation, and surface variation, the fourth method is based on projected distance (TCN$^+$14). We can observe the results of these methods in Figure 7.3.

The three methods shown in Figure 7.3(a), 7.3(b), and 7.3(c), present similar performance; the edges are thick and detect points that do not belong to sharp features; this is because the normal and the curvature present problems to identify these types of features in contrast with the combination of features presented in the proposed method. Our method and the projected distance method have similar performance (Figure 7.3(d) and 7.3(e) because both include the projected distance. Still, we can observe that our method outperforms the projected distance method, detecting edges that projected dis-

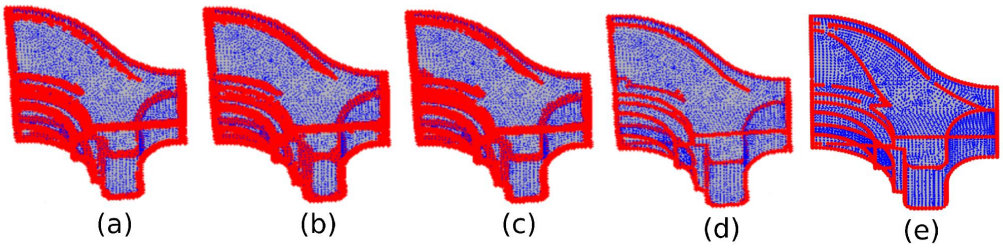


Figure 7.3: Sharp features extraction using different methods for the Fandisk model. (a) Normal. (b) Mean curvature. (c) Surface variation. (d) Projected distance. (e) Proposed method.

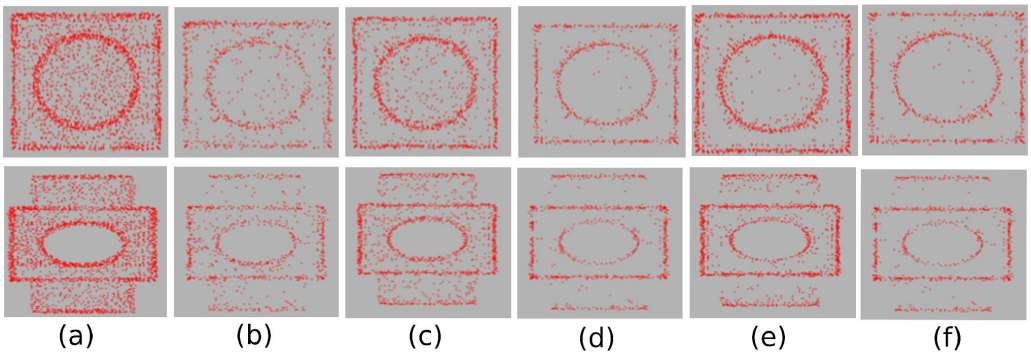


Figure 7.4: Block noisy model, contaminated with noise 5% of the mean distance between the points. (a)-(c)-(e) Projected distance. (b)-(d)-(f) Proposed method.

tance cannot detect. The performance of our method is better than the projected distance when it operates on noisy models, it can be seen in Figure 7.4. The multi-scale strategy proposed in our method, overcome the multi-scale projected distance method. Our method outperforms the projected distance method when both use only one scale ($k = 1$), as we can appreciate in Figure 7.4(a) vs. Figure 7.4(b). After k=10 scales, our method continues to outperforming the projected distant method, Figure 7.4(c) vs. Figure 7.4(d). Finally, our method reaches a good denoising result after $k = 10$ scales Figure 7.4(f), but the projected distance method reaches acceptable results after $k = 33$ scales Figure 7.4(e); our method continues to overcome the projected distance method. We compare our method with the mesh-based method proposed by Ohtake (OBS04), first-row in Figure 7.5. We observe how some sharp features are missing by the method proposed by Ohtake; on the other hand, our method detects the sharp features as it is shown second-row in Figure 7.5.

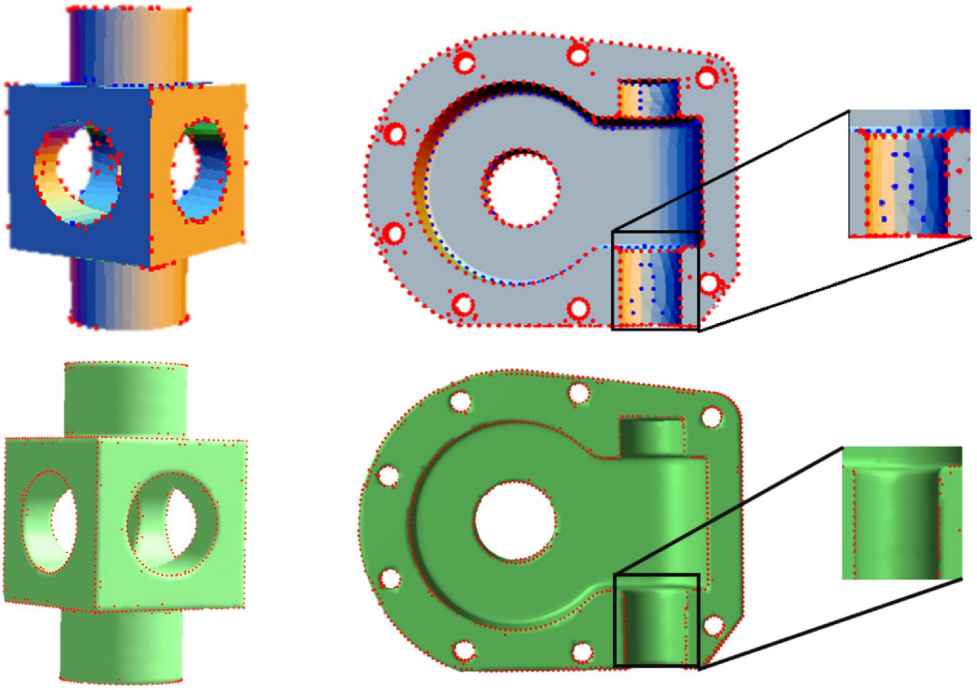


Figure 7.5: Block model, first row, Ohtake method. Second row, our method.

Figure 7.6 illustrates the sharp feature detection results with different free-form point sets. The subtle details and sharp features are faithfully recovered.

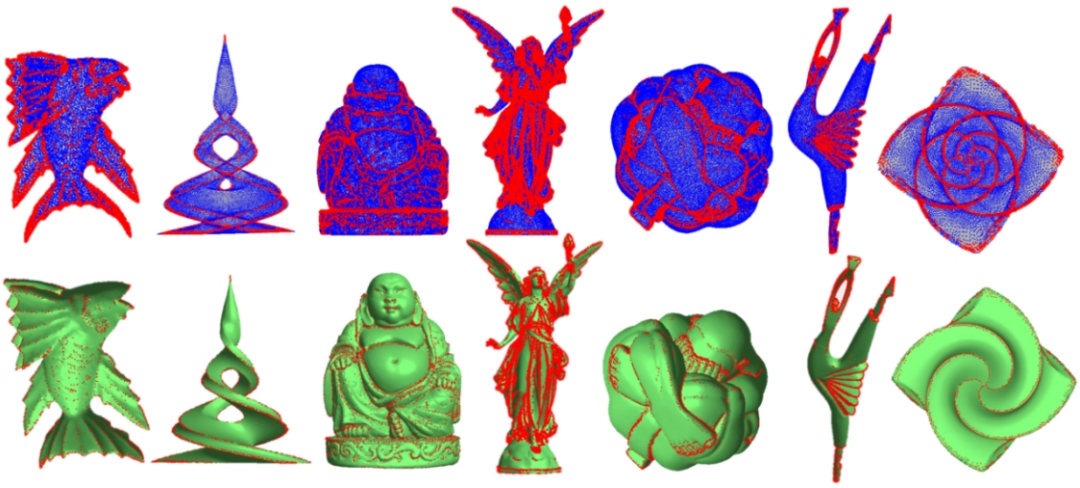


Figure 7.6: Results of Free-form object sharp feature detection using the proposed method.

7.4.3 | Robust to Noise

To show the robustness of the proposed method to different levels of noise, we show in Figure 7.7(b), (c) and (d) columns, the pyramid model contaminated with Gaussian noise with zero mean and standard deviation of 2%, 3% and 5% of the mean distance between the points, respectively. We observe how the proposed method keeps the sharp features in the pyramid model, despite the noise levels. Incorporating a multi-scale schema in our algorithm contributes significantly to detect the sharp features in the presence of noise.

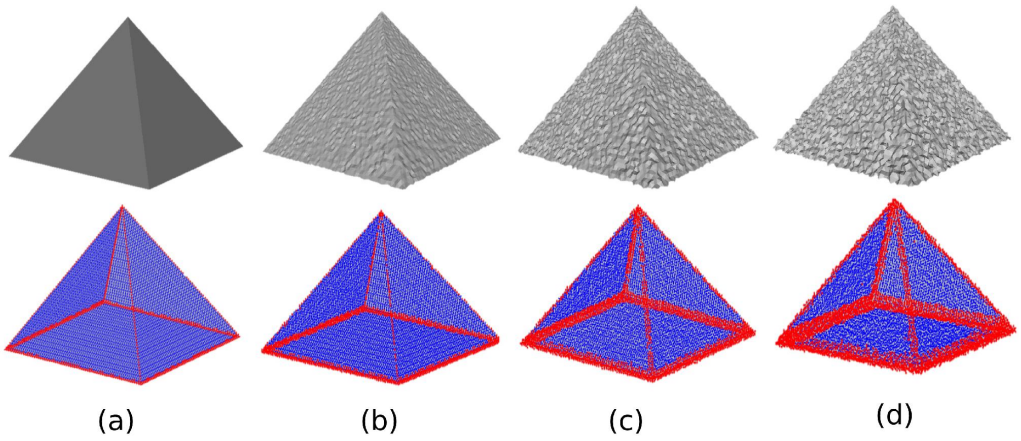


Figure 7.7: Pyramid model contaminated with noise 3% (b), 4% (c), and 5% (d) of mean distance between the points, first row. Clean model first column (a). Feature detection (a)-(d), second row.

Figure 7.8 illustrates our feature extraction results for point sets with 5% of noise. We observe the robustness of the proposed multi-scale scheme to detect sharp features in different noisy models.

In Figure 7.9, a set of mechanical models shows the results of applying the proposed method. The method performs well on shapes with sharp features.

7.4.4 | Hole Detection

We applied the proposed method to a variety of free-form surfaces. Figure 7.10 illustrates our hole detection results for the Bunny, Bimba, and Armadillo point sets. The holes in the models were created artificially, erasing points over different regions. A point belongs to a boundary if none of its neighbors are included inside the surface. In such a setting, boundary points, therefore, correspond either to the peripheral points of

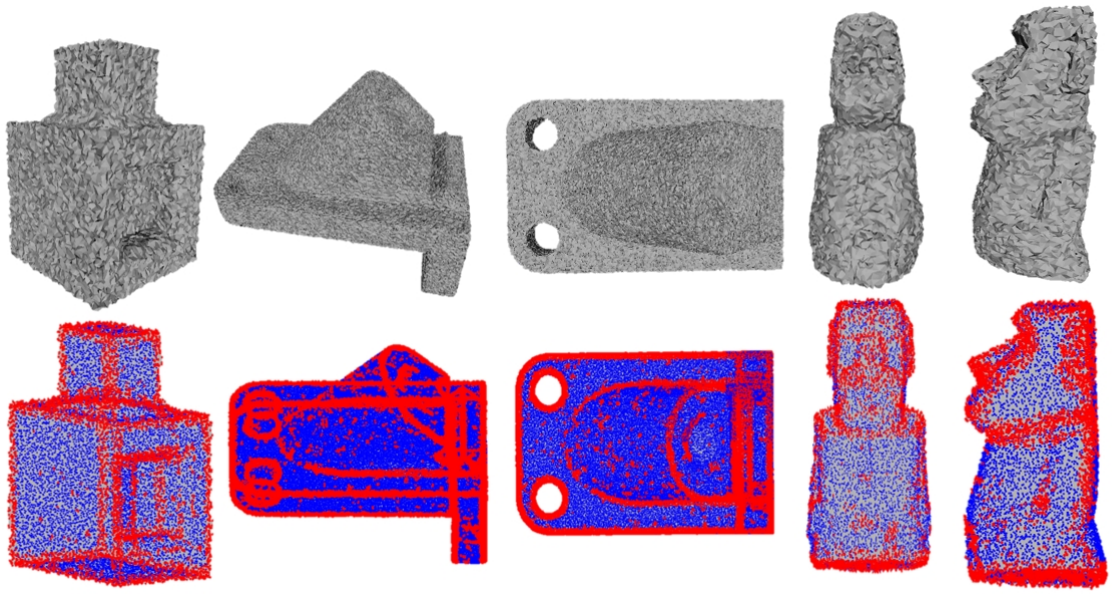


Figure 7.8: Sharp features extracted by the proposed method. Point sets with noise level 5% of the mean distance between the points.

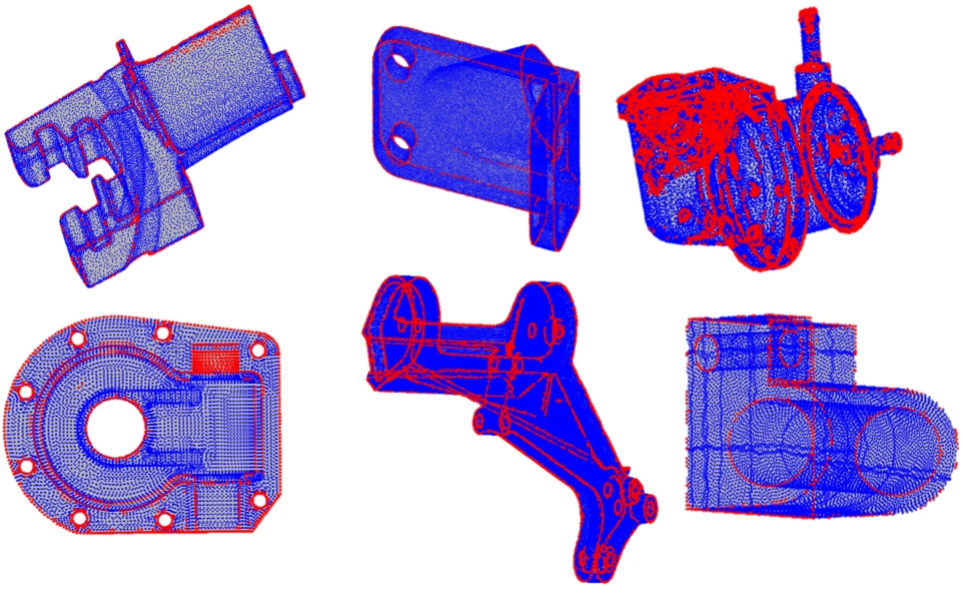


Figure 7.9: Sharp features extracted by the proposed method, over mechanical parts.

an open surface or to the boundary of holes. The proposed method can handle large point clouds as the Armadillo model with 172,387 points. We can observe how the proposed method detects the hole contour faithfully.

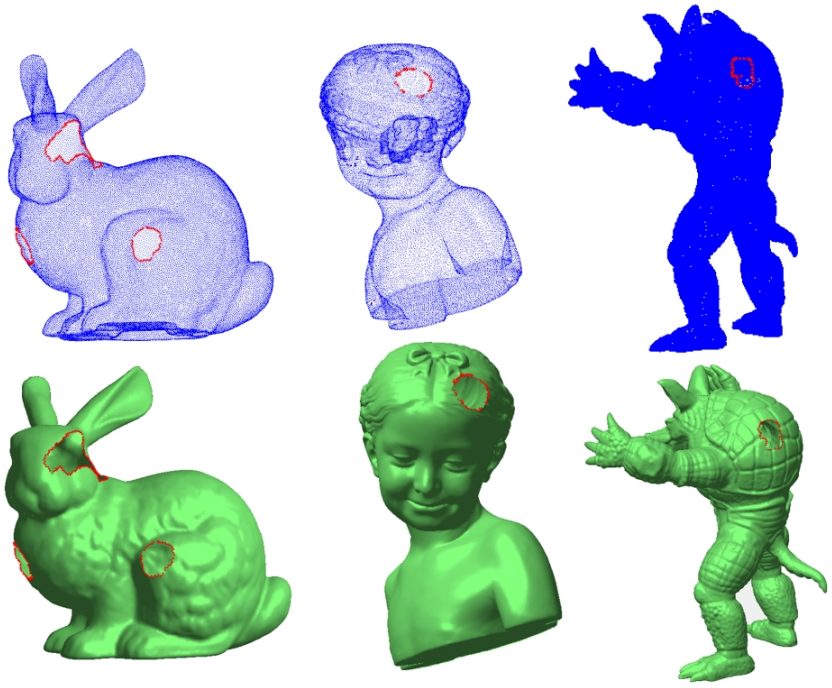


Figure 7.10: Hole contour extracted by the proposed method, over free form surfaces.

7.5 | Conclusion

In this chapter, we have presented a novel method for detection sharp features on 3D point sets based on dictionary learning and sparse coding. The dictionary is built at multiple scales to do the method robust to different levels of noise. We compute the neighborhood at different sizes to estimate the scales. The main contribution of our work is to use the sparse matrix to analyze the structure of the point set and determine what feature vectors correspond to sharp feature points. One point is considered belonging to a sharp feature if its feature vector is reconstructed with many atoms from the dictionary; on the other hand, if the feature vector is reconstructed with few atoms, the point is not considered a sharp feature. The previous is carried out counting the number of non-zero elements of each column of the sparse matrix and verifying if this count exceeds a threshold. The proposed method is simple to implement and relatively fast, is robust to different levels of noise, due to the dictionary learning process and the multi-scale scheme, outperforming other methods of the state-of-the-art.

8

Conclusions And Future Work

This thesis made several contributions to solving the existing problems in the different stages of the point cloud pre-processing. We have focused on sparsity-based modeling applied to 3d point clouds to tackle the problems presented in this work using the same solution framework. The thesis is divided into four parts, and we will show the conclusion and contributions to each part separately. Apart from specific conclusions, our overall conclusion is that we believe that sparse modeling is still a valuable tool that has shown great potential in the field of geometry processing, and can contribute to the development of techniques and algorithms in the surface reconstruction pipeline. It has proved to be versatile for modeling different problems involved in the point cloud pre-processing stages. Its robustness to outliers and noise, and the ability to describe and keep sharp features are attractive for many types of research in the field of geometric processing. Applying sparse models directly to point cloud is difficult due to the lack of topological information between the points, which makes it a challenging task; find the right representation for each problem in this thesis was the key to the success of the proposed methods. Using sparsity in the geometric processing field has been reflected in the increasing number of works that aim at enhancing it. We believe that our work can help to inspire solutions to other problems.

For specific conclusions, we compile a summary of each as follows. Using dispersion in both fidelity and regularization terms shows the capability of the sparse modeling to deal with sharp features and outliers in a unique solution. Combining the L1 median – L1 norm and its solution with the alternating minimization strategy using the proximal gradient algorithm, show convergence and stability, besides a simple implementation. Integrating both point positions and point normals, into the L1 median – L1 norm minimization framework, allow decoupling features from noise giving robustness to the method. Our method can operate over irregular surface sampling and can handle

impulsive noise achieving better results than the competing methods. The method depends on some empirical parameters σ_h and σ_d defined by the user, and which we tuned manually to obtain the desired results. How to determine these parameters continues to be a challenge and is a direction we are going to investigate in our future research. The implementation of a global solution for the cost function is another issue to be examined in future work.

We present a novel saliency detection method, relating the Minimum Description Length principle with the level of sparsity and the measure of the residual error. Based on the sparse representation length of the spatial patches, we use the code length as a metric of sparsity determining when a patch is salient or not respect to its surroundings patches. The proposed method detects saliency assuming the parsimony of data representation and establishing a relation between Information theory and the visual perception theory. For future studies, we plan to investigate how the incorporation of high-level information in the form of semantic cues into the point cloud saliency detection allows us to identify salience globally on the point cloud.

In point cloud simplification and feature detection, one of the main contributions of our work is to use the sparse matrix to analyze the structure of point set to gather evidence from local geometry to infer global properties about the objects. When the point cloud representation is very sparse, it means the current point cloud model has found the intrinsic structure in the input point cloud. In the context of point cloud simplification and feature detection, it means the model can help with better-sampling points and edges detection (removing distortions). Since sharp features are often sparse, sparsity formulations capture this observation well, reflecting in a versatile tool for modeling these types of problems. As future work, we propose examining ways determine automatically the choice of the regularization parameter λ and the size of the dictionary S. As future work, we plan to consider the automation of our method without user intervention to set the rarity parameter β. Furthermore, we will focus on using our method as a previous step to find feature lines on point sets and point cloud denoising.

A

Cost Function Derivatives in the Optimization

In this appendix we go through the mathematical expressions of the two terms of the cost function described in Section 4.4.2 and shown in Equation 4.5.

In Equation 4.5, the weighting factor $\psi(\cdot)$ is left out of this derivation for the sake of simplicity since it is just taken as a constant both in the objective function term and in its derivatives

A.1 | Fidelity Term Derivative for $\mathbf{n_0}$

The term E_f, as shown in Equation 4.3 when $\tau = 0$, and imposing the constraint $\|\mathbf{n}\| = 1$, we get

$$E_f = \sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left\| \mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) \right\| \psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j))\theta(\|\mathbf{p}_i - \mathbf{p}_j\|) \tag{A.1}$$

Now, using the constraint $\|\mathbf{n}\| = 1$, we compose the Lagrange form of E_f, and taking the derivative with respect to $\mathbf{n}$, we can obtain:

$$\frac{\partial E_f}{\partial \mathbf{n}} = \frac{\partial}{\partial \mathbf{n}} \left\{ \sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left\| \mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) \right\| \psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j))\theta(\|\mathbf{p}_i - \mathbf{p}_j\|) + \frac{\lambda}{2}(1 - \|n\|^2) \right\} \tag{A.2}$$

setting $\frac{\partial E_f}{\partial \mathbf{n}} = 0$ we get,

$$\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j))\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j)\|} (\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^T\mathbf{n} - \lambda\mathbf{n} = 0$$

Setting $w_i = \frac{\psi(\mathbf{n}^T(\mathbf{p}_i-\mathbf{p}_j))\theta(\|\mathbf{p}_i-\mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i-\mathbf{p}_j)\|}$,

$$\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} w_i(\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^T\mathbf{n} = \lambda\mathbf{n} \tag{A.3}$$

$$Cm(\mathbf{n})\mathbf{n} = \lambda\mathbf{n}$$

Where $Cm = \sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} w_i(\mathbf{p}_i - \mathbf{p}_j)(\mathbf{p}_i - \mathbf{p}_j)^T$

A.2 | Cost Function Derivative for $\mathbf{n_i}$

The E_f term in the cost function, as shown in Equation 4.5 is as follow and taking the derivative with respect to $\mathbf{n}$, we can obtain:

$$\frac{\partial E_f}{\partial \mathbf{n}_i} = \frac{\partial}{\partial \mathbf{n}_i}\left\{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left\|\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau\right\| \psi(\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)\right\} \tag{A.4}$$

$$\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left(\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau\right)(\mathbf{p}_i - \mathbf{p}_j)^T \frac{\psi(\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau\|}$$

Setting $\eta_i = \frac{\psi(\mathbf{n}_i^T(\mathbf{p}_i-\mathbf{p}_j)-\tau)\theta(\|\mathbf{p}_i-\mathbf{p}_j\|)}{\|\mathbf{n}_i^T(\mathbf{p}_i-\mathbf{p}_j)-\tau\|}$, we have

$$\nabla E_f(\mathbf{n_i}) = \sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \eta_i \left(\mathbf{n}_i^T(\mathbf{p}_i - \mathbf{p}_j) - \tau\right)(\mathbf{p}_i - \mathbf{p}_j)^T \tag{A.5}$$

The E_{reg} term in the cost function as shown in Equation 4.5, its derivative was calculated in section 4.4.2.2 using the Equation 4.11

A.3 | Cost Function Derivative for τ_i

The E_f term in the cost function, as shown in Equation 4.5 is as follow and taking the derivative with respect to $\boldsymbol{\tau}$, we can obtain:

$$\frac{\partial E_f}{\partial \boldsymbol{\tau}_i} = \frac{\partial}{\partial \boldsymbol{\tau}_i}\left\{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\right\| \psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)\right\} \tag{A.6}$$

setting $\frac{\partial E_f}{\partial \boldsymbol{\tau}} = 0$ we get,

$$-\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \left(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\right) \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\|} = 0 \Rightarrow$$

$$-\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\|}$$
$$+\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \tau_i \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\|} = 0 \Rightarrow$$

Solving this equation for τ_i^{k+1} yields the following recurrence equation

$$\tau_i \sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\|} =$$
$$\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i\|} \Rightarrow$$

$$\tau_i^{k+1} = \frac{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j))}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k\|}}{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k\|}}$$

Setting $\eta_i = \frac{\psi(\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k)\theta(\|\mathbf{p}_i - \mathbf{p}_j\|)}{\|\mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j) - \tau_i^k\|}$ and $h_j = \mathbf{n}^T(\mathbf{p}_i - \mathbf{p}_j)$, we have

$$\tau_i^{k+1} = \frac{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \eta_i h_j}{\sum_{\mathbf{p}_j \in \mathbf{N_g}(\mathbf{p_i})} \eta_i} \tag{A.7}$$

Bibliography

M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, January 2003.

Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45(2):15:1–15:33, March 2013.

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

M. Attene, B. Falcidieno, J. Rossignac, and M. Spagnuolo. Sharpen amp;Bend: recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):181–192, March 2005.

Oytun Akman and Pieter Jonker. Computing Saliency Map from Spatial Information in Point Cloud Data. In *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, pages 290–299. Springer, Berlin, Heidelberg, December 2010.

Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. L1 Sparse Reconstruction of Sharp Point Set Surfaces. *ACM Trans. Graph.*, 29(5):135:1–135:12, November 2010.

G. An, T. Watanabe, and M. Kakimoto. Mesh Simplification Using Hybrid Saliency. In *2016 International Conference on Cyberworlds (CW)*, pages 231–234, September 2016.

Simon Brezovnik, Miran Brezovnik, Simon Klančnik, Ivo Pahole, and Karl Gotlih. A Reverse Engineering Technique for Creating Virtual Robots. *Strojniški vestnik - Journal of Mechanical Engineering*, 55(6):347–355, 2009.

A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, June 2005. ISSN: 1063-6919.

D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo. Fast and Robust Edge Extraction in Unorganized Point Clouds. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, November 2015.

A. Borji and L. Itti. Exploiting local and global patch rarities for saliency detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–485, June 2012.

C. Bao, H. Ji, Y. Quan, and Z. Shen. Dictionary Learning for Sparse Coding: Algorithms and Convergence Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1356–1369, July 2016.

F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October 1999. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

Pál Benkő, Ralph R. Martin, and Tamás Várady. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839–851, September 2001.

John Branch, Flavio Prieto, and Pierre Boulanger. Automatic Hole-Filling of Triangular Meshes Using Local Radial Basis Function. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 727–734, June 2006.

Gerhard H. Bendels, Ruwen Schnabel, and Reinhard Klein. Detecting Holes in Point Set Surfaces. *Journal of WSCG*, 14, February 2006.

Igor Budak, Mirko Soković, Janez Kopač, and Janko Hodolič. Point data pre-processing based on fuzzy logic for reverse engineering modelling. *Journal of Mechanical Engineering*, 55(12):755–765, 2009.

Neil D. B. Bruce and John K. Tsotsos. Saliency Based on Information Maximization. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 155–162, Cambridge, MA, USA, 2005. MIT Press.

Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. State of the Art in Surface Reconstruction from Point Clouds. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*, pages 161–185. The Eurographics Association, 2014.

Marcel Campen, Marco Attene, and Leif Kobbelt. A Practical Guide to Polygon Mesh Repairing. *EUROGRAPHICS*, page pages 50, 2012.

U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, pages 397–405, October 2000.

Y.-P. Cao, T. Ju, J. Xu, and S.-M. Hu. Extracting Sharp Features from RGB-D Images. *Computer Graphics Forum*, 36(8):138–152, December 2017.

E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, February 2006.

Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling Points on 3D Surface Meshes. *ACM Trans. Graph.*, 31(4):29:1–29:12, July 2012.

Siheng Chen, Dong Tian, Chen Feng, Anthony Vetro, and Jelena Kovačević. Fast Resampling of Three-Dimensional Point Clouds via Graphs. *IEEE Transactions on Signal Processing*, 66(3):666–681, February 2018. Conference Name: IEEE Transactions on Signal Processing.

J. Cao, S. Wushour, X. Yao, N. Li, J. Liang, and X. Liang. Sharp feature extraction in point clouds. *IET Image Processing*, 6(7):863–869, October 2012.

Yonghui Chen and Lihua Yue. A method for dynamic simplification of massive point cloud. In *2016 IEEE International Conference on Industrial Technology (ICIT)*, pages 1690–1693, March 2016.

Somnath Dutta, Sumandeep Banerjee, Prabir K. Biswas, and Partha Bhowmick. Mesh Denoising Using Multi-scale Curvature-Based Saliency. In *Computer Vision - ACCV 2014 Workshops*, Lecture Notes in Computer Science, pages 507–516. Springer, Cham, November 2014.

Chinthaka Dinesh, Gene Cheung, Ivan V. Bajic, and Cheng Yang. Fast 3D Point Cloud Denoising via Bipartite Graph Approximation & Total Variation. *arXiv:1804.10831 [eess]*, April 2018. arXiv: 1804.10831.

Chaojing Duan, Siheng Chen, and Jelena Kovacevic. Weighted Multi-Projection: 3D Point Cloud Denoising With Tangent Planes. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 725–729, November 2018.

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, February 2014.

D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, January 2006.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, August 2007. Conference Name: IEEE Transactions on Image Processing.

Jean-Emmanuel Deschaud and François Goulette. Point cloud non local denoising using local surface descriptor similarity. In *ISPRS Technical Commission III Symposium, PCV 2010-Photogrametric Computer Vision and Image Analysis XXXVIII*, pages 109–114, int-Mandé, France, S, 2010.

D. L. Donoho. Compressed Sensing. *IEEE Trans. Inf. Theor.*, 52(4):1289–1306, April 2006.

James R. Diebel, Sebastian Thrun, and Michael Brünig. A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics (TOG)*, 25(1):39–59, January 2006.

Kris Demarsin, Denis Vanderstraeten, Tim Volodine, Dirk Roose, Kris Demarsin, Denis Vanderstraeten, Tim Volodine, Dirk Roose, Kris Demarsin, Denis V, Tim Volodine, and Dirk Roose. *Detection of Closed Sharp Feature Lines in Point Clouds for Reverse Engineering Applications*. Springer, Berlin, Heidelberg, 2006.

K. Engan, S.O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 5, pages 2443–2446 vol.5, March 1999. ISSN: 1520-6149.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, April 2004.

M. Elad, J. L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis*, 19(3):340–358, November 2005.

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

Thierry Guillemot, Andres Almansa, and Tamy Boubekeur. Non Local Point Set Surfaces. In *Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 3DIMPVT '12, pages 324–331, USA, October 2012. IEEE Computer Society.

Ran Gal and Daniel Cohen-Or. Salient Geometric Features for Partial Shape Matching and Similarity. *ACM Trans. Graph.*, 25(1):130–150, January 2006.

Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Transactions on Graphics*, 26(3):23–es, July 2007.

Shun Guo, Donghui Guo, Lifei Chen, and Qingshan Jiang. A L1-regularized feature selection method for local dimension reduction on microarray data. *Computational Biology and Chemistry*, 67:92–101, April 2017.

Yi Gong and Yuan-Fang Wang. Multi-View Stereo Point Clouds Visualization. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Song Wang, Kim Kyungnam, Bedrich Benes, Kenneth Moreland, Christoph Borst, Stephen DiVerdi, Chiang Yi-Jen, and Jiang Ming, editors, *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 281–290, Berlin, Heidelberg, 2011. Springer.

Stefan Gumhold, Xinlong Wang, and Rob S. MacLeod. Feature Extraction From Point Clouds. In *IMR - 10th International Meshing Roundtable, Sandia National Laboratories*, pages pp.293–305, 2001.

Yu Guo, Fei Wang, and Jingmin Xin. Point-wise saliency detection on 3D point clouds via covariance descriptors. *The Visual Computer*, pages 1–14, June 2017.

Kenneth M. Hanson. Introduction to Bayesian image analysis. In *Medical Imaging 1993: Image Processing*, volume 1898, pages 716–731. International Society for Optics and Photonics, September 1993.

Laurent Husson, Thomas Bodin, Giorgio Spada, Gaël Choblet, and Corné Kreemer. Bayesian surface reconstruction of geodetic uplift rates: Mapping the global fingerprint of Glacial Isostatic Adjustment. *Journal of Geodynamics*, 122:25–40, December 2018.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *ACM SIGGRAPH Computer Graphics*, 26(2):71–78, July 1992.

A. Hubeli and M. Gross. Multiresolution feature extraction for unstructured meshes. In *Proceedings Visualization, 2001. VIS '01.*, pages 287–294, October 2001.

Huiyan Han, Xie Han, Fusheng Sun, and Chunyan Huang. Point cloud simplification with preserved edge based on normal vector. *Optik - International Journal for Light and Electron Optics*, 126(19):2157–2162, October 2015.

Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics*, 28(5):1–7, December 2009.

Lei He and Scott Schaefer. Mesh denoising via L0 minimization. *ACM Transactions on Graphics (TOG)*, 32(4):64:1–64:8, July 2013.

Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics*, 32(4):65:1–65:8, July 2013.

Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics*, 32(1):9:1–9:12, February 2013.

Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, March 2001.

L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 1998.

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

Chunyang Ji, Ying Li, Jiahao Fan, and Shumei Lan. A Novel Simplification Method for 3D Geometric Point Cloud Based on the Importance of Point. *IEEE Access*, 7:129029–129042, 2019. Conference Name: IEEE Access.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 1986.

S. W. Jeong and J. Y. Sim. Saliency Detection for 3D Surface Geometry Using Semi-regular Meshes. *IEEE Transactions on Multimedia*, 19(12):2692–2705, December 2017.

P. Jenke, M. Wand, M. Bokeloh, A. Schilling, and W. Straßer. Bayesian Point Cloud Reconstruction. *Computer Graphics Forum*, 25(3):379–388, 2006.

Xue Jiao, Tieru Wu, and Xuzhou Qin. Mesh segmentation by combining mesh saliency with spectral clustering. *Journal of Computational and Applied Mathematics*, 329:134–146, February 2018.

Shixiang Jia, Caiming Zhang, Xuemei Li, and Yuanfeng Zhou. Mesh resizing based on hierarchical saliency detection. *Graphical Models*, 76(5):355–362, September 2014.

R. Kalboussi, M. Abdellaoui, and A. Douik. A spatiotemporal model for video saliency detection. In *2016 International Image Processing, Applications and Systems (IPAS)*, pages 1–6, November 2016.

Georgios A. Kordelas, Petros Daras, Patrycia Klavdianos, Ebroul Izquierdo, and Qianni Zhang. Accurate stereo 3D point cloud generation suitable for multi-view stereo reconstruction. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 307–310, December 2014.

Christof Koch and Tomaso Poggio. Predicting the visual world: silence is golden. *Nature Neuroscience*, 2(1):9–10, January 1999.

Youngmin Kim, Amitabh Varshney, David W. Jacobs, and François Guimbretière. Mesh Saliency and Human Eye Fixations. *ACM Trans. Appl. Percept.*, 7(2):12:1–12:13, February 2010.

Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free Projection for Geometry Reconstruction. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. event-place: San Diego, California.

Xuequan Lu, Wenzhi Chen, and Scott Schaefer. Robust mesh denoising via vertex pre-filtering and L1-median normal filtering. *Computer Aided Geometric Design*, 54:49–60, May 2017.

Xuequan Lu, Zhigang Deng, and Wenzhi Chen. A Robust Scheme for Feature-Preserving Mesh Denoising. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1181–1194, March 2016.

Manfred Lau, Kapil Dev, Weiqi Shi, Julie Dorsey, and Holly Rushmeier. Tactile Mesh Saliency. *ACM Trans. Graph.*, 35(4):52:1–52:11, July 2016.

Qi Li, Xiang Huang, Shuanggao Li, and Zhengping Deng. Feature extraction from point clouds for rigid aircraft part inspection using an improved Harris algorithm. *Measurement Science and Technology*, 29(11):115202, 2018.

M. Limper, A. Kuijper, and D. W. Fellner. Mesh Saliency Analysis via Local Curvature Entropy. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: Short Papers*, EG '16, pages 13–16, Goslar Germany, Germany, 2016. Eurographics Association.

Esmeide Leal and Nallig Leal. Point Cloud Denoising Using Robust Principal Component Analysis. In *Proceedings of the First International Conference on Computer Graphics Theory and Applications*, pages 51–58, Setúbal, Portugal, 2006. SciTePress - Science and and Technology Publications.

Nallig E. Leal, Esmeide A. Leal, and John W. Branch. Simplificación robusta de nubes de puntos usando análisis de componentes principales y algoritmos genéticos. *Avances en Sistemas e Informática*, 6(3):45–50, September 2009.

Nallig Leal, Esmeide Leal, and Sanchez-Torres German. A Linear Programming Approach for 3D Point Cloud Simplification. *International Journal of Computer Science,*, 44:60–67, 2017.

Shifan Liu, Jin Liang, Maodong Ren, JingBin He, Chunyuan Gong, Wang Lu, and Zehua Miao. An edge-sensitive simplification method for scanned point clouds. *Measurement Science and Technology*, 31(4):045203, January 2020. Publisher: IOP Publishing.

Xian-yong Liu, Li-zhuang Ma, and Li-gang Liu. P2: A robust and rotationally invariant shape descriptor with applications to mesh saliency. *Applied Mathematics-A Journal of Chinese Universities*, 31(1):53–67, March 2016.

Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 131–144, USA, July 2000. ACM Press/Addison-Wesley Publishing Co.

Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Shiyao Jin. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34(2):94–106, April 2010.

G. Leifman, E. Shtrom, and A. Tal. Surface regions of interest for viewpoint selection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 414–421, June 2012.

Xiuping Liu, Pingping Tao, Junjie Cao, He Chen, and Changqing Zou. Mesh saliency detection via double absorbing Markov chain in feature space. *The Visual Computer*, 32(9):1121–1132, September 2016.

Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh Saliency. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 659–666, New York, NY, USA, 2005. ACM.

Qingshan Liu and Jun Wang. L1-Minimization Algorithms for Sparse Signal Reconstruction Based on a Projection Neural Network. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):698–707, March 2016.

David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., USA, 2002.

Bin Liao, Chunxia Xiao, Liqiang Jin, and Hongbo Fu. Efficient feature-preserving local projection operator for geometry reconstruction. *Computer-Aided Design*, 45(5):861–874, May 2013.

Yin Li, Yue Zhou, Lei Xu, Xiaochao Yang, and Jie Yang. Incremental sparse saliency detection. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3093–3096, November 2009.

Abdelaaziz Mahdaoui, Aziz Bouazi, Abdallah Marhraoui Hsaini, and El Hassan Sbai. Comparison of K-Means and Fuzzy C-Means Algorithms on Simplification of 3D Point Cloud Based on Entropy Estimation. *Advances in Science, Technology and Engineering Systems Journal*, 2:38–44, December 2017. Library Catalog: astesj.com.

E. Mattei and A. Castrodad. Point Cloud Denoising via Moving RPCA. *Computer Graphics Forum*, 36(8):123–137, 2017.

Niloy J. Mitra and An Nguyen. Estimating Surface Normals in Noisy Point Cloud Data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 322–328, New York, NY, USA, 2003. ACM.

Boris Mederos, Luiz Velho, and Luiz Henrique de Figueiredo. Robust smoothing of noisy point clouds. In *Proc. Siam Conference on Geometric Design and Computing*, 2003.

Claudio Mura, Gregory Wyss, and Renato Pajarola. Robust normal estimation in unstructured 3D point clouds by selective normal space exploration. *The Visual Computer*, 34(6):961–971, June 2018.

Anass Nouri, Christophe Charrier, and Olivier Lézoray. Multi-scale mesh saliency with local adaptive patches for viewpoint selection. *Signal Processing: Image Communication*, 38:151–166, October 2015.

Andrew Y. Ng. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM. event-place: Banff, Alberta, Canada.

Huan Ni, Xiangguo Lin, Xiaogang Ning, Jixian Zhang, 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, September 2016.

Van Sinh Nguyen, Trong Hai Trinh, and Manh Ha Tran. Hole Boundary Detection of a Surface of 3D Point Clouds. In *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pages 124–129, November 2015. DOI: 10.1109/ACOMP.2015.12.

Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley Lines on Meshes via Implicit Surface Fitting. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 609–612, New York, NY, USA, 2004. ACM.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997.

A. C. Oztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.

Neal Parikh and Stephen Boyd. Proximal Algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

Rasmus R. Paulsen, Jakob Andreas Baerentzen, and Rasmus Larsen. Markov Random Field Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):636–646, July 2010.

Mark Pauly and Markus Gross. Spectral processing of point-sampled geometry. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 379–386, New York, NY, USA, August 2001. Association for Computing Machinery.

M. Pauly, M. Gross, and L.P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, 2002. VIS 2002.*, pages 163–170, October 2002.

Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale Feature Extraction on Point-Sampled Surfaces. *Computer Graphics Forum*, 22(3):281–289, 2003.

Junkun Qi, Wei Hu, and Zongming Guo. Feature Preserving and Uniformity-Controllable Point Cloud Simplification on Graph. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 284–289, July 2019. ISSN: 1945-7871.

G. Rosman, A. Dubrovina, and R. Kimmel. Patch-Collaborative Spectral Point-Cloud Denoising: Patch-Collaborative Spectral Point-Cloud Denoising. *Computer Graphics Forum*, 32(8):1–12, December 2013.

Nadejda Roubtsova and Jean-Yves Guillemaut. A Bayesian framework for enhanced geometric reconstruction of complex objects by Helmholtz stereopsis. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 3, pages 335–342, January 2014.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. In *Physica D*, volume 60, pages 259–268. Elsevier Science Publishers B. V., January 1992.

I. Ramirez and G. Sapiro. An MDL Framework for Sparse Coding and Dictionary Learning. *IEEE Transactions on Signal Processing*, 60(6):2913–2927, June 2012.

German Sanchez and John William Branch. Toward an automatic hole characterization for surface correction. In *Proceedings of the 6th international conference on Advances in visual computing - Volume Part I*, ISVC'10, pages 602–611, Las Vegas, NV, USA, November 2010. Springer-Verlag.

Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. Robust filtering of noisy scattered point data. In *Proceedings of the Second Eurographics / IEEE VGTC conference on Point-Based Graphics*, SPBG'05, pages 71–77, New York, USA, June 2005. Eurographics Association.

Kripasindhu Sarkar, Florian Bernard, Kiran Varanasi, Christian Theobalt, and Didier Stricker. Structured Low-Rank Matrix Factorization for Point-Cloud Denoising. In *2018 International Conference on 3D Vision (3DV)*, pages 444–453, September 2018. ISSN: 2475-7888, 2378-3826.

Muhammad Shoaib, Joono Cheong, Younghwan Kim, and Hyeonjoong Cho. Fractal bubble algorithm for simplification of 3D point cloud data. *Journal of Intelligent & Fuzzy Systems*, 37(6):7815–7830, January 2019. Publisher: IOS Press.

Guruprasad Somasundaram, Anoop Cherian, Vassilios Morellas, and Nikolaos Papanikolopoulos. Action recognition using global spatio-temporal features derived from sparse representations. *Computer Vision and Image Understanding*, 123:1–13, June 2014.

Shaobing Chen and D. Donoho. Basis pursuit. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44 vol.1, October 1994. ISSN: 1058-6393.

Batchimeg Sosorbaram, Tadahiro Fujimoto, and Norishige Chiba. Simplification of Point Set Surfaces using Bilateral Filter and Multi-Sized Splats. *The Journal of the Society for Art and Science*, 9(3):140–153, 2010.

László Szirmay-Kalos, Balázs Tóth, and Gábor Jakab. Efficient Bregman iteration in fully 3D PET. In *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pages 1–7, November 2014.

Bao-Quan Shi, Jin Liang, and Qing Liu. Adaptive simplification of point cloud using k-means clustering. *Computer-Aided Design*, 43(8):910–922, August 2011.

Ran Song, Yonghuai Liu, Ralph R. Martin, and Karina Rodriguez Echavarria. Local-to-global mesh saliency. *The Visual Computer*, 34(3):323–336, November 2016.

Ran Song, Yonghuai Liu, Ralph R. Martin, and Paul L. Rosin. Mesh Saliency via Spectral Processing. *ACM Trans. Graph.*, 33(1):6:1–6:17, February 2014.

E. Shtrom, G. Leifman, and A. Tal. Saliency Detection in Large Point Sets. In *2013 IEEE International Conference on Computer Vision*, pages 3591–3598, December 2013.

Yann Schoenenberger, Johan Paratte, and Pierre Vandergheynst. Graph-based denoising for time-varying point clouds. In *2015 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, July 2015. ISSN: 2161-2021.

Aparajithan Sampath and Jie Shan. Building Boundary Tracing and Regularization from Airborne Lidar Point Clouds. *American Society for Photogrammetry and Remote Sensing*, 7:pp. 805–812(8), 2007.

Qiqi Shen, Yun Sheng, Congkun Chen, Guixu Zhang, and Hassan Ugail. A PDE patch-based spectral method for progressive mesh compression and mesh denoising. *The Visual Computer*, 34(11):1563–1577, November 2018.

Yujing Sun, Scott Schaefer, and Wenping Wang. Denoising point sets via L0 minimization. *Computer Aided Geometric Design*, 35-36:2–15, May 2015.

Syeda Mariam Ahmed, Yan Zhi Tan, Chee Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7350–7355, October 2018.

Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA, September 1995. Association for Computing Machinery.

Pingping Tao, Junjie Cao, Shuhua Li, Xiuping Liu, and Ligang Liu. Mesh saliency via ranking unsalient patches in a descriptor space. *Computers & Graphics*, 46:264–274, February 2015.

T. Tran, V. Cao, V. T. Nguyen, S. Ali, and D. Laurendeau. Automatic method for sharp feature extraction from 3D data of man-made objects. In *2014 International Conference on Computer Graphics Theory and Applications (GRAPP)*, pages 1–8, January 2014.

F. P. Tasse, J. Kosinka, and N. Dodgson. Cluster-Based Point Set Saliency. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 163–171, December 2015.

F. P. Tasse, J. Kosinka, and N. A. Dodgson. Quantitative Analysis of Saliency Models. In *SIGGRAPH ASIA 2016 Technical Briefs*, SA '16, pages 19:1–19:4, New York, NY, USA, 2016. ACM.

Flora Ponjou Tasse, Jiří Kosinka, and Neil Dodgson. How Well Do Saliency-based Features Perform for Shape Retrieval? *Comput. Graph.*, 59(C):57–67, October 2016.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, January 1998.

Javier Turek. *Topics in Sparse Representation Modeling and Applications*. PhD thesis, Technion — Israel Institute of Technology, The address of the publisher, 3 2015.

W. E. Vinje and J. L. Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science (New York, N.Y.)*, 287(5456):1273–1276, February 2000.

J. Vollmer, R. Mencl, and H. Müller. Improved Laplacian Smoothing of Noisy Surface Meshes. *Computer Graphics Forum*, 18(3):131–138, 1999.

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

Kouki Watanabe and Alexander G. Belyaev. Detection of Salient Curvature Features on Polygonal Surfaces. *Computer Graphics Forum*, 20(3):385–392, September 2001.

Xiao-chao Wang, Jun-jie Cao, Xiu-ping Liu, Bao-jun Li, Xi-quan Shi, and Yi-zhen Sun. Feature detection of triangular meshes via neighbor supporting. *Journal of Zhejiang University SCIENCE C*, 13(6):440–451, June 2012.

Wang Weiming, Chao Haiyuan, Tong Jing, Yang Zhouwang, Tong Xin, Li Hang, Liu Xiuping, and Liu Ligang. Saliency-Preserving Slicing Optimization for Effective 3D Printing. *Computer Graphics Forum*, 34(6):148–160, September 2015.

Christopher Weber, Stefanie Hahmann, and Hans Hagen. Sharp Feature Detection in Point Clouds. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, pages 175–186, USA, June 2010. IEEE Computer Society.

Shengfa Wang, Nannan Li, Shuai Li, Zhongxuan Luo, Zhixun Su, and Hong Qin. Multi-scale mesh saliency based on low-rank and sparse analysis in shape feature space. *Computer Aided Geometric Design*, 35-36:206–214, May 2015.

Peng-Shuai Wang, Yang Liu, and Xin Tong. Mesh denoising via cascaded normal regression. *ACM Transactions on Graphics (TOG)*, 35(6):232:1–232:12, November 2016.

Jeremy M. Wolfe. Guided Search 2.0 A revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238, June 1994.

Aaron Wetzler, Guy Rosman, and Ron Kimmel. Patch-space Beltrami denoising of 3D point clouds. In *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, pages 1–5, November 2012.

Jinliang Wu, Xiaoyong Shen, Wei Zhu, and Ligang Liu. Mesh saliency with global rarity. *Graphical Models*, 75(5):255–264, September 2013.

Yuan Wang, Jiajing Wang, Xiuwan Chen, Tianxing Chu, Maolin Liu, Ting Yang, Yuan Wang, Jiajing Wang, Xiuwan Chen, Tianxing Chu, Maolin Liu, and Ting Yang. Feature Surface Extraction and Reconstruction from Industrial Components Using Multistep Segmentation and Optimization. *Remote Sensing*, 10(7):1073, July 2018.

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 (TOG)*, 33(2):18:1–18:12, April 2014.

Mingqiang Wei, Jinze Yu, Wai-Man Pang, Jun Wang, Jing Qin, Ligang Liu, and Pheng-Ann Heng. Bi-Normal Filtering for Mesh Denoising. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):43–55, January 2015.

Wei Xuan, Xianghong Hua, Xijiang Chen, Jingui Zou, and Xiaoxing He. A New Progressive Simplification Method for Point Cloud Using Local Entropy of Normal Angle. *Journal of the Indian Society of Remote Sensing*, 2018.

Linlin Xu, Ruimin Wang, Juyong Zhang, Zhouwang Yang, Jiansong Deng, Falai Chen, and Ligang Liu. Survey on sparsity in geometric modeling and processing. *Graphical Models*, 82:160–180, November 2015.

H. Xianfeng, C. Xiaoguang, Z. Fan, and G. Jianya. Side Ratio Constrain BasedPrecise Boundary Tracing Algorithm for Discrete Point Clouds. *TheInternational Archives of the Photogrammetry, Remote Sensing andSpatial Information Sciences, Volume. XXXVII, Part. B3b*, 2008.

Jiangyong Xu, Mingquan Zhou, Zhongke Wu, Wuyang Shui, and Sajid Ali. Robust surface segmentation and edge feature lines extraction from fractured fragments of relics. *Journal of Computational Design and Engineering*, 2(2):79–87, April 2015.

Shiyao Xiong, Juyong Zhang, Jianmin Zheng, Jianfei Cai, and Ligang Liu. Robust surface reconstruction via dictionary learning. *ACM Transactions on Graphics (TOG)*, 33(6):201:1–201:12, November 2014.

Shin Yoshizawa, A. Belyaev, and H.-P. Seidel. Smoothing by Example: Mesh Denoising by Averaging with Similarity-Based Weights. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 9–9, June 2006.

Sunil Kumar Yadav, Ulrich Reitebuch, Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics. *Computers & Graphics*, 74:234–243, August 2018.

Haifeng Yu, Rui Wang, Junli Chen, Liang Liu, and Wanggen Wan. Saliency computation and simplification of point cloud data. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pages 1350–1353. IEEE, December 2012.

Zhiqiang Yu, Taiyong Wang, Ting Guo, Hongbin Li, and Jingchuan Dong. Robust point cloud normal estimation via neighborhood reconstruction. *Advances in Mechanical Engineering*, 11(4):1687814019836043, April 2019.

J. Zhang, J. Cao, X. Liu, C. He, B. Li, and L. Liu. Multi-Normal Estimation via Pair Consistency Voting. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2018.

Jin Zeng, Gene Cheung, Michael Ng, Jiahao Pang, and Cheng Yang. 3D Point Cloud Denoising Using Graph Laplacian Regularization of a Low Dimensional Manifold Model. *IEEE Transactions on Image Processing*, 29:3474–3489, 2020.

Wangyu Zhang, Bailin Deng, Juyong Zhang, Sofien Bouaziz, and Ligang Liu. Guided Mesh Normal Filtering. *Computer Graphics Forum*, 34(7):23–34, 2015.

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, October 2011.

Lingli Zhu, Antero Kukko, Juho-Pekka Virtanen, Juha Hyyppä, Harri Kaartinen, Hannu Hyyppä, and Tuomas Turppa. Multisource Point Clouds, Point Simplification and Surface Reconstruction. *Remote Sensing*, 11(22):2659, January 2019.

Yitian Zhao, Yonghuai Liu, Yongjun Wang, Baogang Wei, Jian Yang, Yifan Zhao, and Yongtian Wang. Region-based saliency estimation for 3D shape analysis and understanding. *Neurocomputing*, 197:1–13, July 2016.

Yinglong Zheng, Guiqing Li, Shihao Wu, Yuxin Liu, and Yuefang Gao. Guided point cloud denoising via sharp feature skeletons. *The Visual Computer*, 33(6):857–867, June 2017.

Yinglong Zheng, Guiqing Li, Xuemiao Xu, Shihao Wu, and Yongwei Nie. Rolling normal filtering for point clouds. *Computer Aided Geometric Design*, 62:16–28, May 2018.

Kun Zhang, Shiquan Qiao, Xiaohong Wang, Yongtao Yang, and Yongqiang Zhang. Feature-Preserved Point Cloud Simplification Based on Natural Quadric Shape Models. *Applied Sciences*, 9(10):2130, January 2019.

Youwei Zhang, R. Rohling, and D.K. Pai. Direct surface extraction from 3D freehand ultrasound images. In *IEEE Visualization, 2002. VIS 2002.*, pages 45–52, October 2002. DOI: 10.1109/VISUAL.2002.1183755.

Faisal Zaman, Ya Ping Wong, and Boon Yian Ng. Density-Based Denoising of Point Cloud. In Haidi Ibrahim, Shahid Iqbal, Soo Siang Teoh, and Mohd Tafir Mustaffa, editors, *9th International Conference on Robotic, Vision, Signal Processing and Power Applications*, Lecture Notes in Electrical Engineering, pages 287–295, Singapore, 2017. Springer.

Yufu Zang, Bisheng Yang, Fuxun Liang, and Xiongwu Xiao. Novel Adaptive Laser Scanning Method for Point Clouds of Free-Form Objects. *Sensors (Basel, Switzerland)*, 18(7), July 2018.