Doctoral Dissertation
Doctoral Program in Electronics Engineering (33rd cycle)

Joint geometry and color denoising for 3D point clouds

Muhammad Abeer Irfan

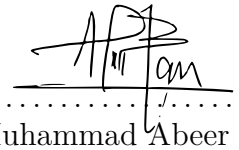
* * * * * *

Supervisors
Prof. Enrico Magli, Supervisor

Doctoral Examination Committee:
Thomas Maugey, Referee, INRIA
Prof. Simone Milani, Referee, Università degli studi di Padova
Prof. Alessandro Rizzo, Politecnico di Torino
Prof. Lorenzo Galleani, Politecnico di Torino
Prof. Cristian Perra, Università degli studi di Cagliari

Politecnico di Torino
2021

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.



Muhammad Abeer Irfan
Turin, 2021

Abstract

It is easy to collect a large amount of data of 3D scenes using 3D scanners in just a few seconds in the modern era. This data consists of an unordered collection of points in 3D space sampled from the surface of a 3D object; such data is called a 3D point cloud. This groundbreaking technology results in a growing number of applications such as 3D model reconstruction, 3D broadcasting, culture and heritage reconstruction, and navigation of unmanned vehicles. These modern point cloud acquisition sensors often suffer from noise. In this thesis, we propose several novel techniques for point cloud denoising to improve their quality.

The novelty in this research work is to study the correlation between the geometry and color attributes of a point cloud and then take advantage of this correlation and employ it as a dynamic tool for various tasks, i.e., geometry denoising, color denoising, and combined geometry and color denoising. The concepts of graph theory have been used. A graph is constructed for each point cloud, where the joint geometry/color graph of each point is a node and the weighted connections between them are the edges. These algorithms are then analyzed in terms of efficiency and performance. We devised two different approaches for point cloud denoising. One is based on the convex minimization problem, and the other is on spectral graph wavelets transforms.

Conventional point cloud denoising techniques are geometrical methods based on graph representation considering only the geometry attribute of a point in a point cloud. Few successful designs include the surface estimation of the point cloud from the noisy observation. Later, the projection of the noisy points renders the point cloud on the graph and employs the graph-based regularization technique. All these procedures result in a traditional optimization problem.

In this thesis, after an introduction section where the definition and applications of a point cloud are presented, the basic concepts of neural network and graph theory are presented. The current state-of-art techniques, a study of the relationship between the geometry and color of a point cloud, and some basic graph theory concepts are summarized in later sections. The project's progress from the construction of joint geometry and color k-NN graph to the architecture presentation is explained and analyzed in detail. Finally, the performance evaluations of the proposed techniques are reported, the quantitative and qualitative analyses are done to assess the obtained results. In particular, the point-to-point distance and cloud-to-mesh distance are taken into account to evaluate the geometry denoising results and perform comparisons with other approaches. For evaluating obtained results of color denoising scheme, Mean squared error (MSE) and Peak Signal-to-noise Ratio (PSNR) are considered. The proposed methods outperform the existing state-of-art methods.

The presented algorithms contribute to the new field of digital geometry processing and help to address the demand for efficient point cloud processing techniques.

Acknowledgements

First of all, I would like to thank Almighty ALLAH for all the blessings endowed upon me. Without ALLAH's desire, I wouldn't have managed this too far. At the end of my overwhelming experience as a Ph.D. student, I'd like to thank plenty of people for the vital role they played in my personal and professional growth. I consider myself very lucky to have such a kind mentor and supervisor, Prof Enrico Magli: he is a fantastic person, and working with him has been an excellent experience. I want to express my gratitude to him. He introduced me to the field of signal processing and particularly to the point cloud denoising using graphs. During my last three academic years, he always motivated and helped me to direct my research work and Ph.D. studies; I'd like to thank him for trusting me and giving me endless support.

Besides, I'd also like to thank all the researchers and colleagues of the Image Processing and Learning group (IPL). I am very grateful to my friends Syed Ali Hassan and Waqas Ahmad for their encouragement and support.

I would like to express my profound gratitude to all my family members from the core of my heart for their unlimited support. I am always indebted to my parents for their everlasting love, prayers, the education they gave me, and for letting me free to make my own decisions during my rational life. I also thank my wife for her support, understanding, and encouragement. Thanks forever for believing in me.

Finally, I would like to thank the Government of Pakistan, particularly the Higher Education Commission (HEC) Pakistan, for providing me the opportunity and financial support under the HRDI-UESTPs/UETs program. This is an excellent program to produce quality researchers to improve the universities' academic strength in Pakistan.

Dedication

Each challenging work requires efforts and also guidance of elders and close ones. I dedicate my sincere efforts to my sweet and loving father(late), mother, and wife. Their love, encouragement, and day & night prayers make me able to get success and honor. I also dedicate this thesis to my children (Muhammad Shaheer Irfan and Hareem Irfan) to whom Pakistan's future belongs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

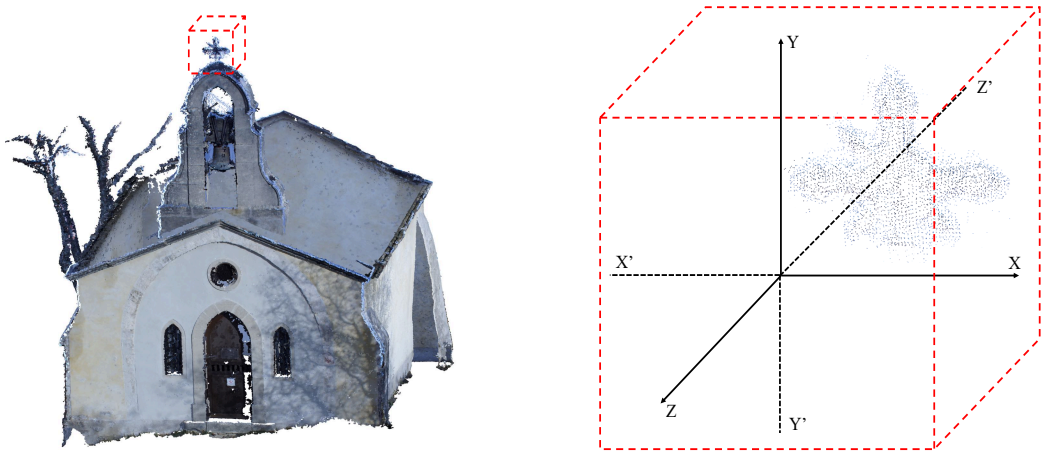
1.1 What is a point cloud?

A 3D point cloud [106, 2, 109] is a collection of points, and a novel primitive representation for an object, environments, buildings, and cities, etc depicted in Figure 1.1. The 3D geometry (x, y, z) represents the data of the sampled points of the analyzed object. There are some other essential attributes of the surface of the analyzed shape, such as color, transparency, and normals, etc. It has become prevalent in numerous field of research [2], such as object reconstruction [92, 104] and recognition [103] due to its versatility, simplicity, and robust representation capability. Furthermore, the point cloud does not need to maintain or store the topological consistency [59] and polygon-mesh connectivity [34]. Manipulating and processing of point clouds, therefore, shows better performance and inexpensive overhead. These striking advantages make point cloud processing a hot research topic.

1.2 3D acquisition techniques

There are many techniques to obtain the point clouds of 3D objects. Several approaches are employed depending on the size and geometry of the object and required scan's precision. Devices based on tactile measurement compute an object's surface using a touch probe by touching it with the surface. However, very precise contact-based measurement systems are typically slow. On the other hand, laser-based systems are slightly less reliable yet significantly faster, projecting a laser beam on the object's surface and using the triangulation principle to infer the distance of the object.

Scanners with a structured light project 2D patterns onto the object's surface and measure the 3D geometry of points by examining the pattern's deformation.



(a) (b)

Figure 1.1: Example of point cloud (a) A real-world point cloud model: House without roof (b) illustration of the points in 3D space.

The advantage of such scanners is their speed, and so they can be helpful in scanning moving objects. Laser radars belong to the family of scanning systems that determine the distance of light beam to an object based on a time-of-flight (ToF) signal. ToF is the time of a signal, i.e., laser pulse, to travel back and forth between the object and the laser scanner. Radar systems scan distant and massive objects such as buildings and mountains to preserve topographic data. X-rays have been used to build point clouds by Computed tomography (CT) scanners. In contrast to the other scanners, these scanners can scan solid objects containing hollow parts, such as human organs and engine blocks. Finally, 3D scenes can be reconstructed in photogrammetry-based systems from several images captured from distinct viewpoints. These systems are generally used in architectural applications and the film-making industry.

The point cloud obtained with these techniques, though, unavoidably suffers from significant noise in geometry and includes outliers [62, 134] due to the intrinsic noise of the acquisition device [84], sensors limitation [92], the artifacts in the scene, or reflective characteristics of the surface [138]. Hence, filtering operations on raw point clouds are required to obtain the point clouds that are suitable for additional processing.

1.3 Applications of point clouds

There are numerous applications of a point cloud. In the construction industry, the 3D point cloud data are employed for 3D model reconstruction and inspection of geometry quality. Besides these two major applications, the other applications

include:

- Tracking of construction progress.
- Analysis of building performance.
- Construction safety management.
- Topographic and bathymetric maps.
- Digital elevation models (DEMs).
- Digital terrain models (DTMs).
- Renovation of a building.
- Construction automation.
- Animation and virutal reality.
- Culture heritage application.
- Robot navigation.
- Autonomous vehicles.
- Medical imaging.
- Aeronautics.

1.4 Problem Statement

With the improvement of computer graphics, computer vision technology, and optical components, in addition to laser scanning sensors, economical low-cost RGB-D cameras have been developed, such as the Astra, Astra S, Astra Pro, Intel RealSense [115, 33, 15], and Microsoft Kinect [68, 57]. With the advent of RGB-D cameras, it is relatively easy to generate the point cloud of an object. However, the point cloud obtained with these cameras will have significant noise in geometry and color, exhibiting artifacts because of various viewing angles, reflective material or characteristics of the surfaces of the objects, light intensities, as well as the limitations of sensors [121]. Several types of geometric noise can occur during the acquisition of a point cloud that includes mixed pixels, range drift, wrap-around, multiple inconsistent scans, and smoothing noise. In this thesis, we have dealt with noise smoothing. The noise added both to the color and geometry attribute of a point cloud is additive noise. Since the Gaussian noise model is the most established model for additive noise [130], this thesis treats the Gaussian noise case mathematically. Alongside the geometry of objects and scenes, which is essential for many applications, surface colors and details play a vital role in several virtual and augmented reality applications [144]. The color is a necessary attribute of a point cloud, and it has been considered an essential feature for point cloud segmentation [126, 142, 14] and retrieval of 3D models [76, 116]. Noise in the colors of a point cloud may lead to a wrong segmentation. In recent works, geometry denoising has received a lot of attention. In contrast, there is just one work for denoising the color of a point cloud using Graph Laplacian regularizer (GLR) coupled with alternating direction method of multipliers [20]; yet, numerous applications are employing both the geometry and color attribute of a point cloud.

In current literature, the geometry of a point cloud is expressed as a graph, which is used for denoising by convex optimization [112]. In this paper, we propose the joint use of the geometry and the color attribute of the points in a point cloud to remove noise. We note that the color attribute is a powerful and very informative feature correlated with the geometry, as also observed in [14, 20, 79]. Knowledge of the color can be exploited to improve the denoising process for geometry noise. Indeed, geometry and color can be jointly employed to remove geometry *and* color noise.

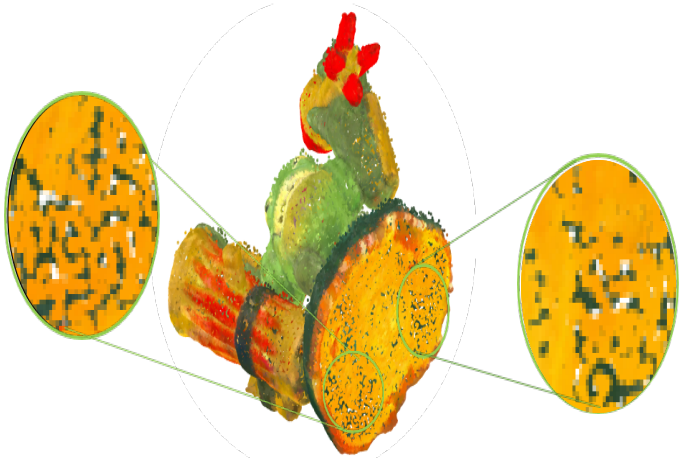


Figure 1.2: Green_monster model: Geometry denoised from geometry-only graph. The noisy points are moved towards their nearest neighbors rather than their correct positions, opening holes in the surface [112].

1.5 Contributions

Due to many point cloud applications in various research fields and among these applications, some are very sensitive and require the accurate acquisition of the point cloud, such as autonomous driving, virtual reality, etc. However, with the advancement in digital technology, there are still limitations in the sensors and extrinsic sources of noise such as reflection, refraction, etc., due to which accurate acquisition is not possible; hence, point cloud denoising has recently gained a lot of attention. Several techniques have been developed for the geometry denoising of point clouds based on the geometry information of the neighboring points. The disadvantage of these techniques is that holes are typically formed in the resulting denoised point cloud (see Fig. 1.2) as the actual location of a point is estimated based on the noisy geometry. This can lead to errors in estimating the local surface as the manifold's information is based only on geometry. Despite various applications associated with the color attribute of the point cloud as anticipated in

Section. 1.4 there are few works at all considering the problem of denoising the color attribute of a point cloud.

In this thesis, the focus of the research is to denoise the geometry and color attributes of a point cloud. The point cloud denoising problem faces some severe issues of creating artifacts, as discussed here. Many solutions are devised to solve the denoising problem with fewer adverse artifacts.

The first contribution towards point cloud denoising is providing statistical analysis to assess the *correlation* between color and geometry of a point cloud, and then move beyond the state-of-the-art technique and propose a novel approach employing the graph-based optimization, taking advantage of the correlation between color and geometry, and using it as a powerful tool for several various tasks as follow:

- Denoising of the color attribute of a point cloud using convex optimization.
- Denoising of the geometry attribute of a point cloud based on graph optimization.
- Denoising of combined geometry and color attribute point cloud exploiting the joint geometry and color k-NN graph.

The proposed technique is based on the notion that the correct location of a point also depends on the color attribute and not only the geometry of the neighboring points, and the correct color also depends on the geometry of the neighbors. The algorithm is explained in [49] and [48].

Our second contribution is to develop an algorithm that is non-iterative and less computationally intensive than the graph-based convex optimization. We propose a set-up for denoising point clouds based on spectral graph wavelet transform (SGW) that jointly exploits geometry and color to perform denoising of geometry attribute in graph spectral domain. The proposed technique is based on the design of a joint geometry and color graph that compacts smooth graph signals' energy in low-frequency bands. A soft-thresholding is then applied for the noise removal from the spectral graph wavelet coefficients. The denoising technique based on SGW is presented in [51].

A data-driven adaptive soft-thresholding-based technique is the third contribution of this thesis. In the algorithm described in [51], prior knowledge of standard deviation is required for performing soft-thresholding, which can work fine for the synthetic point clouds. Still, in real-world point clouds, the standard deviation is unknown. In this technique, we used adaptive soft-thresholding, in which the prior knowledge of standard deviation is not needed.

We use the same framework for solving the color denoising problem of a point cloud in a computationally efficient way with a different set of weights with respect to the approach discussed in [50] where the point cloud color is denoised using an iterative technique. Furthermore, the algorithm in [17] does not perform color denoising.

1.6 Organization of the thesis

The thesis is organized in the following chapters:

- Chapter 2 presents the classification of noise in point clouds. It also gives a detailed literature review of point cloud denoising techniques, discussing the advantages and the problems these algorithms face.
- Chapter 3 presents a comprehensive statistical analysis performed to study the correlation between geometry and color attributes of both the real-world and synthetic point cloud.
- Chapter 4 presents the preliminaries of the graph and construction of joint geometry and color k-NN graph construction. It also discussed the dataset used for the experimentation and the evaluation metrics for both the color and geometry denoising of the point cloud.
- Chapter 5 presents a detailed discussion on the experimental setup. The steps involved in implementing the proposed point cloud denoising algorithm using graph-based convex optimization are discussed in this chapter. In particular, various k-NN graphs have been constructed from the joint geometry and color with different proportion of contribution for different application, respectively. Then, Tikhonov regularization is applied to the constructed graph to estimate the exact geometry, color and combine geometry and color. It then shows extensive subjective and objective experimental results on synthetic and real-world point clouds to analyze the denoising algorithm.
- Chapter 6 presents the preliminaries of the spectral graph wavelets. It also discusses the experimental setup and detailed implementation of the denoising technique based on the graph wavelets. The selection of the data-drive adaptive soft-thresholding method is also presented in this chapter.
- Chapter 7 concludes the work. It also states the possible future work in this field.

The work presented in this thesis had led to the following publications:

1. Irfan, Muhammad Abeer, and Enrico Magli. "Point Cloud Denoising using Joint Geometry/Color Graph Wavelets." 2020 IEEE Workshop on Signal Processing Systems (SiPS). IEEE, 2020.
2. Irfan, Muhammad Abeer, and Enrico Magli. "3D point cloud denoising using a joint geometry and color k-NN graph." 2020 28th European Signal Processing Conference (EUSIPCO). IEEE, 2021.

3. Irfan, Muhammad Abeer, and Enrico Magli. "Exploiting color for graph-based 3D point cloud denoising." Journal of Visual Communication and Image Representation 75 (2021).

4. Irfan, Muhammad Abeer, and Enrico Magli. "Joint Geometry and Color Point Cloud Denoising Based on Graph Wavelets." IEEE Access 9 (2021): 21149-21166.

Chapter 2

Background on point cloud denoising

In the current literature, several categorizations of point cloud processing are available addressing various tasks such as point cloud segmentation, classification, 3D retrieval, and denoising [100, 25, 102, 37, 33, 126, 76], stating the increasing attention towards point cloud and its application. In this chapter, the particular study is presented on the denoising problem of the point cloud. The significance of point cloud denoising, along with the other possible approaches, is discussed here. The most related point cloud denoising algorithms highlighting the differences and mutual features among them are presented.

2.1 Point clouds

As discussed in the chapter 1, a point cloud is a 3D model acquired from a scanning process. To indicate the term point cloud in this thesis, we present the following definition.

Definition 2.1.1 (Point Cloud)**.** A point cloud $\mathcal{P}$ is a random collection of points $\{p_i\}_{i=1}^{n}$ in 3D Euclidean space, representing the surface of an object resulting from the scanning of that object.

Mathematically, it is appropriate to imagine a point cloud as a group of points sampled with adequate density from some connected and dense smooth surface $\mathcal{S}$[18]. In reality, the point cloud algorithms require assumptions about the sampling density. Otherwise, it is not easy to preserve the topology and geometry of the original object.

However, in practice, the mathematical viewpoint is usually unrealistic. The points in the point cloud are estimated with an inevitable error that results in the sampling of points close to the manifold rather than on the surface. Furthermore, the sampling density of a target object also depends on the scanning process.

Sometimes the term point cloud refers to the collection of points defining a curve in space.

2.2 Related Work

Point cloud denoising techniques can be classified into two categories: outlier removal and noise removal, i.e., surface smoothing techniques.

2.2.1 Outlier removal techniques

Outlier removal techniques can be further classified into two main approaches: statistical and model-based.

Statistical approaches

The main objective of the statistical-based algorithms is to eliminate the outliers by examining the distribution of each point to its neighbors or the number of neighbors to the corresponding point. The statistical-based method described in [108] measures the mean distance of each point from its corresponding points in proximity. The standard deviation σ and mean μ identify reasonable intervals of the global distances. The point is counted as an outlier whose mean distance is outside the defined range and removed from the point cloud.

Radius outlier removal (ROR) is an alternative approach that is extensively used for outlier removal. It is based on the number of neighbors, where each point's neighbors are calculated in a fixed radius. Those points with a number of neighbors are less than a specific threshold are considered outliers and eliminated from the point cloud. [106].

Model-based approaches

These techniques are based on the notion that the outliers usually are distant from an object's surface. The idea is to approximate the unknown surface of the object with some model, e.g., a plane, square or sphere, etc., and then measure the distance of each point to the surface of the model. The points with a significant distance are considered as outliers and removed from the point cloud [31]. A progressive plane technique is explained in [46], where a plane is estimated from the average normal and given 3D geometry coordinates. A least-square plane fitting method is used for the distance measure between each point to the plane and build a progressive plane; a hybrid technique is presented in [136] based on [46]. The problem with these algorithms is that details can be missed in the object with complex geometry as it is hard to approximate the complex areas with simple models.

2.2.2 Surface smoothing techniques

The surface smoothing technique is an intensive research domain and an essential part of the processing pipeline for many applications. The main surface smoothing techniques for 3D point cloud can be grouped into the following several classes, in which four categories are from [61].

Locally Optimal Projection-based methods

Locally optimal projection (LOP) is a projection-based method used for noise removal by adjusting each point's position in a noisy point cloud via different projection strategies.

Motivated by the idea of L_1 median [8, 117, 43, 45] from statistics, a parameterization-free projection operation i.e., LOP has been introduced in [74]. This method tries to iteratively project a subset of the input point onto the point cloud for noise removal. However, suppose there is a highly non-uniform input point cloud. In that case, LOP projection leads to non-uniformity, which is not desirable in normal estimation and shape features preservation and normal estimation. A variant of LOP, i.e., weighted LOP (WLOP), is used to generate an equally distributed point set presented in [43], in which locally adaptive density weights of an individual point are incorporated. In LOP, an essential factor that controls the smoothness is the support size h of the weight function. An appropriate selection of h is required for effective point cloud denoising. At a large value of h, the point cloud may shrink, while too small a value provides the least denoising results. To solve the aforementioned issue, another technique described in [137], where only the z coordinate of the point set is projected, and by employing re-projection, x and y coordinates are computed. This approach gains better smoothness and well preservation of the geometric structure.

To address the stated problems, the authors of [69] add a feature preservation weight to the formula that penalized significant variation in the geometry similarity into the term E_1 of L_1 median, while preserving the repulsion term E_2 unaltered to set the feature-preserving locally optimal projection (FLOP). Hence, LOP is an isotropic operator; thus, anisotropic LOP [44] remodels WLOP using an anisotropic weighting function better preservation of sharp features.

Since the above techniques are computationally expensive, an efficient variant of LOP is introduced by [99], where Gaussian Mixture Model (GMM) is produced by regularizing the hierarchical Expectation-Maximization (EM) algorithm is employed to describe the density of point cloud. Later, they re-defined and assessed the attractive force to get the continuous LOP (CLOP). Subsequently, CLOP has been applied to the Gaussians to speedup the process. Due to the use of local operators, these projection-based methods are also affected by over-smoothing [118, 143].

MLS-based methods

As point-based models do not provide a mathematical definition of a smooth surface, a local fitting surface is generally determined to approximate the sample points in various applications. The first work is proposed by [67], where a projection operator is devised based on moving least-squares (MLS). Later, numerous implementations of the projection method on point-based geometry are presented in [3]; their approach begins from a 3D point with a shorter distance from the point cloud. It then projects it onto the underlying surface locally approximated by a polynomial. To determine the appropriate polynomial, a reference plane is estimated to fit the surface around the points by minimizing the energy function defined based on a weighted least-squares technique. The reference plane is then employed to provide a bivariate polynomial based on a second least-squares fitting.

Following the work presented by [3], numerous improved MLS-based approaches are offered by the researchers [60, 73, 72, 66]. Most of these methods need to reconstruct a local surface from a specific neighborhood of the test point. Still, it is challenging to identify the MLS projection operation's right neighborhood, particularly for the point clouds having under-sampling density. The nearest points can be practically defined by calculating the Euclidean distances between the test and neighborhood points. The close neighborhood of the point commonly has a shorter Euclidean distance than the others. An alternative solution is to compute the nearest neighborhood of a test point and then calculate the average of k neighborhood [58]. Furthermore, the point sets obtained with scanning devices generally consist of noise and uneven samples, causing notable approximation errors.

The disadvantage of MLS is that finding a reference plane involves a non-linear optimization [59] which makes it computationally costly. Therefore, a simple projection method has been proposed in [4], in which the weighted average position of a point defines the reference planes. Since MLS is a low-pass filter that helps in smoothing the shape features of a point cloud, based on the M-estimator and MLS methods, a new smoothing operator is computed via an effective numerical optimization procedure in [80] that preserve the salient features.

Several extensions of MLS, such as robust MLS [107] and robust implicit MLS [89], have also been proposed. The robust moving least-squares technique is based on a forward-search model used to treat the noise, sharp features, and outliers. They incorporate the neighborhood of points into a subgroup of smooth regions of the surface, and then the MLS mechanism is performed on points. However, this approach is very time-consuming and requires very dense point clouds. In the above techniques, it can be observed that in the regions of high curvature, the plane fitting operation is unstable for a sampling rate below a threshold; thus, spherical fitting denoising based on MLS algebraic point set surfaces is proposed in [36], along with its variant [35] to improve the stability where planar MLS fails. These MLS-based methods can provide a smooth surface from significantly noisy input but are usually

prone to over-smoothing and are very sensitive to outliers [118, 143].

Sparsity-based surface smoothing

In the context of a point cloud's surface smoothing, several routines employ the statistical notions suited for the point cloud type.

A kernel-based clustering technique is introduced in [110] for point cloud denoising. Initially, the local likelihood L_i calculated on each point is acquired to determine the likelihood function L, modeling the probability of noisy point clouds. Then, the points are moved to high probability positions in an iterative way motivated by the mean shift procedure for point cloud smoothing. This technique performs better in denoising and robust to the outlier detection. Still, their algorithm misses the treatment of sharp features.

A Bayesian statistics was first used by [52] for the point cloud denoising. The authors obtained a $P(D|E)$ determination model, which defines the probability distribution of estimated point cloud E corresponding to the evaluated data D. Later, they set three prior probabilities, like smoothness priors, density priors, and sharp features. A posteriori probability $P(E|D)$ is maximized to remove noise and preserve the features. A robust statistical approach has been presented in [56] for filtering the point cloud. The authors employ an Iterative Least Squares (IRLS) framework to estimates the curvature and assign weights in a single iteration to update the neighborhood around every point. Finally, the estimated curvatures and the final weights are used to improve the normal. The robustly computed normals and curvatures may reject the outliers and point cloud denoising in a feature-preserving fashion based on the global energy minimization method.

[6] offered L_1-sparsity criterion for point cloud denoising. Initially, the orientation of the point is reconstructed by employing the re-weighted L_1 minimization process. The point's position is then restored by considering a local planarity paradigm to retain the shape features. However, this process can gain good outcomes; points on edge are often not recovered [118]. Since L_0 provides the sparse solution with respect to L_1, [118] presented a L_0 minimization process adopted for point cloud denoising by utilizing the same L_0 optimization method. It approximates the normals; then, the points are re-positioned along with directions of normal to preserve the sharp features better.

Neighborhood-based techniques

Neighborhood-based techniques identify a refined position of a point by utilizing the degree of similarities between a point and its neighborhood. The similarity measures have a significant impact on efficiency and effectiveness [111]. In the following schemes, the similarity can be determined by various attributes such as normals, positions of points.

The bilateral filter, which is an edge-preserving [125] and smoothing filter introduced initially in [91], is practised for the denoising of 3D meshes [28, 54, 64]. However, a mesh generation process is required for these techniques, which usually suffers from noise [81]. The bilateral filter is directly applied to the point cloud [78, 40, 113, 104]. The range and spatial weights are represented by w_r and w_s, respectively.

$$w_r = \exp\left(-\frac{(I(i,j) - I(x,y))^2}{2\sigma_r^2}\right).$$

$$w_s = \exp\left(-\frac{(i-x)^2 + (j-y)^2}{2\sigma_s^2}\right).$$

Here, the neighborhood of(x, y) is represented by (i, j), the intensity at (x, y) is denoted by $I(i, j)$, σ_r and σ_s are the standard Gaussian function.

To overcome the time complexity, Xu et al. [135] substituted the weight in the bilateral filter with a binary function shown in Eq. 2.1 to attain good performance. Still, these filters consider the point clouds containing the intensity attribute. Consequently, normal [35,40–43], as an essential attribute of the point cloud used in the bilateral filter, whose weight is set as a function of spatial position and normal of points, shown in Eq. 2.2.

$$w_r = \begin{cases} 1 & |I(x,y) - I(i,j)| \leq 3, I(x,y) \neq 0 \\ 0 & |I(x,y) - I(i,j)| > 3, I(x,y) \neq 0. \end{cases} \tag{2.1}$$

$$w = f_1[d(p,q)] \,\mathrm{x}\, f_2[c(\mathbf{n}_p, \mathbf{n}_q)]. \tag{2.2}$$

where, the f_1 and f_2 are the Gaussian function with σ_{f_1} and σ_{f_2}. The normal of the corresponding points p and q are represented as $\mathbf{n}_p$ and $\mathbf{n}_q$, respectively. The Euclidean distance $d(p, q)$ defines the distance between point p and its neighboring point q. $c(\mathbf{n}_p, \mathbf{n}_q)$ represents the normal relations at point p and q, like $\|(p-q).\mathbf{n}_p\mathbf{n}_q\|$ [53], $(\mathbf{n}_p - \mathbf{n}_q)^2$ [127], inner product of normal vector $\langle \mathbf{n}_p, \mathbf{n}_q \rangle$ [81], and $\mathbf{n}_p.(\mathbf{n}_p - \mathbf{n}_q)$ [82].

Inspired by the mean shift filtering technique for 2D images, Hu et al. [42] presented a filtering algorithm based on a 3D mean shift anisotropy by considering various attributes of a point cloud such as position, vertex, curvature, and normal. They determine the neighborhood of each point via an adaptive neighboring searching procedure based on clustering. Lastly, they employed the trilateral filter on the computed adaptive neighborhoods to denoise the point cloud and preserved the features. Another algorithm is proposed in [111] based on identical bilateral filtering, where the non-local means employed for 2D images [9] have been extended for 3D point cloud denoising. They established a new non-local similarity measure that considers the local square neighborhoods around the point. This technique provides more precise denoising results and carries better feature preservation.

The algorithm described in [47] considers the intra-patch similarity and color information for designing a robust non-local filter. This technique initially removes the outliers and then carries out a smoothing process and maintains the sharp features. An effective method is presented in [129] performs both the outlier removal and surface smoothing. This paradigm removes the sparse outliers based on the average local neighborhood and relative deviation of a local neighborhood. Then, the normal filtering in an iterative manner is performed. Furthermore, the points are updated to match the filtered normal.

PDEs-based techniques

One of the essential tools extensively used in computer graphics and computer vision is PDEs (Partial Differential Equations) that have been effectively applied to various tasks like image and mesh denoising. PDEs-based point cloud denoising methods are the extension of filtering of triangular meshes.

A PDE-based technique is presented in [11], where the point cloud filtering is done by solving an anisotropic geometric diffusion equation. This technique achieves good smoothing results and also preserves the shape features. An extended work of [123] is presented in [63] and considers the principal and directional curvature and the Weingarten map to get the anisotropic geometric curvature to denoise the point cloud. A Weingarten map is produced from the directional curvature and calculates the principal curvature's eigenvectors and eigenvalues. Finally, a modified Laplacian is obtained by applying the curvature information. This technique can also find the features such as edges.

An anisotropic curvature flow method based on directional curvature and covariance analysis is proposed in [133]. Their method is anti-shrinking, anti-point drifting and provides better performance in surface smoothing and preserving shape features. To extend various procedures on 2D images into point clouds, an arbitrary weighted graph is constructed in [77], which considers the local neighborhood information. In addition, a PDEs morphological and p-laplacian operator [120] was employed in these arbitrary graphs i.e., PDEs on weighted graphs [27] for the point cloud denoising.

The drawback of PDEs-based methods is that they require the computation of partial differential equations that are time-consuming for the 3D point cloud denoising, which scales linearly with the point cloud's complex structure.

Signal processing based techniques

Signal processing techniques can also be utilized for point cloud denoising. Based on the method described in [122] where the mesh is filtered by employing the Laplacian operators, a filtering operator is defined in [71] comprising three features: geometric appearance, non-shrinkage, and locality.

A discrete estimation of the Laplacian to point cloud is introduced in [94]. Their approach balances the shrinkage using the displacement of a point $p's$ neighborhood to deal with shrinkage. Encouraged by Fourier transform, the algorithm presented in [93] used spectral processing for the point cloud filtering. A discrete Fourier transform is applied to achieve a spectral decomposition of a point cloud. The filtering is then carried out by manipulating the frequency spectrum by a Wiener filter.

Recently, graph-based techniques have been used for denoising point clouds. The conventional approach constructs a k nearest neighbor (k-NN) graph as it makes geometric structure explicit [128]. The points in a given point cloud are considered nodes, and each node is connected through edges to its k nearest neighbors with weights that reflect inter-node similarities based on geometric information [112, 22]. A fast 3D point cloud denoising using Bipartite graph approximation is proposed in [21]. Here, the point cloud is decomposed into two set through bipartite graph approximation [139] of a k-NN graph. Then, surface normals of one set were defined via geometric coordinates of the neighboring points. Finally, the denoised point cloud is obtained by employing a convex optimization problem, i.e., Graph Total variation (GTV). However, the graph Laplacian is constructed from the noisy point clouds and cannot exhibit the exact manifold.

Motivated from the low-dimensional manifold model (LDMM) in [141], they took advantage of the self-similarity of the surface patches. The extension of [87] from images to point clouds is non-trivial. A precise coordinate function is required for the correct computation of the manifold dimension. They approximate the manifold dimension in the continuous domain by a graph Laplacian regularizer (GLR).

An iteratively based regularization technique (IBR) is employed to enforce smoothness on the geometry-only graph signal [112]. They explain how the position can be represented as a graph signal that can be filtered using the convex optimization method. Inspired by the traditional wavelet-based signal denoising techniques, a framework is proposed in [17] for manifold denoising based on Spectral Graph Wavelet transform (MSGW). This algorithm exploits the geometry coordinate information for the construction of a graph.

The robust denoising of piece-wise smooth manifolds (RPSM) is performed using a local tangent space-based graph [16]. First, a graph is constructed that encodes the geometric structure locally via tangent space affinity graph on Tensor voting [83]. Later, the manifold denoising is performed by solving a diffusion process on the graph through convex optimization, particularly via Tikhonov regularization. The disadvantage of these approaches is that holes are typically formed in the denoised output (see Fig. 1.2) as the correct position of a point is estimated based on the *noisy* geometry. This can lead to errors in estimating the local surface as the manifold's information is based only on geometry.

Learning based techniques

In the last few years, learning-based techniques [102, 25, 105, 39], particularly the ones based on deep learning, have gained a lot of attention. Extension of traditional Convolutional Neural Network (CNN) to 3D point cloud is not simple and straightforward due to unorganized points in 3D space. Indeed, in point cloud segmentation and shape classification, several schemes have recently been explicitly proposed to deal with point cloud data. PointNet [100] is among the most suitable works in this field, characterized by a simple design. This algorithm has achieved promising outcomes.

Several contributions exploit the same scheme presented in [100] to address other point cloud processes. One of the most work is PCPNet [37], where the local shape features are computed, such as normals of the points. In the current literature, a few designs are proposed to extend the work of PointNet to point cloud denoising. PointCleanNet [101] employs a similar method to PointNet to approximate the points' correct vectors in the noisy point cloud. In [25], the authors apply a neural network equivalent to the PointNet scheme to determine a reference plane for each noisy point. Then the denoised point cloud is obtained by projecting the noisy point onto the respective reference plane.

In PointProNet [105], point cloud denoising is performed by utilizing the similar architecture of PointNet for the approximation of local directions of the surface. However, the local frame approximation precision delimits the accuracy of this technique. The drawback of all these methods similar to PointNet is that they consider the individual points and use a global symmetric aggregate function and ignore the neighborhood's local structure. To address this problem, PointCleanNet takes the input local patches in place of a whole point cloud. Still, this technique is limited as the network cannot learn hierarchical features, such as conventional CNNs. These deep denoising techniques are supervised since they need pair of noisy and reference ground-truth point clouds. In practice, the noisy point clouds are generated by adding noise to the reference point clouds. To address this problem, an unsupervised approach is presented in [39], which works under the hypothesis that a noisy pixel is random distributed around a clean pixel value that assists in appropriate learning on this distribution to converge finally to an actual value. However, this supposition is invalid for the unstructured points and also cannot preserve the sharp features due to the lack of feature information during the training step.

Recently, Graph-convolutional networks (GCNs) have shown better performance for the classification and segmentation tasks. Particularly, Dynamic Graph CNN (DGCNN) [131] presents the concept of the dynamic graph update in the hidden layers of a GCN. However, the denoising task is distinct from the segmentation and classification problem presented in [131], which depends on global features rather

than localized representations. Primarily, numerous factors such as spatial transformer block make DGCNN inconsistent for denoising the point cloud since its aim is canonical global representation. However, the denoising task is mainly concerned with the local representation of the neighborhood of point that also linearly increases the computational complexity with the size of point clouds. Furthermore, the graph convolutional operator behaves uncertain in the presence of a substantial amount of noise, as it employs the max operator in an aggregate way. A graph-convolutional-based method that removes the outliers and denoised the point cloud in a single model is proposed in [98], where the graph-convolutional layer in their architecture learns the feature, imitating the standard CNN behavior.

Other techniques

There are various other techniques for point cloud denoising. A set of potential functions is developed by [119] to group together the oriented particles (3D point cloud) into a surface-like structure. Each point has its local coordinate framework, defining both the local tangent plane and the normal of the corresponding surface. This technique helps in the construction of the surface by moving the points in their proximity.

Based on Voxel Grid (VG), a filtering algorithm is presented in [41], where a 3D voxel grid is constructed for the point cloud. Then, a point is selected in each voxel to estimate all the points that lie in the corresponding voxel. The centroid or center of the voxel is usually used as the estimation. The VG-based algorithm typically results in geometric information. The point cloud denoising method generates a quadtree (a data structure used to organize a point cloud). Such representation is to improve the neighborhood searching, which is crucial for efficient smoothing procedures. Later, the projection or neighborhood-based techniques) are employed for the point cloud denoising.

A similarity-based filtering technique for a point cloud denoising is presented in [19]. The input point cloud is decomposed into a smooth piece produced using the high-frequency term and mean curvature. Then, a non-local approach based on L_2-distance similarity is used for filtering the high-frequency components. This technique relied on the density of the point cloud and cannot discriminate structure and texture noise.

There are some hybrid schemes, typically two or more filtering methods, to treat raw point clouds. An iterative procedure is developed in [75] to process the point cloud. First, a WLOP operator is applied for efficient noise filtering. Then, the outliers are detected and eliminated from the input raw point cloud using the mean-shift operator. Nevertheless, their algorithm faces difficulty in recovering sharp features and is computationally expensive.

A density-based approach proposed in [138] for point cloud denoising. They

first applied the particle-swam optimization method for the approximation of kernel density estimation (KDE). Further, the point cloud is filtered using a mean-shift clustering scheme utilizing KDE. Finally, the remaining noise was reduced by employing the bilateral filter. This algorithm shows robustness to the highly noisy point cloud.

Chapter 3

Statistical Analysis

This chapter presents the statistical analysis to study the correlation between geometry and color in a point cloud and use that information for the crucial step of graph construction in our proposed point cloud denoising algorithms. The idea of using joint geometry/color graph construction is based on the notion that on a smooth surface, the color is typically smooth, and the correct location of a point also depends on the color attribute and not only the geometry of the neighboring points, and the accurate color also depends on the geometry of the neighbors. For the statistical analysis, a multivariate platform is employed to examine every variable's relationship to the rest of the attributes of a point cloud. We then summarize the linear relationship's strength between each pair of a response variable through correlation tables. To identify the dependencies among the variables, scatterplot matrices were computed. We employed other procedures to examine multiple variables, such as pairwise correlations and covariance matrices.

3.1 Pearson product-moment correlation

Correlation is an assessment of a monotonic relationship between the two variables. A monotonic association between two variables is either (a) directly proportional: the value of one variable increases, so does the other; or (b) indirectly proportional: the value of one variable increases with the decrease of the second variable. In correlated data, the higher value of a variable tends to be lower (negatively correlated) or higher (positively correlated) values of the second variable, and vice versa. A linear relationship between the two variables is a specific case of a monotonic relationship.

A Pearson product-moment correlation (PPMC) attempts to form a line that fits best for two responsive variables' data. The Pearson correlation coefficient, ρ, shows how the data points are far away from the drawn line, i.e., to what extent the data points fit the model or line of best fit). The PPMC coefficient of two

n-dimensional vectors v_1 and v_2 is denoted by $\rho(v_1, v_2)$. To compute the ρ, first, the standard deviation of each random variable is computed as:

$$\varsigma_{v_1} = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(v_{1i} - \overline{v_1})^2}.$$

$$\varsigma_{v_2} = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(v_{2i} - \overline{v_2})^2}.$$

where, $\overline{v_1} = \frac{1}{n}\sum_{i=1}^{n} v_{1i}$ and $\overline{v_2} = \frac{1}{n}\sum_{i=1}^{n} v_{2i}$ represents the arithmetic means of the corresponding variable v_1 and v_2, respectively. The PPMC is then measured as:

$$\rho(v_1, v_2) = \frac{1}{N}\sum_{i=1}^{n}\left(\frac{v_{1i} - \overline{v_1}}{\varsigma_{v_1}}\right)\left(\frac{v_{2i} - \overline{v_2}}{\varsigma_{v_2}}\right). \tag{3.1}$$

If there is an exact linear relationship between two variables, the correlation is 1 or -1 depending on whether the variables are positively or negatively related. If there is no linear relationship, the correlation tends towards zero.

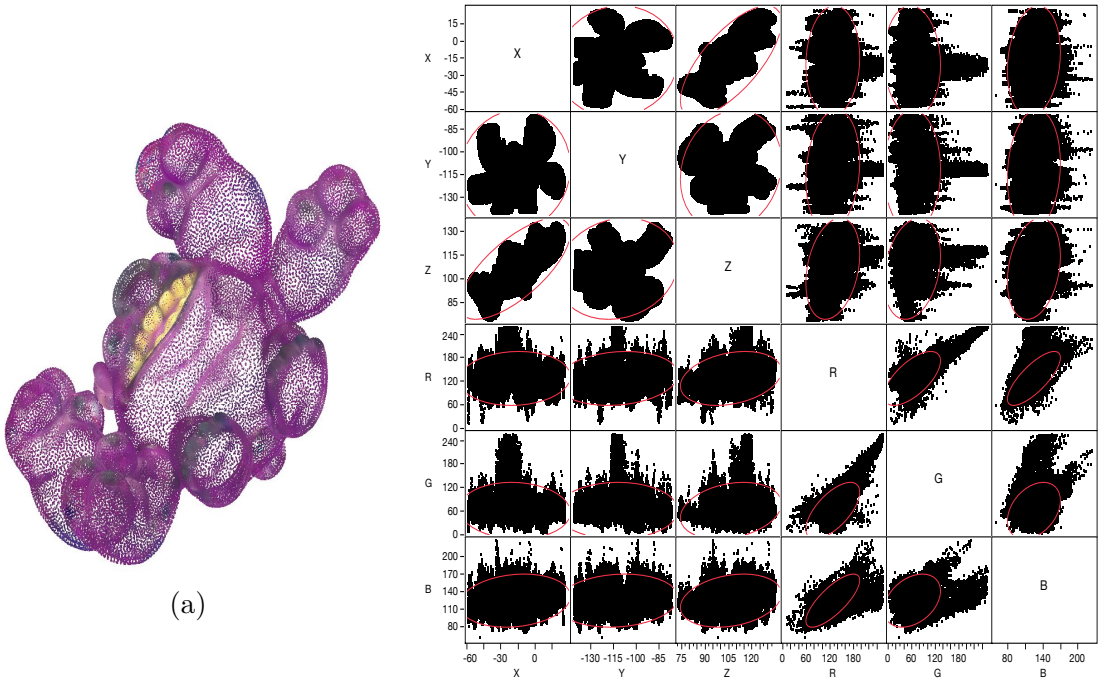
Scatterplots

The first necessary step in all the data analysis procedures is to build a data plot in a very illustrative possible way. A scatterplot is a two-dimensional representation of n pairs of measurements (v_1, v_2) made on the respective random variables v_1 and v_2. The very first bi-variate scatterplot is presented in [65] showing the correlation given by [30]. These plots conveys information about the relationship between v_1 and v_2 [12], such plots are considered as useful tools in the exploratory analysis.

Along with the correlation coefficient, scatterplots are very informative [132]. To understand each geometry's relationship to the individual color component of a point cloud, scatterplots are combined in multiple plots to realize higher-level structure data with more than two variables. Hence, a scatterplot matrix gives the best summary of the data set since they provide a graphical summary of non-linearity, linearity, and separated points [12, 88].

3.2 Correlation between geometry and color

A point cloud is represented as $\mathcal{P} = \{p_1, p_2, p_3, ..., p_N\}$ with $p_i \in \mathbb{R}^6$ containing 3D geometry and RGB color information for point p_i. The six-dimensional feature of each point is $p_i = [X_i, C_i]$, where $X_i = [x_i\ y_i\ z_i] \in \mathbb{R}^3$ is the geometric coordinate vector and $C_i = [R_i\ G_i\ B_i] \in \mathbb{R}^3$ are the color attributes. We compute the linear correlation coefficient ρ between each independent variable of X_i and C_i.



(a) (b)

Figure 3.1: (a) 4arms_monster ground-truth model; (b) Scatterplot Matrix of the correlation result of the geometry and color of (a) shown in Table 3.1.

Table 3.1: Correlation ρ of two n-dimensional variables of geometry and color of *4arms_monster* ground-truth model.

	x	y	z	R	G	B
x	1.00	0.07	0.71	0.07	0.06	0.16
y	0.07	1.00	0.14	0.15	0.01	0.12
z	0.71	0.14	1.00	0.28	0.25	0.25
R	0.07	0.15	0.28	1.00	0.65	0.77
G	0.06	0.01	0.25	0.65	1.00	0.36
B	0.16	0.12	0.25	0.77	0.36	1.00

In Figure 3.1, 95% bivariate normal density ellipse is shown in each scatterplot box. Assuming that each pair of variables has a bivariate normal distribution, this

ellipse encloses approximately 95% of the points. The narrowness of the ellipse reflects the degree of correlation of the variables.

The results in Table 3.1 and Figure 3.1 show the relationship of individual geometry coordinate to itself (diagonal values of ρ in Table 3.1 and to other variables. They show that there exists some correlation between each geometry coordinate and every individual color attribute of a point.

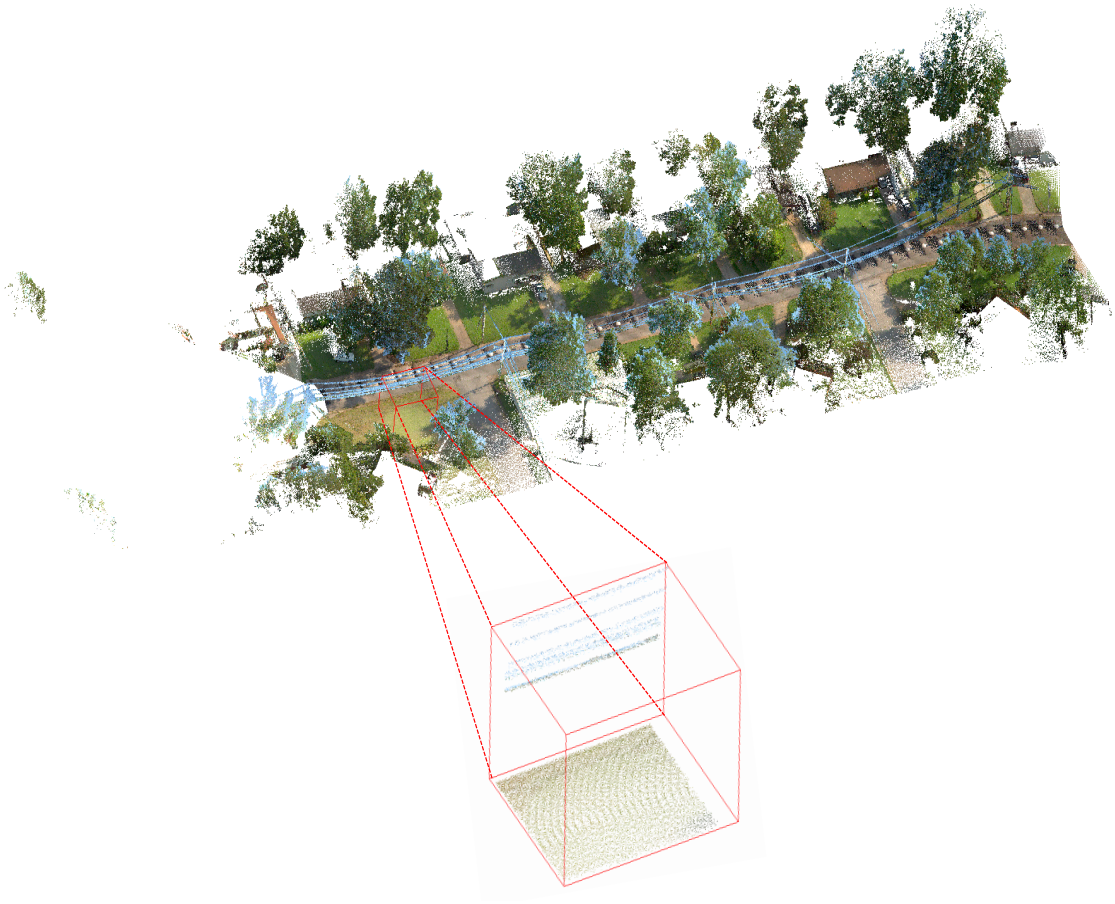


Figure 3.2: MastinLake9_001_colorized0: LiDAR model.

We further investigate the correlation between geometry and color on a point cloud acquired from the heterogeneous system. A LiDAR image *MastinLake9_001_ colorized0 model* has been taken from the sampled dataset provided by LiDARUSA [70]. The LiDAR model Figure 3.2 contains over 10 million points; we then take a small patch and compute the linear correlations between the color and geometry attributes. The correlation results in Table 3.2 and the scatterplot box shown in Figure 3.3. The density ellipses indicate that both the geometry and color components are highly correlated.

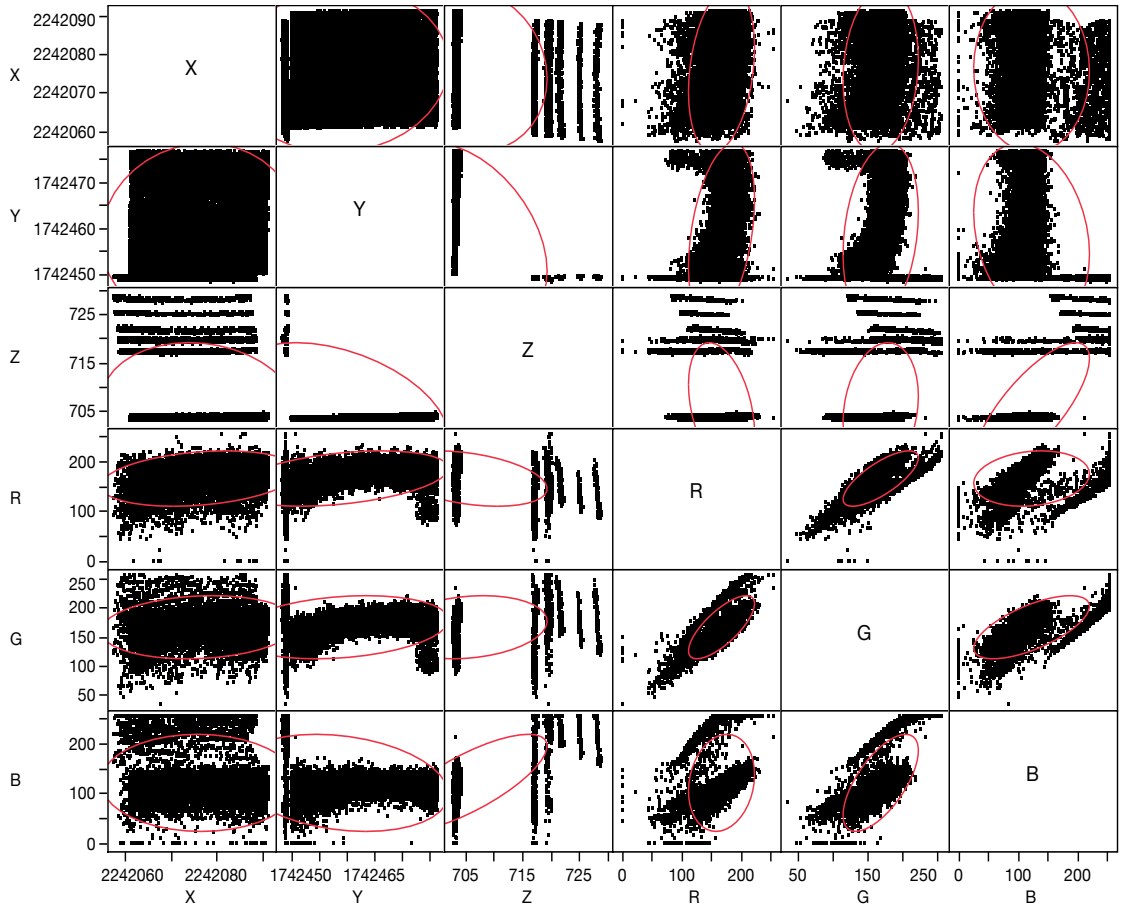


Figure 3.3: Scattorplotbox of MastinLake9_001_colorized0 model.

Table 3.2: Correlation ρ of two n-dimensional variables of geometry and color of *MastinLake9_001_colorized0* model.

	x	y	z	R	G	B
x	1.00	0.06	-0.10	0.27	0.13	-0.02
y	0.06	1.00	-0.42	0.37	0.23	-0.23
z	-0.10	-0.42	1.00	-0.37	0.19	0.76
R	0.27	0.37	-0.37	1.00	0.75	0.17
G	0.13	0.23	0.19	0.75	1.00	0.67
B	-0.02	-0.23	0.76	0.17	0.67	1.00

3.3 Covariance

In order to quantify the relationship between the geometry and color more precisely and gain some intuition, it is helpful to compute the covariance between the individual color component for a different region within a point cloud.

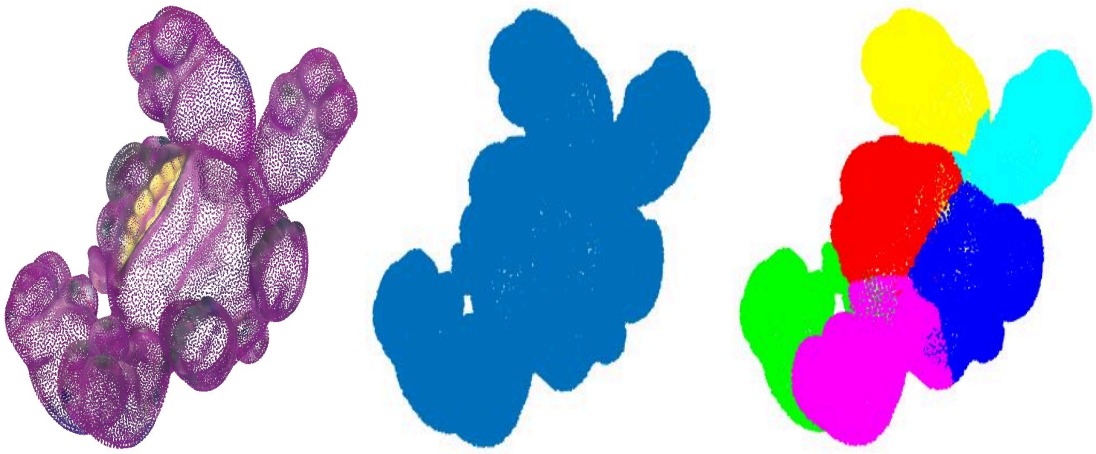
The covariance is a measurement of the correlation of variation in two random variables v_1 and v_2, with a joint normal distribution. The covariance can be computed as:

$$cov(v_1, v_2) = \frac{1}{n-1} \sum (v_{1i} - \bar{v_1})(v_{2i} - \bar{v_2}).$$

Where, $\bar{v_1}$ and $\bar{v_2}$ are the mean values of respective variables. If there is a linear relationship between two variables, then the covariance can be positive or negative subject to the variables' association has a positive or negative slope. Covariance is zero, if v_1 and v_2 are independent, i.e, not correlated.

The covariance is computed initially for a full point cloud (singleton cluster) to understand how much the geometry and color attributes are associated. Then, the k-mean clustering is performed based on geometry information for an optimized choice of k using the algorithm described in [124]. The pair-wise covariance between colors in each cluster is computed and compared with the singleton cluster's covariance. Figure 3.4-a and Figure 3.4-b represent a singleton cluster and k clusters of 4arms_monstre model, respectively.

The intuition is that within a geometry-based cluster, the covariance among the color components should be less than the covariance of the singleton cluster, which means that the color is less varying within close proximity and is highly correlated. Table. 3.4-a reports the pair-wise covariance of color for a singleton cluster. The covariance among the individual color components in each cluster is shown in Tables. 3.4(b-g). It can be seen that the colors have less covariance for all the clusters with respect to the covariance in a singleton cluster except the results shown in Table. 3.4-f, where the covariance is higher, that may be due to the dynamic range of colors inside the cluster.



(a) (b) (c)

Figure 3.4: (a) 4arms_monstre model: (b) Singleton cluster and (c) k-mean clustering of (a) with $k = 6$.

Table 3.3: Comparison of covariance between pair-wise color components of Arco_valentino model: for (a) Singleton cluster (b) highly correlated cluster and (c) less correlated cluster.

	R	G	B
R	2672.51	2373.41	2103.21
G	2373.41	2187.40	1984.80
B	2103.21	1984.80	1857.42

(a)

	R	G	B
R	597.82	561.12	545.37
G	561.12	563.91	563.41
B	545.37	563.41	586.12

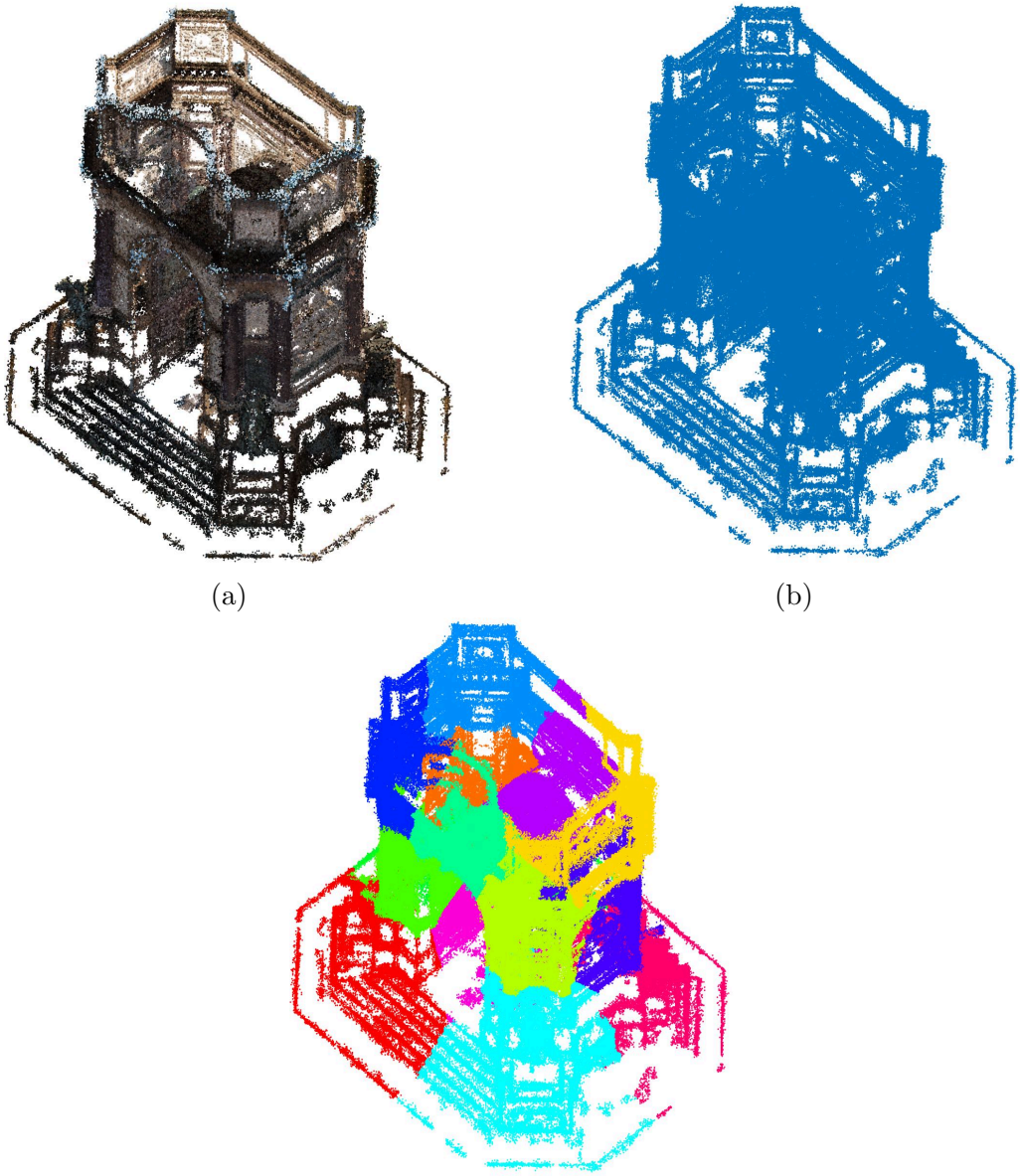
(b)

	R	G	B
R	4358.04	4006.77	3600.47
G	4006.77	3803.53	3499.75
B	3600.47	3499.75	3325.05

(c)

The same analysis has been performed to verify and understand the relationship

between geometry and color in real-world point clouds. Figure 3.5-a and Figure 3.5-b are the singleton cluster and k clusters of *Arco_valentino* model, respectively.



(a) (b)

(c)

Figure 3.5: (a) Arco_valentino model: (b)Singleton cluster and (b) $k$mean clustering of (a) for $k = 14$.

It is observed that the color covariance in each cluster is less with respect to the covariance in a singleton cluster in a similar fashion as seen for synthetic point clouds. The results of the real-world point cloud analysis depict that the geometry

and color are closely related to each other in close proximity.

Table 3.4: Comparison of covariance between pair-wise color components of 4arms_monstre model: for (a) Singleton cluster shown in Figure 3.4b and (b-g) individual cluster shown in Figure 3.4c.

	R	G	B
R	787.69	539.95	401.29
G	539.95	865.78	196.08
B	401.29	196.08	345.63

(a)

	R	G	B
R	609.73	355.56	379.11
G	355.56	494.22	288.51
B	379.12	288.51	325.90

(b)

	R	G	B
R	416.74	133.19	256.34
G	133.19	264.56	94.52
B	256.34	94.52	212.72

(c)

	R	G	B
R	585.75	223.74	438.67
G	223.74	311.18	177.29
B	438.67	177.29	362.83

(d)

	R	G	B
R	544.74	198.17	290.70
G	198.17	387.39	162.12
B	290.70	162.12	260.61

(e)

	R	G	B
R	853.32	779.21	596.92
G	779.21	901.56	207.09
B	596.92	207.09	571.66

(f)

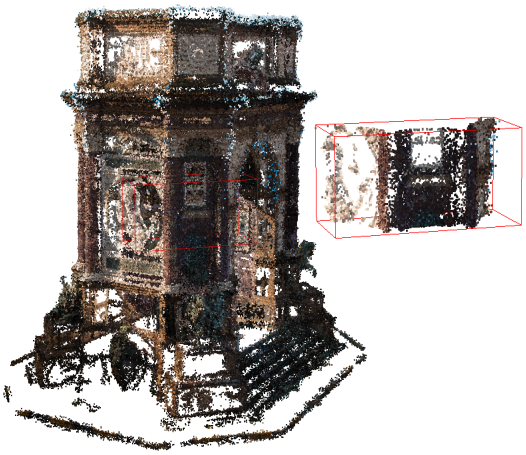
	R	G	B
R	372.99	152.97	233.94
G	152.97	297.62	133.53
B	233.94	133.53	203.93

(g)

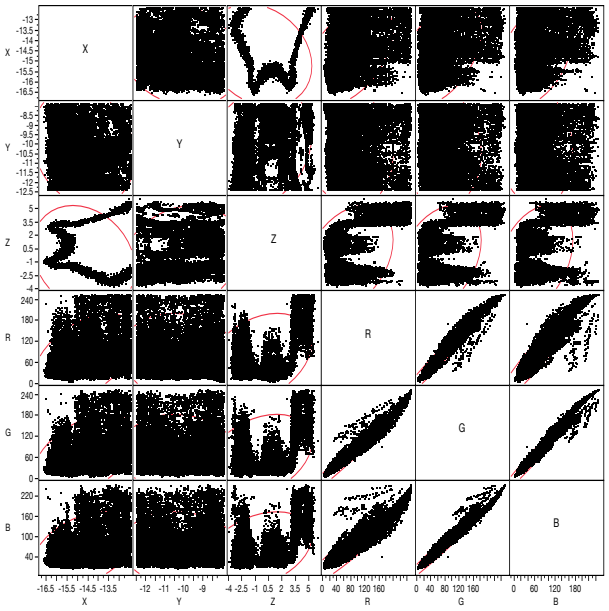
In order to present the covariance result of Arco_valentino to depict the results of all the clusters, the highly correlated cluster and the only inadequate cluster

with less correlation in terms of covariance to the corresponding singleton cluster are shown in Table. 3.3-b and Table. 3.3-c, respectively.

Furthermore, to get more meaningful results, a planar patch is taken into account from *Arco_Valentino* shown in Figure 3.6a to compute the linear correlation between geometry and color. The results in Table 3.5 and the scatterplot shown in Figure 3.6b indicates that the data points for the individual geometry and color component are tightly clustered, which depicts that the geometry and color are highly correlated.



(a)



(b)

Figure 3.6: (a) Arco_valentino model; (b) Scatterplot Matrix of the correlation result of the geometry and color of (a) shown in Table. 3.5.

By performing statistical analysis, a conclusion can be drawn that the geometry and color attribute in a point cloud is highly correlated, varying from point cloud to point cloud. Based on this statistical analysis, we can use both geometry and color attributes for geometry-only and color-only denoising a point cloud.

Table 3.5: Correlation ρ of two n-dimensional variables of geometry and color of a planar patch in *Arco_valentino* model.

	x	y	z	R	G	B
x	1.00	-0.17	-0.21	0.49	0.47	0.41
y	-0.17	1.00	0.20	0.12	0.11	0.09
z	-0.21	0.20	1.00	0.35	0.34	0.34
R	0.49	0.12	0.35	1.00	0.98	0.94
G	0.47	0.11	0.34	0.98	1.00	0.99
B	0.41	0.09	0.34	0.94	0.99	1.00

Chapter 4

Graph Signal Processing

In this chapter, a concise introduction to the Graph theory is provided in order to present the essential concepts and the processes employed in our proposed algorithms.

4.1 Preliminaries of Graph

A generic graph $\mathcal{G}$ is composed of a set of points called *nodes* or *vertices*, and *edges* are the connections between them, usually denoted by $\mathcal{V}$ and $\mathcal{E} \subset \mathrm{V} \times \mathrm{V}, respectively.$

A graph $\mathcal{G}$ can be directed or undirected. In the case of a directed graph, for instance, if vertex A is connected to the vertex B, then B is connected to A; unless the edge's direction is specified by an arrow as shown in Figure 4.1-a and Figure 4.1-b.

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of set of N vertices represented by $\mathcal{V} = \{v_j\}$ for $j \in \{1,2,...,N\}$, and the aggregation of all the edges $\mathcal{E} = \{e_{ij}\}$ for $i, j \in \{1,2,....,N\}$. A matrix $M = N \times N$ defines the *adjacency matrix* with the elements m_{ij} equal to 1 if an edge exists between the node i and the node j, otherwise the value is 0.

The *adjacency matrix* of the undirected graph shown in Figure 4.1-b is shown as an example. It is axiomatic that the adjacency matrix for an undirected graph would be symmetric.

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Weight is a particular quantity associated to each edge of the graph, adding some information to the signal's representation; a graph with this information is called a weighted graph.

Consider an undirected weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ having a finite set of m vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ defined as $(i, j, w_{i,j})$, where $i, j \in \mathcal{V}$ and each edge has a non-negative weight $w_{i,j} \in \mathbb{R}^+$ that reflects the affinity between node i and j. The corresponding *adjacency matrix* $\mathbf{W}(i, j) = w_{i,j}$ is a real symmetric $m \times m$ matrix, that can be defined as:

$$\mathbf{W}_{i,j} = \begin{cases} 0 & \text{if } e_{ij} \notin \mathcal{E} \\ w_{ij} & \text{if } e_{ij} \in \mathcal{E}. \end{cases}$$

Here, w_{ij} denotes the weight associated to the edge e_{ij}. The adjacency matrix of the weighted graph shown in Figure 4.1-c is reported as:

$$\mathbf{W} = \begin{bmatrix} 0 & 0.31 & 0 & 0 & 0.14 & 0.75 \\ 0.31 & 0 & 0.56 & 0 & 0 & 0.48 \\ 0 & 0.56 & 0 & 0.20 & 0 & 0 \\ 0 & 0 & 0.20 & 0 & 0 & 0 \\ 0.67 & 0 & 0 & 0 & 0 & 0 \\ 0.75 & 0.48 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After a brief description of weighted and adjacency matrix, the degree matrix can be defined as the diagonal degree matrix $\mathbf{D}$ with entries $\mathbf{D}_{i,j} = \sum_j w_{i,j}$. The degree matrix of the undirected weighted graph in Figure 4.1-c is:

$$\mathbf{D} = \begin{bmatrix} 1.20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.35 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.76 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.67 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.23 \end{bmatrix}$$

The edge direction is considered in a directed graph; different sign is associated to the edge for different directions.

Given $\mathbf{D}$ and $\mathbf{W}$, the combinatorial graph Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

The graph signal $g(\mathcal{G})$ for a given graph $\mathcal{G}$ is defined on the vertices of a graph, as $g : \mathcal{V} \to \mathbb{R}^D$ for some dimension D.

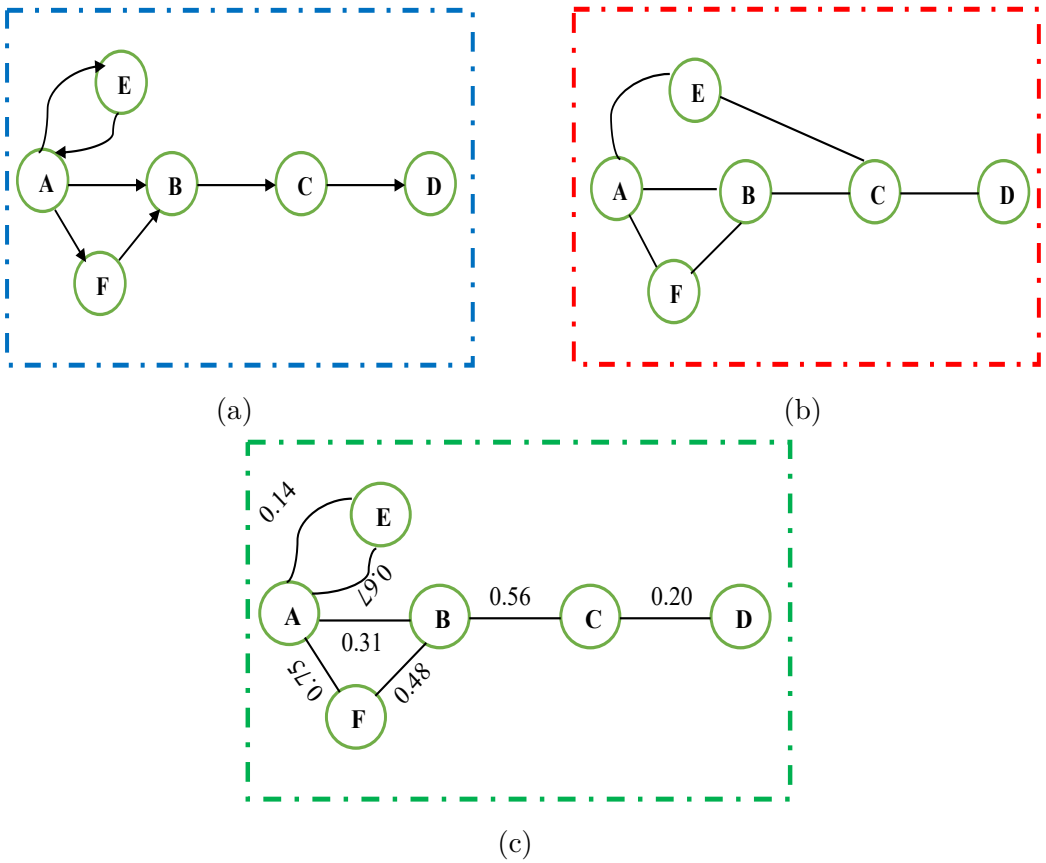


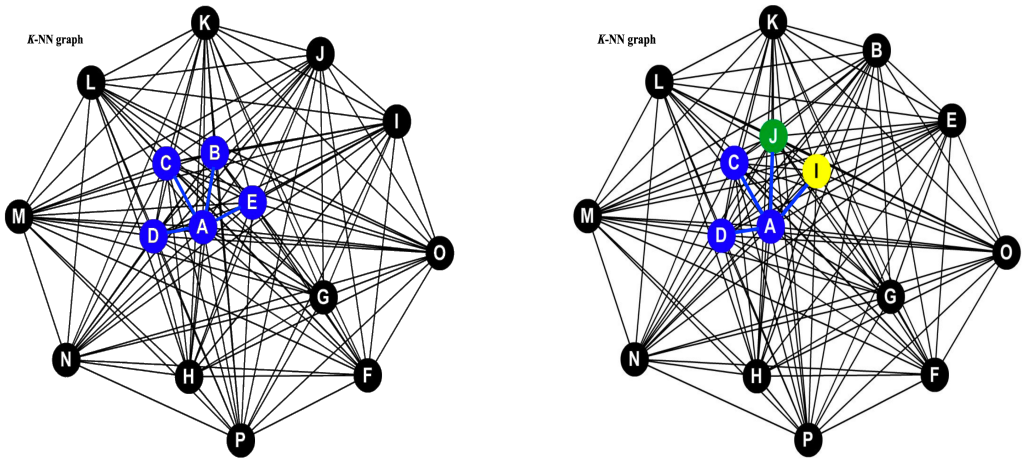
Figure 4.1: Examples of different types of graphs: (a) Directed graph, (b) Undirected graph, and (c) Weighted graph.

4.1.1 Joint geometry/color k-NN graph

A common approach is to construct a k nearest neighbor (k-NN) graph based on Euclidean distance to make geometric structure explicit [128]. We generate the k-NN graph based on both coordinates proximity and similarity in color of points $p_i \in \mathcal{P}$.

Construction of k-NN color-only graph is not a good choice as geometrically distant points may have similar color but different semantic content, which may lead to the construction of a wrong graph. In contrast, geometry-only k-NN graph generates artifacts as anticipated in Sec. 1. Figure 4.2 illustrates how color can positively affect the graph construction by generating a set of neighbors that is more semantically meaningful than a geometry-only graph. The joint geometry and color graph is helpful as it exploits more information about the point cloud. To construct such graph for a given point cloud $\mathcal{P}$, every vertex is connected through an edge to its k nearest neighbors with an associated weight, which is computed using some metric. In this context, we choose to use the Euclidean distance. A common choice

for weighting function is to use threshold Gaussian kernel [114]:



(a) (b)

Figure 4.2: (a) Illustration of a joint geometry and color k-NN graph for $k = 4$; node A is connected to the nodes that are closer to it by geometric and color distance, blue colored nodes are analogous to same color within proximity. (b) Illustration of a geometry only k-NN graph for $k = 4$; node A is connected to nodes that are within its proximity regardless of color of each connected node.

$$w_{i,j} = \begin{cases} \exp\left(-\frac{\|X_i - X_j\|^2}{2\theta_X^2} - \frac{\|C_i - C_j\|^2}{2\theta_C^2}\right) & \text{if } p_j \in \phi_k(i) \\ & \text{or } p_i \in \phi_k(j) \\ 0 \qquad \text{otherwise.} \end{cases} \tag{4.1}$$

Here, θ_X and θ_C determine the relative contribution of geometry and color in the construction of joint geometry and color graph. $\phi_k(i)$ is the set of k nearest neighbors to point p_i, and $\phi_k(j)$ is the set of k nearest neighbors to point p_j. The resulting k-NN graph is denoted as $\mathcal{G}$.

4.2 Datasets

The denoising algorithms have been applied to both the synthetic and real-world point clouds. For the synthetic point clouds, Greyc 3D colored mesh dataset [85] is used, which contains 16 real objects with different colors, textures, and geometric structures. The static real-world point cloud dataset is available in the JPEG PLENO (GTI-UTM) database for evaluating the proposed denoising algorithms [29].

4.3 Evaluation Metrics

The evaluation and comparison of the proposed algorithm and the other competing algorithms are made using appropriate evaluation metrics for both the color and geometry denoising.

4.3.1 Color denoising

The metrics used for the objective evaluation of the proposed color denoising algorithm are mean-squared-error (MSE) and peak signal-to-noise ratio (PSNR):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{C}_i - \widehat{\boldsymbol{C}}_i\|^2. \tag{4.2}$$

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right). \tag{4.3}$$

Where $\boldsymbol{C}_i$ and $\widehat{\boldsymbol{C}}_i$ represent the color attribute of the points in ground-truth and denoised point cloud, respectively, N is the number of points in a point cloud $\mathcal{P}$.

4.3.2 Geometry denoising

For image denoising, quality metrics are based on a one-to-one correspondence between ground-truth and denoised data samples. However, in the case of point clouds, such constraint would be practically too restrictive. The Hausdorff distance overcomes the notion of a vertex to vertex distance [10, 5, 26, 86]. A triangular mesh $\mathcal{M}$ consists of a set $\boldsymbol{p}_i$ of points and a set $\mathcal{T}$ of triangles defining how the vertices from $\boldsymbol{p}_i$ are associated together, denoted by $\mathcal{M} = (\boldsymbol{p}_i, \mathcal{T})$. We consider a mesh corresponding to $\boldsymbol{p}_i$ as $\mathcal{M} = (\boldsymbol{p}_i, \mathcal{T})$ and a denoised point cloud $\widehat{\boldsymbol{p}}_i$. We are interested in measuring the distances between sets of points in the two-point clouds.

Hausdorff distance

The distance $d(\widehat{\boldsymbol{X}}_i, \mathcal{M})$ between a given point $\widehat{\boldsymbol{X}}_i \in \widehat{\boldsymbol{p}}_i$ and any point $\boldsymbol{X}_i \in \mathcal{M}$ is defined as:

$$d(\widehat{\boldsymbol{X}}_i, \mathcal{M}) = \min_{\boldsymbol{X}_i \in \mathcal{M}} \|\boldsymbol{X}_i - \widehat{\boldsymbol{X}}_i\|_2. \tag{4.4}$$

The Hausdorff distance between $\widehat{\boldsymbol{X}}$ and $\mathcal{M}$ is denoted as $d_H(\widehat{\boldsymbol{X}}, \mathcal{M})$ and is given by:

$$d_H(\widehat{\boldsymbol{X}}, \mathcal{M}) = \max_{\widehat{\boldsymbol{X}}_i \in \widehat{\boldsymbol{p}}_i} d(\widehat{\boldsymbol{X}}_i, \mathcal{M}). \tag{4.5}$$

The $d_H(\widehat{\boldsymbol{X}}, \mathcal{M})$ and $d_H(\mathcal{M}, \widehat{\boldsymbol{X}})$ are referred to as forward and backward distance, respectively. These distances are not symmetrical, i.e., $d_H(\widehat{\boldsymbol{X}}, \mathcal{M}) \neq d_H(\mathcal{M}, \widehat{\boldsymbol{X}})$. The symmetrical Hausdorff distance $d_S(\widehat{\boldsymbol{X}}, \mathcal{M})$ can be computed as:

$$d_S(\widehat{\boldsymbol{X}}, \mathcal{M}) = \max\left(d_H(\widehat{\boldsymbol{X}}, \mathcal{M}), d_H(\mathcal{M}, \widehat{\boldsymbol{X}})\right). \tag{4.6}$$

The distance between any point $\boldsymbol{X}_i$ belonging to $\mathcal{M}$ and $\widehat{\boldsymbol{X}}$ can be computed analytically, as it can be reduced to the minimum of the distances between $\boldsymbol{X}_i$ and all the triangles $T \in \mathcal{T}$. If the orthogonal projection $\widehat{\boldsymbol{X}}_i$ of $\boldsymbol{X}_i$ on the plane of T is inside the triangle, the point-to-triangle distance is nothing but a point-to-plane distance. When the projection lies outside T, the point-to-triangle distance is the distance between $\boldsymbol{X}_i$ and the closest point $\widehat{\boldsymbol{X}}_i^{'}$ of T, which lies necessarily on one of the sides of T [10, 5, 26].
The point-to-mesh distance in Eq. 4.4 can also be used to calculate the mean distance d_m between $\widehat{\boldsymbol{X}}$ and $\mathcal{M}$:

$$d_m(\widehat{\boldsymbol{X}}, \mathcal{M}) = \frac{1}{N} \sum_{\widehat{\boldsymbol{X}}_i \in \widehat{\boldsymbol{X}}} d(\widehat{\boldsymbol{X}}_i, \mathcal{M}). \tag{4.7}$$

$$\zeta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - d_m)^2}. \tag{4.8}$$

Where $d_i = \|\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i\|^2$; ζ represents the standard deviation of the distance between the point $\widehat{\boldsymbol{p}}_i$ and the corresponding point $\boldsymbol{p}_i$. We computed the Hausdorff distance using the cloud-to-mesh (C2M) metric in CloudCompare [32]. The ground-truth 3D models act as the reference meshes to their respective denoised point clouds. The outputs of the C2M metric are d_H, d_m.

For further analysis of the proposed algorithm with respect to the other algorithms, we have also computed the Mean-squared-error (MSE) and Mean city-block distance (MCD). Assume $\mathcal{Q}$ and $\mathcal{Q}'$ represent the geometry of the noise-free and denoised point cloud respectively, where $\mathcal{Q} = \{\boldsymbol{q}_i\}_{i=1}^{N_1}$ $\mathcal{Q}'=\{\boldsymbol{q}_i'\}_{i=1}^{N_2}$, such that $\boldsymbol{q}_i$, $\boldsymbol{q}_i'$ $\in \mathbb{R}^3$. We define distance metrics as follows.

MSE: It is computed as an average of the squared Euclidean distance between each point in $\mathcal{Q}$ and its corresponding nearest point in $\mathcal{Q}'$, and also between each point in $\mathcal{Q}'$ and its corresponding nearest point in $\mathcal{Q}$:

$$\text{MSE} = \frac{1}{2N_1}\sum_{\boldsymbol{q}_i \in \mathcal{Q}} \min_{\boldsymbol{q}_i' \in \mathcal{Q}'} \|\boldsymbol{q}_i - \boldsymbol{q}_i'\|_2^2 + \frac{1}{2N_2}\sum_{\boldsymbol{q}_i' \in \mathcal{Q}'} \min_{\boldsymbol{q}_i \in \mathcal{Q}} \|\boldsymbol{q}_i' - \boldsymbol{q}_i\|_2^2. \tag{4.9}$$

MCD: MCD uses l_1 norm instead of l_2 norm.

$$\text{MCD} = \frac{1}{2N_1}\sum_{\boldsymbol{q}_i \in \mathcal{Q}} \min_{\boldsymbol{q}_i' \in \mathcal{Q}'} \|\boldsymbol{q}_i - \boldsymbol{q}_i'\| + \frac{1}{2N_2}\sum_{\boldsymbol{q}_i' \in \mathcal{Q}'} \min_{\boldsymbol{q}_i \in \mathcal{Q}} \|\boldsymbol{q}_i' - \boldsymbol{q}_i\|. \tag{4.10}$$

Chapter 5

Point cloud denoising using convex optimization

In this chapter, we will discuss the denoising of both the geometry and color attribute of a point cloud by employing graph-based optimization technique, taking advantage of the *correlation* between geometry and color described in Sec. 3.2, and using it as a powerful tool for several tasks, i.e., color denoising, geometry denoising, and combined geometry and color denoising. We explain how the geometric coordinates can be represented as a graph signal that can be filtered utilizing convex optimization techniques.

5.1 Proposed method - Graph construction

We have constructed the k- nearest neighbors (k-NN) graph using the Eq. 4.1 in the following three different settings for each task.

5.1.1 Joint geometry/color graph construction from noisy geometry and noise-free color

In the setting considered here, the given point cloud $\mathcal{P}$ contains noisy geometry and noise-free color information for a point $\boldsymbol{p}_i$. The proposed approach described in Eq. 4.1 generates a k-NN graph based on color similarity and geometry proximity in a 3D plane, with a suitable choice of θ_X and θ_C. The resulting k-NN graph is denoted as $\mathcal{G}_2$.

5.1.2 Joint geometry/color graph construction from noise-free geometry and noisy color

For color denoising of a point cloud, a joint k-NN graph constructed from the noise-free geometry and noisy color of the points is used to denoise the color attribute of a point cloud. The resulting k-NN graph is defined as $\mathcal{G}_1$.

5.1.3 A joint geometry/color graph construction from noisy geometry and noisy color

A joint k-NN graph constructed from the noisy geometry and color is used to denoise both the geometry and color of a point cloud. The given noisy point cloud $\mathcal{P}$ consists of noisy geometry and noisy color information for a point $\boldsymbol{p}_i$, with a potentially different choice of θ_X and θ_C. The resulting k-NN graph is denoted as $\mathcal{G}_3$.

Having introduced the mechanism of graph construction for the different denoising scenarios we are interested in, we now present the proposed approach for denoising the color-only, geometry-only, and combined geometry and color noise of a point cloud.

The very first step is to remove the outliers from the entire point cloud. The outliers have distinct characteristics in that each outlier has a sparse neighborhood; therefore, the detection and removal of outliers are density-based [112, 31]. In the following, we follow the ROR approach, in which a sphere with a predefined radius r is formed, having each point $\boldsymbol{p}_i$ as its center. u_i denotes the number of points contained in each sphere. We then calculate $\bar{u} = \frac{\sum_{i=1}^{n} u_i}{N}$, where N is the total number of points. A point $\boldsymbol{p}_j$ is considered as an outlier if $u_j < \bar{u}$.

5.2 Geometry denoising

For geometry denoising, we exploit the graph $\mathcal{G}_2$ constructed from both geometry and color information of the noisy point cloud defined in Sec. 4.1.1. We define a graph signal $g(\mathcal{G}_2)$, where each vertex of $\mathcal{G}_2$ is associated with the geometry information $\boldsymbol{X}_i$ and color information $\boldsymbol{C}_i$ of the point cloud. The point $\boldsymbol{p}_i$ can be expressed as $\boldsymbol{p}_i = [\boldsymbol{X}_i + \boldsymbol{Z}_i, \boldsymbol{C}_i]$, $\boldsymbol{X}_i$ being the unknown true geometry of $\boldsymbol{p}_i$, $\boldsymbol{Z}_i$ the geometry noise, and $\boldsymbol{C}_i$ the color attribute, with $\boldsymbol{X}_i, \boldsymbol{Z}_i, \boldsymbol{C}_i \in \mathbb{R}^3$. The objective is to estimate $\boldsymbol{X}_i$ for each point. This can be done using the following denoising algorithm described in Eq. 5.1, which moves the points closer to their exact location based on a smoothness assumption applied to the joint geometry and color information embedded in a graph $\mathcal{G}_2$. A graph $\mathcal{G}_2$ can be considered

an approximation of a 3D manifold. Therefore, the regularity of the combined geometry and color is associated with the proximity of the points to the manifold.

$$\widehat{\boldsymbol{X}} = \arg\min_{X} \|X - g\|_2^2 + \gamma \|\nabla_{\mathcal{G}_2} X\|_2^2. \tag{5.1}$$

Here, $X = X(\mathcal{G}_2)$ is a graph signal on $\mathcal{G}_2$ containing geometry values on each node; the estimated denoised geometry for the whole point cloud is referred to as $\widehat{\boldsymbol{X}}$. The convex optimization problem in Eq. 5.1 promotes the smoothness of the graph signal X defined on $\mathcal{G}_2$ utilizing the combined geometry and color k-NN graph. The fidelity term in Eq. 5.1 enforces the denoised points to move to their observed position. The smoothness of the graph can be measured by graph gradient $\nabla_{\mathcal{G}_2} X$ of the signal X. This can also be done by Total variation using Eq. 5.2. This yields the denoised point cloud $\widehat{\boldsymbol{p}}_i = [\widehat{\boldsymbol{X}}_i \; \boldsymbol{C}_i]$, where $\widehat{\boldsymbol{X}}_i$ is the denoised geometry for point $\boldsymbol{p}_i$.

$$\widehat{\boldsymbol{X}} = \arg\min_{X} \|X - g\|_2^2 + \gamma \|\nabla_{\mathcal{G}_2} X\|_1. \tag{5.2}$$

The alternative direction method of multipliers (ADMM) [7] can also handle this problem efficiently.

5.3 Color denoising

The algorithm presented in the following has the objective to perform color denoising by exploiting the graph $\mathcal{G}_1$ from geometric and color information of the noisy point cloud. We define a graph signal $g(\mathcal{G}_1)$, where each vertex of $\mathcal{G}_1$ is associated with the geometry $\boldsymbol{X}_i$ and color $\boldsymbol{C}_i$ of the corresponding point $\boldsymbol{p}_i$ of a point cloud $\mathcal{P}$. The input noisy point cloud consists of both geometry and color. Each point can be expressed as $\boldsymbol{p}_i = [\boldsymbol{X}_i, \; \boldsymbol{C}_i + \boldsymbol{W}_i]$, $\boldsymbol{X}_i$ being the geometry, $\boldsymbol{C}_i$ being the unknown true color, and $\boldsymbol{W}_i$ the color noise. The objective is to estimate $\boldsymbol{C}_i$ for each point of the point cloud.

For denoising, we exploit the regularity of the color and its correlation with the proximity of the points. Graph gradient can be used to measure the degree of smoothness of a graph signal [114]. In the following, we propose a convex optimization technique that enforces the regularity of the denoised color attributes on $\mathcal{G}_1$. In particular, the denoising problem can be written as follows:

$$\widehat{\boldsymbol{C}} = \arg\min_{C} \|C - g\|_2^2 + \gamma \|\nabla_{\mathcal{G}_1} C\|_2^2. \tag{5.3}$$

Here, $C = C(\mathcal{G}_1)$ is a graph signal on $\mathcal{G}_1$ containing an RGB color attribute on each node; the estimated denoised color for the whole point cloud is referred to as $\widehat{\boldsymbol{C}}$, the observed noisy signal is represented by the graph signal g defined above, γ is a parameter for regularization and $\nabla_{\mathcal{G}_1} C$ represents the gradient of the graph

signal C on the graph $\mathcal{G}_1$ (see [114]). Eq. 5.3 has two terms; the first is a fidelity term that enforces the denoised point to be not too far from its observed color, while the second term promotes smoothness of the denoised point cloud on $\mathcal{G}_1$ via Tikhonov regularization. The same method can also employ Total Variation (TV) regularization with the constraint that the underlying manifold of a point cloud is piece-wise smooth. This leads to the following convex optimization problem:

$$\widehat{\boldsymbol{C}} = \arg\min_{C} \|C - g\|_2^2 + \gamma \|\nabla_{\mathcal{G}_1} C\|_1. \tag{5.4}$$

The problems in Eq. 5.3 and 5.4 can be solved by alternating direction method of multipliers (ADMM) [7]. This yields the denoised point cloud $\widehat{\boldsymbol{p}}_i = [\boldsymbol{X}_i \ \widehat{\boldsymbol{C}}_i]$, where $\widehat{\boldsymbol{C}}_i$ is the denoised color attribute for point $\boldsymbol{p}_i$.

5.4 Combined geometry and color denoising

Up to this point, we have taken advantage of the correlation between geometry and color to remove one type of noise, either on the geometry or on the color.

In the following, we consider the case of simultaneous denoising of geometry and color noise employing the constructed k-NN graph $\mathcal{G}_3$ described in Sec. 5.1.3. The point $\boldsymbol{p}_i$ can be expressed as $\boldsymbol{p}_i = [\boldsymbol{X}_i + \boldsymbol{Z}_i, \boldsymbol{C}_i + \boldsymbol{W}_i]$, $\boldsymbol{X}_i$ being the unknown true geometry and $\boldsymbol{C}_i$ the noisy RGB color of $\boldsymbol{p}_i$, $\boldsymbol{Z}_i$ the geometry noise, and $\boldsymbol{W}_i$ is the color noise, with $\boldsymbol{X}_i, \boldsymbol{Z}_i, \boldsymbol{C}_i, \boldsymbol{W}_i \in \mathbb{R}^3$. The objective is to estimate $\boldsymbol{X}_i$ and $\boldsymbol{C}_i$ for each point. In the proposed algorithm, we use a weighting procedure for each component in the feature vector representing a point in the cloud to generate a weighted input signal $\tilde{\boldsymbol{X}}_i = [\Omega_1(\boldsymbol{X_i} + \boldsymbol{Z_i}), \Omega_2(\boldsymbol{C_i} + \boldsymbol{W_i})]$ where, Ω_1 and $\Omega_2 \in \mathbb{R}$. $\tilde{\boldsymbol{X}}_i$ is then denoised using Eq. 5.5, which moves the points closer to their actual position and their true color based on the smoothness assumption in graph $\mathcal{G}_3$. The weights Ω_1 and Ω_2 enforce the fidelity term in Eq. 5.5 that the geometry and color of the denoised points do not move too far from their observed position and their actual color.

$$\widehat{\tilde{\boldsymbol{X}}} = \arg\min_{\tilde{X}} \|\tilde{X} - g\|_2^2 + \gamma \|\nabla_{\mathcal{G}_3} \tilde{X}\|_2^2. \tag{5.5}$$

Here, $\tilde{X} = \tilde{X}(\mathcal{G}_3)$ is a graph signal on $\mathcal{G}_3$ containing a geometry value and color value on each node; $\widehat{\tilde{\boldsymbol{X}}}$ represents the estimated geometry and color denoised values. The convex optimization problem in Eq. 5.5 promotes the smoothness of the graph signal $\tilde{X}$ defined on $\mathcal{G}_3$ utilizing the combined geometry and color k-NN graph. The smoothness of the graph can be measured by graph gradient $\nabla_{\mathcal{G}_3} \tilde{X}$ of the signal $\tilde{X}$. This yields the denoised point cloud $\widehat{\boldsymbol{p}}_i = [\widehat{\boldsymbol{X}}_i \ \widehat{\boldsymbol{C}}_i]$, with $\widehat{\boldsymbol{X}}_i$ being obtained from the first three components of $\widehat{\tilde{\boldsymbol{X}}}$ divided by Ω_1, and $\widehat{\boldsymbol{C}}_i$ being obtained from the last three components of $\widehat{\tilde{\boldsymbol{X}}}$ divided by Ω_2.

Table 5.1: Parameter setting of the proposed denoising techniques for both the synthetic and natural point cloud models with different noise levels.

Techniques	Noise level in color distribution	Noise added in color attribute	Noise added in geometry attribute	Noise level in geometry distribution	k (Synthetic point clouds)	γ (Synthetic point clouds)	k (Natural point clouds)	γ (Natural point clouds)	θ_X	θ_C	Ω_1	Ω_2
Color denoising	$\sigma = 10$	100%	0%	—	8	0.05	25	1.5	0.88	3.5	—	—
	$\sigma = 15$				8	0.08						
	$\sigma = 20$				8	0.13						
	$\sigma = 25$				8	0.16						
	$\sigma = 30$				8	0.20						
	$\sigma = 40$				8	0.28						
Geometry denoising	—	0%	50%	$\sigma = 0.3$	15	0.075	10	0.075	1.0	2.4	—	—
				$\sigma = 0.4$	15	0.075						
				$\sigma = 0.5$	15	0.1						
Combined geometry & color denoising	$\sigma = 20$	100%	50%	$\sigma = 0.3$	100	0.025	70	0.075	0.7	14	0.20	0.80
	$\sigma = 30$			$\sigma = 0.4$	100	0.025						
	$\sigma = 40$			$\sigma = 0.5$	100	0.04						

5.5 Experimental results

This section focuses on the analysis of the experimental results, both subjective and objective, on static real-world and synthetic point cloud datasets discussed in Sec. 4.2.

5.5.1 Experimental setup

For our experiments, we assume that the noise follows a uniform and Gaussian distribution for geometry and color of a point cloud, respectively. This is a common assumption, see e.g. [55, 20]. The graph signal processing in our denoising algorithm has been implemented using GSPBOX [95], and for the convex optimization, we have used UNLocBox [96].

For the outlier removal we have set $\epsilon = 0.01$, $k = 5$ and $\tau = 1$ as in [112]. The parameters setting that provides the best results for the proposed algorithms for both the synthetic and real-world point clouds is shown in Tab. 5.1.

5.5.2 Visual analysis of geometry denoising algorithm

The visual results of the geometry denoising algorithm applied to *Greyc* datasets and natural point clouds are discussed here.

Geometry denoising of natural point clouds

We show a visual comparison between the point cloud denoised by the proposed algorithm and denoised using a graph constructed from only geometry as in [112].

The experiment is performed on real-world natural point clouds, for which we do not have a noiseless reference; hence, the results are only qualitative. Fig. 5.1a and Fig. 5.2a show the point clouds with real noise; it can be seen that the points with the same color are typically in a small neighborhood. Fig. 5.1b and Fig. 5.2b are the resulting output after outlier removal. Fig. 5.1c and Fig. 5.2c depict the denoised point cloud using the proposed algorithm. Here the noisy points are moved close to their original position by exploiting the correlation between the geometric coordinates and the color attribute. Fig. 5.1d and Fig. 5.2d show the denoised point cloud using the geometry-only graph approach in [112]; it can be seen in the same region that, using no color information, the noisy points are not moved to their correct location, leaving gaps as anticipated in Fig. 1.2, and generally providing a noisier result near object boundaries. As can be seen, the proposed method does not generally create gaps and provides a visually more natural result.

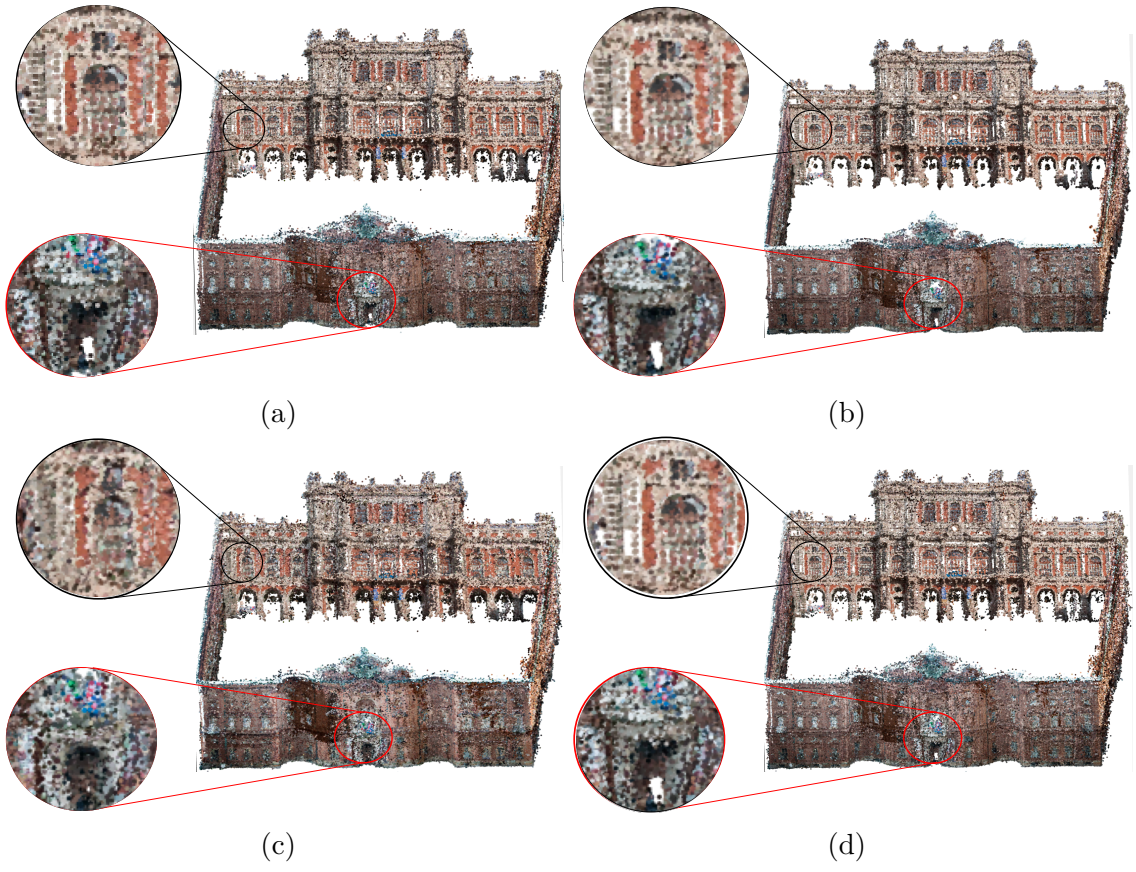
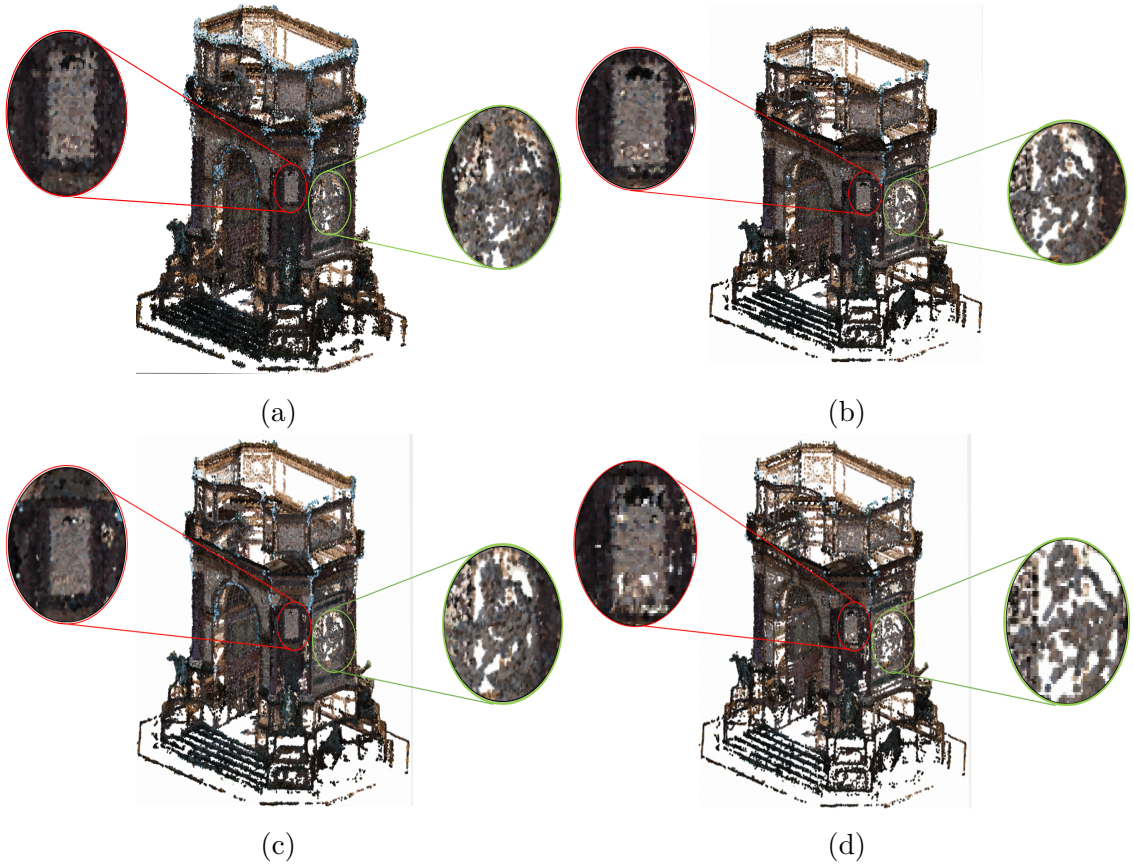


Figure 5.1: Palazzo_Carignano_Dense model illustration. (a) noisy input, (b) outlier-free input, geometry denoised results by (c) proposed algorithm, and (d) geometry-only graph [112].



(a) (b) (c) (d)

Figure 5.2: Arco_Valentino model illustration. (a) noisy input, (b) outlier-free input, geometry denoised results by (c) proposed algorithm, and (d) geometry-only graph [112].

Geometry denoising of LiDAR image

The proposed denoising algorithm has also been applied to the point cloud acquired by the LiDAR sensor from a dataset provided in [70]. The input point cloud model is shown in Fig. 5.3, where we can see noise in the acquired LiDAR. The noise can be seen as the dispersal of points on the trees and power lines with different colors, which may be due to the adverse effects of atmospheric particles or weather conditions such as snow or rain. The denoising result obtained from the proposed algorithm is shown in Fig. 5.4, where we can see that the points are moved to their actual position, and minimize the noise. Furthermore, in the highlighted region, we can see a less adverse effect of the creation of holes in the denoised output. However, using the proposed algorithm for the denoising of point cloud acquired from LiDAR is computationally expensive especially for a very densely populated point cloud. The computational cost of the proposed algorithm and the algorithm described in [112] for point clouds from various acquisition system are

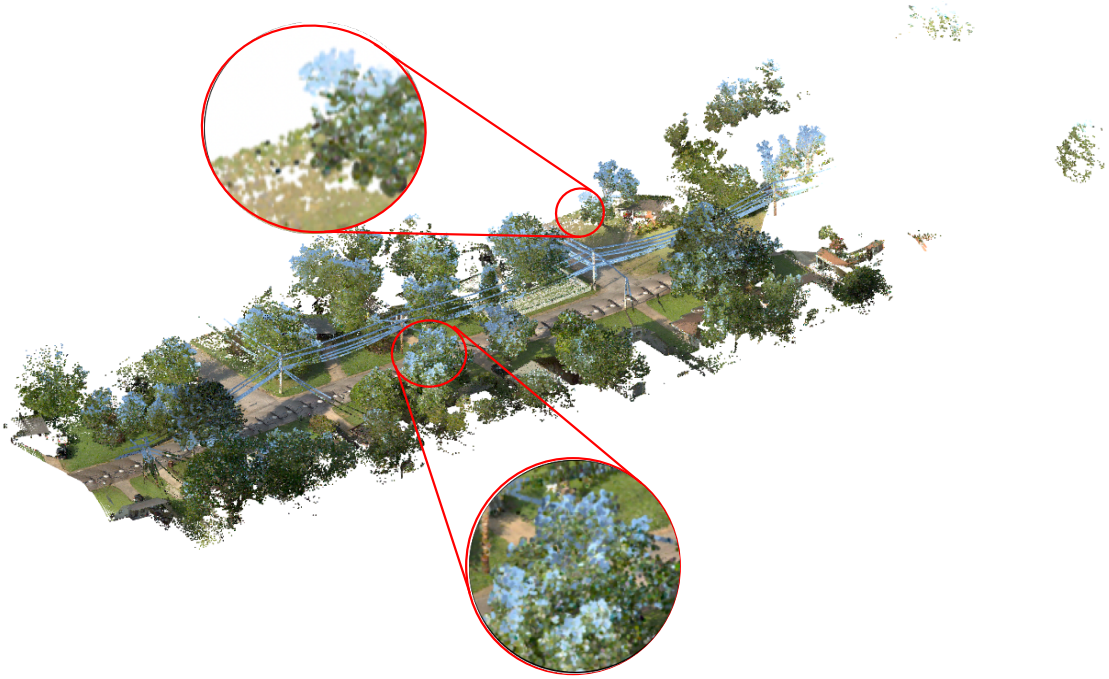


Figure 5.3: Input LiDAR model: MastinLake9_001_colorized0.

shown in Table. 5.12. Fig. 5.5 depicts the denoising result using the algorithm, where the geometry is considered for the construction of k-NN graph. We can see that the input point cloud is over-regularized and hence causes the adverse effects of leaving gaps.

Geometry denoising of ToF point clouds

To check the robustness of our proposed algorithm, we applied our technique on point cloud generated using Helio2 ToF (Time-of-Flight) 3D camera. The sample point cloud is available on [1]. The input model *3 Plastic Laundry Detergent on Carpet* (color overlay using Triton 3.2MP camera) is shown in Fig. 5.6a. The synthetic geometry noise is then added with $\sigma = 0.3$. Fig. 5.6b represents the noisy ToF model. We can see that the model is having inconsistent and lacks smoothness. The denoised point clouds obtained using the proposed is shown in Fig. 5.6c. We can see that the points are relocated to their actual position with fewer adverse artifacts. The denoised point cloud using geometry-only algorithm [112] is depicted by Fig. 5.6d. It can be seen that the algorithm tried to stretch the points too much around their neighbors, which caused the creation of large holes in the output. It can also be seen that the resulting point cloud from the proposed algorithm is quite smoother than the noisy point cloud and the output point cloud of the technique considering only geometry for the graph construction.

Figure 5.4: Denoised LiDAR model with proposed algorithm.

Geometry denoising of synthetic point clouds

The proposed denoising approach has also been applied to noise-free point clouds from the *Greyc* dataset [85], corrupted with uniform zero-mean synthetic geometry noise applied to 50% of the points with $\sigma = 0.3, 0.4$, and 0.5. Fig. 5.7a and Fig. 5.8a show the noise-free point clouds. Fig. 5.7b and Fig. 5.8b show the noisy point clouds with $\sigma = 0.4$. The denoised point clouds obtained by our proposed algorithm are shown in Fig. 5.7c and 5.8c; it can be seen that the geometry noise has been regularized, and the noisy points are moved close to their original positions. The resulting denoised point clouds using the geometry-only algorithm [112] are shown in Fig. 5.7d and 5.8d. The output point clouds of MSGW [17] are shown in Fig. 5.7e and Fig. 5.8e. It can be seen that the geometry is not quite as much regularized; moreover, as anticipated in Sec. 1, these approaches have an adverse effect on overall geometry, as they tend to open holes in the denoised point clouds. Overall, it can be seen from the qualitative results of both the real-world and synthetic point clouds that the point clouds denoised by the proposed algorithm have better quality and fewer artifacts.

We have also compared the proposed algorithm with RPSM [16]. Due to the limitations of RPSM as anticipated in Sec. 4.3, we performed experiments on sub-sampled point clouds. The sub-sampling performed here is on a spatial basis, setting a minimum distance between the two points equal to 0.80. The larger the

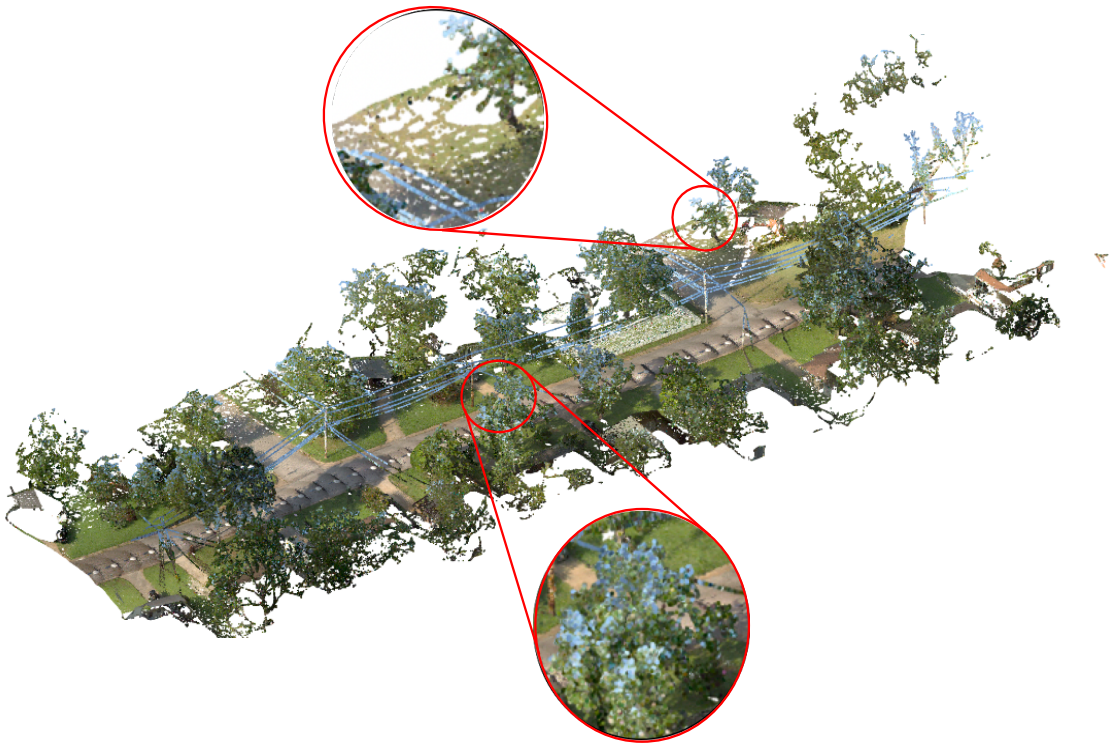


Figure 5.5: Denoised LiDAR model with geometry-only algorithm [112].

distance, the fewer points will be retained in the sub-sampled output. The number of points in the sub-sampled clouds is around 20000 on average. The subjective results are shown in Fig. 5.9 and Fig. 5.10. The proposed algorithm and RPSM [16] are applied to the noisy inputs shown in Fig. 5.7b and Fig. 5.8b; the reference noise-free point clouds are shown in Fig. 5.7a and Fig. 5.8a. It can be seen that, while RPSM yields rather natural denoised color, it also tends to over-regularize the point cloud and thereby opens large holes.

5.5.3 Visual analysis of color denoising algorithm

The subjective analysis of color denoising algorithms has been performed for both synthetic and natural point clouds.

Color denoising of natural point clouds

The visual results of the proposed color denoising algorithm on natural point clouds are presented here. There is not much literature available for color denoising of the point cloud to the best of our knowledge, as anticipated in Sec. 1. Here, the qualitative results show the comparison between the proposed algorithm using Tikhonov and TV regularization. Fig. 5.11a and Fig. 5.12a show the real noisy

Figure 5.6: 3 Plastic Laundry Detergent on Carpet model: (a) ground-truth (b) noisy input, geometry denoised results by (c) proposed algorithm using Tikhonov regularization (d) geometry-only graph [112].

point clouds; it can be seen that the details are not very clear due to a large amount of color noise. In some areas, two distinct regions are overlapped with each other due to the color noise.

Fig. 5.11b and Fig. 5.12b show the resulting point clouds after outlier removal. For the color denoising, the outliers must be removed before applying the proposed algorithm as they may lead to construct a wrong graph, which in turn affects the

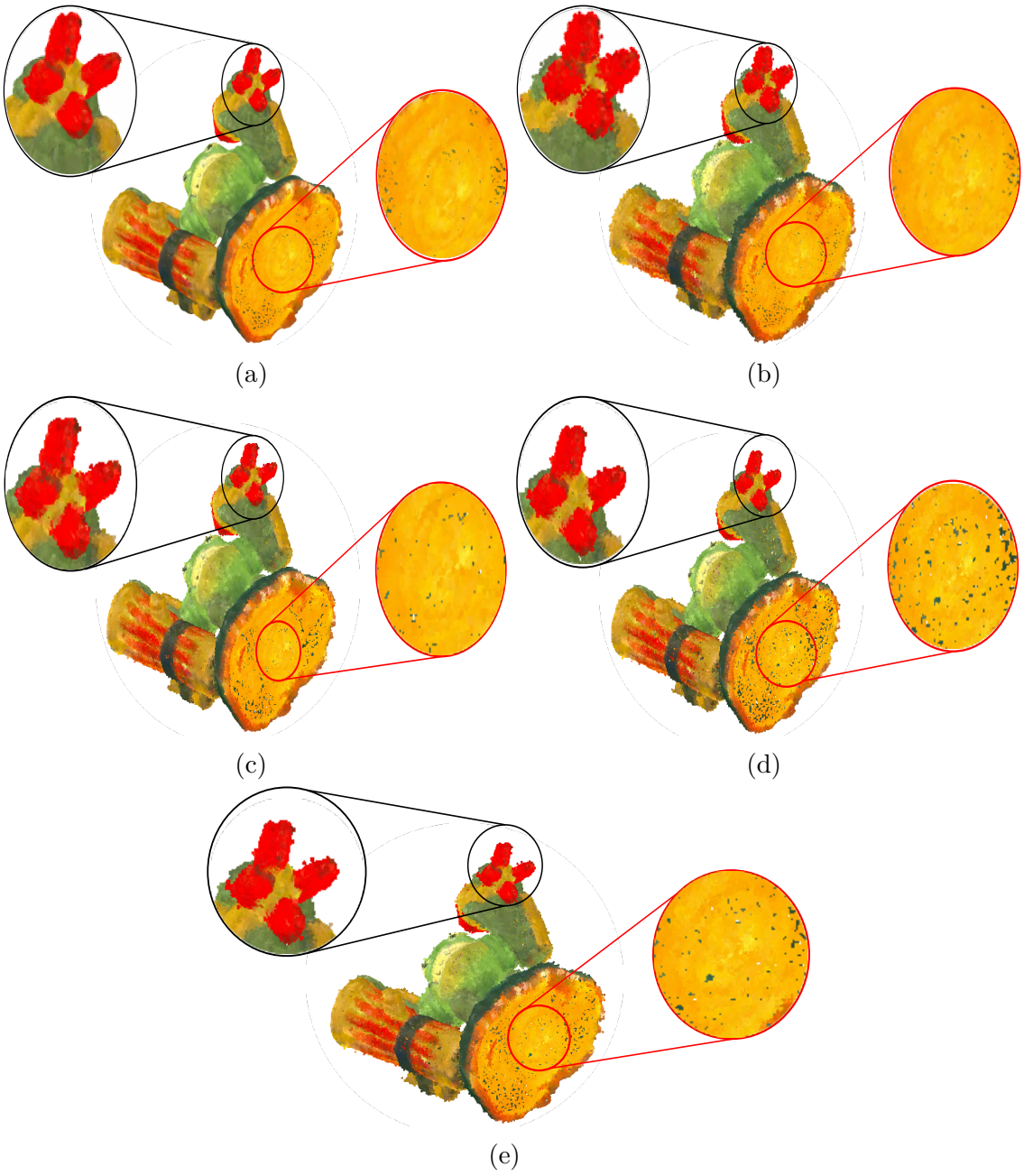


Figure 5.7: Green_monster model: (a) ground-truth (b) noisy input, geometry denoised results by (c) proposed algorithm using Tikhonov regularization (d) geometry-only graph [112], and (e) MSGW [17].

color denoising process. Fig. 5.11c and Fig. 5.12c depict the point cloud denoised using the proposed algorithm. Here, the colors are much smoother and natural by exploiting the relation of the color of the points within proximity. Due to the

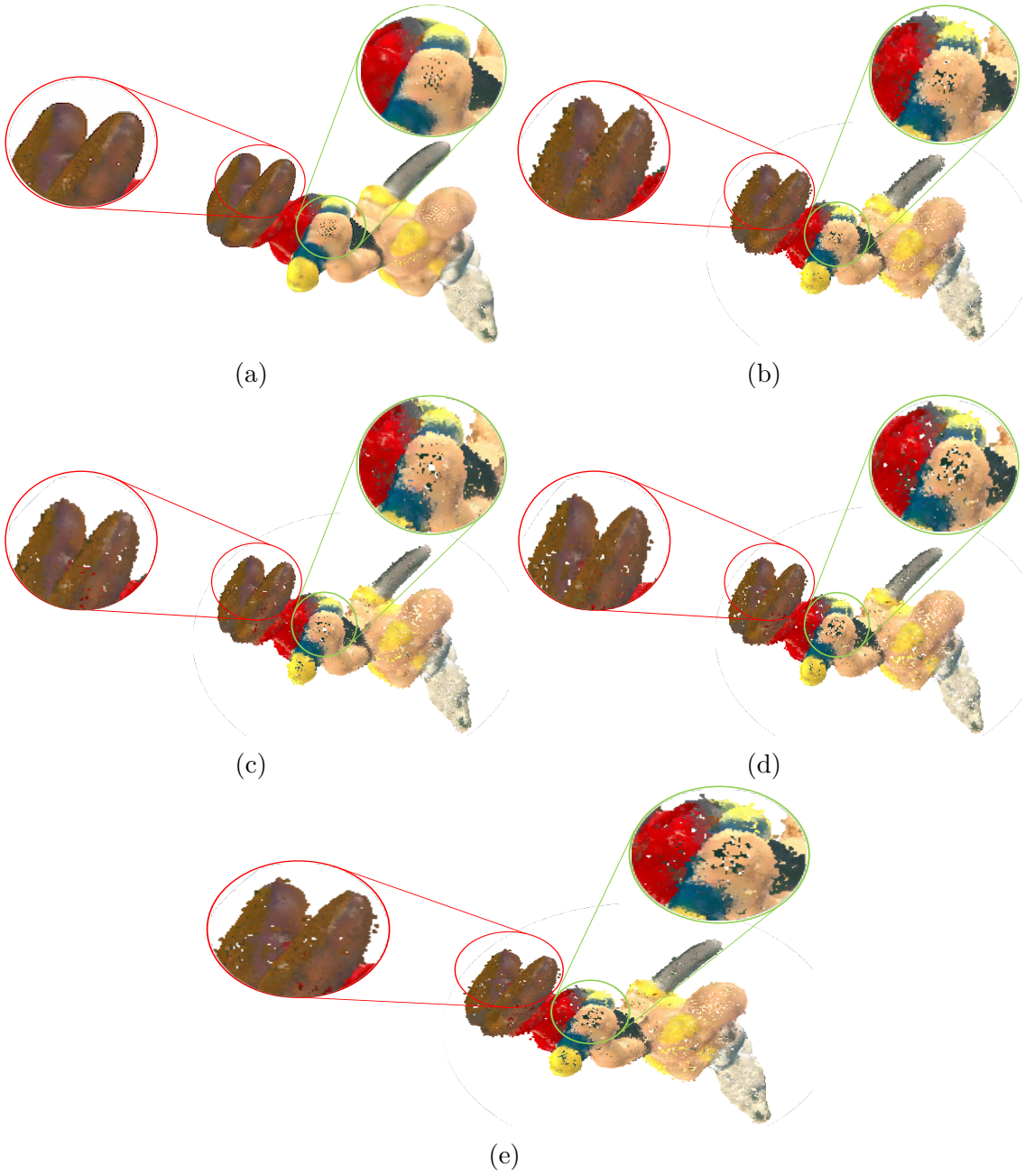
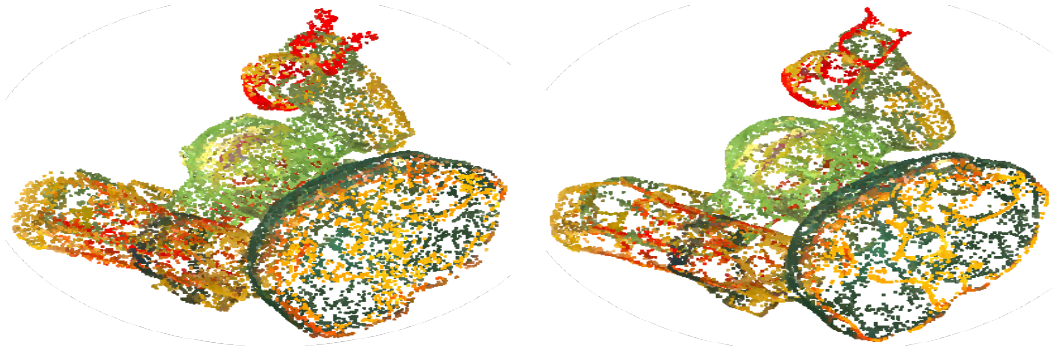


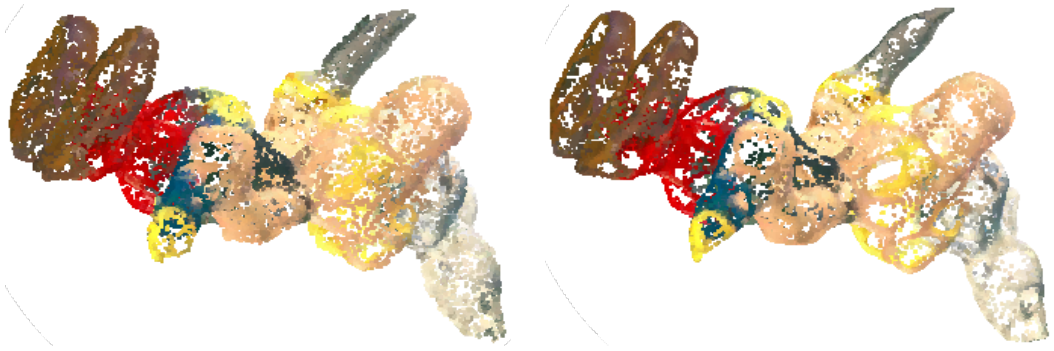
Figure 5.8: Asterix model: (a) ground-truth (b) noisy input, geometry denoised results by (c) proposed algorithm using Tikhonov regularization (d) geometry-only graph [112], and (e) MSGW [17].

noise in the point cloud, details are missing, and one can not see the contours in the real point cloud. The denoised point clouds look sharper in comparison to the input noisy point clouds. The color denoising procedure helps to preserve object



(a) (b)

Figure 5.9: Green_monster model: (a) denoised results by proposed algorithm using Tikhonov regularizatoin, and (b) RPSM [16].



(a) (b)

Figure 5.10: Asterix model: (a) denoised results by proposed algorithm using Tikhonov regularization, and (b) RPSM [16].

boundaries. Fig. 5.11d and Fig. 5.12d show the denoised point cloud using TV; it can be seen that the color is still noisy, and there is a lack of details in the output point clouds. TV is not very effective at enforcing color smoothness in comparison to the proposed algorithm using Tikhonov regularization.

Color denoising of synthetic point clouds

The proposed algorithm for color denoising has been applied to noise-free point clouds affected by synthetic color noise; Gaussian distribution is used to add noise to the color attribute of every point in a reference point cloud while keeping the geometry noise-free. Fig. 5.13a and Fig. 5.14a present the ground-truth point cloud having noise-free geometry and color. Fig. 5.13b and Fig. 5.14b show the point cloud affected by Gaussian noise with $\mu = 0$ and $\sigma = 30$; adding noise to the color affects the details and causes blurring of boundaries. Fig. 5.13c and 5.14c depict

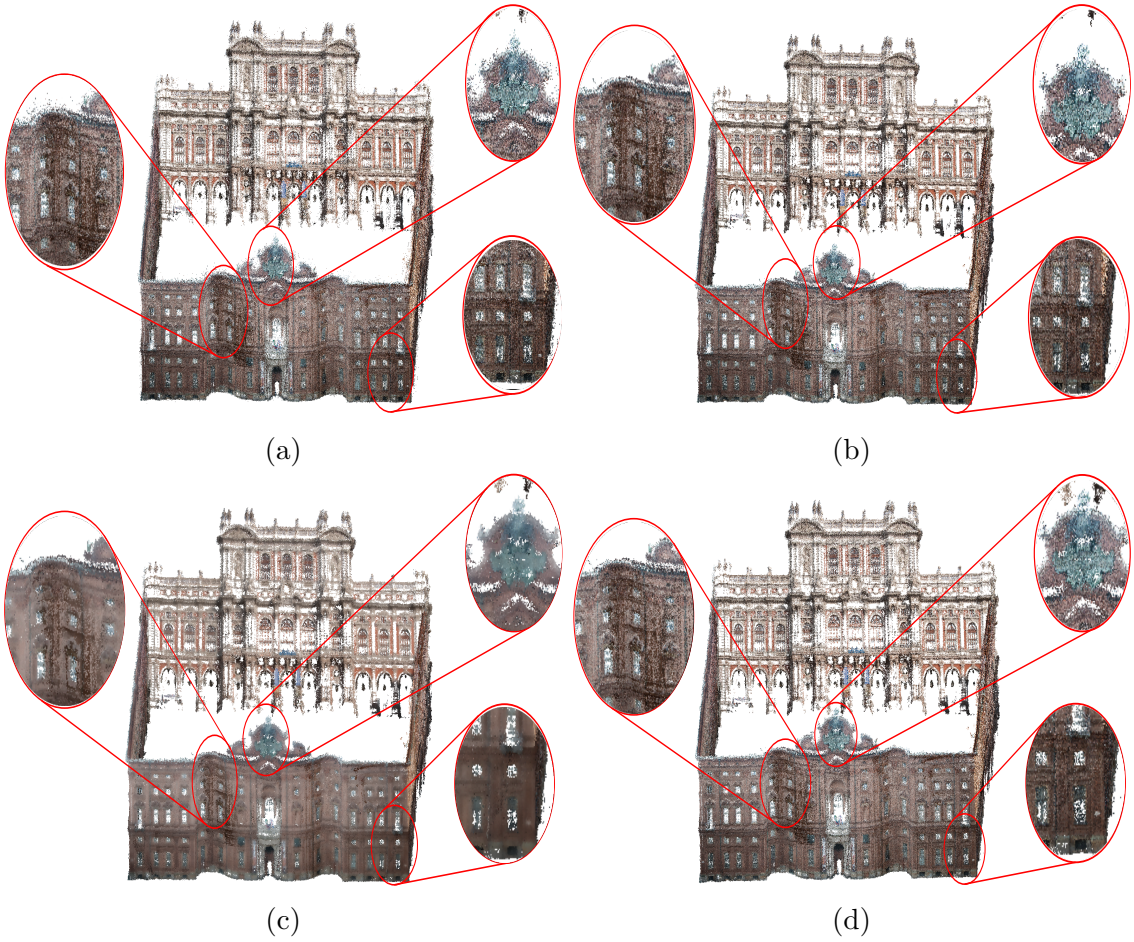


Figure 5.11: Palazzo_Carignano_Dense model illustration. (a) noisy input, (b) outlier-free input, color denoised results by (c) proposed algorithm using Tikhonov regularization, and (d) using TV.

the denoised output of the proposed algorithm using Tikhonov regularization. The color of the output point cloud is denoised by exploiting the correlation of color within the proximity, and the points in the k-neighborhood have a high probability of having a similar color as the surface has smooth color. Fig. 5.13d and Fig. 5.14d illustrate the denoised output using TV. The output point clouds are still noisy, and the details are not preserved. The TV technique is the least effective in terms of color denoising.

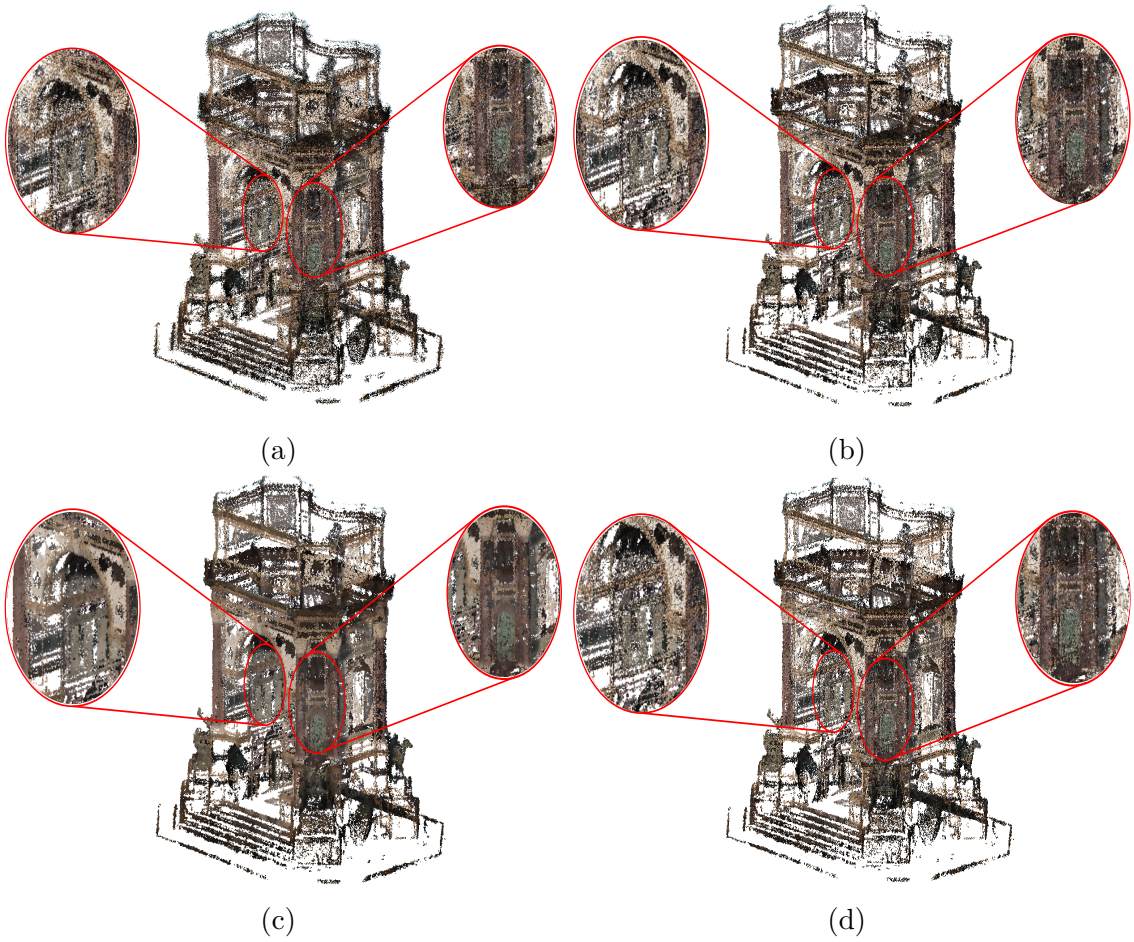


Figure 5.12: Arco_Valentino model illustration. (a) noisy input, (b) outlier-free input, color denoised results by (c) proposed algorithm using Tikhonov regularization, and (d) using TV.

5.5.4 Visual analysis of combined geometry and color denoising algorithm

Geometry and color denoising of natural point clouds

The proposed denoising algorithm is applied to the natural point cloud with real noise in geometry and color. The visual results of the comparison between the point cloud denoising by the proposed technique using Tikhonov and TV are described here. Fig. 5.15a and Fig. 5.16a show a real-world point cloud containing noise both in geometry and color. Due to the noise, the structural information in the real point clouds is lost, and fine details are not visible. Fig. 5.15b and Fig. 5.16b are the outcomes after applying the outlier removal algorithm, eliminating the noisiest points, thereby helping in the construction of a useful k-NN graph. Fig. 5.15c and

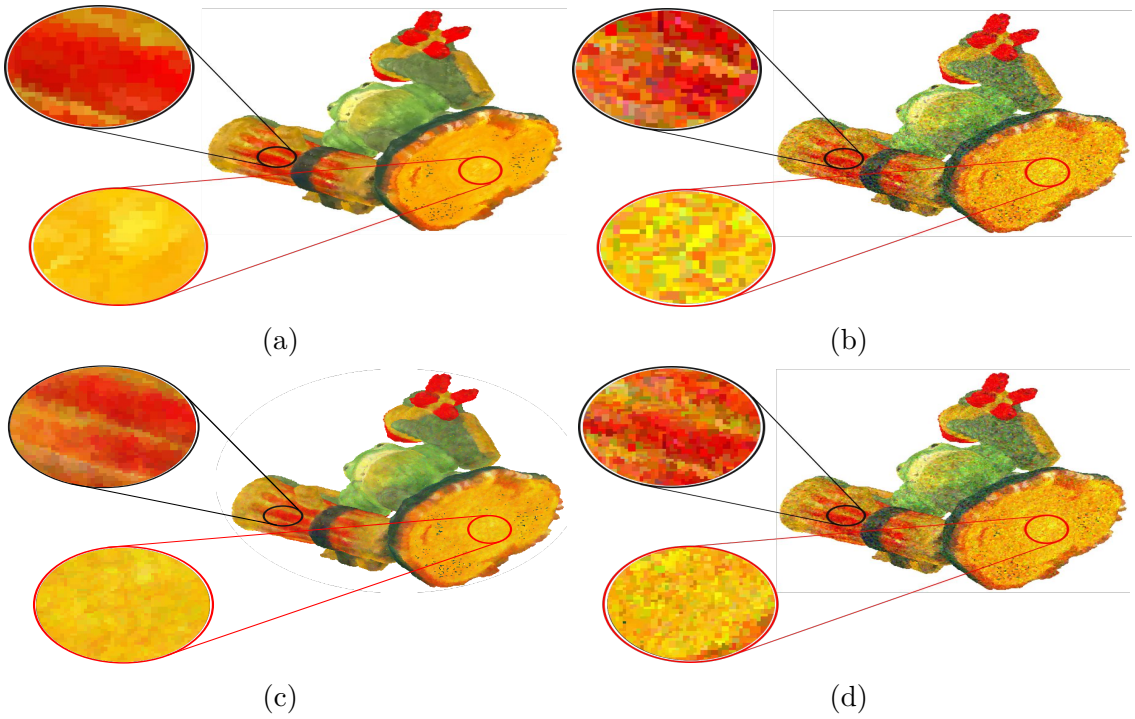


Figure 5.13: Green_monster model: (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0$ and $\sigma = 30$, color denoised results by (c) proposed algorithm using Tikhonov, and (d) using TV.

Fig. 5.16c depict the denoised point clouds obtained by the proposed algorithm using Tikhonov regularization. The geometry noise is removed by moving the noisy points closer to their original position, and also, the color becomes smoother by exploiting the correlation of geometry and color of points in a point cloud. Structural information is preserved in the resulting point cloud, and one can see more details, which were partly hidden in the noisy point cloud. Fig. 5.15d and Fig. 5.16d show the denoised point cloud by using TV. The TV also performs geometry denoising but has a minimal denoising effect on the color noise; it can be seen that the details are not very well-defined in the output point clouds because the colors are not smooth enough.

Geometry and color denoising of synthetic point clouds

Here we present the subjective results of the proposed combined geometry and color denoising approach applied to noise-free point clouds affected by synthetic geometry and color noise. The noise in the geometry is added to 50% of the points using a zero-mean uniform noise with $\mu = 0$ and $\sigma = 0.3$, 0.4, and 0.5; the color of each point is corrupted by zero-mean Gaussian noise with $\sigma = 20$, 30, and 40. For clarity, the subjective results of combined geometry and color denoising are shown

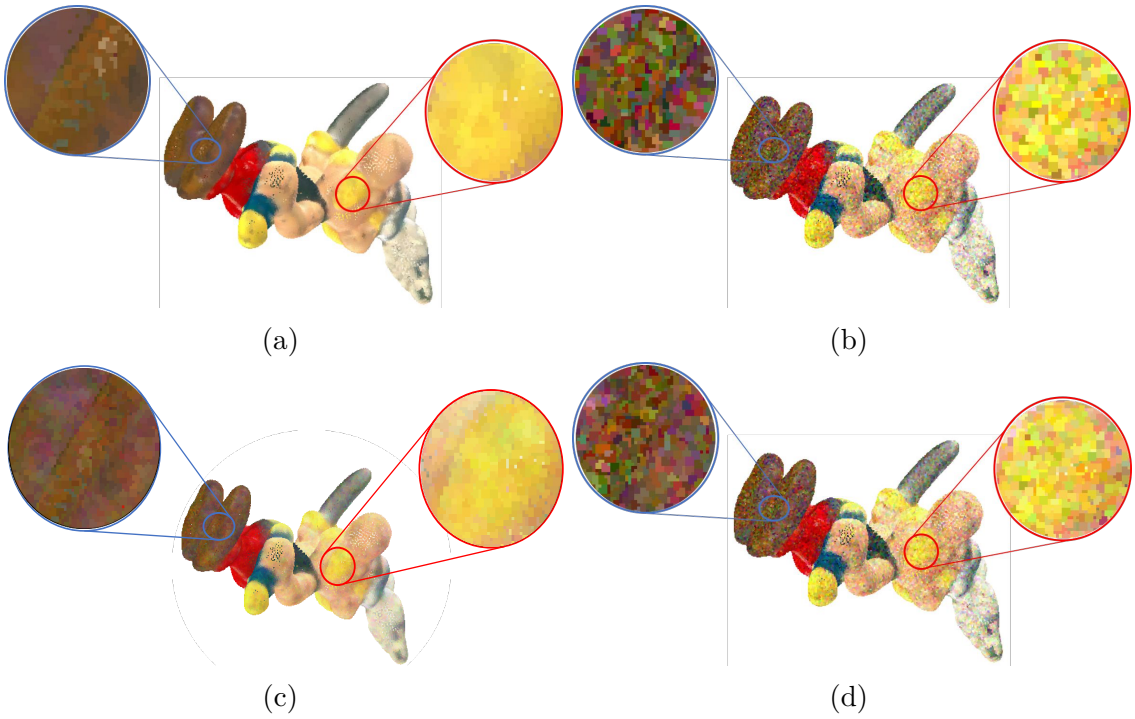
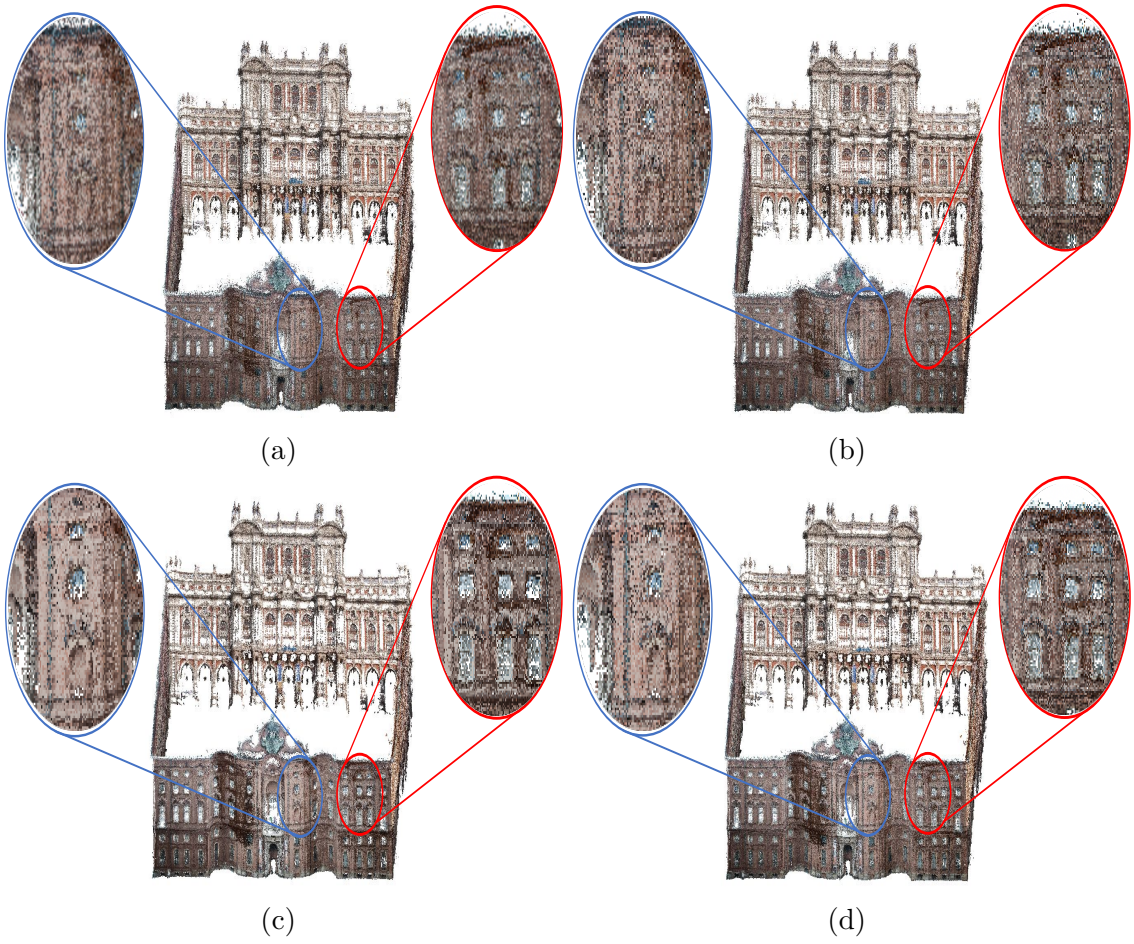


Figure 5.14: Asterix model: (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0$ and $\sigma = 30$, color denoised results by (c) proposed algorithm using Tikhonov, and (d) using TV.

in separate figures focusing on the color and geometry individually. The effect of color denoising in the combined geometry and color denoising algorithm is shown in Fig. 5.17 and Fig. 5.18. The noise-free point cloud is shown in Fig. 5.17a and Fig. 5.18a. Fig. 5.17b and Fig. 5.18b depict the noisy point clouds highlighting only the color noise added in the noise-free synthetic point cloud. The denoised point clouds obtained by our proposed algorithm using Tikhonov regularization are shown in Fig. 5.17c and 5.18c, highlighting the color denoising effect in combined geometry and color denoising. It can be seen that the denoised output is cleaned from the color noise, and the colors in the output point cloud are closer to the exact color of the ground-truth input point cloud. Fig. 5.17d and Fig. 5.18d are the resulting denoised point clouds of the proposed algorithm using TV. The TV has a small effect at removing the noise from the color.

The effect of the proposed algorithm for geometry denoising can be seen in Fig. 5.19 and Fig. 5.20. The noise-free input point clouds are shown in Fig. 5.19a and Fig. 5.20a. The noisy point clouds highlighting only the geometry noise can be seen in Fig. 5.19b and Fig. 5.20b. The outcome of the proposed denoising algorithm is depicted in Fig. 5.19c and Fig. 5.20c, focusing only on the denoising effect on geometry. It can be seen that the geometry noise has been regularized, and the



(a) (b) (c) (d)

Figure 5.15: Palazzo_Carignano_Dense model illustration. (a) noisy input, (b) outlier-free input, combined geometry and color denoised results by (c) proposed algorithm using Tikhonov regularizarion, and (d) using TV.

noisy points are moved close to their original positions; moreover, as anticipated in Sec. 1, our approach avoids generating artifacts by leaving fewer holes in the output denoised point cloud. Fig. 5.19d and Fig. 5.20d are the denoised output of TV regularization. The TV also helps in regularizing the noisy points. Overall we can perceive from the qualitative results of the synthetic point clouds that the point clouds denoised by the proposed algorithm using Tikhonov have better quality both in color and geometry denoising and exhibit fewer artifacts.

5.5.5 Objective evaluation on Greyc color mesh dataset

The quantitative evaluation has also been performed on the *Greyc* noise-free synthetic point clouds dataset.

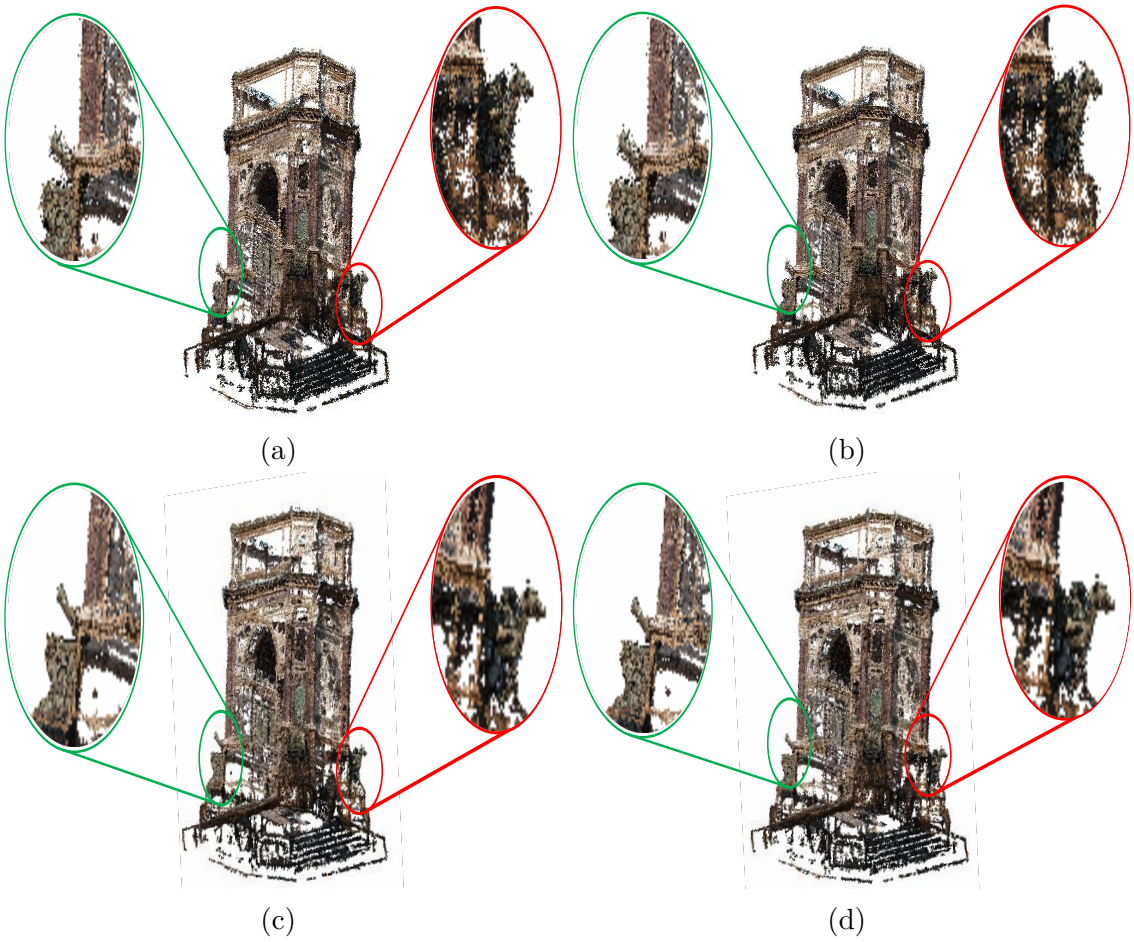


Figure 5.16: Arco_Valentino model illustration. (a) noisy input, (b) outlier-free input, combined geometry and color denoised results by (c) proposed algorithm using Tikhonov regularizarion, and (d) using TV.

Color denoising

The color attribute of each point cloud is corrupted with Gaussian noise applied to each point in a point cloud with $\sigma = 20, 30$, and 40. The MSE and PSNR comparisons between the proposed color denoising algorithm using Tikhonov and TV regularization are shown in Tab. 5.2 and Tab. 5.3. The results of both metrics show that the proposed technique via Tikhonov regularization performed far better than the TV regularization. The average gain in MSE and PSNR is demonstrated in Fig. 5.21 for three different noise levels. It can be seen that with the increase in the intensity of noise level, the proposed color denoising method using Tikhonov performs much better than the TV regularization.

Gaussian noise with zero-mean and $\sigma = 10, 15, 20$, and 25 is added to the color attribute of the noise-free point cloud models of *Greyc* dataset for the comparison

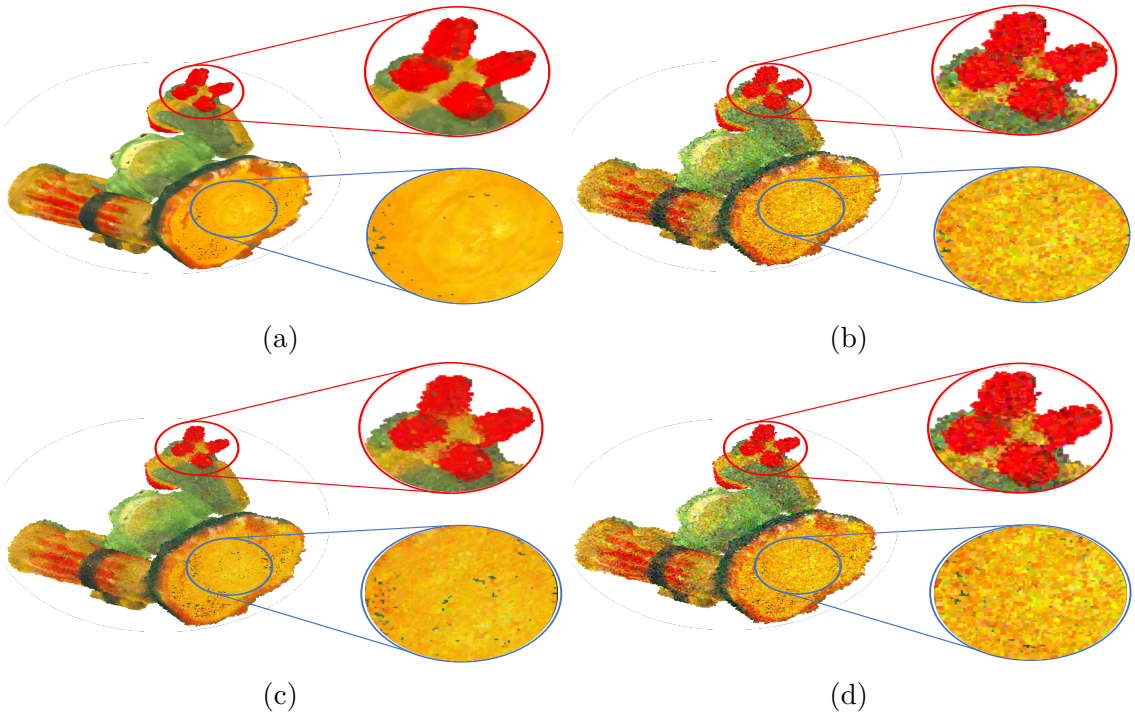


Figure 5.17: Green_monster model: Highlighting only the color denoising effect in combined geometry and color denoising algorithm: (a) ground-truth (b) noisy point cloud with $\sigma = 30$ in color attribute, color denoised results in combined geometry and color by (c) proposed algorithm using Tikhonov, and (d) using TV.

Table 5.2: MSE comparison of color denoising algorithm for *Greyc dataset*.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 20$	Noisy	398.25	361.28	373.52	393.64	339.57	397.24	365.38	383.29	368.56	387.52	321.09	375.24	309.14	335.40	377.39	398.13
	Proposed	**77.75**	**103.56**	**145.23**	**82.80**	**130.65**	**63.04**	**91.64**	**150.11**	**85.83**	**77.90**	**95.91**	**89.56**	**149.53**	**93.75**	**82.52**	**84.77**
	TV	303.05	279.02	289.48	298.13	286.05	301.50	279.04	299.68	279.21	293.61	242.14	283.15	241.15	254.99	293.32	302.99
$\sigma = 30$	Noisy	869.27	781.84	816.58	867.14	722.06	875.25	796.19	850.03	797.42	850.88	692.54	810.94	645.11	713.61	811.92	882.35
	Proposed	**109.74**	**148.53**	**214.89**	**120.77**	**189.47**	**89.12**	**138.24**	**217.08**	**130.09**	**112.13**	**157.44**	**136.40**	**250.35**	**148.58**	**133.63**	**123.06**
	TV	590.97	540.25	564.67	589.46	548.87	594.55	542.77	596.83	539.70	575.89	467.70	544.00	449.27	485.12	561.85	602.88
$\sigma = 40$	Noisy	1506.10	1329.00	1384.60	1462.90	1241.90	1469.80	1347.80	1462.10	1355.50	1462.60	1171.90	1372.90	1092.10	1217.80	1376.20	1519.90
	Proposed	**137.31**	**210.90**	**281.89**	**149.67**	**262.51**	**133.94**	**184.53**	**279.98**	**175.90**	**144.71**	**228.37**	**185.25**	**385.53**	**213.93**	**242.06**	**154.26**
	TV	1119.50	995.38	1033.80	1081.70	979.85	1088.40	1000.40	1107.10	1000.90	1083.80	862.00	1005.10	816.02	903.14	1026.80	1131.90

of the proposed color denoising algorithm with GLR [20, 90] and GTV [13]. Quantitative results in terms of PSNR are shown in Tab. 5.4, where GTV and GLR show

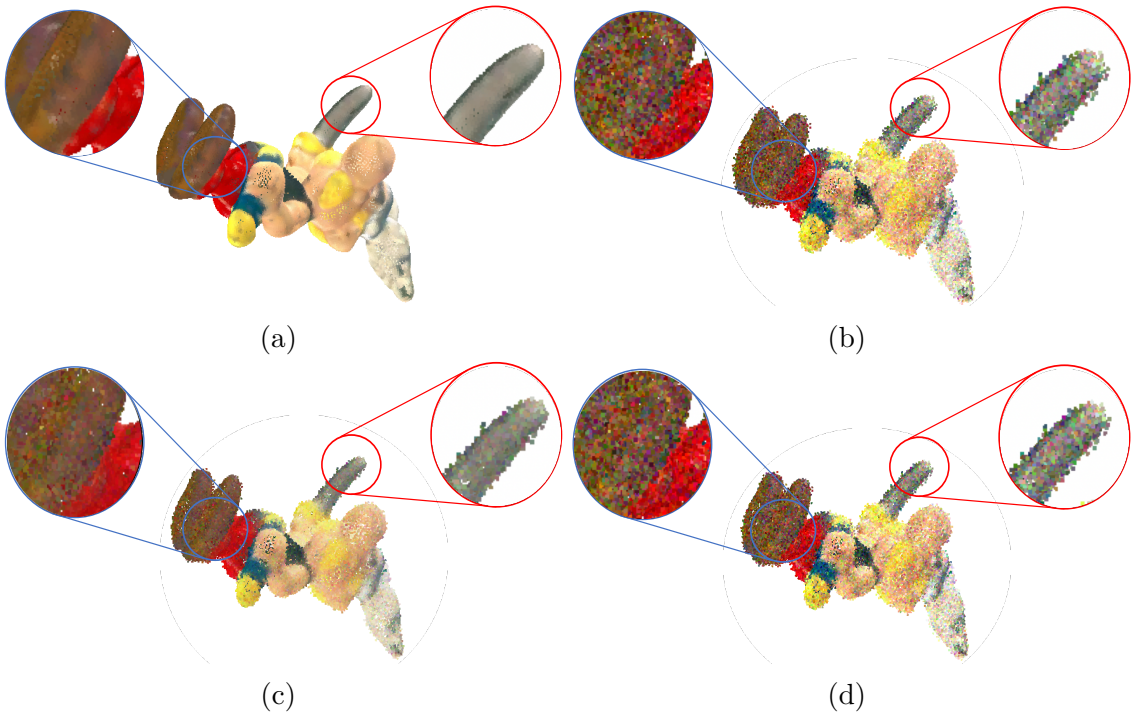


Figure 5.18: Asterix model: Highlighting only the color denoising effect in combined geometry and color denoising algorithm: (a) ground-truth (b) noisy point cloud with $\sigma = 30$ in color attribute, color denoised results in combined geometry and color by (c) proposed algorithm using Tikhonov, and (d) using TV.

Table 5.3: PSNR comparison of color denoising algorithm for *Greyc dataset*.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 20$	Noisy	22.13	22.55	22.41	22.18	22.82	22.14	22.50	22.30	22.47	22.25	23.07	22.39	23.23	22.88	22.36	22.13
	Proposed	**29.22**	**27.98**	**26.53**	**28.95**	**26.97**	**30.13**	**28.50**	**26.38**	**28.80**	**29.19**	**28.33**	**28.61**	**26.43**	**28.41**	**28.96**	**28.84**
	TV	23.32	23.67	23.52	23.39	23.57	23.34	23.67	23.36	23.67	23.45	24.29	23.61	24.31	24.07	23.46	23.32
$\sigma = 30$	Noisy	18.74	19.20	19.01	18.75	19.55	18.71	19.12	18.84	19.11	18.83	19.73	19.04	20.03	19.60	19.04	18.67
	Proposed	**27.72**	**26.41**	**24.81**	**27.31**	**25.35**	**28.63**	**26.72**	**24.76**	**26.98**	**27.63**	**26.15**	**26.78**	**24.14**	**26.33**	**26.87**	**27.23**
	TV	20.42	20.81	20.61	20.43	20.74	20.39	20.79	20.37	20.81	20.53	21.43	20.78	21.60	21.27	20.64	20.33
$\sigma = 40$	Noisy	16.35	16.90	16.72	16.48	17.19	16.46	16.84	16.48	16.81	16.48	17.44	16.75	17.75	17.28	16.74	16.31
	Proposed	**26.36**	**24.90**	**23.63**	**26.38**	**23.95**	**26.86**	**25.47**	**23.66**	**25.68**	**26.53**	**24.54**	**25.45**	**22.27**	**24.83**	**24.29**	**26.25**
	TV	17.64	18.15	17.97	17.79	18.22	17.76	18.13	17.69	18.13	17.78	18.78	18.11	19.01	18.57	18.01	17.59

the highest average PSNR value for noise level $\sigma = 10$ and $\sigma = 15$, respectively. With the increase in the noise level, the proposed algorithm performs better than

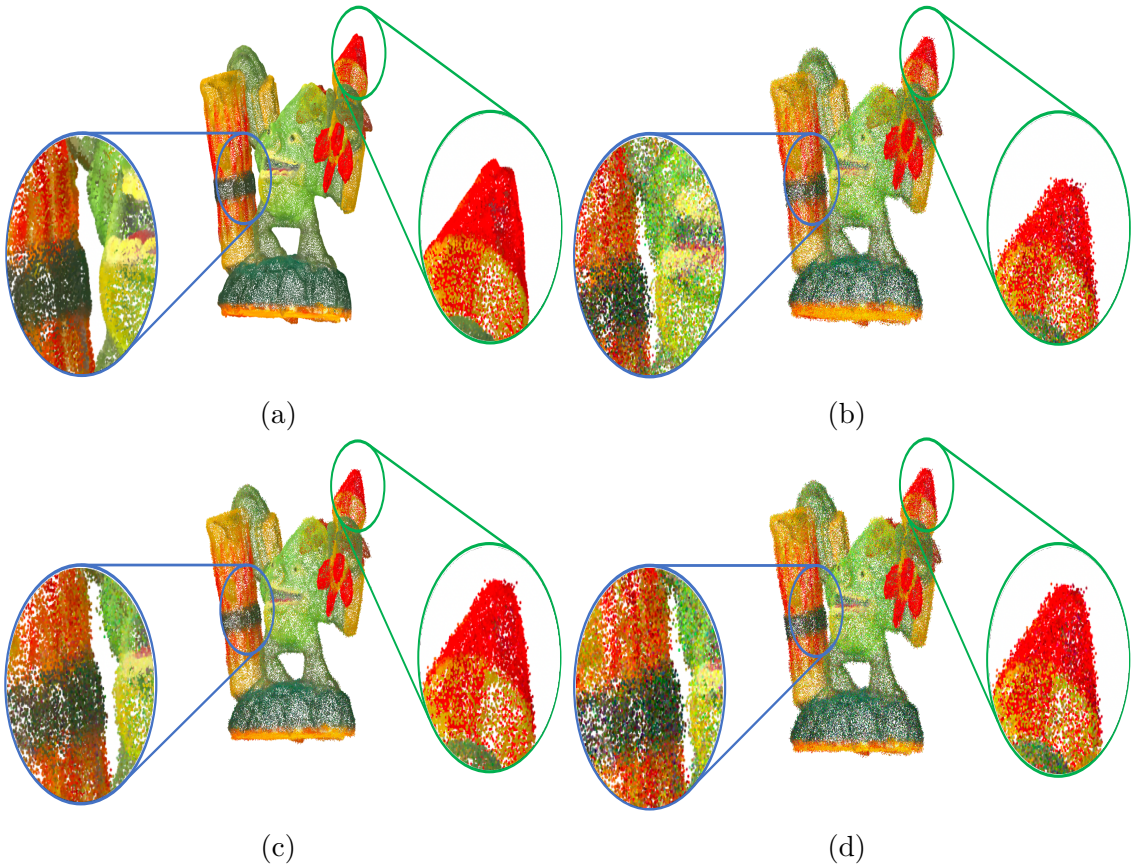


Figure 5.19: Green_monster model: Same output point cloud as Fig. 5.17, highlighting geometry denoised effect in combined geometry and color denoising technique, (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0, \sigma = 0.4$ in geometry attribute, geometry denoised results in combined geometry and color by (c) proposed algorithm, and (d) using TV.

GLR and GTV, with an average PSNR increased by 0.25dB and 0.09dB, respectively. Besides PSNR, the average execution time (AET) per noise level for the proposed, GLR, and GTV algorithms is also shown in Tab. 5.4. The proposed method is faster because it denoises the color using smaller γ, which in turn performs regularization in a smaller number of iterations. AET has been measured in MATLAB 2017b on a 3.5Ghz MacBook Pro with an Intel Core i7 processor and 16GB memory.

Geometry denoising

Each point cloud has been corrupted with uniform synthetic geometry noise, applied to 50% of the points with $\sigma = 0.3, 0.4$, and 0.5. The comparison between the proposed algorithm and the denoising approach used in [112] and MSGW [17]

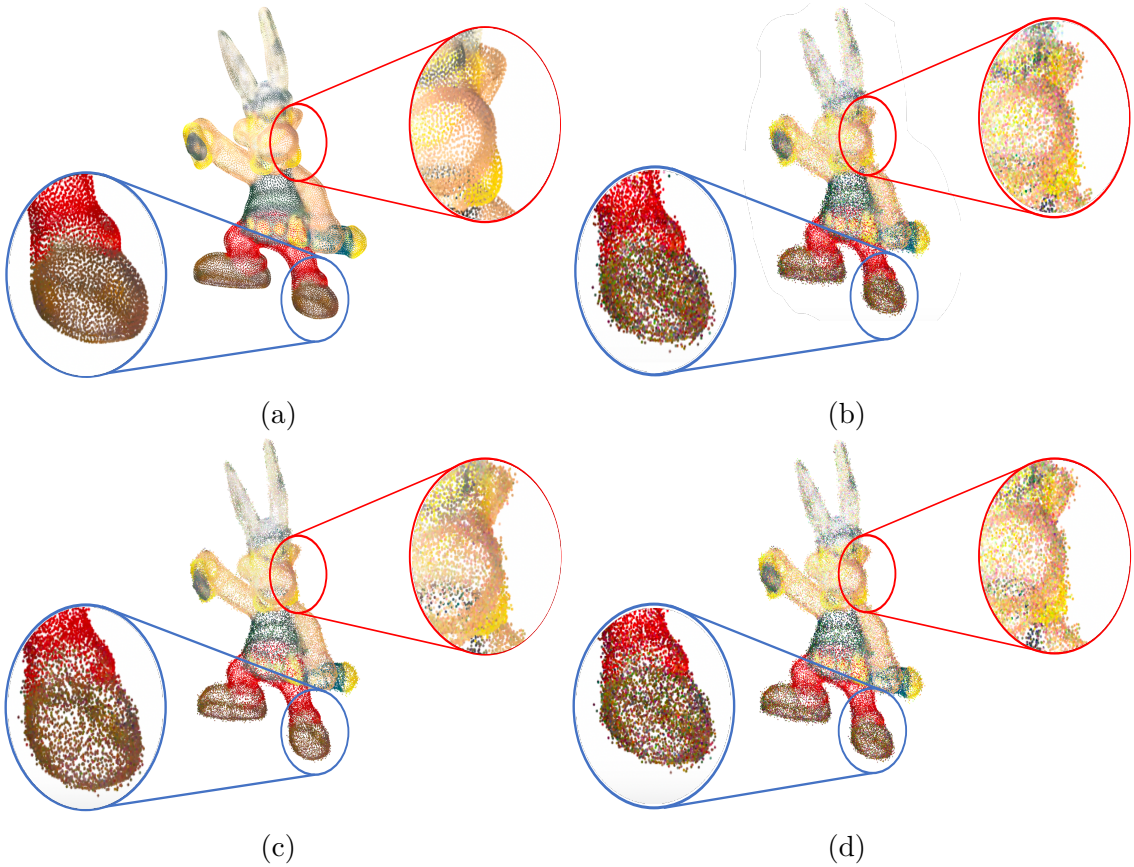


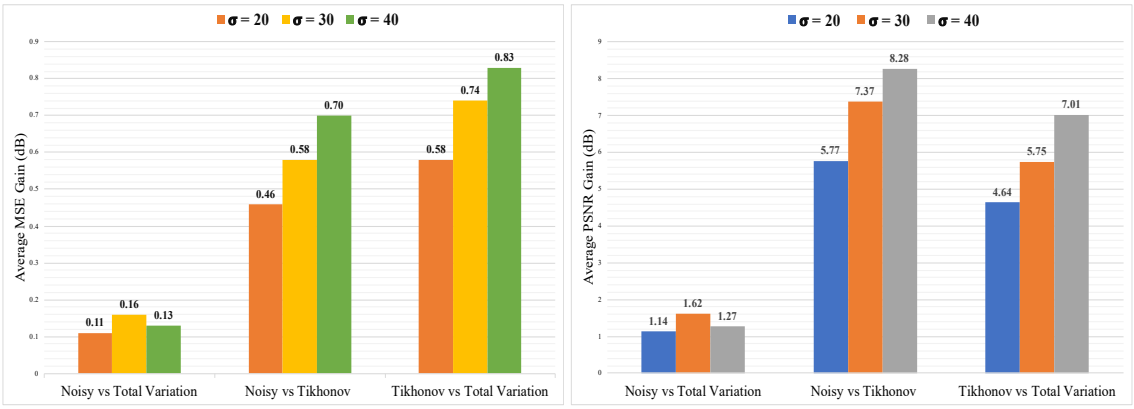
Figure 5.20: Asterix model: Same output point cloud as Fig. 5.18, highlighting geometry denoised effect in combined geometry and color denoising technique, (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0, \sigma = 0.4$ in geometry attribute, geometry denoised results in combined geometry and color by (c) proposed algorithm, and (d) using TV.

is shown in Tab. 5.7. The results show that the proposed denoising technique performs better than [112] and [17] in terms of all the metrics in the cloud to mesh distance at all noise levels. For $\sigma = 0.3$, the *Green monster, Mario, Pokeman ball, and Statue* show good results for the geometry-only algorithm [112]. Still, with the increase in noise intensity, the proposed algorithm performs better for all the models in the *Greyc* mesh dataset. The gain is more significant as the noise level increases, showing that the proposed denoising method is indeed better at removing geometry noise.

To check the robustness of the proposed algorithm to other noise distribution, Gaussian noise is added to each point cloud of the *Greyc* dataset with $\sigma = 0.2$,0.3, and 0.4. The comparative results are shown in Tab. 5.8, proving that the proposed algorithm yields good results with respect to the denoising techniques described

Table 5.4: Color denoising comparison for Gaussian noise $\sigma = 10, 15, 20$, and 25 with GLR-based [20] and GTV-based [13] in terms of PSNR and AET (s).

Model	$\sigma = 10$				$\sigma = 15$				$\sigma = 20$				$\sigma = 25$				AET (s)		
	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Proposed	GLR	GTV
Asterix	28.38	31.32	**32.03**	31.61	24.94	29.21	**29.66**	29.53	22.53	27.98	**28.10**	27.56	20.68	26.69	26.12	**27.05**	**2.55**	5.20	211.00
Duck	28.53	**30.77**	30.40	30.60	25.20	**28.63**	28.42	28.14	22.71	**26.97**	26.89	26.35	20.90	**25.81**	25.68	25.04	**0.792**	1.70	58.00
Green_Dinosaur	28.14	33.17	33.28	**33.36**	24.60	31.28	**31.64**	31.31	22.13	30.13	30.30	**30.34**	20.23	29.23	28.52	**29.62**	**3.99**	11.20	389.00
Red_Horse	28.29	**32.43**	32.15	32.20	24.74	**30.17**	29.72	30.01	22.35	**28.96**	27.85	28.57	20.47	**27.38**	25.66	27.33	**8.13**	19.50	851.00
Average	28.30	31.92	31.87	**31.94**	24.85	29.82	**29.86**	29.75	22.44	**28.51**	28.29	28.20	20.61	**27.27**	26.495	27.26	**3.86**	9.40	377.25



(a) (b)

Figure 5.21: (a): Average gain in MSE (dB) for the color denoising algorithm (b): Average gain in PSNR (dB) for the color denoising algorithm.

in [112] and [17] at all noise levels for C2M distance. At noise level $\sigma = 0.2$, the geometry-only [112] algorithm performs better for *Asterix*, *Pokeman ball*, and *Rabbit* models; however, only the *duck* shows good results for MSGW [17] for all the noise intensities. The proposed denoising algorithm performs better for all the models except *Duck* as the noise level increases.

For performance comparison between the proposed algorithm and RPSM [16], we need a ground-truth mesh for each point cloud of the Greyc dataset [85]. Due to the large memory requirement of RPSM, we performed experiments on sub-sampled point clouds, for which the sub-sampled ground-truth mesh is not available. Hence, we employ on these sub-sampled point clouds the same metrics as in [140]

Table 5.5: MSE comparison on sub-sampled *Greyc* dataset.

Model	$\sigma = 0.3$		$\sigma = 0.4$		$\sigma = 0.5$	
	Proposed	RPSM [16]	Proposed	RPSM [16]	Proposed	RPSM [16]
4arms Monstre	**0.29**	0.48	**0.34**	0.51	**0.42**	0.52
Asterix	**0.26**	0.31	**0.28**	0.34	**0.30**	0.37
Cable car	**0.36**	0.74	**0.38**	0.80	**0.40**	0.82
Dragon	**0.25**	0.44	**0.26**	0.48	**0.28**	0.50
Green monster	**0.33**	0.62	**0.36**	0.65	**0.37**	0.67
Rabbit	**0.31**	0.58	**0.32**	0.60	**0.35**	0.63
Red horse	**0.34**	0.51	**0.36**	0.55	**0.39**	0.58

Table 5.6: MCD comparison on sub-sampled *Greyc* dataset.

Model	$\sigma = 0.3$		$\sigma = 0.4$		$\sigma = 0.5$	
	Proposed	RPSM [16]	Proposed	RPSM [16]	Proposed	RPSM [16]
4arms Monstre	**0.43**	0.70	**0.57**	0.90	**0.62**	1.00
Asterix	**0.38**	0.45	**0.41**	0.48	**0.44**	0.53
Cable car	**0.51**	0.92	**0.54**	0.98	**0.57**	1.05
Dragon	**0.37**	0.65	**0.37**	0.68	**0.42**	0.70
Green monster	**0.49**	0.71	**0.52**	0.77	**0.55**	0.80
Rabbit	**0.46**	0.80	**0.49**	0.82	**0.51**	0.88
Red horse	**0.50**	0.75	**0.54**	0.78	**0.57**	0.81

also described in Sec. 4.3.2. Tab. 5.5 and Tab. 5.6 show the MSE and MCD comparisons between the proposed algorithm and RPSM [16] on sub-sampled point clouds. The results clearly indicate that the proposed algorithm performs better than RPSM [16].

Table 5.7: C2M metric comparison of the proposed geometry denoising algorithm with the geometry-only graph approach [112] and MSGW [17] for uniform noise.

Noise level in geometry	Methods	Parameters	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 0.3$	Proposed	d_H	**0.96**	0.97	**1.17**	**1.29**	**1.56**	**1.50**	1.18	**0.94**	**1.17**	1.45	1.18	**0.98**	1.01	**1.32**	**1.27**	0.90
		d_m	$<$ **0.01**	**0.01**	**0.03**	**0.01**	**0.02**	$\approx$ **0.00**	0.02	$<$ **0.01**	**0.03**	**0.01**	0.04	**0.03**	0.03	$<$ **0.01**	$<$ **0.01**	0.01
		ζ	**0.07**	**0.07**	**0.16**	**0.11**	**0.18**	**0.04**	0.10	**0.06**	**0.14**	**0.13**	0.15	**0.14**	0.11	**0.06**	**0.10**	0.09
	Geometry-only [112]	d_H	0.96	**0.75**	1.17	1.29	1.56	1.50	**0.75**	0.94	1.17	**1.02**	**1.02**	**0.98**	**0.64**	1.33	1.28	**0.90**
		d_m	0.01	0.02	0.04	0.02	0.02	$<$ 0.01	**0.02**	$<$ 0.01	0.03	0.02	**0.04**	**0.03**	**0.03**	$<$ 0.01	0.02	**0.01**
		ζ	0.08	0.08	0.18	0.14	0.19	0.08	**0.10**	0.08	0.15	0.15	**0.15**	**0.14**	**0.11**	0.08	0.14	**0.08**
	MSGW [17]	d_H	1.10	0.98	1.31	1.55	1.97	1.86	1.25	1.31	1.29	1.20	1.11	1.34	0.99	1.59	1.65	1.50
		d_m	0.02	0.10	0.09	0.08	0.11	0.01	0.08	0.01	0.07	0.05	0.07	0.18	0.07	0.01	0.04	0.01
		ζ	0.07	0.13	0.27	0.22	0.35	0.13	0.17	0.10	0.20	0.21	0.19	0.26	0.15	0.10	0.18	0.10
$\sigma = 0.4$	Proposed	d_H	**0.96**	**0.97**	**1.17**	**1.29**	**1.56**	**1.50**	**1.18**	**0.94**	**1.17**	**1.45**	**1.18**	**0.98**	**1.01**	**1.32**	**1.27**	**0.90**
		d_m	**0.01**	**0.02**	**0.05**	**0.03**	**0.03**	$<$ **0.01**	**0.04**	**0.01**	**0.05**	**0.03**	**0.07**	**0.05**	**0.03**	**0.01**	**0.02**	**0.02**
		ζ	**0.09**	**0.11**	**0.21**	**0.17**	**0.20**	**0.09**	**0.15**	**0.10**	**0.20**	**0.19**	**0.20**	**0.19**	**0.13**	**0.11**	**0.17**	**0.12**
	Geometry Only [112]	d_H	1.18	0.97	1.17	1.29	1.57	2.12	**1.18**	1.32	1.18	1.45	1.32	1.20	1.01	1.32	1.80	1.27
		d_m	0.02	0.03	0.06	0.04	0.03	0.01	**0.04**	0.01	0.06	0.04	0.08	0.06	0.04	0.02	0.03	0.02
		ζ	0.14	0.11	0.22	0.19	0.21	0.13	**0.15**	0.11	0.21	0.20	0.21	0.20	0.14	0.12	0.20	0.12
	MSGW [17]	d_H	1.26	1.25	1.48	1.82	2.13	2.12	1.91	1.87	1.56	1.76	1.68785	1.86	1.35	1.86	1.87	1.79
		d_m	0.04	0.18	0.14	0.13	0.18	0.02	0.13	0.01	0.14	0.10	0.15	0.27	0.09	0.02	0.06	0.02
		ζ	0.17	0.23	0.31	0.32	0.50	0.19	0.23	0.11	0.31	0.30	0.29	0.32	0.19	0.15	0.28	0.13
$\sigma = 0.5$	Proposed	d_H	**1.37**	**1.07**	**1.43**	**1.30**	**1.59**	**1.51**	**1.19**	**1.33**	**1.18**	**1.46**	**1.32**	**1.55**	**1.11**	**1.32**	**1.80**	**1.28**
		d_m	**0.02**	**0.04**	**0.09**	**0.06**	**0.03**	**0.01**	**0.07**	**0.02**	**0.08**	**0.05**	**0.11**	**0.08**	**0.05**	**0.02**	**0.05**	**0.03**
		ζ	**0.14**	**0.14**	**0.26**	**0.23**	**0.21**	**0.14**	**0.18**	**0.15**	**0.25**	**0.22**	**0.23**	**0.23**	**0.15**	**0.16**	**0.24**	**0.16**
	Geometry Only [112]	d_H	1.53	1.31	1.44	1.32	1.63	2.13	1.34	1.35	1.44	1.47	1.45	1.56	1.12	1.33	1.89	1.28
		d_m	0.03	0.05	0.12	0.07	0.05	0.02	0.08	0.03	0.09	0.07	0.12	0.10	0.06	0.03	0.07	0.04
		ζ	0.14	0.15	0.26	0.24	0.27	0.16	0.20	0.19	0.26	0.25	0.25	0.25	0.16	0.16	0.245	0.17
	MSGW [17]	d_H	2.05	1.77	1.75	1.98	2.45	2.50	1.99	1.89	1.86	1.77	1.96	1.87	1.63	2.09	1.90	1.48
		d_m	0.04	0.18	0.14	0.14	0.18	0.03	0.15	0.04	0.16	0.12	0.17	0.30	0.09	0.03	0.10	0.03
		ζ	0.18	0.24	0.32	0.33	0.51	0.20	0.26	0.24	0.34	0.51	0.32	0.34	0.19	0.17	0.29	0.20

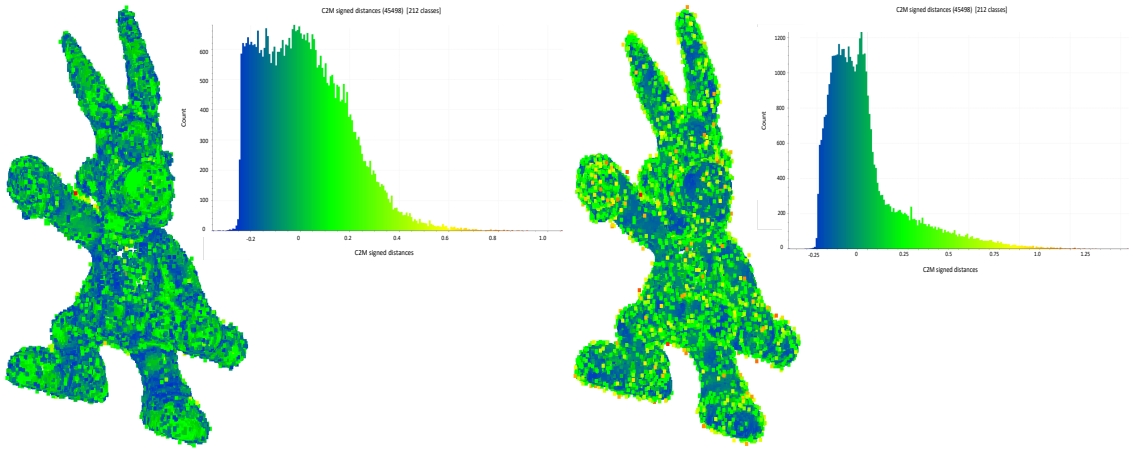
Combined geometry and color denoising

The geometry and color of each reference point cloud have been altered with uniform synthetic geometry noise and Gaussian color noise, respectively. The geometry of 50% of the points in each point cloud is affected with noise level $\sigma = 0.3, 0.4,$ and 0.5; however, the color of each point in a point cloud is affected with $\sigma = 20, 30,$ and 40. The C2M metric with respect to the corresponding reference point cloud has been computed for the denoised point cloud by the proposed algorithm using

Table 5.8: C2M metric comparison of the proposed geometry denoising algorithm with the geometry-only graph approach [112] and MSGW [17] for Gaussian noise.

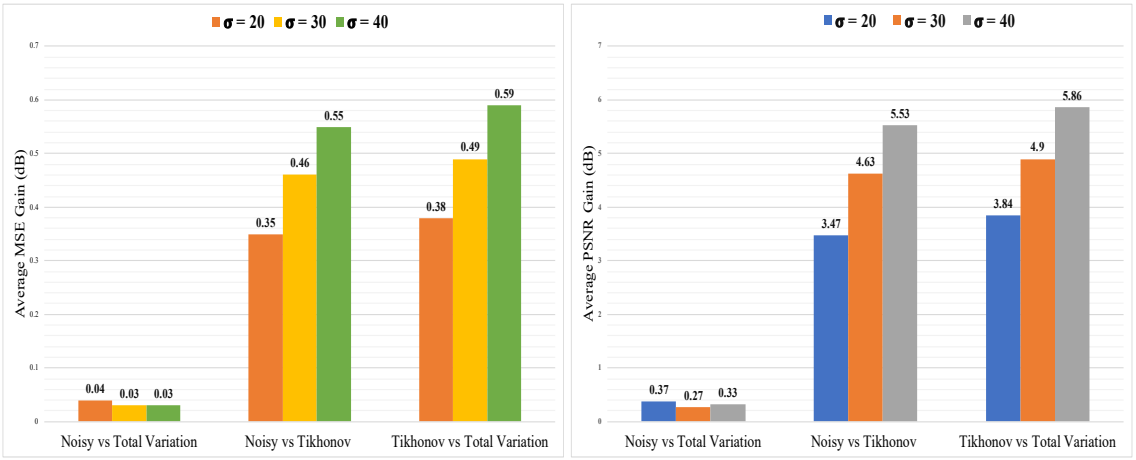
Noise level in geometry	Methods	Parameters	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 0.2$	Proposed	$\boldsymbol{d_H}$	**0.68**	0.61	**1.16**	**0.91**	1.56	**1.49**	**1.06**	**0.94**	**1.17**	**1.02**	**0.98**	**1.18**	0.64	0.93	**1.26**	**0.90**
		$\boldsymbol{d_m}$	**< 0.01**	0.01	**0.02**	**0.01**	0.02	**< 0.01**	**0.01**	**< 0.01**	**0.01**	**0.01**	**0.02**	**0.02**	0.03	< 0.01	**< 0.01**	**< 0.01**
		ζ	**0.04**	0.06	**0.12**	**0.08**	0.19	**0.02**	**0.07**	**0.03**	**0.11**	**0.09**	**0.10**	**0.12**	0.11	0.05	**0.07**	**0.05**
	Geometry Only [112]	$\boldsymbol{d_H}$	0.78	**0.61**	1.27	0.98	1.56	1.71	1.17	0.94	1.38	1.23	0.98	1.07	**0.45**	**0.93**	1.47	1.09
		$\boldsymbol{d_m}$	0.01	**0.01**	0.04	0.02	0.02	< 0.01	0.01	< 0.01	0.02	0.01	0.02	0.03	**0.03**	**< 0.01**	0.01	< 0.01
		ζ	0.05	**0.05**	0.16	0.12	0.19	0.05	0.09	0.53	0.12	0.13	0.12	0.14	**0.11**	**0.04**	0.12	0.07
	MSGW [17]	$\boldsymbol{d_H}$	1.37	0.97	1.18	0.92	**1.56**	1.51	1.07	0.95	1.41	1.02	0.99	1.18	0.91	0.94	1.81	1.28
		$\boldsymbol{d_m}$	0.02	0.02	0.03	0.02	**0.02**	< 0.01	0.02	0.01	0.03	0.02	0.03	0.05	0.04	0.02	0.01	0.02
		ζ	0.11	0.09	0.17	0.12	**0.16**	0.07	0.11	0.11	0.15	0.12	0.15	0.17	0.12	0.12	0.13	0.12
$\sigma = 0.3$	Proposed	$\boldsymbol{d_H}$	**0.96**	**1.05**	**1.17**	**1.28**	2.21	**1.50**	**1.18**	**1.30**	**1.17**	**1.03**	**1.38**	**1.19**	0.90	**1.32**	**1.27**	**0.90**
		$\boldsymbol{d_m}$	**0.01**	**0.02**	**0.04**	**0.03**	0.05	**< 0.01**	**< 0.01**	**< 0.01**	**0.03**	**0.02**	**0.04**	**0.06**	0.04	**0.01**	**0.02**	**0.01**
		ζ	**0.08**	**0.09**	**0.19**	**0.15**	0.27	**0.08**	**0.12**	**0.07**	**0.17**	**0.16**	**0.17**	**0.18**	0.13	**0.08**	**0.14**	**0.08**
	Geometry Only [112]	$\boldsymbol{d_H}$	0.96	1.06	1.66	1.29	1.57	1.96	1.60	1.32	1.66	1.65	1.58	1.33	**0.87**	1.32	1.68	1.28
		$\boldsymbol{d_m}$	0.01	0.04	0.07	0.04	0.05	0.01	0.04	0.01	0.05	0.03	0.06	0.09	**0.03**	0.01	0.02	0.01
		ζ	0.10	0.10	0.21	0.17	0.24	0.11	0.13	0.08	0.19	0.16	0.18	0.20	**0.12**	0.09	0.17	0.10
	MSGW [17]	$\boldsymbol{d_H}$	1.53	1.31	1.67	1.83	**1.56**	2.13	1.20	1.88	1.86	1.50	1.56	1.34	1.30	1.88	1.98	1.82
		$\boldsymbol{d_m}$	0.03	0.03	0.05	0.035	**0.03**	0.01	0.05	0.02	0.06	0.03	0.07	0.10	0.05	0.03	0.02	0.03
		ζ	0.14	0.12	0.23	0.16	**0.23**	0.09	0.16	0.14	0.22	0.18	0.22	0.23	0.14	0.16	0.16	0.15
$\sigma = 0.4$	Proposed	$\boldsymbol{d_H}$	**1.13**	**1.48**	**1.34**	**1.53**	2.41	**1.83**	**1.37**	**1.58**	**1.29**	**1.19**	**1.61**	**1.33**	**0.98**	**1.46**	**1.35**	**1.02**
		$\boldsymbol{d_m}$	**0.02**	**0.05**	**0.07**	**0.05**	0.07	**0.01**	**0.05**	**0.02**	**0.05**	**0.03**	**0.06**	**0.07**	**0.05**	**0.01**	**0.03**	**0.01**
		ζ	**0.12**	**0.15**	**0.22**	**0.19**	0.38	**0.11**	**0.14**	**0.13**	**0.23**	**0.18**	**0.21**	**0.20**	**0.17**	**0.11**	**0.16**	**0.13**
	Geometry Only [112]	$\boldsymbol{d_H}$	1.24	1.55	1.89	1.56	2.04	1.94	1.75	1.59	1.92	1.84	1.79	1.50	0.99	1.48	1.73	1.39
		$\boldsymbol{d_m}$	0.03	0.06	0.10	0.06	0.08	0.02	0.07	0.03	0.08	0.04	0.08	0.11	0.06	0.02	0.04	0.03
		ζ	0.15	0.18	0.26	0.23	0.31	0.15	0.16	0.15	0.27	0.19	0.29	0.24	0.17	0.13	0.22	0.18
	MSGW [17]	$\boldsymbol{d_H}$	1.66	1.77	1.94	2.01	**1.97**	2.39	1.46	1.84	2.17	1.76	1.83	1.51	1.47	2.07	2.03	1.97
		$\boldsymbol{d_m}$	0.05	0.06	0.09	0.06	**0.05**	0.01	0.08	0.05	0.09	0.05	0.11	0.13	0.07	0.06	0.04	0.04
		ζ	0.19	0.21	0.27	0.21	**0.29**	0.13	0.19	0.20	0.30	0.21	0.32	0.28	0.19	0.22	0.19	0.26

Tikhonov regularization and using TV. The comparative quantitative results of geometry denoising in a combined geometry and color denoising algorithm are shown in Tab. 5.9. To better understand the results, we mapped the C2M distance on the reference point cloud. The distances are represented by the color scale; blue, green,



(a) (b)

Figure 5.22: Asterix_model (noise level $\mu = 0$ and $\sigma = 0.5$) (a): C2M metric of denoised point cloud by proposed algorithm using Tikhonov regularization (b): C2M metric of denoised point cloud using TV.



(a) (b)

Figure 5.23: (a): Average gain in MSE (dB) for the color denoising in combined geometry and color denoising algorithm (b): Average gain in PSNR (dB) for the color denoising in combined geometry and color denoising algorithm.

yellow, and red display the range of distances from minimum to maximum. Fig. 5.22 shows the results of C2M distance for the *Asterix_model*. The performance evaluation of the proposed algorithm for color denoising has been done by computing the MSE and PSNR. The results of MSE and PSNR comparison with different noise levels are shown in Tab. 5.10 and Tab. 5.11, respectively. Fig. 5.23 shows the

average gain in MSE and PSNR, which further verifies the proposed algorithm's better performance. The results show that the proposed technique using Tikhonov regularization performs significantly better than the TV for noise levels $\sigma = 0.3, 0.4,$ and 0.5 except for the *Duck*. The results also indicate that the proposed algorithm is robust to the increase in noise intensity.

Table 5.9: C2M metric comparison between the proposed combined geometry and color denoising algorithm using Tikhonov regularization and TV.

Noise level in geometry	Methods	Parameters	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\boldsymbol{\sigma} = 0.3$	Proposed algorithm using Tikhonov regularization	$\boldsymbol{d_H}$	**0.68**	**0.61**	1.17	0.91	2.71	**1.50**	**0.52**	**0.94**	**0.82**	**1.02**	**0.82**	**0.69**	**0.45**	**0.93**	**1.27**	**0.89**
		$\boldsymbol{d_m}$	$<$ **0.01**	$<$ **0.01**	0.05	0.03	0.37	$\approx$ **0.00**	$<$ **0.01**	$\approx$ **0.00**	**0.01**	**0.01**	**0.02**	**0.01**	**0.03**	$\approx$ **0.00**	**0.01**	$<$ **0.01**
		ζ	**0.04**	**0.06**	0.19	0.15	0.68	**0.03**	**0.07**	**0.02**	**0.10**	**0.10**	**0.11**	**0.10**	**0.11**	**0.03**	**0.11**	**0.04**
	TV regularization	$\boldsymbol{d_H}$	0.96	0.61	**0.82**	**0.91**	**1.56**	1.50	0.75	0.94	1.17	1.02	0.83	0.97	0.64	0.93	1.27	0.90
		$\boldsymbol{d_m}$	0.01	0.01	**0.04**	**0.02**	**0.07**	$<$ 0.01	0.02	$<$ 0.01	0.03	0.02	0.05	0.03	0.03	$<$ 0.01	0.01	0.01
		ζ	0.08	0.07	**0.19**	**0.14**	**0.31**	0.05	0.11	0.06	0.15	0.14	0.16	0.15	0.12	0.08	0.13	0.09
$\boldsymbol{\sigma} = 0.4$	Proposed algorithm using Tikhonov regularization	$\boldsymbol{d_H}$	**0.68**	**0.74**	**1.16**	**0.91**	2.71	**1.50**	**0.74**	**0.98**	**0.83**	**1.02**	**1.02**	**0.98**	**0.63**	**0.93**	**1.27**	**0.89**
		$\boldsymbol{d_m}$	$<$ **0.01**	**0.01**	**0.06**	**0.03**	0.37	$<$ **0.01**	**0.02**	$<$ **0.01**	**0.02**	**0.02**	**0.04**	**0.03**	**0.03**	$<$ **0.01**	**0.02**	$<$ **0.01**
		ζ	**0.05**	**0.08**	**0.21**	**0.17**	0.68	**0.05**	**0.10**	**0.04**	**0.14**	**0.13**	**0.15**	**0.13**	**0.11**	**0.05**	**0.14**	**0.06**
	TV regularization	$\boldsymbol{d_H}$	0.96	0.87	1.43	1.29	**1.61**	1.54	1.06	1.08	1.18	1.02	1.18	0.98	1.01	1.32	1.80	0.90
		$\boldsymbol{d_m}$	0.02	0.02	0.06	0.04	**0.07**	0.01	0.05	0.01	0.05	0.04	0.08	0.06	0.05	0.02	0.03	0.02
		ζ	0.12	0.10	0.22	0.19	**0.33**	0.09	0.16	0.10	0.20	0.19	0.21	0.20	0.14	0.12	0.19	0.13
$\boldsymbol{\sigma} = 0.5$	Proposed algorithm using Tikhonov regularization	$\boldsymbol{d_H}$	**0.96**	**0.96**	**1.43**	**0.94**	3.50	**1.53**	**1.05**	**0.99**	**1.17**	**1.02**	**1.02**	**0.98**	**0.89**	**0.93**	**1.29**	**0.91**
		$\boldsymbol{d_m}$	$<$ **0.01**	**0.03**	**0.09**	**0.06**	0.61	$<$ **0.01**	**0.04**	$<$ **0.01**	**0.04**	**0.03**	**0.06**	**0.04**	**0.04**	$<$ **0.01**	**0.03**	$<$ **0.01**
		ζ	**0.08**	**0.12**	**0.26**	**0.23**	0.83	**0.07**	**0.13**	**0.05**	**0.18**	**0.17**	**0.18**	**0.16**	**0.12**	**0.06**	**0.19**	**0.06**
	TV regularization	$\boldsymbol{d_H}$	1.37	1.07	1.65	1.30	**1.69**	1.50	1.30	1.33	1.18	1.45	1.45	1.56	1.11	1.32	1.81	1.27
		$\boldsymbol{d_m}$	0.03	0.04	0.10	0.07	**0.08**	0.01	0.08	0.02	0.09	0.06	0.11	0.09	0.06	0.03	0.05	0.03
		ζ	0.14	0.14	0.26	0.26	**0.34**	0.13	0.20	0.14	0.26	0.25	0.24	0.24	0.16	0.16	0.25	0.16

Table 5.10: MSE comparison of combined geometry and color denoising algorithm for *Greyc* dataset with three noise levels in color attribute.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 20$	Noisy	396.81	361.15	374.29	393.14	339.42	397.91	364.35	385.17	368.52	386.45	321.67	376.17	309.85	333.85	378.42	396.90
	Tikhonov	**107.03**	**164.03**	**172.73**	**116.33**	564.41	**97.96**	**125.90**	**173.33**	**127.41**	**105.68**	**130.65**	**121.52**	**269.01**	**119.79**	**219.17**	**119.52**
	TV	360.47	333.08	344.70	358.37	**320.00**	361.15	334.29	354.34	338.20	352.53	294.67	344.65	285.11	303.66	350.36	361.43
$\sigma = 30$	Noisy	876.66	782.13	810.77	863.43	728.82	872.76	793.86	841.43	801.55	848.09	689.96	813.16	652.34	716.01	814.97	882.11
	Tikhonov	**199.35**	**269.90**	**275.74**	**202.90**	**647.76**	**174.94**	**235.39**	**267.61**	**242.29**	**200.79**	**252.53**	**251.17**	**385.45**	**216.69**	**316.62**	**215.98**
	TV	820.72	737.84	764.08	809.03	693.63	815.83	748.37	792.41	755.33	795.76	649.93	766.46	613.50	670.00	769.39	826.59
$\sigma = 40$	Noisy	1505.90	1337.20	1384.60	1459.90	1243.80	1476.30	1353.50	1467.20	1357.00	1458.90	1167.20	1368.70	1088.30	1209.70	1380.00	1523.40
	Tikhonov	**241.45**	**386.44**	**395.22**	**246.54**	**884.03**	**202.77**	**321.99**	**375.98**	**340.35**	**254.89**	**372.51**	**350.60**	**658.49**	**308.36**	**445.18**	**275.12**
	TV	1390.60	1244.00	1286.80	1347.20	1162.90	1358.90	1258.70	1362.30	1262.80	1350.50	1087.20	1275.20	1009.70	1116.70	1283.10	1408.20

Table 5.11: PSNR comparison of combined geometry and color denoising algorithm for *Greyc* dataset with three noise levels in color attribute.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 20$	Noisy	22.15	22.55	22.40	22.19	22.82	22.13	22.52	22.27	22.47	22.26	23.06	22.38	23.22	22.90	22.35	22.14
	Tikhonov	**27.84**	**25.99**	**25.76**	**27.47**	20.61	**28.22**	**27.13**	**25.74**	**27.08**	**27.89**	**26.97**	**27.28**	**23.83**	**27.35**	**24.72**	**27.36**
	TV	22.56	22.91	22.76	22.59	**23.08**	22.55	22.89	22.64	22.84	22.66	23.44	22.76	23.58	23.31	22.69	22.55
$\sigma = 30$	Noisy	18.70	19.20	19.04	18.77	19.50	18.72	19.13	18.88	19.09	18.85	19.74	19.03	19.99	19.58	19.02	18.68
	Tikhonov	**25.13**	**23.82**	**23.73**	**25.06**	**20.02**	**25.70**	**24.41**	**23.86**	**24.29**	**25.10**	**24.11**	**24.13**	**22.27**	**24.77**	**23.13**	**24.79**
	TV	18.99	19.45	19.30	19.05	19.72	19.01	19.39	19.14	19.35	19.12	20.00	19.29	20.25	19.87	19.27	18.96
$\sigma = 40$	Noisy	16.35	16.87	16.72	16.49	17.18	16.44	16.82	16.47	16.81	16.49	17.46	16.77	17.76	17.30	16.73	16.30
	Tikhonov	**24.30**	**22.26**	**22.16**	**24.21**	**18.67**	**25.06**	**23.05**	**22.38**	**22.37**	**22.81**	**24.07**	**22.42**	**19.95**	**23.24**	**21.65**	**23.74**
	TV	16.70	17.18	17.046	16.84	17.48	16.80	17.13	16.79	17.12	16.83	17.77	17.08	18.09	17.65	17.05	16.64

Table 5.12: The computational cost of the proposed algorithm and IBR [112] for different types of point clouds.

Uniform Noise			0.3		0.4		0.5	
Methods		No of Points	Proposed	IBR	Proposed	IBR	Proposed	IBR
Synthetic Point clouds	4arms_monstre	83063	1.56	1.32	1.85	1.43	1.91	1.85
	Asterix	45498	0.82	0.68	1.01	0.76	0.94	0.82
	Cable car	267828	7.54	4.76	5.78	5.10	5.81	4.88
	Dragon	101369	2.32	2.18	1.85	1.66	1.93	1.51
	Duck	19247	0.41	0.33	0.39	0.29	0.36	0.31
	Green dinasour	87999	1.71	1.43	2.37	1.35	1.71	1.34
	Green monster	204250	5.59	4.11	5.43	4.69	4.71	3.72
	Horse	137831	3.75	3.32	2.86	2.37	2.86	2.38
	Jaguar	130423	3.19	2.22	2.49	2.02	2.63	2.19
	Long dinasour	114005	2.43	1.89	2.20	1.79	2.14	1.85
	Mario	199281	4.72	3.93	4.32	3.49	5.41	3.76
	Mario car	216375	4.63	3.86	4.78	3.93	4.88	4.40
	Pokemon Ball	25427	0.49	0.40	0.54	0.57	0.48	0.39
	Rabbit	157534	3.27	2.74	3.42	3.10	3.44	2.81
	Red Horse	156671	3.17	2.57	3.20	2.70	3.83	3.33
	Statue	89256	1.68	1.36	1.74	1.49	1.94	1.50
	Average		2.96	**2.32**	2.76	**2.30**	2.81	**2.31**

			Proposed	IBR
Real-world Point clouds	Arco_valentino	1530552	108.23	116.22
	Palazzo_carignano	4203962	173.93	190.37
LiDAR	MastinLake9_001	10001543	1048.10	1455.09

Chapter 6

Joint geometry and color point cloud denoising based on graph wavelets

In this chapter, we describe a novel non-iterative set-up for the denoising of point cloud based on spectral graph wavelet transform (SGW) that jointly exploits geometry and color to perform denoising of geometry and color attributes in graph spectral domain. The design framework is based on the construction of a joint geometry and color graph that compacts smooth graph signals' energy in the low-frequency bands. We then apply soft-thresholding to remove the noise from the spectral graph wavelet coefficients.

6.1 Spectral Graph Wavelet Preliminaries

In this section, we set up a mathematical notation and the definition of SGW [38]. The Graph Fourier Transform (GFT) $\hat{g}$ for a function $g(\mathcal{G})$ is defined as $\hat{g}(l) = \langle \chi_l, g \rangle = \sum_{i=1}^{m-1} g(i)\chi_l(i)$, where the eigenvectors of the graph Laplacian $\mathbf{L}$ are represented by χ_l and eigenvalues are denoted as λ_l for $l = 0, \ldots, m-1$. SGW [38] establishes a scaling operator in the Graph Fourier domain based on λ_l. Particularly, SGW are determined using a kernel f, the wavelet operator $T_f = f(\mathbf{L})$ acts on a given function g by modulating each Fourier mode as:

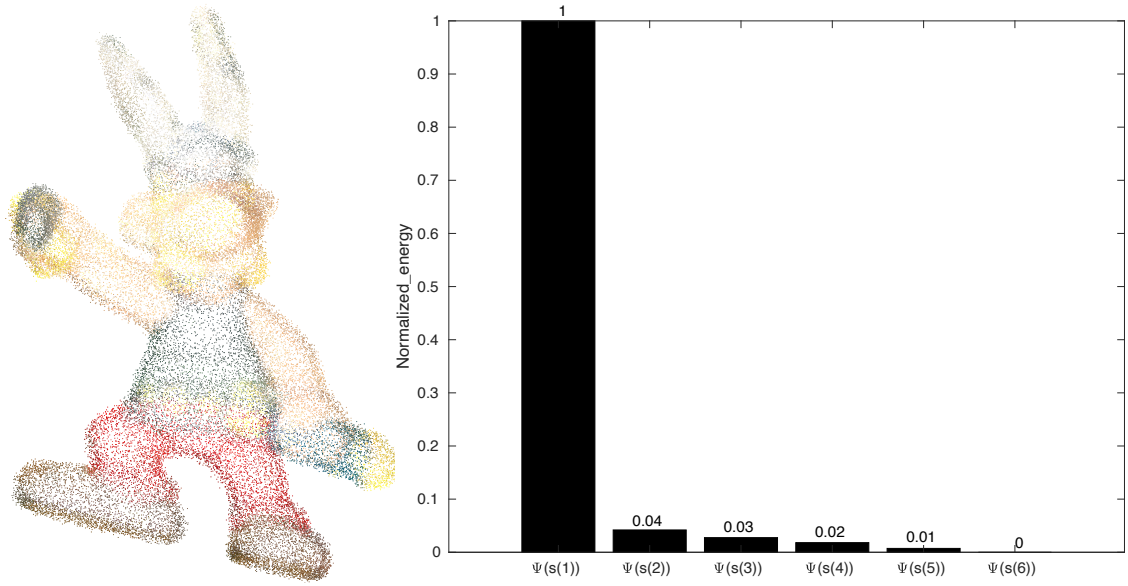
$$\widehat{T_f\, g}(l) = f(\lambda_l)\hat{g}(l)\,. \tag{6.1}$$

The inverse transform is defined as:

$$(T_f\; g)(n) = \sum_{l=0}^{m-1} f(\lambda_l)\hat{g}(l)\chi_l(n)\,. \tag{6.2}$$

At scale s, the wavelet operator is defined as $T_f^s = f(s\mathbf{L})$. SGW is then calculated by localizing the operators T_f^s by enforcing them to the impulse δ on a single vertex: $\psi_{s,n} = T_f^s \delta_n$. For a given graph signal g, the wavelet coefficients are determined by taking the inner product with these wavelets as $\Psi_g(s,n) = \langle \psi_{s,n}\, , g \rangle$. $\Psi_g(s,n)$ can be estimated from the wavelet operators T_f^s using the orthonormality of χ as:

$$\Psi_g(s,n) = (T_f^s g)(n) = \sum_{l=0}^{m-1} f(s\lambda_l)\hat{g}(l)\chi_l(n)\,. \tag{6.3}$$



(a) (b)

Figure 6.1: (a) Noisy Asterix model with $\mu = 0, \sigma = 0.2$ (b) Normalized energy of $\Psi_{X_i}(s(i), n)$, where $1 \leq i \leq I$ and $n = 1 \ldots . N$.

The naive technique for calculating SGW from Eq. 6.3 needs explicit computation of the entire χ, which scales inadequately for large graphs. Another method for direct calculation of SGW transform is to diagonalize $\mathbf{L}$, which is possible only for the small graphs having less than a thousand vertices [38]. For the real-world and synthetic point clouds consisting of a large number of points, the SGW can be computed with a fast algorithm via low order polynomials, which estimate the scaled generating kernels. The wavelet coefficients can be determined at every single scale by applying a polynomial of $\mathbf{L}$ to the underlying data. This leads to lower computational costs when the graph is sparse. To define the transform in graph spectral domain, direct calculation of Laplacian operator via diagonalization is infeasible for problems with size exceeding a few thousand vertices.

Table 6.1: Parameter values used for the proposed SGW using soft-thresholding, MSGW [17], IBR [112] and RPSM [16].

Parameter	Proposed	MSGW [17]	IBR [112]	RPSM [16]
k	30	60	20	10
θ_X	0.8	–	–	–
θ_C	0.2	–	–	–
γ	–	–	0.50 ($\sigma = 0.2$, 0.3) 0.70 ($\sigma = 0.4$)	–
τ	0.2 ($\sigma = 0.2$) 0.3 ($\sigma = 0.3$) 0.4 ($\sigma = 0.4$)	0.2 ($\sigma = 0.2$) 0.3 ($\sigma = 0.3$) 0.4 ($\sigma = 0.4$)	–	–

Hence, the wavelet and scaling coefficients of a given input $[m \times D]$ graph signal are efficiently calculated by employing Chebyshev polynomial approximation [97, 38] to avoid the need for explicit diagonalization of the graph Laplacian and then mapped these coefficients to $[m * (I - 1) \times D]$, where I denotes the number of wavelet decomposition levels.

In order to compare the proposed geometry denoising algorithm with MSGW [17] and IBR [112], outlier removal is needed. Outliers have distinct features i.e., that they have a sparse neighborhood; therefore, outliers detection and removal are density-based [112, 31]. We follow the ROR method, in which a sphere with a radius r is formed having each point p_i as its centre. The sphere contains u_i number of points. We then compute $\bar{u} = \frac{\sum_{i=1}^{N} u_i}{N}$, where N is the total number of points. A point p_j is considered as an outlier if $u_j < \bar{u}$.

6.2 Geometry denoising

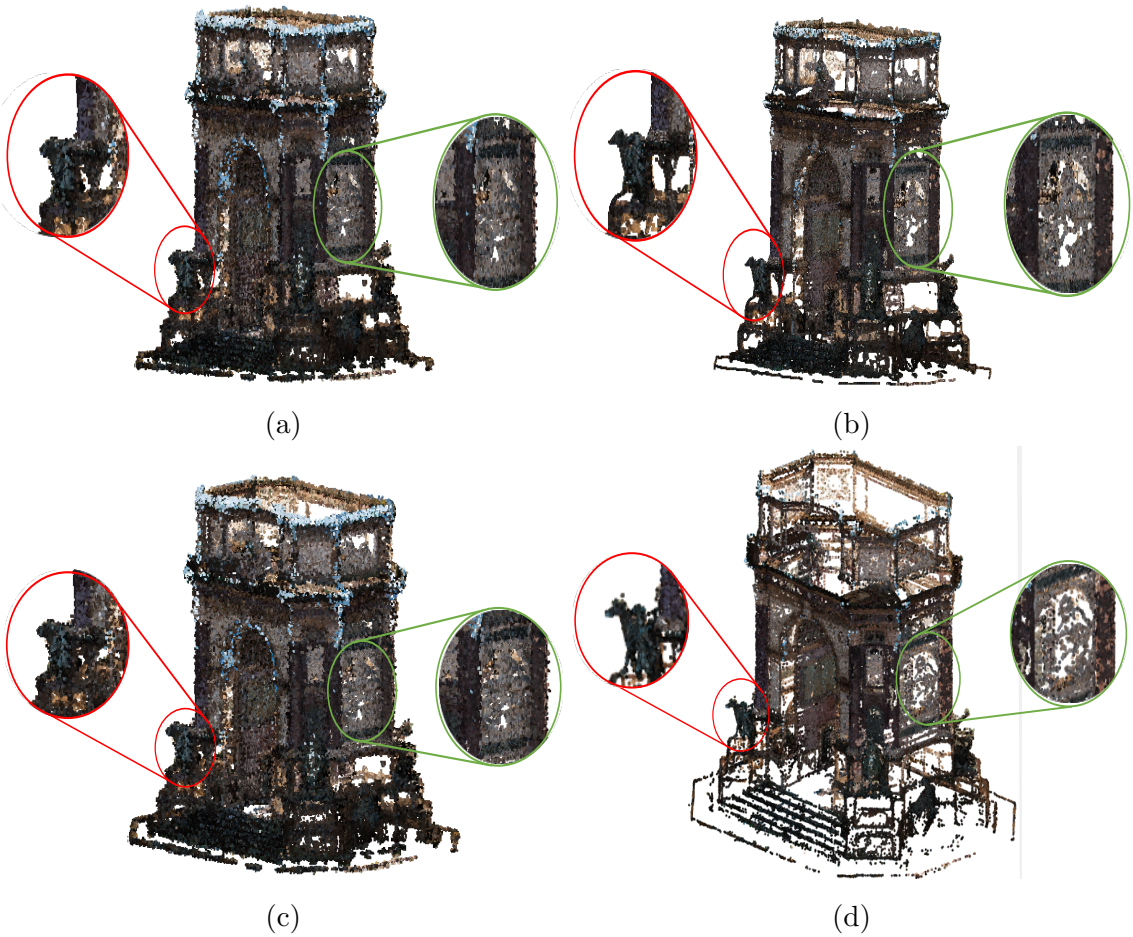
The given noisy coordinates of each point p_i can be described as $\boldsymbol{p}_i = [\boldsymbol{X}_i + \boldsymbol{n}_i, \boldsymbol{C}_i]$, $\boldsymbol{X}_i$ being the unknown true position of a point $\boldsymbol{p}_i$ and $\boldsymbol{n}_i$ is the geometry noise, with $\boldsymbol{X}_i, \boldsymbol{n}_i \in \mathbb{R}^3$. The objective is to estimate x_i for each point in a point cloud. This can be performed using the proposed denoising technique based on SGW. After the construction of an undirected graph $\mathcal{G}$ using Eq. 4.1 and defining the graph Laplacian $\mathbf{L}$ from $\mathbf{W}$, we take the SGW transform using low order polynomials to establish tight vertex localization of SGW coefficients.

For the respective noisy signal $\boldsymbol{X}$, SGW coefficients are computed for each SGW

band, i.e., $\Psi_{X_i}(s(i))$ for $1 \leq i \leq I$, preserving all the wavelet and scaling coefficients, which corresponds to a low-frequency wavelet band s. Denoising of $\Psi_{X_i}(s(i), n)$ is then performed by soft-thresholding based on the property that energy of the signal is concentrated in the low-frequency spectral wavelet bands. The distribution of the energy of signal for a noisy *Asterix* model is shown in Fig. 6.1-b. Finally, the denoised point cloud $\mathcal{Q}$ is obtained by taking an inverse spectral wavelet transform of the denoised SGW coefficients $\Psi^*_{X_i}(s(i))$.

6.2.1 Experimental Setup

We set $I = 6$ wavelet decomposition levels, and preserve the wavelet coefficients of $s(1)$ scale and then preserve the wavelet coefficients which are above the threshold τ for the corresponding noise level in the $s(i)$ for $2 \leq i \leq I$.



(a) (b) (c) (d)

Figure 6.2: Arco_Valentino model. (a) Noisy input, denoised results by (b) Proposed algorithm based on SGW using soft-thresholding, (c) MSGW [17] applied to outlier-free input, and (d) IBR performed after the outlier removal step [112].

In order to denoise the point cloud locally by approximating the spectral wavelet coefficients, the Chebyshev polynomials approximation order is set to $k/2$ for a k-NN graph. All the parameters used in this algorithm are given in Tab. 6.1.

6.2.2 Denoising of real-world point clouds

We show a visual comparison for the point cloud denoised by the proposed algorithm with a graph constructed from only geometry and regularization as in [112] and the algorithm described in [17]. The experiment is performed on real-world natural point clouds, for which we do not have a noiseless reference; hence the results are only qualitative.

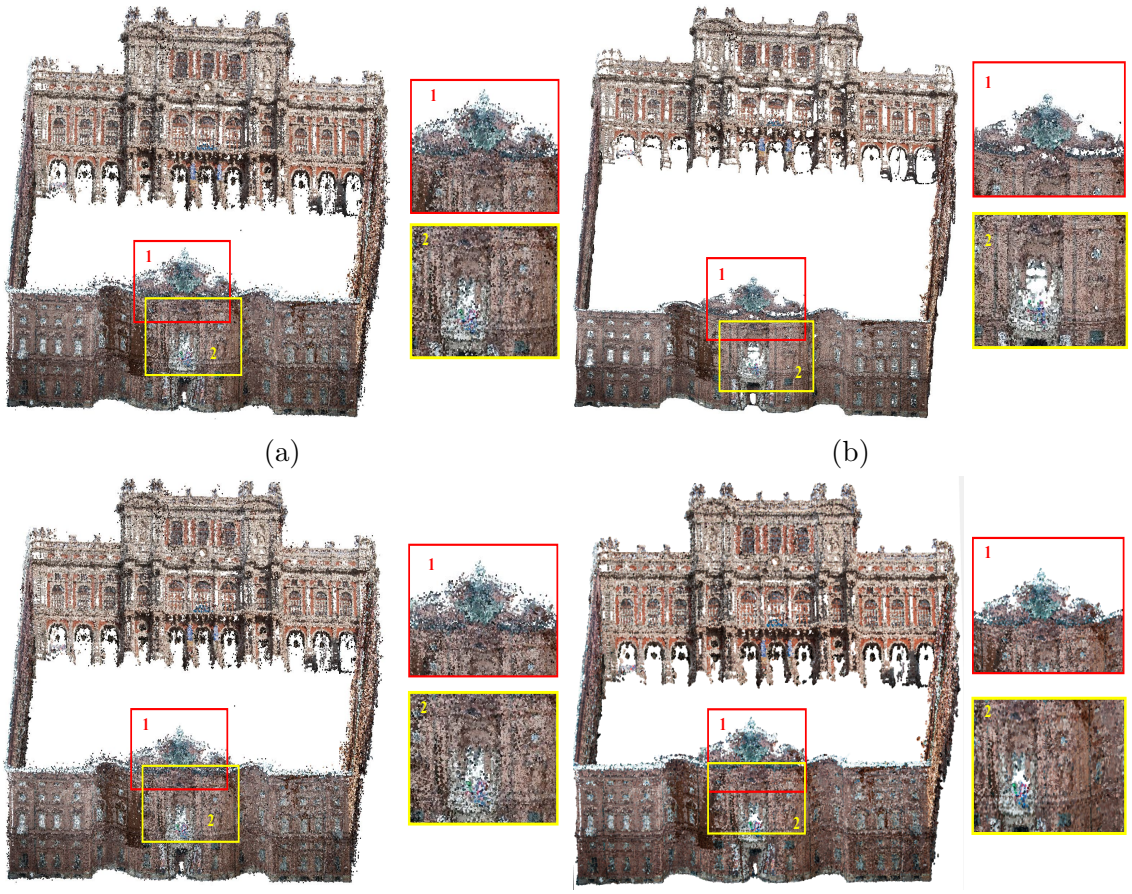


Figure 6.3: Palazzo_Carignano model. (a) Noisy input, denoised results by (b) Proposed algorithm based on SGW using soft-thresholding, (c) MSGW [17] applied to outlier-free input, and (d) IBR performed after the outlier removal step [112].

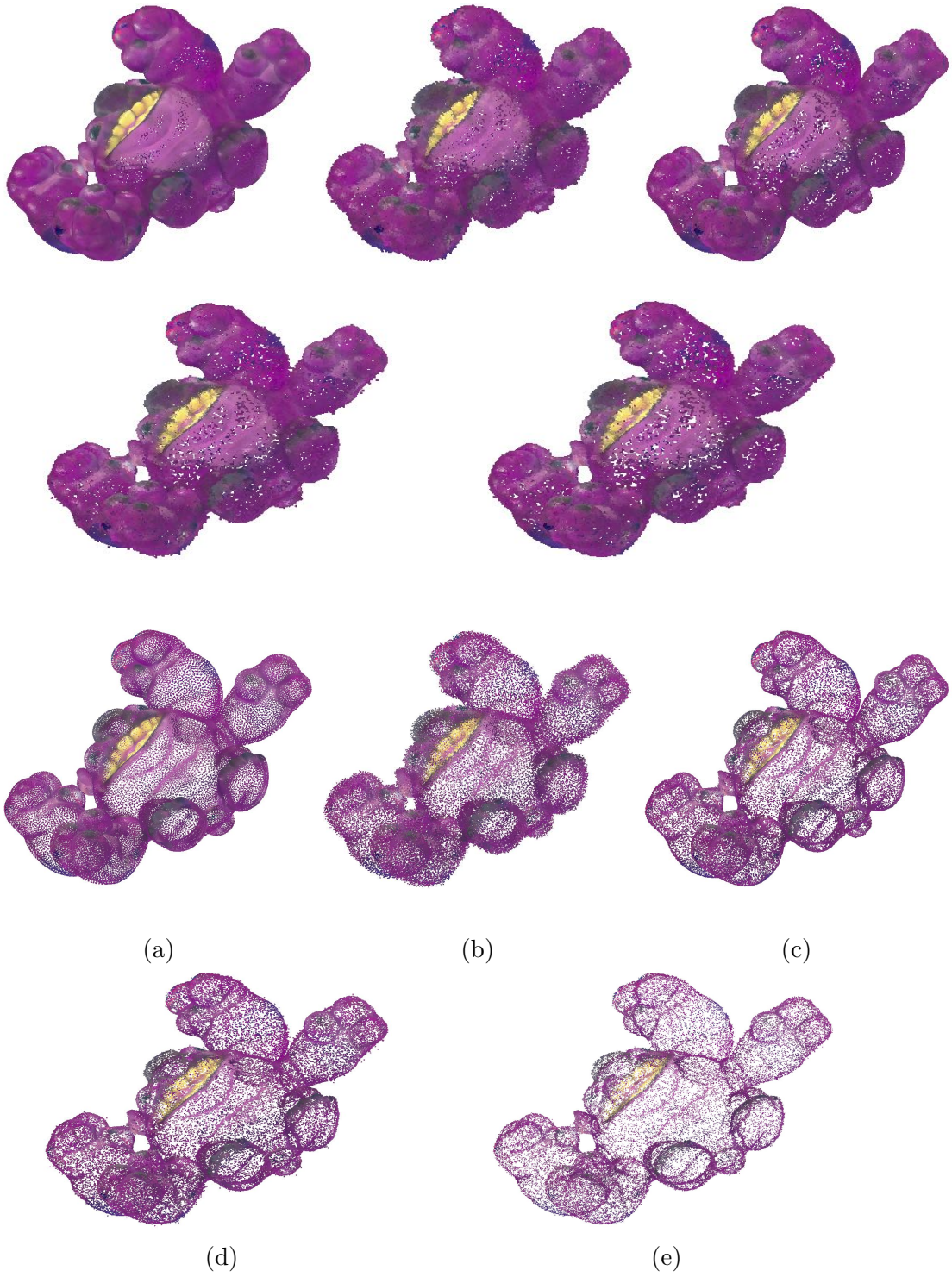
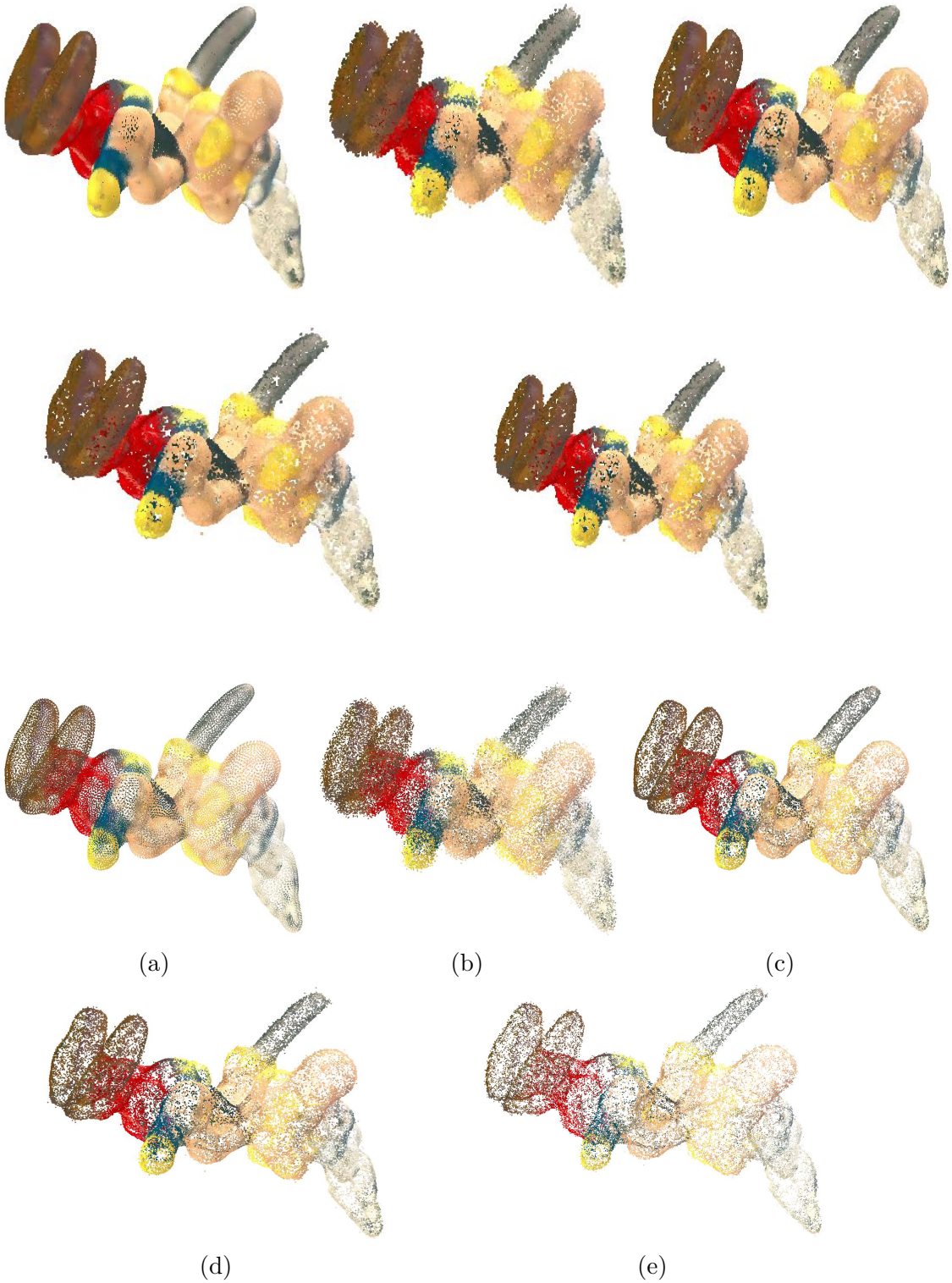


Figure 6.4: 4arms_monster models with color: (a) ground-truth, (b) noisy input ($\mu = 0$ and $\sigma = 0.3$), denoised results by (c) proposed algorithm based on SGW using soft-thresholding at $\tau = 0.3$, (d) MSGW [17], and (e) IBR [112].

Fig. 6.2-a and Fig. 6.3-a show the point clouds with real noise; it can be seen that the points with the same color are typically spread in a small neighborhood during the acquisition process, which may blur the details. Fig. 6.2-b and Fig. 6.3-b reports the resulting denoised point cloud using the proposed algorithm. Here the noisy points are moved close to their original position by exploiting the correlation of their geometry and color, hence preserving the details hidden in the noisy point cloud. Fig. 6.2-c and Fig. 6.3-c show the denoised point cloud using [17]; it can be seen that the algorithm is less effective at denoising the geometry even after applying the outlier removal step prior to perform denoising. Fig. 6.2-d and Fig. 6.3-d show the denoised output using IBR [112] applied to the outlier-free input; it can be seen in the same region that, using no color information, the noisy points are not moved to their correct location, enlarging gaps and generally providing a noisier result near object boundaries. This algorithm is iterative, and for real-world point cloud, it is very computationally expensive; moreover, the outlier removal step is required for both MSGW and IBR, which makes these algorithms more computational complex.

6.2.3 Denoising of synthetic point clouds

The proposed denoising approach has also been applied to noise-free point clouds from the *Greyc* dataset [85], corrupted with zero-mean Gaussian synthetic geometry noise applied to the points with $\sigma = 0.2$, 0.3 and 0.4. In Fig. 6.4 and Fig. 6.5, the results are shown in two rows for *4arms_monstre* and *Asterix* models respectively. The top row of each point cloud model highlights the adverse effect, i.e., creating holes in the resulting denoised point cloud using MSGW and IBR with respect to the proposed algorithm. In contrast, the effect of geometry denoising is highlighted in the second row of each point cloud model. To visualize the first row of each Fig. 6.4 and Fig. 6.5, the axes of each point cloud is scaled by a factor of 1/4, and for the visualization of the second row, the axes of an individual point cloud are scaled by a factor of 4. Fig. 6.4-a and Fig. 6.5-a show the noise-free point cloud models. Fig. 6.4-b and Fig. 6.5-b show the noisy point clouds with $\sigma = 0.3$. The denoised point clouds obtained by the proposed algorithm are shown in Fig. 6.4-c and Fig. 6.5-c; it can be seen that the geometry noise has been regularized, and the noisy points are moved close to their original positions with less adverse effect of creating holes. Fig. 6.4-d and Fig. 6.5-d shows the denoised output of MSGW [17]; it can be seen that the geometry is not denoised properly and also causes the opening of holes in the resulting output. The resulting denoised point clouds using the IBR algorithm [112] are shown in Fig. 6.4-e and Fig.6.5-e. It can be seen that the geometry is not quite as much regularized with respect to the proposed algorithm, while still better than MSGW [17]; moreover, the resulting point cloud of the IBR algorithm [112] has big holes; the reason is the value of γ parameter, higher γ is required for denoising well, but it causes severe artifacts.



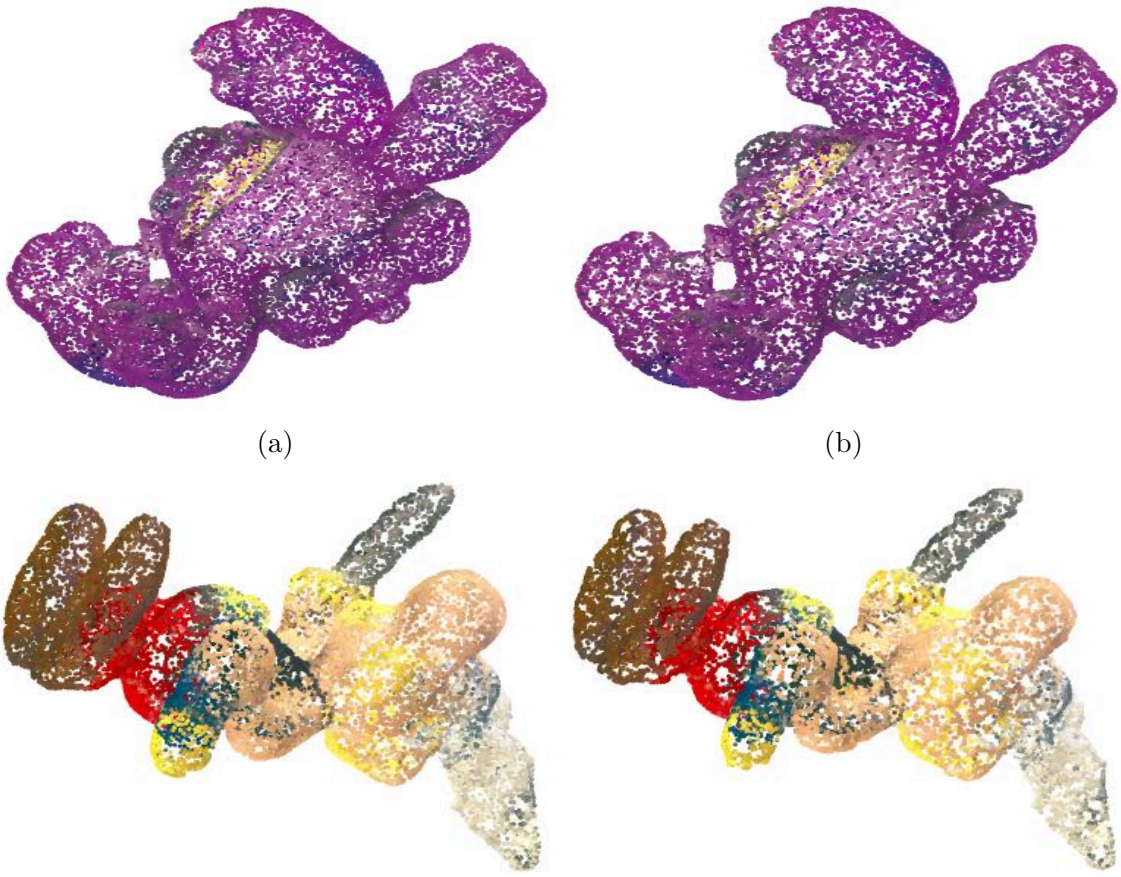
(a) (b) (c)

(d) (e)

Figure 6.5: Asterix models with color: (a) ground-truth, (b) noisy input ($\mu = 0$ and $\sigma = 0.3$), denoised results by (c) proposed algorithm based on SGW using soft-thresholding at $\tau = 0.3$, (d) MSGW [17], and (e) IBR [112].

Overall, it can be seen from the qualitative results of both real-world and synthetic point clouds that the point clouds denoised by the proposed algorithm have better quality and fewer artifacts.

The comparative analysis has also been performed concerning RPSM [16], and the results are shown in Fig. 6.6. In this particular case, we performed the experiment on sub-sampled point clouds from the same dataset due to the large memory requirement of RPSM. The sub-sampling performed here is on a spatial basis, where the average minimum distance between the two points in each point cloud is set to 0.80. The number of points in each point cloud model of Greyc dataset [85] is around 20,000 on average.



(a) (b)

(c) (d)

Figure 6.6: 4arms_monstre model with noise of ($\mu = 0$ and $\sigma = 0.3$): (a) Denoised results by proposed algorithm based on SGW using soft-thresholding at $\tau = 0.3$, (b) RPSM [16]. Asterix model with noise of ($\mu = 0$ and $\sigma = 0.3$): (c) denoised results by proposed algorithm based on SGW using soft-thresholding at $\tau = 0.3$, and (d) RSPM [16].

The proposed algorithm and RPSM [16] are applied to the sub-sampled noisy inputs shown in Fig. 6.5-b; the ground-truth point clouds are shown in Fig. 6.5-a. The denoised outputs of the proposed algorithm are shown in Fig. 6.6-a, and Fig. 6.6-c, and the resulting denoised outputs of RPSM [16] are shown in Fig. 6.6-b and Fig. 6.6-d; it can be seen that RPSM [16] over-regularized the sub-sampled point clouds which tends to generate the holes in the resulting denoised point cloud.

Table 6.2: MSE and MCD comparison of various algorithms for *Greyc* dataset.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
MSE																		
$\sigma = 0.2$	Proposed	**0.35**	**0.27**	**0.20**	**0.20**	0.73	**0.36**	**0.17**	**0.41**	**0.13**	**0.16**	**0.13**	**0.13**	0.38	**0.37**	**0.21**	**0.36**	**0.29**
	MSGW[17]	0.36	**0.27**	0.23	0.23	0.85	0.37	0.18	0.43	0.14	0.17	0.14	0.14	**0.36**	0.37	0.23	0.37	0.30
	IBR [112]	0.38	0.30	0.29	0.31	**0.63**	0.41	0.23	0.45	0.20	0.24	0.19	0.19	0.41	0.40	0.30	0.37	0.33
$\sigma = 0.3$	Proposed	**0.36**	**0.27**	**0.22**	**0.21**	0.76	**0.37**	**0.18**	**0.42**	**0.14**	**0.17**	**0.14**	**0.15**	**0.40**	**0.38**	**0.22**	**0.36**	**0.30**
	MSGW[17]	0.38	0.28	0.24	0.24	0.92	0.39	0.18	0.44	0.15	0.18	0.15	0.15	**0.40**	0.40	0.25	0.38	0.32
	IBR [112]	0.39	0.30	0.30	0.32	**0.71**	0.42	0.24	0.46	0.20	0.25	0.20	0.19	0.44	0.42	0.32	0.38	0.35
$\sigma = 0.4$	Proposed	**0.37**	**0.27**	**0.26**	**0.28**	**1.15**	**0.40**	**0.19**	**0.44**	**0.15**	**0.19**	**0.16**	**0.16**	**0.41**	**0.41**	**0.28**	**0.38**	**0.34**
	MSGW[17]	0.38	0.28	0.30	0.32	1.35	0.43	0.21	0.45	0.16	0.21	0.17	0.17	0.41	0.41	0.32	0.39	0.37
	IBR [112]	0.39	0.31	0.31	0.33	1.16	0.43	0.24	0.45	0.21	0.26	0.21	0.20	0.46	0.42	0.33	0.38	0.38
MCD																		
$\sigma = 0.2$	Proposed	**0.51**	**0.39**	**0.30**	**0.30**	1.07	**0.52**	**0.25**	**0.60**	**0.19**	**0.24**	**0.19**	**0.19**	0.56	**0.53**	**0.31**	**0.52**	**0.42**
	MSGW[17]	0.53	0.39	0.33	0.33	1.24	0.53	0.26	0.61	0.20	0.25	0.21	0.20	**0.52**	0.54	0.34	0.53	0.44
	IBR [112]	0.55	0.43	0.42	0.44	**0.94**	0.59	0.33	0.65	0.29	0.35	0.28	0.28	0.61	0.59	0.44	0.54	0.48
$\sigma = 0.3$	Proposed	**0.52**	**0.39**	**0.32**	**0.31**	1.13	**0.54**	**0.27**	**0.61**	**0.21**	**0.25**	**0.21**	**0.21**	0.59	**0.55**	**0.32**	**0.53**	**0.44**
	MSGW[17]	0.54	0.40	0.35	0.35	1.36	0.56	0.27	0.63	0.21	0.26	0.22	0.22	**0.58**	0.58	0.36	0.55	0.47
	IBR [112]	0.56	0.44	0.43	0.46	**1.06**	0.61	0.34	0.66	0.30	0.36	0.29	0.28	0.65	0.61	0.46	0.55	0.50
$\sigma = 0.4$	Proposed	**0.54**	**0.40**	**0.38**	**0.40**	**1.68**	**0.58**	**0.28**	**0.63**	**0.23**	**0.28**	**0.23**	**0.23**	**0.60**	**0.59**	**0.40**	**0.53**	**0.50**
	MSGW[17]	0.54	0.41	0.43	0.45	1.96	0.61	0.31	0.64	0.24	0.30	0.24	0.25	0.61	0.59	0.47	0.56	0.54
	IBR [112]	0.57	0.45	0.44	0.47	1.71	0.62	0.35	0.68	0.31	0.37	0.30	0.29	0.67	0.61	0.48	0.55	0.55

Table 6.3: MSE and MCD comparison between proposed algorithm and RPSM [16] on sub-sampled *Greyc* dataset for different noise levels.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
MSE																		
$\sigma = 0.2$	Proposed	**0.51**	**0.34**	**0.73**	**0.52**	**0.80**	**0.58**	**0.49**	**0.65**	**0.38**	**0.47**	**0.45**	**0.44**	**0.41**	**0.65**	**0.63**	**0.52**	**0.54**
	RPSM [16]	0.90	0.74	1.25	1.02	1.33	0.99	0.87	1.19	0.78	0.74	1.02	0.86	0.73	0.94	1.18	1.03	0.97
$\sigma = 0.3$	Proposed	**0.54**	**0.36**	**0.77**	**0.53**	**0.84**	**0.61**	**0.51**	**0.68**	**0.39**	**0.49**	**0.46**	**0.47**	**0.44**	**0.68**	**0.66**	**0.54**	**0.56**
	RPSM [16]	0.95	0.76	1.34	1.07	1.35	1.04	0.93	1.24	0.83	0.80	1.07	0.92	0.79	1.00	1.26	1.06	1.03
$\sigma = 0.4$	Proposed	**0.56**	**0.37**	**0.80**	**0.56**	**0.87**	**0.64**	**0.52**	**0.71**	**0.41**	**0.50**	**0.48**	**0.48**	**0.46**	**0.69**	**0.68**	**0.56**	**0.58**
	RPSM [16]	0.98	0.79	1.41	1.13	1.39	1.12	0.98	1.28	0.89	0.86	1.13	0.98	0.90	1.05	1.31	1.11	1.08
MCD																		
$\sigma = 0.2$	Proposed	**0.75**	**0.50**	**1.08**	**0.76**	**1.19**	**0.85**	**0.72**	**0.96**	**0.56**	**0.70**	**0.66**	**0.65**	**0.60**	**0.95**	**0.93**	**0.76**	**0.79**
	RPSM [16]	1.44	1.04	1.93	1.57	1.91	1.60	1.32	1.79	1.18	1.24	1.49	1.25	1.52	1.63	1.79	1.56	1.52
$\sigma = 0.3$	Proposed	**0.79**	**0.52**	**1.14**	**0.79**	**1.25**	**0.90**	**0.75**	**1.00**	**0.58**	**0.72**	**0.68**	**0.69**	**0.65**	**0.99**	**0.98**	**0.80**	**0.83**
	RPSM [16]	1.51	1.08	2.05	1.63	1.94	1.67	1.41	1.87	1.24	1.32	1.56	1.34	1.60	1.72	1.90	1.61	1.59
$\sigma = 0.4$	Proposed	**0.82**	**0.53**	**1.18**	**0.82**	**1.30**	**0.95**	**0.77**	**1.05**	**0.60**	**0.74**	**0.70**	**0.70**	**0.67**	**1.02**	**1.00**	**0.82**	**0.85**
	RPSM [16]	1.55	1.12	2.15	1.71	2.00	1.77	1.47	1.92	1.32	1.40	1.63	1.41	1.74	1.78	1.97	1.66	1.66

6.2.4 Objective evaluation on Greyc color mesh database

The proposed method has also been verified via quantitative evaluation on the complete *Greyc* dataset. Each point cloud has been corrupted with zero-mean Gaussian synthetic geometry noise, applied to each point with $\sigma = 0.2$, 0.3, and 0.4. The MSE and MCD comparisons between the proposed algorithm and the denoising approaches used in IBR [112] using total variation regularization and MSGW in [17] is shown in Tab. 6.2. The results show that the proposed denoising technique performed better than MSGW [17] and IBR [112] for all the models for noise level $\sigma = 0.2$, 0.3 and 0.4 except *duck* (where IBR performed better) and *Pokeman_ball* (where MSGW performed better).

The objective evaluation has also been performed for the comparison between the proposed algorithm and RPSM [16] on the sub-sampled point cloud models of *Greyc* dataset [85]. Tab. 6.3 shows the MSE and MCD comparison; it can be seen that the proposed algorithm outperformed the RPSM [16] in terms of both metrics for $\sigma = 0.2$, 0.3 and 0.4.

The average MSE and MCD (last column in Tab. 6.2 and Tab. 6.3) shows that the gain is more significant as the noise level increases, indicating that the proposed denoising methods are indeed better at removing geometry noise.

6.3 Denoising using adaptive data-driven thresholding

In the algorithm described in Sec. 6.2, prior knowledge of the standard deviation is needed to perform the soft-thresholding, which can work when noise is synthetically added. Still, in the real-world point clouds, the standard deviation is unknown. We used the same framework for solving the geometry and color denoising problem of a point cloud and used a data-driven adaptive soft-thresholding technique, in which the prior knowledge of the standard deviation is not required.

Table 6.4: Parameter values used for the proposed geometry denoising algorithm based on SGW using data-driven adaptive soft-thresholding, MSGW [17], IBR [112] and RPSM [16].

Parameter	Proposed	MSGW [17]	IBR [112]	RPSM [16]
k (Synthetic Point cloud)	30	60	20	10
k (Natural Point cloud)	10	20	20	–
θ_X	0.8	–	–	–
θ_C	0.2	–	–	–
γ (Synthetic Point cloud)	–	–	0.50 ($\sigma = 0.2$, 0.3) 0.70 ($\sigma = 0.4$)	–
γ (Natural Point cloud)	–	–	0.1	–
τ (Synthetic Point cloud)	–	0.2 ($\sigma = 0.2$) 0.3 ($\sigma = 0.3$) 0.4 ($\sigma = 0.4$)	–	–
τ (Natural Point cloud)	–	0.4	–	–

Table 6.5: Parameters values used for proposed color denoising based on SGW using data-driven adaptive soft-thresholding and other various algorithms.

Parameters	Proposed	Tikhonov & TV	GLR [20]	GTV [20]
k (Synthetic point cloud)	8	20	8	8
k (Natural point cloud)	8	25	–	–
θ_X	5	0.88	3	2
θ_C	50	3.5	100	100
γ (Synthetic point cloud)	–	0.05 ($\sigma = 10$) 0.07 ($\sigma = 15$) 0.10 ($\sigma = 20$) 0.13 ($\sigma = 25$) 0.19 ($\sigma = 30$) 0.26 ($\sigma = 40$)	0.3 ($\sigma = 10$, 15) 0.5 ($\sigma = 20$,30)	15 ($\sigma = 10$, 15) 25 ($\sigma = 20$,30)
γ (Natural point cloud)	–	1.5	–	–
ρ , t	–	–	–	5 ,0.1

6.3.1 Selection of data-driven adaptive soft-thresholding

Here, we focus on the estimation of parameters σ_X and σ_C for geometry and color denoising, respectively, which in turn yields a measure of individual thresholds $\tau(\sigma_X)$ and $\tau(\sigma_C)$ for geometry and color denoising application. These estimated thresholds are adaptive to various sub-band characteristics.

The geometry/color noise variance σ^2 is required to be estimated initially, which can be done using robust median estimator as in [24, 23]:

$$\hat{\sigma} = \frac{\text{Median}(|\Psi_{X_i}(s(i))|)}{0.6745}, \quad \text{for } 5 \leq i \leq I\,. \tag{6.4}$$

The variance of the transform coefficients $\sigma^2_{\Psi_{X_i}}$ can be found empirically:

$$\hat{\sigma}^2_{\Psi_{X_i}} = \frac{1}{n}\sum_{i=1}^{3n} \Psi^2_{X_i}\,, \tag{6.5}$$

where $n = m \times (I-1)$. The threshold is thus computed using:

$$\tau(\hat{\sigma}_X) = \frac{\hat{\sigma}^2}{\hat{\sigma}_X}\,, \tag{6.6}$$

where

$$\hat{\sigma}_X = \sqrt{\max(\hat{\sigma}^2_{\Psi_{X_i}} - \hat{\sigma}^2), 0}\,. \tag{6.7}$$

To compute $\tau(\sigma_C)$, we need to replace Ψ_{X_i} by Ψ_{C_i} in Eq. 6.4, Eq. 6.5, and Eq. 6.7, and replace σ_X by σ_C in Eq. 6.6.

6.3.2 Color denoising

For the color denoising problem, we assume that the geometry of the point cloud is noise-free or has been denoised using one of the existing denoising algorithms [112, 17, 16, 20].

We perform color denoising by exploiting the graph $\mathcal{G}$ from geometry and color information of the noisy point cloud. Each vertex of $\mathcal{G}$ is associated with the geometry X_i and color C_i information of the corresponding point p_i. The input noisy point cloud consists of both geometry and color. Each point can be expressed as $p_i = [X_i, C_i + n_i]$, X_i being the geometry, C_i being the unknown true color and W_i the color noise. The objective is to estimate C_i for each point of the point cloud. After the construction, the graph $\mathcal{G}$ using Eq. 4.1 with the appropriate choice of θ_X and θ_C for the color denoising, we compute the SGW coefficients for each SGW band, i.e., $\Psi_{C_i}(s(i))$, where $1 \leq i \leq I$ for the corresponding noisy signal C_i and then denoising is performed by following the procedure described in Sec. 6.2.

6.3.3 Setup for experiments

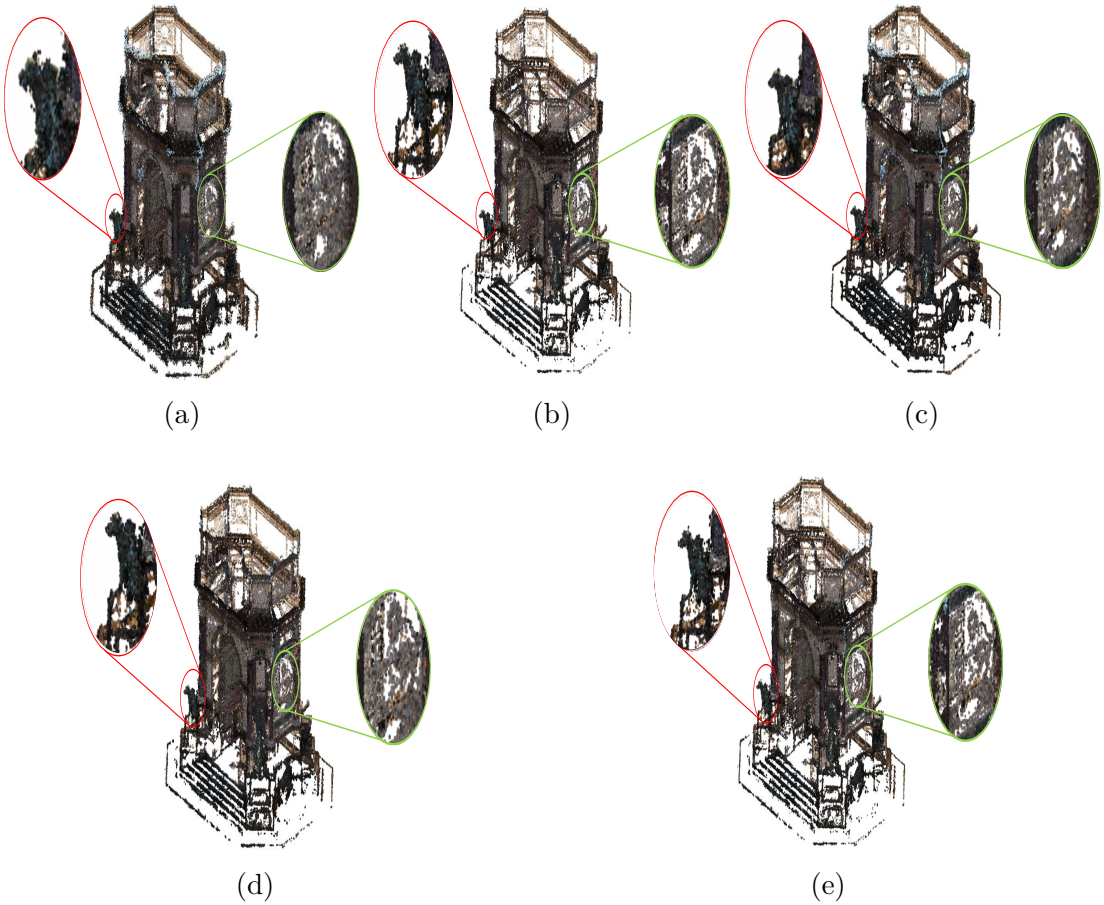
We have set six (6) wavelet decomposition levels, and maintain the wavelet coefficients of a scale $s(1)$ and then preserve the wavelet coefficients that exceeds the threshold $\tau(\sigma_X)$, which is calculated using Eq. 6.6 for the corresponding noise level in $s(i)$ for $2 \leq i \leq I$. The whole set of parameters used in this paper required for geometry-only and color-only denoising scenario are given in Tab. 6.4 and Tab. 6.5, respectively.

6.3.4 Visual results of geometry denoising algorithm

The visual inspection of geometry denoising algorithms has been performed for both synthetic and natural point clouds.

Geometry denoising of natural point clouds

We show a visual comparison of the proposed point cloud denoised algorithm with the denoising technique that constructs a graph from geometry only and performs regularization as in [112], along with the algorithm described in [17]. The proposed algorithm is applied to real-world natural point clouds, for which we do not have a noiseless reference; thereby, the results are only qualitative. Fig. 6.7-a and Fig. 6.8-a present the point clouds with real noise; it can be observed that the points with similar color are typically dispersed in a small neighborhood in the acquisition process, which may blur the details.



(a) (b) (c)

(d) (e)

Figure 6.7: Arco_Valentino model. (a) Noisy input (b) outlier-free input, denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) MSGW [17] applied to outlier-free input, and (e) IBR performed after the outlier removal step [112].

The resulting output after outlier removal is shown in Fig. 6.7-b and Fig. 6.8-b, a prior step required for denoising algorithms such as MSGW [17] and IBR [112].

The resulting denoised point cloud using the proposed method is reported in Fig. 6.2-c and Fig. 6.8-c. Here, it can be seen that the noisy points are relocated close to their true position by taking advantage of the correlation of their geometry and color, thus preserving the details which were hidden in the noisy input point cloud.

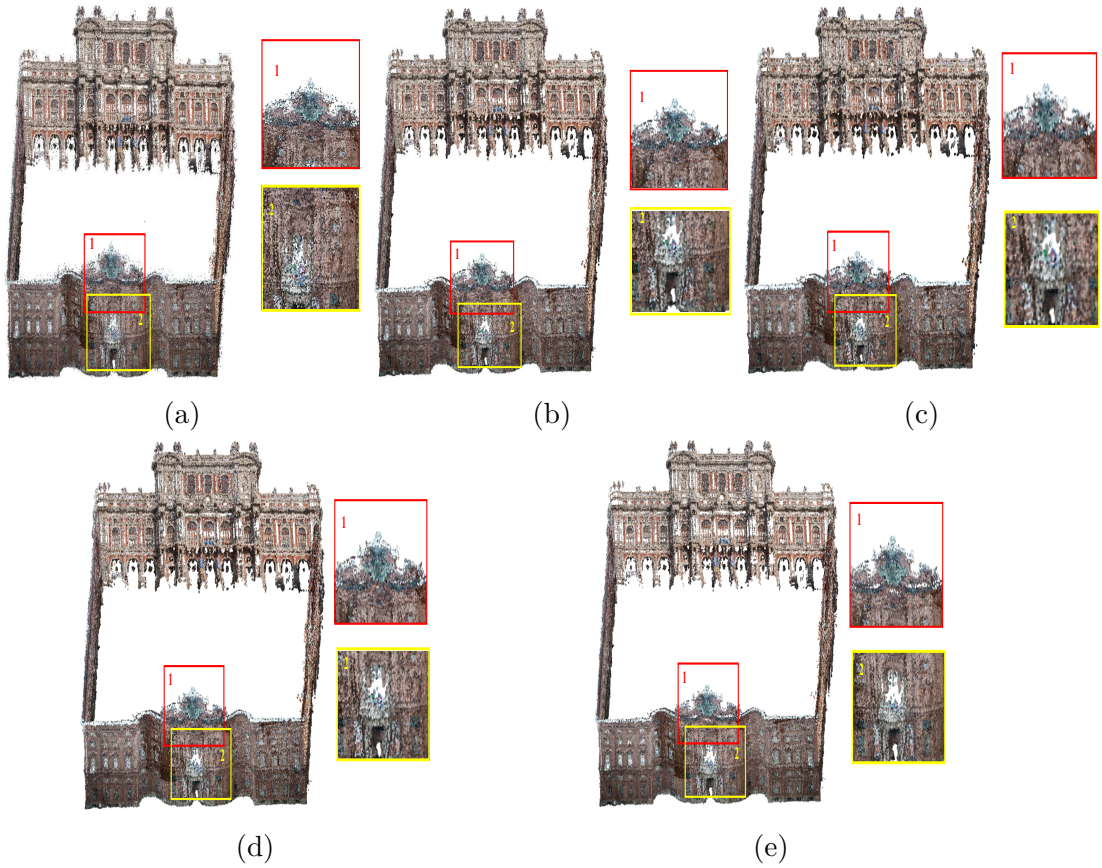


Figure 6.8: Palazzo_Carignano model. (a) Noisy input (b) outlier-free input, denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) MSGW [17] applied to outlier-free input, and (e) IBR performed after the outlier removal step [112].

The adaptive thresholding is very helpful for denoising the real-world point cloud, for which prior knowledge of the standard deviation of the noise is not required and is estimated using Eq. 6.4. Fig. 6.7-d and Fig. 6.8-d show the denoised point cloud using MSGW; it appears that the technique is not preserving the details effectively in the geometry irrespective of employing the outlier removal step before performing denoising.

Furthermore, prior knowledge of noise-level (σ) is required to apply soft-thresholding in [51], which is unknown for real-world point clouds, and geometry denoising depends on the selection of the σ for soft-thresholding.

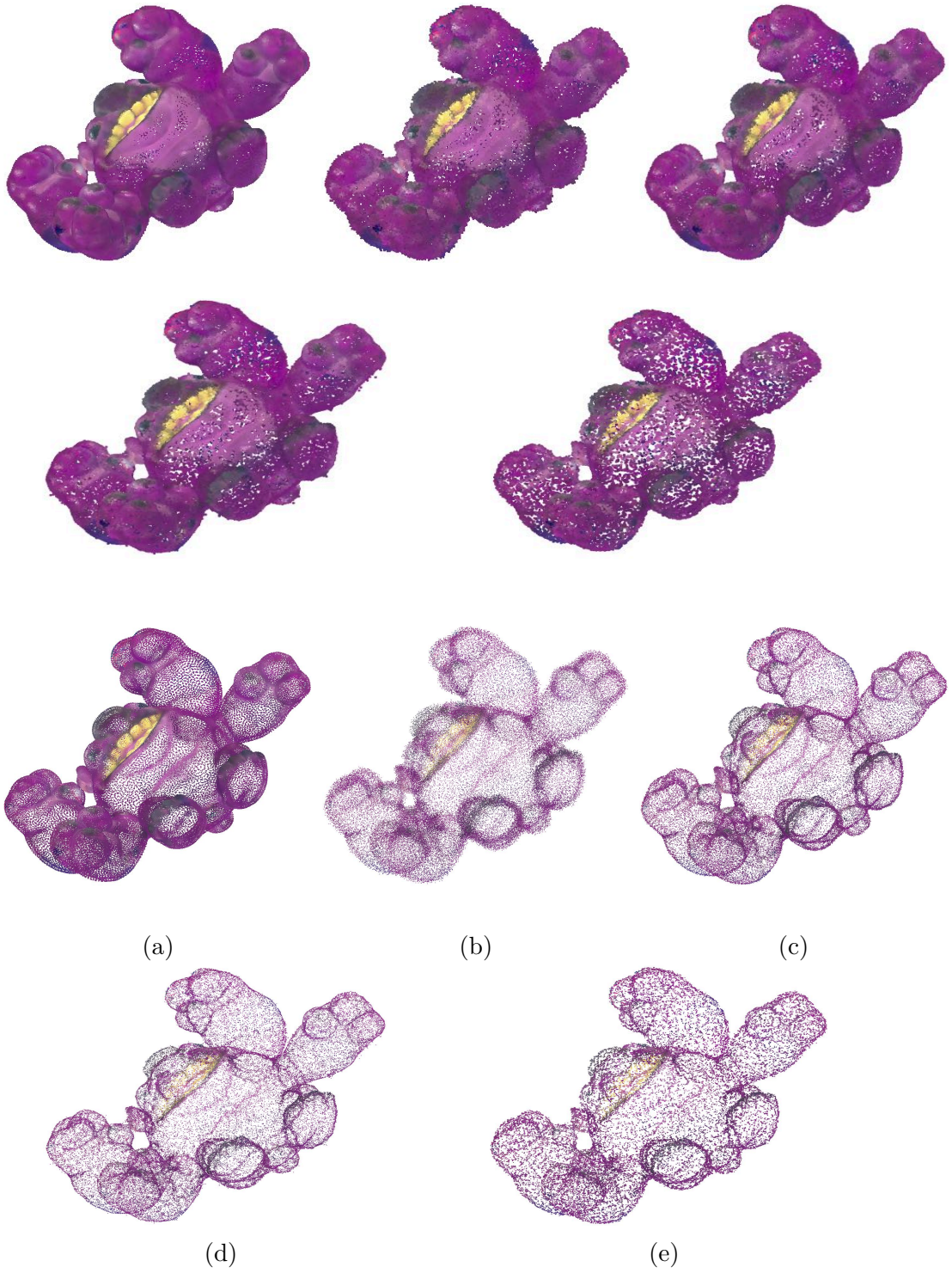


Figure 6.9: 4arms_monstre model with color: (a) ground-truth, (b) noisy input ($\mu = 0$ and $\sigma = 0.3$), denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) MSGW [17], and (e) IBR [112].

Fig. 6.7-e and Fig. 6.8-e depict the resulting denoised output using IBR [112] applied to the outlier-free input; it can be noted in the same region that, considering no color information, the noisy points are not relocated to their original location with respect to the proposed algorithm. However, IBR performs better than MSGW in denoising but enlarges gaps due to over-regularization and typically provides a noisier result near object boundaries.

This is an iterative-based algorithm, and for real-world point cloud, it is computationally very expensive; furthermore, the outlier removal step is essential for both MSGW and IBR, which turns these into more computational complex algorithms.

Geometry denoising of LiDAR

For the visual comparison of proposed denoising technique with the algorithm defined in [17], we performed experiments on LiDAR image shown in Fig. 6.10. We, then denoised the input point cloud using proposed and MSGW [17] algorithms, the outputs are shown in Fig. 6.11 and Fig. 6.12 where, we have highlighted a couple of regions to understand both the denoising and adverse effects i.e., (creation of holes) of the respective techniques. We can see that the proposed algorithm is relatively performs better with less side effects. The wavelet based approach for denoising the LiDAR is very less computationally expensive in comparison with the iterative based techniques. The computational cost for different types of point clouds, denoised with various algorithms are shown in Table. 6.12.

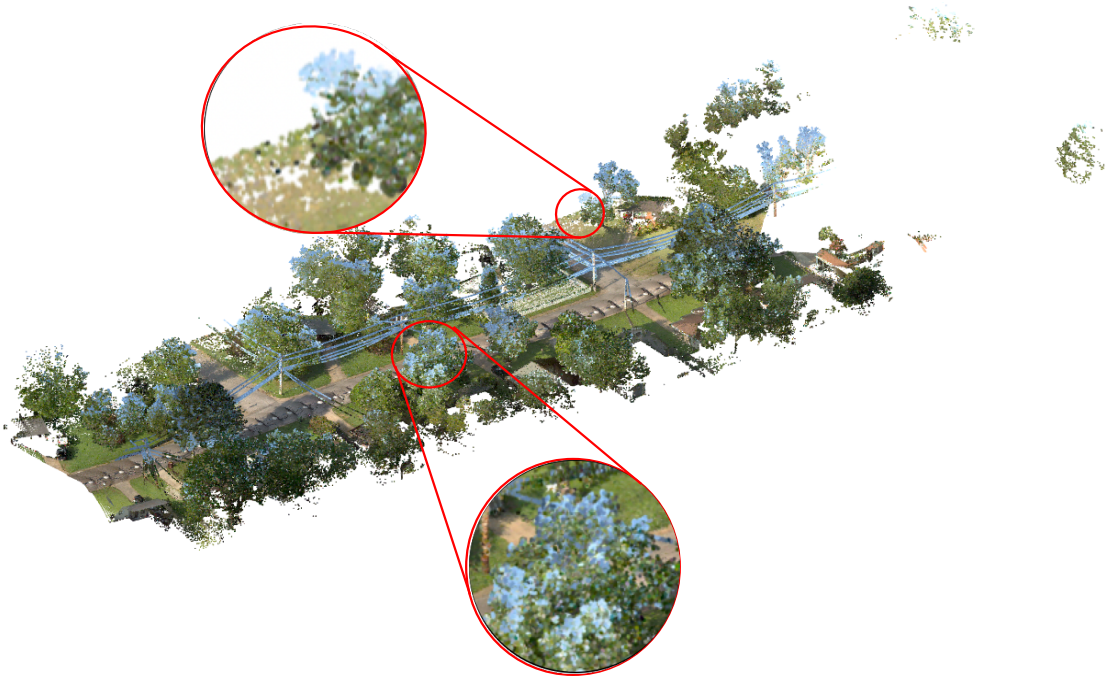


Figure 6.10: Input LiDAR model: MastinLake9_001_colorized0.

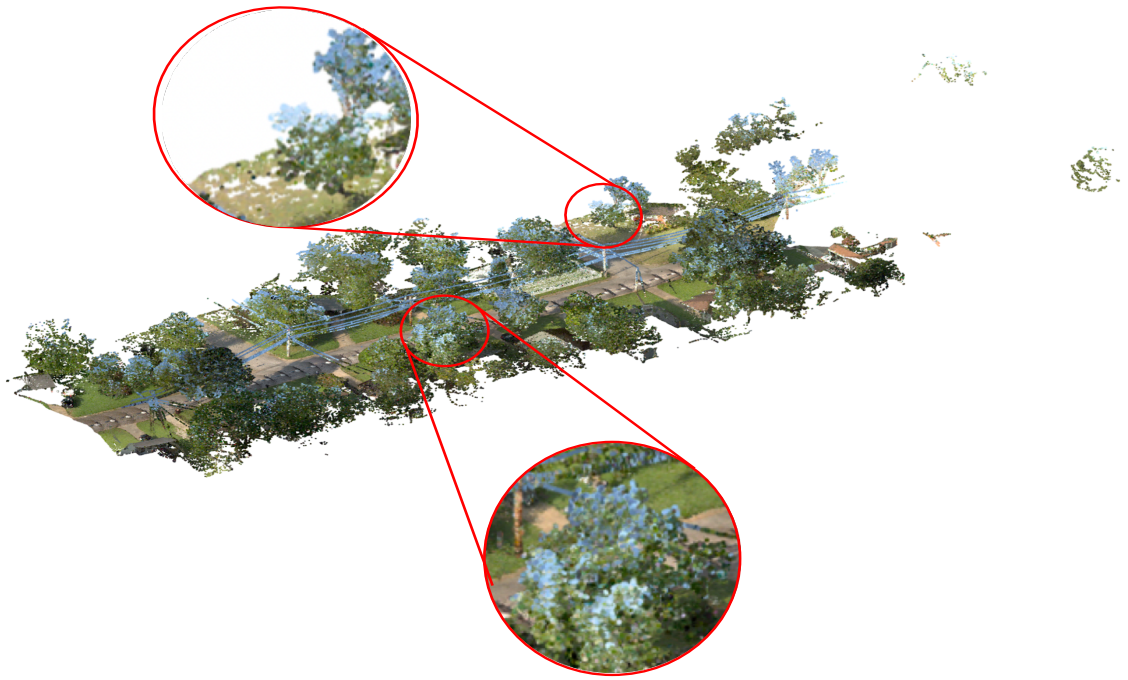


Figure 6.11: Denoised LiDAR model with proposed algorithm.

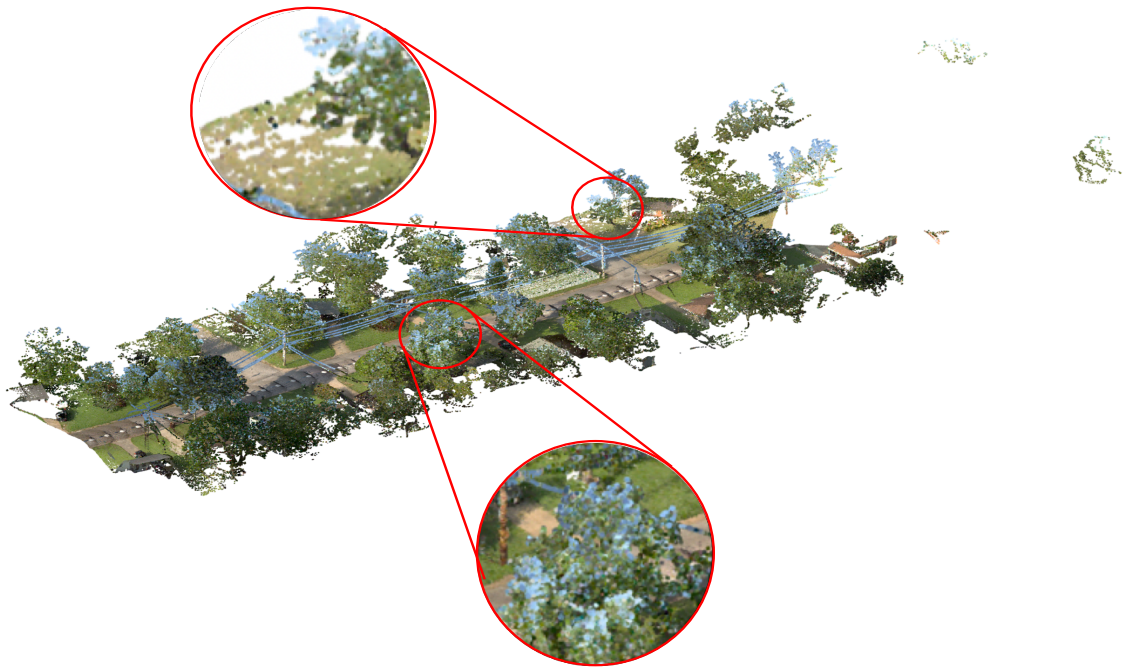


Figure 6.12: Denoised LiDAR model with MSGW [17].

Geometry denoising of ToF point clouds

To perform the proposed algorithm for geometry denoising on point cloud generated from ToF, we take the sample point cloud *3 Plastic Laundry Detergent on Carpet* generated using Helio2 ToF 3D camera, and color overlay is done with Triton 3.2MP camera. The ground-truth ToF is shown in Fig. 6.13a. Gaussian synthetic geometry noise with $\mu = 0$ and $\sigma = 0.3$ is then added to each point in a ground-truth ToF. The denoising results with proposed algorithm and MSGW [17] are shown in Fig. 6.13c and Fig. 6.13d, respectively. The denoising results indicate that the proposed algorithm is slightly better in denoising with fewer adverse effects than the MSGW. However, in terms of overall smoothness, the proposed algorithm is far better than MSGW. Thus, we can say that adding an extra feature in the graph construction can help in obtaining better denoising results.

Geometry denoising of synthetic point clouds

The experiments have also been performed on noise-free point clouds from the *Greyc* dataset [85], corrupted with Gaussian synthetic geometry noise with $\mu = 0$ and $\sigma = 0.2$, 0.3, and 0.4. In Fig. 6.5, the results are presented in two rows for *4arms_monstre* and *Asterix* models, respectively. For each point cloud model in Fig. 6.5, the first row of figures is their natural representation. The artifacts (i.e., hole formation) can be seen in the resulting denoised point clouds using MSGW and IBR with reference to the proposed algorithm. Moreover, an alternate view is provided in the second and fourth rows; they are the zoomed-in versions of the point clouds shown with the identical size in the first and third row. This sort of visual representation is more sparse and allows to differentiate better the noise removal and fine details at the boundaries. Fig. 6.5-a shows the ground-truth point cloud models. Fig. 6.5-b presents the noisy point clouds with $\sigma = 0.3$. The resulting denoised point clouds using the proposed algorithm are shown in Fig. 6.5-c; it can be observed that the geometry noise has been regularized, and the noisy points are proximate to their actual positions with the less adverse effect of holes formation. The denoised output of MSGW is shown in Fig. 6.5-d; it can be clearly seen that the geometry noise is not properly removed and also causes the opening of holes in the output point clouds. Fig. 6.5-e depicts the resulting denoised point clouds obtained by the IBR algorithm. It is evident that the geometry is not entirely regularized with respect to the proposed algorithm, while still better than MSGW; nevertheless, the resulting output of the IBR algorithm has large holes; the reason is the γ value, higher γ is appropriate for denoising well, but it causes severe artifacts. In general, it can be noted from the visual results of both real-world and synthetic point clouds that the denoised point clouds using the proposed algorithm have superior quality and fewer artifacts.

A comparative study has also been conducted with respect to RPSM [16], and the outcomes are shown in Fig. 6.15. In this specific case, we have applied the

Figure 6.13: 3 Plastic Laundry Detergent on Carpet model: (a) ground-truth (b) noisy input, geometry denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding (d) MSGW [17].

proposed algorithm on sub-sampled point clouds from the same dataset because of the large memory requirement of RPSM as anticipated in Sec. 5.5.5. The sub-sampling performed here is on a spatial basis, where the average minimum distance between the two points in each point cloud is set to 0.80. The number of points in each point cloud model of the Greyc dataset [85] is around 20,000 on average.

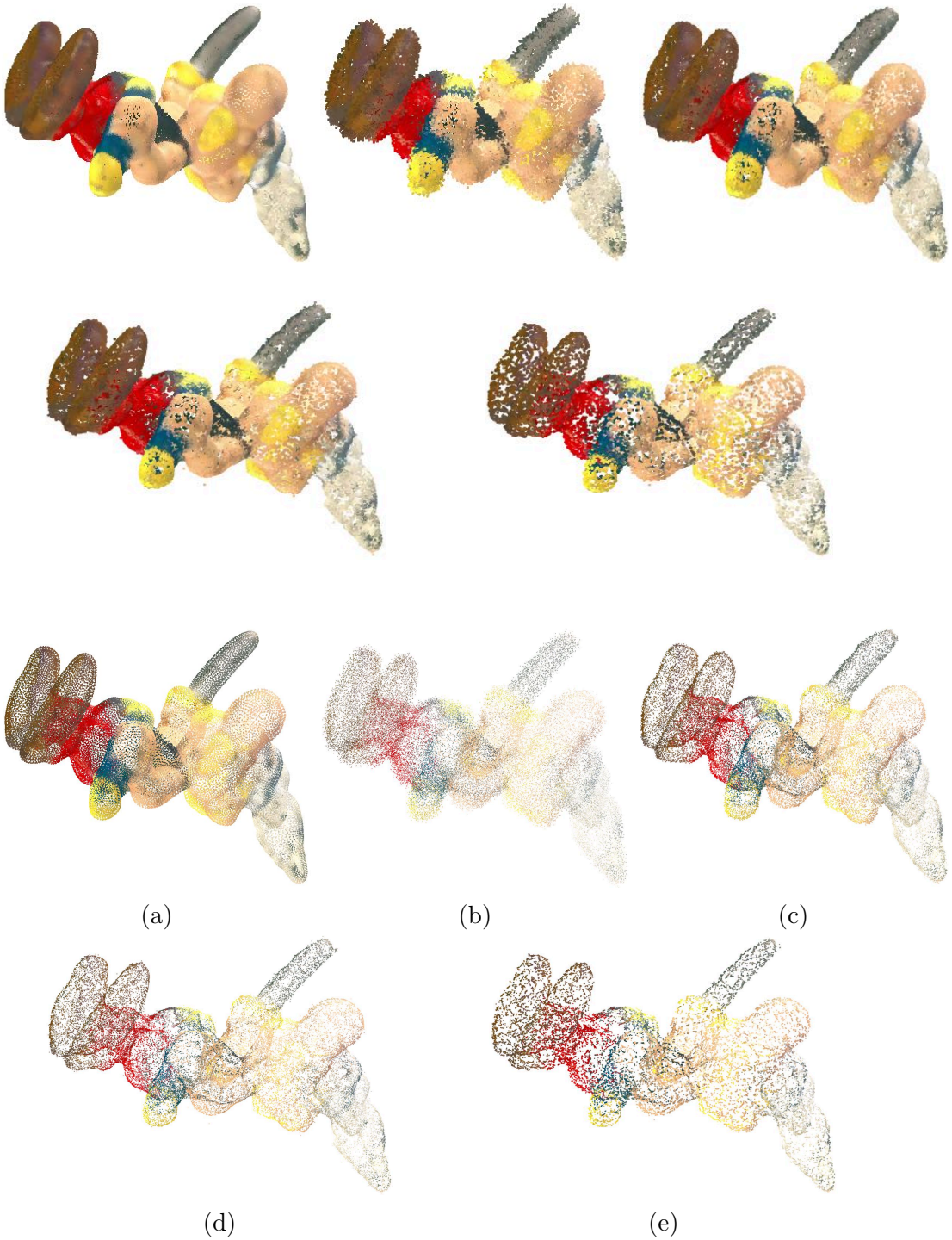
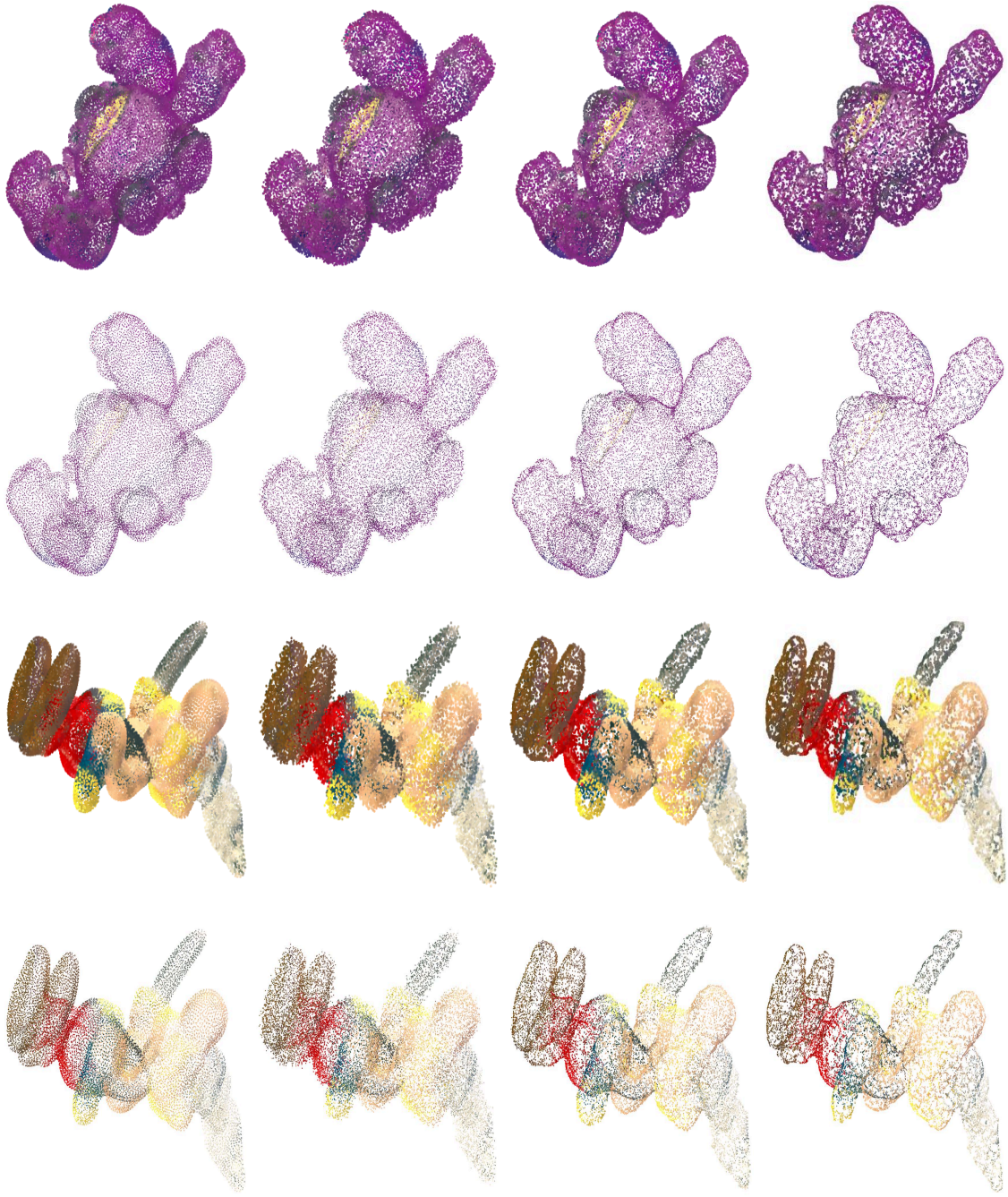


Figure 6.14: Asterix model with color: (a) ground-truth, (b) noisy input ($\mu = 0$ and $\sigma = 0.3$), denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) MSGW [17], and (e) IBR [112].



(a) (b) (c) (d)

Figure 6.15: Sub-sampled point cloud models with color: (a) ground-truth, (b) noisy input ($\mu = 0$ and $\sigma = 0.3$), denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, and (d) RPSM [16].

The results of the proposed algorithm and RPSM [16] on sub-sampled *4arms_monstre*

and *Asterix* are presented in two rows for each point, respectively. The proposed algorithm and RPSM [16] are applied to the sub-sampled noisy inputs shown in Fig. 6.15-b; the corresponding ground-truth point clouds are shown in Fig. 6.15-a.

The denoised outputs of the proposed algorithm are shown in Fig. 6.15-c; it can be observed that the proposed algorithm performs better at geometry denoising with very few artifacts, and the resulting denoised outputs of RPSM [16] are shown in Fig. 6.15-d; it can be seen that RPSM [16] over-regularizes the sub-sampled point clouds which tends to generate the holes in the resulting denoised point cloud.

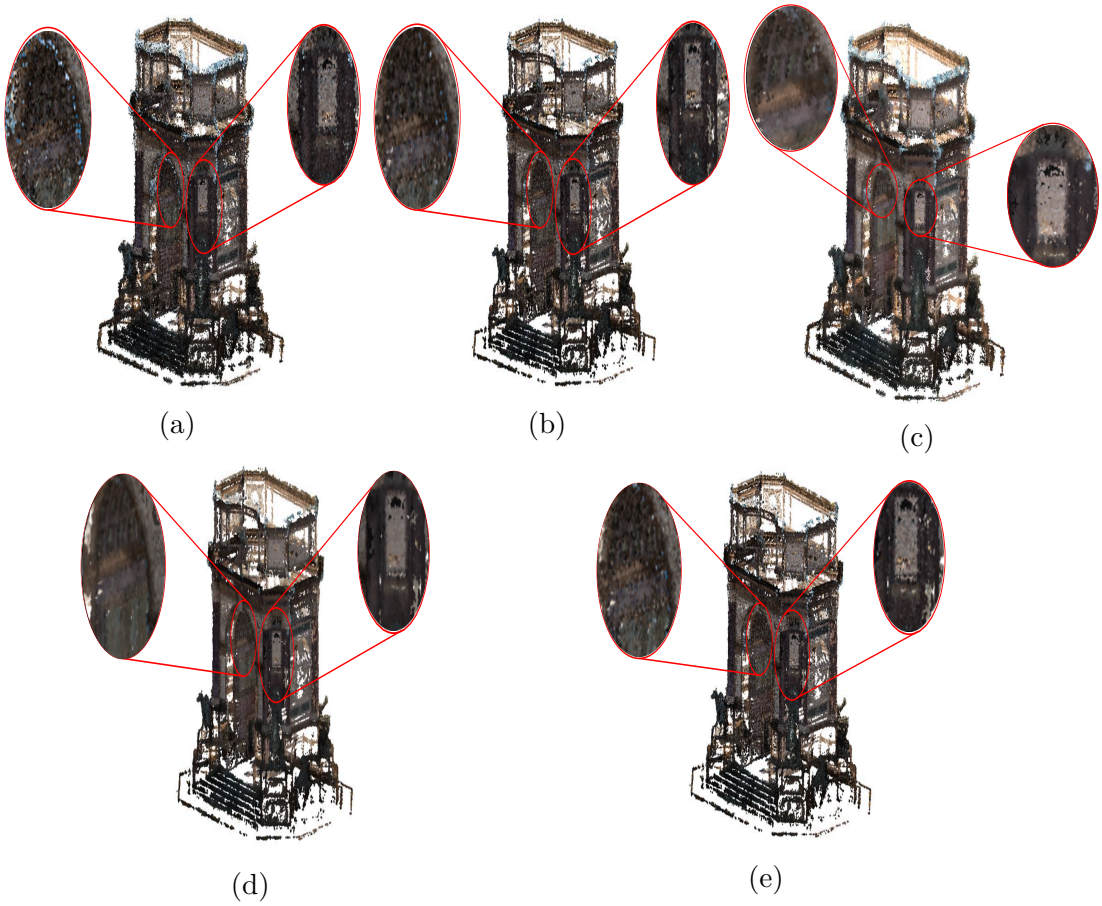


Figure 6.16: Arco_Valentino model illustration. (a) noisy input, (b) outlier-free input, color denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) using Tikhonov regularization, and (e) using TV.

6.3.5 Visual results of color denoising algorithm

The visual inspection of color denoising algorithms has been performed for both synthetic and real-world point clouds. The proposed color denoising technique has

been compared with the color denoising method described in [50], where the color of the point clouds is denoised using a different set of parameters θ_X and θ_C. Graph gradient was used to measure the degree of smoothness of a geometry/color graph signal. The color denoising technique [50] is an iterative convex optimization technique that enforces the regularity of the denoised color attributes on a defined graph.

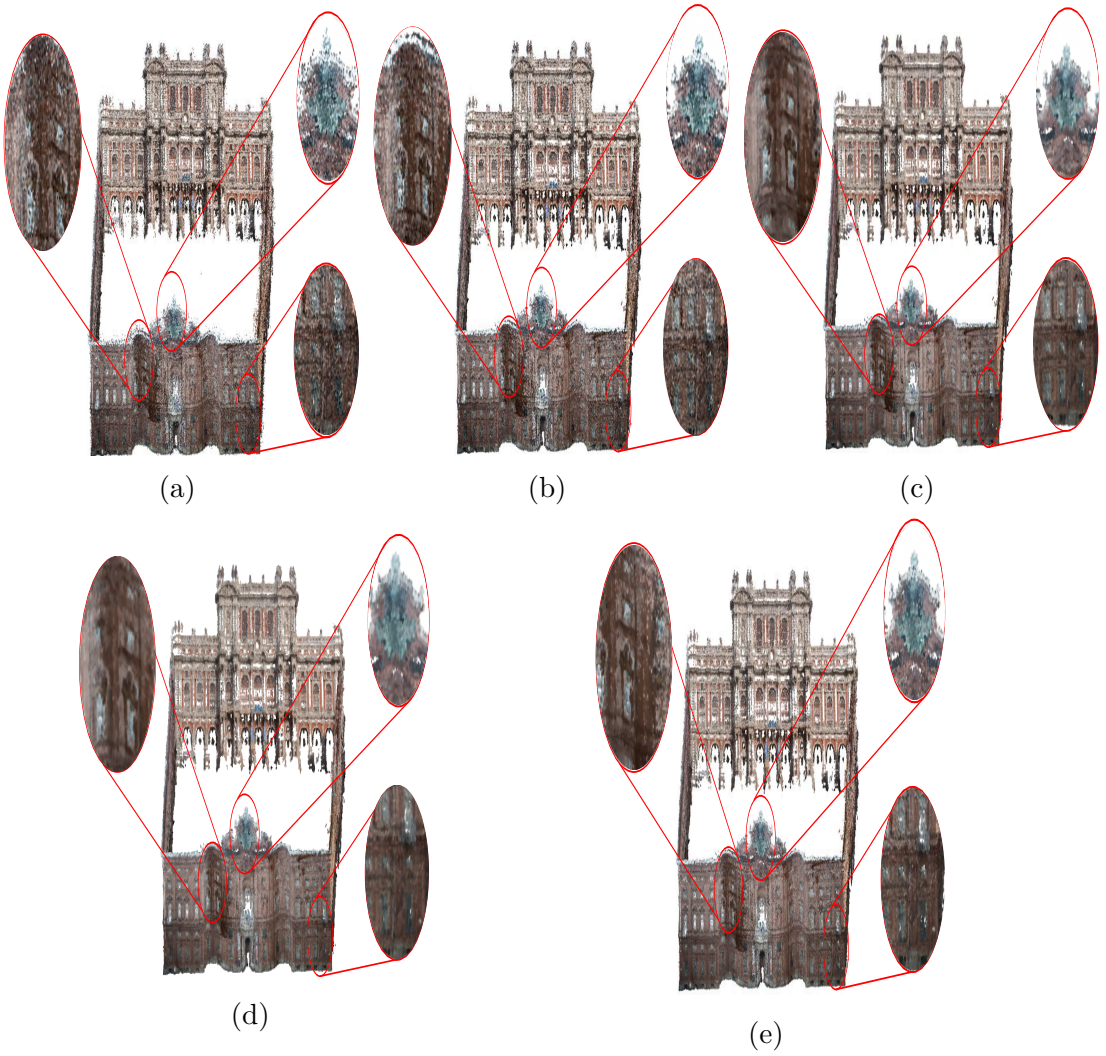


Figure 6.17: Palazzo_Carignano_Dense model illustration. (a) noisy input, (b) outlier-free input, color denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) using Tikhonov regularization, and (e) using TV.

Color denoising of real-world point clouds

The visual results of the proposed color denoising algorithm on real-world point clouds are presented here. There is not much literature available for color denoising of the point cloud as anticipated in Sec. 1. Here, the qualitative results show the comparison between the proposed algorithm using SGW with Tikhonov and TV-based regularization. Fig. 6.16-a and 6.17-a represent the real-world noisy point clouds; it can be observed that the details are not very clear due to the existence of a large amount of color noise.

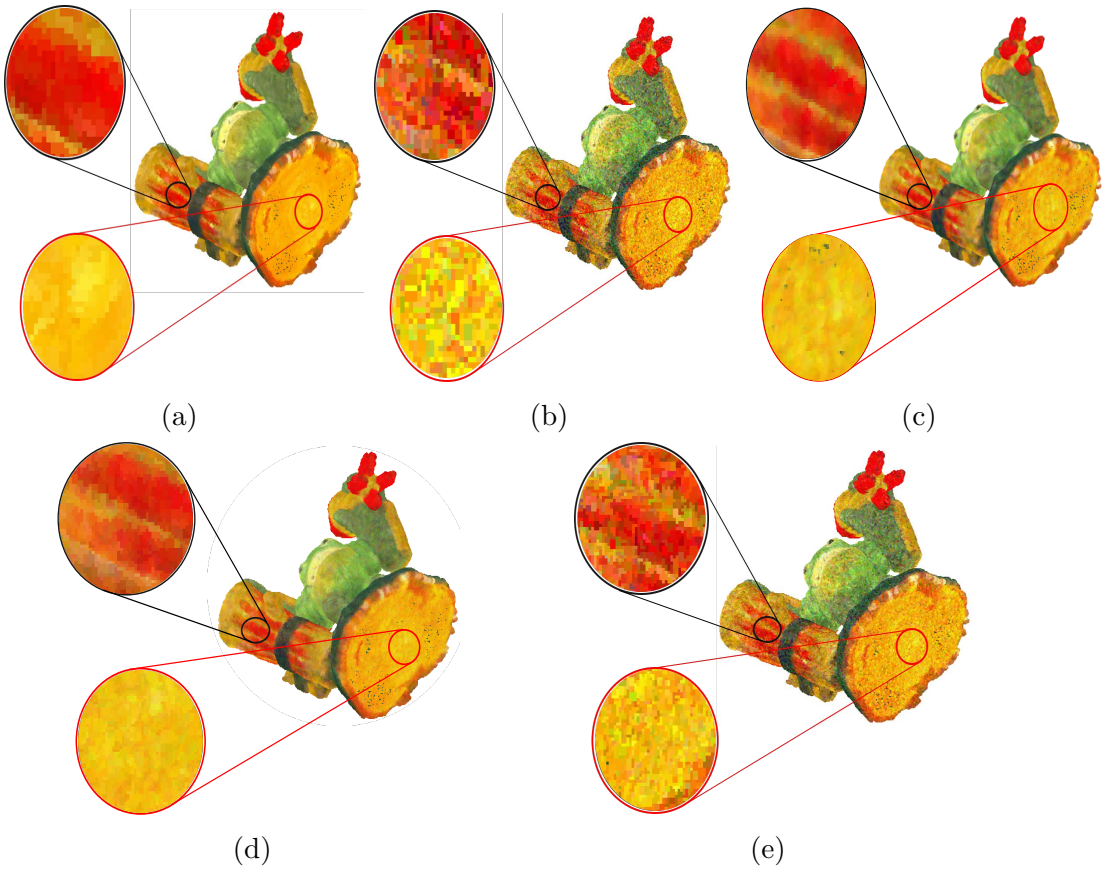


Figure 6.18: Green_monster model: (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0$ and $\sigma = 30$, color denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) using Tikhonov regularization, and (e) using TV.

In certain areas, two different regions are overlapped with each other as an effect of color noise. Fig. 6.16-b and 6.17-b show the resulting point clouds after outlier removal, which is required to get better color denoising results using Tikhonov and TV regularization. Fig. 6.16-c and 6.17-c depict the point cloud denoised using the

proposed algorithm. Here, the colors are much smoother and natural by exploiting the relation of the color of the points within proximity. Due to the noise in the point cloud, details are missing, and one can not see the contours in the real-world point cloud. The denoised point clouds look sharper in comparison to the input noisy point clouds.

The color denoising procedure helps to preserve object boundaries. Fig. 6.16-d and Fig. 6.17-d show the denoised result using Tikhonov regularization; it can be seen that the details are quite similar to the proposed algorithm but a little over-smoothed. Fig. 6.16-e and Fig. 6.17-e show the denoised point cloud using TV; it can be seen that the color is still noisy, and there is a lack of details in the output point clouds. TV is not very effective at enforcing color smoothness in comparison to the proposed algorithm. Tikhonov and TV are iterative-based and parameter oriented techniques, which tend to be computationally expensive.

Color denoising of synthetic point clouds

The proposed algorithm for color denoising has been applied to noise-free point clouds affected by synthetic color noise; Gaussian distribution is used to add noise to the color attribute of every point in a reference point cloud while keeping the geometry noise-free. Fig. 6.18-a and 6.19-a present the ground-truth *Green_Monster* and *Asterix* point cloud models, respectively having noise-free geometry and color. Fig. 6.18-b and 6.19-b show the point cloud affected by Gaussian noise distribution with $\mu = 0$ and $\sigma = 30$; adding noise to the color affects the details and causes blurring of the boundaries. Fig. 6.18-c and 6.19-c depict the denoised output of the proposed algorithm.

The color of the output point cloud is denoised by exploiting the correlation of color within the proximity; the points in the k-neighborhood have a high probability of having a similar color as the surface has smooth color. Fig. 6.18-d and 6.19-d depict the denoised output using Tikhonov regularization; it can be clearly seen, particularly in the highlighted areas of both point cloud models, that there is still noise in color. Fig. 6.18-e and 6.19-e illustrate the denoised output using TV. The output point clouds are still noisy, and the details are not preserved. The TV technique has the least effective in terms of color denoising.

6.3.6 Objective evaluation on Greyc Color mesh dataset

The quantitative evaluation has also been performed on the Greyc noise-free synthetic point clouds dataset.

Geometry denoising

The proposed geometry denoising method has also been verified via quantitative evaluation on the complete *Greyc* dataset. Each point cloud has been corrupted

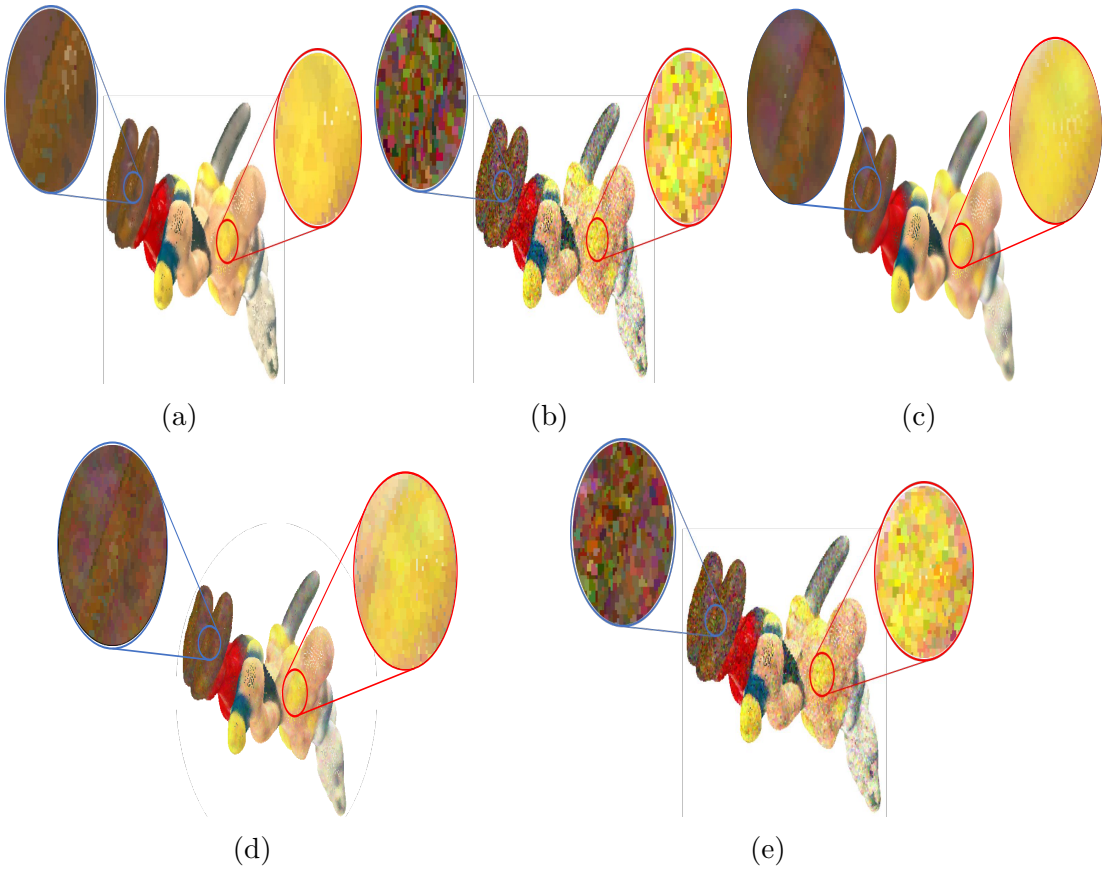


Figure 6.19: Asterix model: (a) ground-truth (b) noisy point cloud with noise level of $\mu = 0$ and $\sigma = 30$, color denoised results by (c) proposed algorithm based on SGW using data-driven adaptive soft-thresholding, (d) using Tikhonov regularization, and (e) using TV.

with zero-mean Gaussian synthetic geometry noise, applied to each point with $\sigma = 0.2$, 0.3, and 0.4. The MSE and MCD comparisons between the proposed algorithm and the denoising approaches used in IBR [112] using TV regularization and MSGW are shown in Tab. 6.6. The results show that the proposed denoising technique performed better than MSGW and IBR for all the models for noise level $\sigma = 0.2$, 0.3, and 0.4 except *Green_Dinasour* and *Red_Horse* (where MSGW performed better) for $\sigma = 0.2$,0.3.

For further verification of the performance of the proposed geometry denoising algorithm, we compute the C2M metric for each denoised point cloud by the proposed algorithm, MSGW, and IBR with respect to the corresponding reference ground-truth point cloud. The comparative C2M metric results are shown in Tab. 6.8, which further verifies the better performance of the proposed algorithm. To better understand the results, we mapped the C2M distance of denoised point cloud

Table 6.6: MSE and MCD comparison of various algorithms for *Greyc* dataset.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
MSE																		
$\sigma = 0.2$	Proposed	**0.33**	**0.26**	**0.21**	**0.22**	**0.61**	0.38	**0.17**	**0.38**	**0.13**	**0.16**	**0.12**	**0.13**	**0.35**	**0.35**	0.26	**0.35**	**0.28**
	MSGW	0.36	0.27	0.23	**0.22**	0.84	**0.36**	**0.17**	0.42	0.14	0.17	0.14	0.14	0.36	0.37	**0.23**	0.37	0.30
	IBR	0.38	0.30	0.29	0.30	0.63	0.40	0.23	0.45	0.20	0.24	0.19	0.19	0.41	0.40	0.30	0.37	0.33
$\sigma = 0.3$	Proposed	**0.34**	**0.27**	**0.23**	**0.23**	**0.68**	0.39	**0.18**	**0.39**	**0.13**	**0.17**	**0.13**	**0.15**	**0.38**	**0.36**	0.28	**0.36**	**0.29**
	MSGW	0.37	0.28	0.24	0.24	0.92	**0.38**	**0.18**	0.44	0.15	0.18	0.15	**0.15**	0.40	0.40	**0.25**	0.38	0.32
	IBR	0.39	0.30	0.30	0.31	0.71	0.42	0.23	0.45	0.20	0.25	0.20	0.19	0.44	0.42	0.32	0.38	0.34
$\sigma = 0.4$	Proposed	**0.35**	**0.28**	**0.25**	**0.26**	**1.12**	**0.39**	**0.21**	**0.40**	**0.15**	**0.18**	**0.15**	**0.16**	**0.40**	**0.38**	**0.29**	**0.36**	**0.33**
	MSGW	0.38	**0.28**	0.30	0.31	1.35	0.42	**0.21**	0.44	0.16	0.21	0.17	0.17	0.41	0.41	0.32	0.39	0.37
	IBR	0.39	0.31	0.30	0.32	1.15	0.42	0.24	0.45	0.21	0.26	0.21	0.20	0.46	0.42	0.33	0.38	0.38
MCD																		
$\sigma = 0.2$	Proposed	**0.49**	**0.38**	**0.30**	**0.31**	**0.85**	0.55	**0.25**	**0.56**	**0.19**	**0.24**	**0.18**	**0.19**	**0.51**	**0.52**	0.38	**0.50**	**0.40**
	MSGW	0.53	0.39	0.33	0.33	1.24	**0.53**	**0.25**	0.61	0.20	0.25	0.20	0.20	0.52	0.54	**0.34**	0.53	0.44
	IBR	0.55	0.43	0.42	0.44	0.94	0.59	0.33	0.65	0.29	0.35	0.28	0.27	0.61	0.58	0.44	0.53	0.48
$\sigma = 0.3$	Proposed	**0.50**	**0.39**	**0.32**	**0.32**	**1.22**	0.57	**0.25**	**0.57**	**0.20**	**0.25**	**0.20**	**0.21**	**0.56**	**0.52**	0.40	**0.51**	**0.44**
	MSGW	0.54	0.40	0.35	0.35	1.36	**0.55**	0.26	0.63	0.21	0.26	0.22	**0.21**	0.58	0.58	**0.36**	0.54	0.46
	IBR	0.56	0.44	0.43	0.46	1.06	0.61	0.34	0.66	0.30	0.36	0.29	0.28	0.65	0.60	0.46	0.55	0.50
$\sigma = 0.4$	Proposed	**0.52**	**0.40**	**0.40**	**0.41**	**1.64**	**0.56**	**0.27**	**0.58**	**0.21**	**0.26**	**0.23**	**0.23**	**0.58**	**0.55**	**0.42**	**0.53**	**0.49**
	MSGW	0.54	0.41	0.43	0.45	1.95	0.61	0.31	0.64	0.24	0.30	0.24	0.25	0.61	0.60	0.47	0.54	0.54
	IBR	0.57	0.45	0.44	0.47	1.71	0.62	0.35	0.68	0.30	0.37	0.30	0.29	0.67	0.60	0.47	0.55	0.55

of *4arms_monstre* on the reference point cloud; this is shown in Fig. 6.21. The distances are represented by the color scale; blue, green, yellow, and red display the range of distances from minimum to maximum. The histogram of the proposed algorithm in Fig. 6.21-a shows that the points move closer to their actual position without moving too far from their corresponding original position of a reference point cloud. We can see in the histograms shown in Fig. 6.21-b and Fig. 6.21-c that the points have deviated too much from their actual position, and also the number of points moved farther are higher for MSGW and IBR.

The objective evaluation has also been performed for the comparison between the proposed algorithm and RPSM [16] on the sub-sampled point cloud models of the *Greyc* dataset [85]. Tab. 6.7 shows the MSE and MCD comparison; it can be seen that the proposed algorithm outperformed RPSM [16] in terms of both metrics

Table 6.7: MSE and MCD comparison between proposed algorithm and RPSM [16] on sub-sampled *Greyc* dataset for different noise levels.

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
MSE																		
$\sigma = 0.2$	Proposed	**0.61**	**0.49**	**0.85**	**0.67**	**0.83**	**0.74**	**0.59**	**0.73**	**0.50**	**0.59**	**0.55**	**0.58**	**0.51**	**0.74**	**0.76**	**0.56**	**0.64**
	RPSM [16]	0.90	0.74	1.2457	1.02	1.33	1.00	0.87	1.19	0.78	0.73	1.02	0.86	0.73	0.94	1.18	1.03	0.97
$\sigma = 0.3$	Proposed	**0.61**	**0.49**	**0.86**	**0.67**	**0.87**	**0.75**	**0.58**	**0.74**	**0.48**	**0.59**	**0.56**	**0.58**	**0.50**	**0.75**	**0.77**	**0.57**	**0.65**
	RPSM [16]	0.95	0.76	1.34	1.06	1.35	1.03	0.93	1.24	0.83	0.80	1.07	0.92	0.79	1.00	1.25	1.06	1.02
$\sigma = 0.4$	Proposed	**0.62**	**0.50**	**0.87**	**0.68**	**0.90**	**0.77**	**0.59**	**0.75**	**0.51**	**0.58**	**0.56**	**0.59**	**0.51**	**0.77**	**0.79**	**0.58**	**0.66**
	RPSM [16]	0.98	0.79	1.41	1.13	1.39	1.12	0.97	1.28	0.89	0.85	1.12	0.98	0.90	1.05	1.30	1.11	1.08
MCD																		
$\sigma = 0.2$	Proposed	**0.89**	**0.70**	**1.24**	**0.98**	**1.24**	**1.09**	**0.86**	**1.07**	**0.72**	**0.85**	**0.80**	**0.83**	**0.74**	**1.08**	**1.11**	**0.83**	**0.94**
	RPSM [16]	1.44	1.04	1.93	1.56	1.91	1.60	1.33	1.79	1.18	1.24	1.48	1.25	1.51	1.63	1.79	1.56	1.51
$\sigma = 0.3$	Proposed	**0.90**	**0.71**	**1.26**	**0.98**	**1.29**	**1.10**	**0.86**	**1.08**	**0.71**	**0.86**	**0.82**	**0.84**	**0.74**	**1.09**	**1.13**	**0.84**	**0.95**
	RPSM [16]	1.51	1.08	2.05	1.63	1.94	1.67	1.41	1.87	1.24	1.32	1.55	1.33	1.60	1.72	1.90	1.60	1.59
$\sigma = 0.4$	Proposed	**0.91**	**0.72**	**1.27**	**0.99**	**1.33**	**1.12**	**0.87**	**1.10**	**0.74**	**0.84**	**0.83**	**0.86**	**0.75**	**1.12**	**1.16**	**0.85**	**0.96**
	RPSM [16]	1.55	1.11	2.15	1.71	2.00	1.77	1.47	1.92	1.32	1.40	1.63	1.41	1.73	1.78	1.97	1.66	1.66

for $\sigma = 0.2$, 0.3, and 0.4. The average MSE and MCD (last column in Tab. 6.6 and Tab. 6.7) shows that the gain is larger as the noise level increases, indicating that the proposed denoising method is better at removing geometry noise.

Color denoising

The color attribute of each point cloud is corrupted with Gaussian noise applied to each point in a point cloud with σ = 20,30,40. The MSE and PSNR comparisons between the proposed color denoising algorithm and the color denoising using Tikhonov and TV regularization are shown in Tab. 6.9 and Tab. 6.10. The results of both metrics show that the proposed technique performed better than the Tikhonov and TV regularization for all the point cloud models for all the noise-level except *cable_car*, *Horse*, which performed better for the noise level of $\sigma = 20, 30$ and *Pokemon_ball* performed better only for noise-level $\sigma = 20$.

The average MSE and PSNR (last column in Tab. 6.9 and Tab. 6.10 and Fig. 6.20) shows that the gain is larger with the increase in the noise-level, showing that the proposed algorithm of color denoising using SGW performed better.

Table 6.8: C2M metric comparison of the proposed geometry denoising algorithm with the IBR and MSGW.

Geometry Noise	Methods	Parameters	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue
$\sigma = 0.2$	Proposed	d_H	**0.68**	**0.40**	**0.82**	**0.91**	**3.13**	1.50	**0.75**	**0.94**	**0.83**	**1.02**	**0.69**	**0.58**	**0.45**	**0.92**	1.27	**0.89**
		d_m	**0.19**	**0.12**	**0.12**	**0.12**	**0.44**	0.17	**0.05**	**0.24**	**0.06**	**0.07**	**0.07**	**0.08**	**0.16**	**0.23**	0.17	**0.25**
		ζ	**0.10**	**0.07**	**0.09**	**0.09**	**0.37**	0.10	**0.04**	**0.12**	**0.05**	**0.05**	**0.06**	**0.07**	**0.09**	**0.12**	0.13	**0.09**
	IBR	d_H	0.96	0.61	1.16	1.29	3.83	1.50	1.18	0.94	0.83	1.12	1.19	0.82	1.00	1.31	1.28	1.26
		d_m	0.30	0.21	0.22	0.24	0.60	0.30	0.09	0.37	0.14	0.17	0.14	0.14	0.35	0.33	0.24	0.29
		ζ	0.11	0.09	0.11	0.09	0.43	0.10	0.61	0.13	0.05	0.06	0.08	0.07	0.15	0.13	0.10	0.12
	MSGW	d_H	0.68	0.86	1.17	0.91	3.53	**1.50**	1.06	0.94	0.83	1.02	1.38	1.31	0.63	0.93	**1.27**	0.90
		d_m	0.25	0.13	0.16	0.17	0.76	**0.16**	0.07	0.29	0.10	0.10	0.09	0.09	0.20	0.28	**0.13**	0.26
		ζ	0.11	0.08	0.12	0.13	0.66	**0.09**	0.05	0.13	0.07	0.07	0.07	0.07	0.10	0.13	**0.09**	0.09
$\sigma = 0.3$	Proposed	d_H	**0.68**	**0.86**	**0.82**	**0.91**	**3.13**	1.50	**0.75**	**0.93**	**0.85**	**1.02**	**0.97**	**0.83**	**0.45**	**0.94**	1.27	**0.89**
		d_m	**0.20**	**0.13**	**0.13**	**0.13**	**0.50**	0.17	**0.05**	**0.25**	**0.07**	**0.07**	**0.08**	**0.09**	**0.16**	**0.24**	0.19	**0.26**
		ζ	**0.09**	**0.07**	**0.10**	**0.10**	**0.41**	0.11	**0.05**	**0.12**	**0.06**	**0.06**	**0.07**	**0.08**	**0.10**	**0.13**	0.15	**0.09**
	IBR	d_H	0.96	0.91	1.165	1.29	3.98	1.51	1.59	0.96	0.92	1.04	1.43	1.18	0.10	1.31	1.29	1.28
		d_m	0.30	0.21	0.23	0.25	0.66	0.31	0.13	0.38	0.15	0.18	0.15	0.15	0.36	0.34	0.24	0.30
		ζ	0.12	0.09	0.13	0.10	0.24	0.10	0.67	0.14	0.06	0.07	0.12	0.13	0.16	0.14	0.10	0.12
	MSGW	d_H	0.68	0.96	1.17	0.91	3.87	**0.15**	1.58	0.93	0.90	1.03	1.39	1.69	0.90	1.31	**1.27**	1.27
		d_m	0.26	0.14	0.17	0.18	0.79	**0.16**	0.08	0.29	0.14	0.15	0.13	0.14	0.22	0.30	**0.13**	0.27
		ζ	0.11	0.09	0.13	0.14	0.68	**0.10**	0.07	0.13	0.10	0.10	0.10	0.10	0.10	0.13	**0.09**	0.10
$\sigma = 0.4$	Proposed	d_H	**0.96**	**0.86**	**1.06**	**0.91**	**3.54**	**1.50**	**1.18**	**0.94**	**1.18**	**1.03**	**1.38**	**1.17**	**0.64**	**0.94**	**1.27**	**1.27**
		d_m	**0.20**	**0.13**	**0.20**	**0.21**	**0.75**	**0.17**	**0.10**	**0.25**	**0.10**	**0.12**	**0.11**	**0.12**	**0.17**	**0.25**	**0.21**	**0.27**
		ζ	**0.12**	**0.08**	**0.15**	**0.15**	**0.64**	**0.12**	**0.10**	**0.13**	**0.08**	**0.09**	**0.09**	**0.10**	**0.10**	**0.13**	**0.16**	**0.11**
	IBR	d_H	1.21	1.17	1.36	1.58	4.02	1.52	1.75	1.32	1.24	1.76	1.53	1.44	1.11	1.61	1.37	1.28
		d_m	0.32	0.22	0.23	0.27	0.69	0.32	0.18	0.39	0.16	0.19	0.16	0.16	0.37	0.36	0.25	0.30
		ζ	0.13	0.09	0.15	0.11	0.26	0.15	0.13	0.16	0.09	0.11	0.12	0.13	0.16	0.15	0.10	0.13
	MSGW	d_H	1.16	1.14	1.22	1.30	4.11	1.51	1.25	1.32	1.23	1.45	1.44	1.38	0.91	1.51	1.28	1.28
		d_m	0.22	0.14	0.21	0.22	0.95	0.18	0.12	0.25	0.14	0.17	0.16	0.15	0.18	0.26	0.22	0.28
		ζ	0.13	0.10	0.16	0.16	0.78	0.13	0.11	0.14	0.08	0.10	0.11	0.12	0.11	0.13	0.17	0.12

To compare the proposed color denoising using SGW with GLR [20] and GTV [20, 13], we added Gaussian noise with zero-mean and $\sigma = 10, 15, 20, 25$ to the color

Table 6.9: MSE comparison of color denoising algorithm for *Greyc dataset.*

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
$\sigma = 20$	Noisy	398.25	361.28	373.52	393.64	339.57	397.24	365.38	383.29	368.57	387.52	321.09	375.24	309.14	335.40	377.39	398.13	367.79
	Proposed	**69.43**	**98.20**	166.40	**71.44**	**128.27**	**63.03**	**85.39**	166.60	**76.68**	**66.00**	**87.33**	**78.41**	156.10	**89.09**	**80.64**	**78.93**	**97.63**
	Tikhonov	77.75	103.56	**145.23**	82.80	130.65	63.04	91.64	**150.11**	85.83	77.90	95.91	89.56	**149.53**	93.75	82.52	84.77	100.29
	Total Variation	303.05	279.02	289.48	298.13	286.05	301.50	279.04	299.68	279.21	293.61	242.14	283.15	241.15	254.99	293.32	302.99	282.91
$\sigma = 30$	Noisy	869.27	781.84	816.58	867.14	722.06	875.25	796.19	850.03	797.43	850.89	692.54	810.94	645.11	713.61	811.93	882.35	798.95
	Proposed	**93.26**	**124.45**	224.75	**99.77**	**166.98**	**81.85**	**131.05**	226.10	**125.85**	**85.24**	**137.52**	**109.78**	**194.21**	**133.74**	**128.09**	**103.39**	**135.38**
	Tikhonov	109.74	148.53	**214.89**	120.77	189.47	89.12	138.24	**217.08**	130.09	112.13	157.44	136.40	250.35	148.58	133.63	123.06	151.22
	Total Variation	590.97	540.25	564.67	589.46	548.87	594.55	542.77	596.83	539.70	575.89	467.70	544.00	449.27	485.12	561.85	602.88	549.67
$\sigma = 40$	Noisy	1506.10	1329.01	1384.60	1462.90	1241.90	1469.80	1347.81	1462.10	1355.51	1462.61	1171.90	1372.91	1092.10	1217.80	1376.21	1519.90	1360.82
	Proposed	**108.93**	**203.40**	**269.15**	**137.18**	**221.99**	**107.32**	**181.40**	**274.05**	**139.83**	**119.45**	**194.93**	**149.54**	**263.86**	**204.24**	**221.17**	**129.75**	**182.89**
	Tikhonov	137.31	210.90	281.89	149.67	262.51	133.94	184.53	279.98	175.90	144.71	228.37	185.25	385.53	213.93	242.06	154.26	210.67
	Total Variation	1119.50	995.38	1033.80	1081.70	979.85	1088.40	1000.40	1107.10	1000.90	1083.80	862.00	1005.10	816.02	903.14	1026.80	1131.90	1014.74

Table 6.10: PSNR comparison of color denoising algorithm for *Greyc dataset.*

Gaussian Noise	Methods	4arms monstre	Asterix	Cable car	Dragon	Duck	Green Dinasour	Green monster	Horse	Jaguar	Long Diansour	Mario	Mario car	Pokeman ball	Rabbit	Red horse	Statue	**Average**
$\sigma = 20$	Noisy	22.13	22.55	22.41	22.18	22.82	22.14	22.50	22.30	22.47	22.25	23.07	22.39	23.23	22.88	22.36	22.13	22.49
	Proposed	**29.72**	**28.21**	25.92	**29.59**	**27.05**	**30.13**	**28.82**	25.91	**29.28**	**29.94**	**28.72**	**29.19**	26.20	**28.63**	**29.07**	**29.19**	**28.47**
	Tikhonov	29.22	27.98	**26.51**	28.95	26.97	30.13	28.51	**26.37**	28.79	29.21	28.31	28.61	**26.38**	28.41	28.97	28.85	28.26
	Total Variation	23.32	23.67	23.52	23.39	23.57	23.34	23.67	23.36	23.67	23.45	24.29	23.61	24.31	24.07	23.46	23.32	23.63
$\sigma = 30$	Noisy	18.74	19.20	19.01	18.75	19.55	18.71	19.12	18.84	19.11	18.83	19.73	19.04	20.03	19.60	19.04	18.67	19.12
	Proposed	**28.43**	**27.18**	24.61	**28.14**	**25.90**	**29.01**	**26.96**	24.59	**27.13**	**28.83**	**26.75**	**27.73**	**25.25**	**26.87**	**27.06**	**27.99**	**27.03**
	Tikhonov	27.73	26.42	**24.81**	27.31	25.36	28.63	26.72	**24.76**	26.99	27.63	26.16	26.78	24.14	26.33	26.88	27.23	26.49
	Total Variation	20.42	20.81	20.61	20.43	20.74	20.39	20.79	20.37	20.81	20.53	21.43	20.78	21.61	21.27	20.64	20.33	20.75
$\sigma = 40$	Noisy	16.35	16.90	16.72	16.48	17.19	16.46	16.84	16.48	16.81	16.48	17.44	16.75	17.75	17.28	16.74	16.31	16.81
	Proposed	**27.76**	**25.05**	**23.83**	**26.76**	**24.67**	**27.82**	**25.54**	**23.75**	**26.67**	**27.36**	**25.23**	**26.38**	**23.92**	**25.03**	**24.68**	**27.00**	**25.72**
	Tikhonov	26.75	24.89	23.63	26.38	23.95	26.86	25.47	23.66	25.68	26.53	24.54	25.45	22.27	24.83	24.29	26.25	25.09
	Total Variation	17.64	18.15	17.99	17.79	18.22	17.76	18.13	17.69	18.13	17.78	18.78	18.11	19.01	18.57	18.01	17.59	18.09

attribute of the noise-free point cloud models of *Greyc* dataset. Quantitative results in terms of PSNR are shown in Tab. 6.11, where the proposed algorithm shows the highest average PSNR value for all the noise levels $\sigma = 10$,15 ,20 and 25. With the

increase in the noise level, the proposed algorithm performs better than GLR and GTV, with an average PSNR increase of 0.49dB and 0.34dB, respectively.

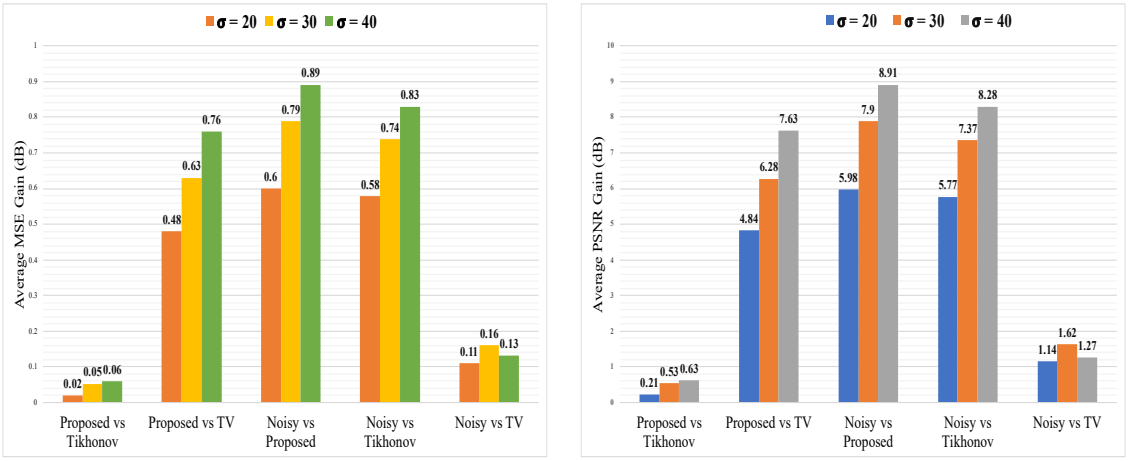
Table 6.11: Color denoising comparison for Gaussian noise $\sigma = 10, 15, 20$, and 25 with GLR-based [20] and GTV-based [13] in terms of PSNR and AET (s).

Model	$\sigma = 10$				$\sigma = 15$				$\sigma = 20$				$\sigma = 25$				AET (s)		
	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Noise	Proposed	GLR	GTV	Proposed	GLR	GTV
Asterix	28.38	31.76	**32.03**	31.61	24.94	**29.69**	29.66	29.53	22.53	**28.21**	28.10	27.56	20.68	**27.09**	26.12	27.05	**1.06**	5.20	211.00
Duck	28.53	**30.83**	30.40	30.60	25.20	**28.51**	28.42	28.14	22.71	**27.05**	26.89	26.35	20.90	**26.21**	25.68	25.04	**0.46**	1.70	58.00
Green_Dinosaur	28.14	33.19	33.28	**33.36**	24.60	31.21	**31.64**	31.31	22.13	30.13	30.30	**30.34**	20.23	29.37	28.52	**29.62**	**1.82**	11.20	389.00
Red_Horse	28.29	**32.79**	32.15	32.20	24.74	**30.59**	29.72	30.01	22.35	**29.07**	27.85	28.57	20.47	**28.25**	25.66	27.33	**3.48**	19.50	851.00
Average	28.30	**32.14**	31.87	31.94	24.85	**30.00**	29.86	29.75	22.44	**28.62**	28.29	28.20	20.61	**27.73**	26.495	27.26	**1.71**	9.40	377.25

Table 6.12: The computational cost of the proposed algorithm and MSGW [17] for different types of point clouds.

Gaussian Noise			0.2		0.3		0.4	
Methods		No of Points	Proposed	MSGW	Proposed	MSGW	Proposed	MSGW
Synthetic Point clouds	4arms_monstre	83063	1.58	4.49	1.52	3.86	1.89	3.96
	Asterix	45498	1.07	2.06	0.83	2.18	1.07	2.11
	Cable car	267828	6.31	10.44	7.18	12.10	6.77	14.88
	Dragon	101369	1.89	4.60	2.26	4.42	1.99	4.39
	Duck	19247	0.38	1.11	0.42	0.96	0.43	0.79
	Green dinasour	87999	1.69	3.95	1.69	3.77	1.63	3.74
	Green monster	204250	4.74	11.66	4.70	13.45	4.96	13.34
	Horse	137831	2.84	6.62	2.79	6.43	3.30	8.41
	Jaguar	130423	2.43	5.95	2.47	6.97	2.86	6.49
	Long dinasour	114005	2.09	5.17	2.10	5.97	2.52	5.20
	Mario	199281	4.20	10.28	4.39	11.54	4.43	11.59
	Mario car	216375	5.29	11.56	4.67	11.78	4.91	13.28
	Pokemon Ball	25427	0.57	1.25	0.54	1.20	0.54	1.28
	Rabbit	157534	3.80	8.08	3.33	8.04	3.47	8.13
	Red Horse	156671	3.50	7.79	3.28	9.33	3.35	9.35
	Statue	89256	1.83	4.04	1.76	4.74	2.03	5.05
	Average		**2.76**	6.50	**2.75**	6.87	**2.88**	7.05

			Proposed	MSGW
Real-world Point clouds	Arco_valentino	1530552	73.87	32.82
	Palazzo_carignano	4203962	159.08	85.25
LiDAR	MastinLake9_001	10001543	665.37	625.86



(a) (b)

Figure 6.20: (a): Average gain in MSE (dB) for the color denoising algorithm (b): Average gain in PSNR (dB) for the color denoising algorithm.

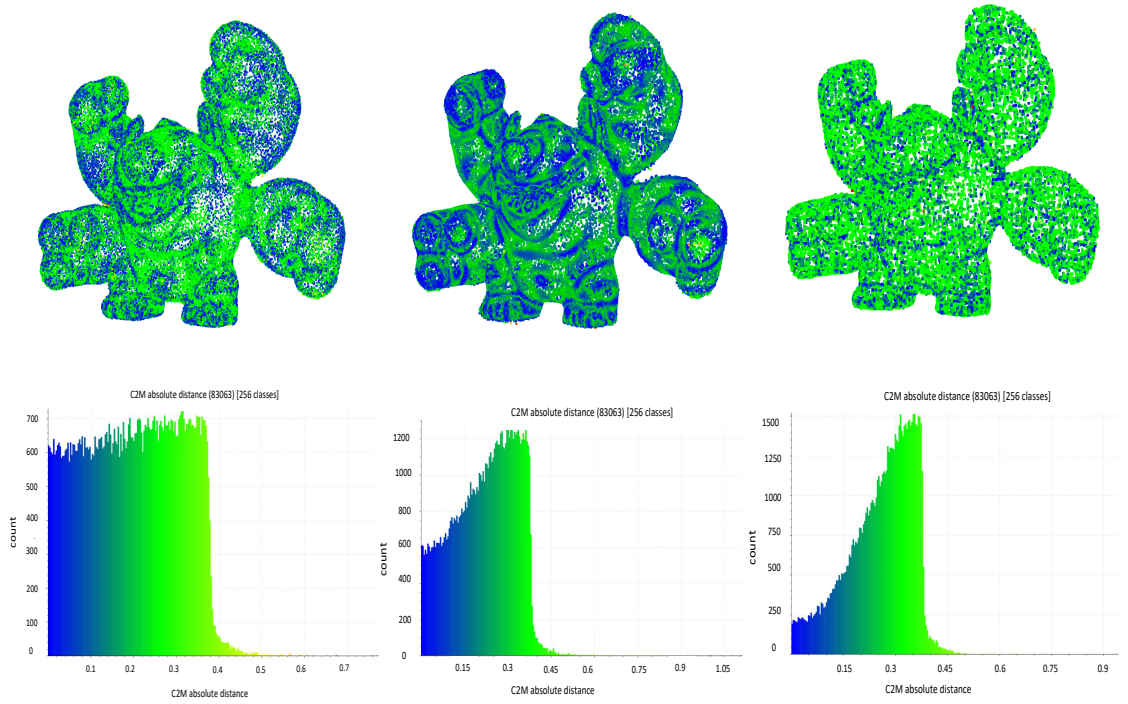


Figure 6.21: 4arms_model (noise level $\mu = 0$ and $\sigma = 0.4$) (a) C2M metric of denoised point cloud by proposed algorithm based on SGW using data-driven adaptive soft-thresholding (b) C2M metric of denoised point cloud using MSGW, and (c) C2M metric of denoised point cloud using IBR.

Besides PSNR, the average execution time (AET) per noise level for the proposed GLR and GTV algorithms is also shown in Tab. 6.11. The proposed method is almost six and over 200 times faster than GLR and GTV, respectively, as it is a non-iterative technique, which is computationally very cheap. AET has been measured on the machine with the exact specification and same MATLAB version as anticipated in 5.5.5.

Chapter 7

Conclusions and Future Work

In this thesis, we have presented a collection of techniques to perform point cloud denoising by exploiting the correlation between geometry and color attribute of a point cloud and construct a joint geometry/color k-NN graph. We tried to avoid the artifacts that appear in the denoised point cloud with the other state-of-the-art algorithms that consider the geometry information only to construct the graph.

Our first proposed algorithm is an efficient way of denoising a point cloud based on graph signal processing. It is based on the notion that in close proximity with a smooth surface, color is typically smooth. We take advantage of this correlation and encode it in a k-NN graph that can be used for several tasks such as denoising only the color, only the geometry, and both of them jointly by merely adapting the parameters to each denoising scenario. Denoising is then performed by employing graph-based Tikhonov regularization on graph signal. The convex optimization problem improves the smoothness of the graph signal defined on the graph. The fidelity term in the Tikhonov regularization drives the denoised points to move to their original observed location in the case of geometry denoising and their true color in color denoising. Graph gradient is used for the measurement of the smoothness of the graph.

Our second algorithm proposed a non-iterative scheme for point cloud denoising using SGW that takes advantage of positive correspondence between the color and geometry and performs denoising in the graph frequency domain. The idea is based on the framework where the joint graph compacts smooth graph signals' energy in low-frequency bands. A soft-thresholding is then applied to eliminate the noise from the spectral graph wavelet coefficients. In this proposed algorithm, prior knowledge of standard deviation works fine for the synthetic point cloud. Still, it is unknown for the real-world point cloud. We extend this framework and apply a data-driven adaptive soft-thresholding for denoising, in which prior information of standard deviation is not required.

7.1 Future work

As earlier explained, the state-of-art point cloud denoising approaches create the artifacts in the resulting denoised point cloud as they only consider the geometry information. Some other algorithms exploit the normal attribute for the point cloud denoising; however, these algorithms are not providing good results for the point cloud with the complex structure. The color attribute can be used to estimate the correct normal of a complex manifold, and then the normals can be used for the denoising problem.

As a future advancement, we can use the data-driven approach to define a new loss function that counters the color attribute of a point. In particular, a possible way is to estimate the actual position of a point corresponding to the color information and then compare it with the reference point in a ground-truth.

Nomenclature

Acronyms / Abbreviations

$CLOP$ Continuous Locally Optimal Projection

$CNNs$ Convolutional Neural Networks

CT Computed Tomography

$DEMs$ Digital Elevation Models

$DGCNN$ Dynamic Graph Convolutional Neural Network

$DTMs$ Digital Terrain Models

EM Expectation-Maximization

$FLOP$ Feature Locally Optimal Projection

GLR Graph Laplacian Regularizer

GMM Gaussian Mixture Model

GTV Graph Total Variation

IBR Iterative-based regularization

$IRLS$ Iterative Least Squares

KDE Kernel Density Estimation

$LDMM$ Low Dimensional Manifold Model

LOP Locally Optimal Projection

MLS Moving Least Squares

MSE Mean-squared error

$MSGW$ Manifold denoising using Spectral Graph Wavelets

$PDEs$ Partial Differential Equations

$PPMC$ Pearson Product-Moment Correlation

$PSNR$ Peak-to-signal ratio

ROR Radius outlier removal

$RPSM$ Robust denoising of piece-wise smooth manifolds

SGW Spectral Graph Wavelet Transform

ToF Time of flight

VG Voxel Grid

$WLOP$ Weighted Locally Optimal Projection

Bibliography

[1] URL: https://thinklucid.com/helios-time-of-flight-tof-camera/.

[2] A. Aldoma et al. "Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation". In: *IEEE Robotics Automation Magazine* 19.3 (2012), pp. 80–91. DOI: 10.1109/MRA.2012.2206675.

[3] Marc Alexa et al. "Computing and rendering point set surfaces". In: *IEEE Transactions on Visualization and Computer Graphics* 9.1 (2003), pp. 3–15.

[4] Nina Amenta and Yong Joo Kil. "Defining point-set surfaces". In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 264–270.

[5] Nicolas Aspert, Diego Santa-Cruz, and Touradj Ebrahimi. "Mesh: Measuring errors between surfaces using the Hausdorff distance". In: *Proceedings. IEEE International Conference on Multimedia and Expo*. Vol. 1. IEEE. 2002, pp. 705–708.

[6] Haim Avron et al. "$_1$-Sparse reconstruction of sharp point set surfaces". In: *ACM Transactions on Graphics* 29.5 (2010), p. 135.

[7] S. Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[8] BM Brown. "Statistical uses of the spatial median". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 45.1 (1983), pp. 25–30.

[9] Antoni Buades, Bartomeu 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)*. Vol. 2. IEEE. 2005, pp. 60–65.

[10] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. "Metro: measuring error on simplified surfaces". In: *Computer Graphics Forum*. Vol. 17. Wiley Online Library. 1998, pp. 167–174.

[11] Ulrich Clarenz, Martin Rumpf, and Alexandru Telea. *Fairing of point based surfaces*. IEEE, 2004.

[12] R Dennis Cook and Sanford Weisberg. *An introduction to regression graphics*. Vol. 405. John Wiley & Sons, 2009.

[13] Camille Couprie et al. “Dual constrained TV-based regularization on graphs”. In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1246–1273.

[14] M. C. Dal, P. Zanuttigh, and G. M. Cortelazzo. “Fusion of geometry and color information for scene segmentation”. In: *IEEE Journal of Selected Topics in Signal Processing* 6.5 (2012), pp. 505–521.

[15] Rupam Das and KB Shiva Kumar. “GeroSim: A simulation framework for gesture driven robotic arm control using Intel RealSense”. In: *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*. IEEE. 2016, pp. 1–5.

[16] S. Deutsch, Antonio Ortega, and Gerard Medioni. “Robust denoising of piece-wise smooth manifolds”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 2786–2790.

[17] Shay Deutsch, Antonio Ortega, and Gerard Medioni. “Manifold denoising based on spectral graph wavelets”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4673–4677.

[18] Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*. Vol. 23. Cambridge University Press, 2006.

[19] Julie Digne. “Similarity based filtering of point clouds”. In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2012, pp. 73–79.

[20] Chinthaka Dinesh, Gene Cheung, and Ivan V Bajić. “3D Point Cloud Color Denoising Using Convex Graph-Signal Smoothness Priors”. In: *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. 2019, pp. 1–6.

[21] Chinthaka Dinesh et al. “Fast 3d point cloud denoising via bipartite graph approximation & total variation”. In: *arXiv preprint arXiv:1804.10831* (2018).

[22] Chinthaka Dinesh et al. “Local 3D point cloud denoising via bipartite graph approximation & total variation”. In: *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. 2018, pp. 1–6.

[23] David L Donoho and Iain M Johnstone. “Adapting to unknown smoothness via wavelet shrinkage”. In: *Journal of the american statistical association* 90.432 (1995), pp. 1200–1224.

[24] David L Donoho and Jain M Johnstone. “Ideal spatial adaptation by wavelet shrinkage”. In: *biometrika* 81.3 (1994), pp. 425–455.

[25] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. "3D point cloud denoising via deep neural network based local surface estimation". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 8553–8557.

[26] David Eberly. "Distance between point and triangle in 3D". In: *Magic Software, http://www. magic-software. com/Documentation/pt3tri3. pdf* (1999).

[27] Abdallah El Chakik, Xavier Desquesnes, and Abderrahim Elmoataz. "Fast 3D surface reconstruction from point clouds using graph-based fronts propagation". In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 309–320.

[28] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. "Bilateral mesh denoising". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 950–953.

[29] Karel Fliegel et al. "3D Visual Content Datasets". In: *3D Visual Content Creation, Coding and Delivery*. Springer, 2019, pp. 299–325.

[30] Francis Galton. "On the Anthropometric Laboratory at the Late International Health Exhibition." In: *The Journal of the Anthropological Institute of Great Britain and Ireland* 14 (1885), pp. 205–221.

[31] X. Gao, W. Hu, and Z. Guo. "Graph-Based Point Cloud Denoising". In: *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*. IEEE. 2018, pp. 1–6.

[32] D Girardeau-Montaut. *CloudCompare*. 2016.

[33] Xuejian Gong, Ming Chen, and Xiaojun Yang. "Point cloud segmentation of 3D scattered parts sampled by RealSense". In: *2017 IEEE International Conference on Information and Automation (ICIA)*. IEEE. 2017, pp. 1–6.

[34] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Elsevier, 2011.

[35] Gaël Guennebaud, Marcel Germann, and Markus Gross. "Dynamic sampling and rendering of algebraic point set surfaces". In: *Computer Graphics Forum*. Vol. 27. Wiley Online Library. 2008, pp. 653–662.

[36] Gaël Guennebaud and Markus Gross. "Algebraic point set surfaces". In: *ACM Transactions on Graphics*. Vol. 26. ACM. 2007, p. 23.

[37] Paul Guerrero et al. "Pcpnet learning local shape properties from raw point clouds". In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 75–85.

[38] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. "Wavelets on graphs via spectral graph theory". In: *Applied and Computational Harmonic Analysis* 30.2 (2011), pp. 129–150.

[39] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. "Total Denoising: Unsupervised learning of 3D point cloud cleaning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 52–60.

[40] Matthias Hernandez, Jongmoo Choi, and Gerard Medioni. "Near laser-scan quality 3-D face reconstruction from a low-quality depth stream". In: *Image and Vision Computing* 36 (2015), pp. 61–69.

[41] Fengjun Hu et al. "Discrete Point Cloud Filtering And Searching Based On VGSO Algorithm." In: *Ecms*. 2013, pp. 850–856.

[42] Guofei Hu, Qunsheng Peng, and A Robin Forrest. "Mean shift denoising of point-sampled surfaces". In: *The Visual Computer* 22.3 (2006), pp. 147–157.

[43] Hui Huang et al. "Consolidation of unorganized point clouds for surface reconstruction". In: *ACM transactions on graphics (TOG)* 28.5 (2009), pp. 1–7.

[44] Hui Huang et al. "Edge-aware point set resampling". In: *ACM Transactions on Graphics* 32.1 (2013), p. 9.

[45] Hui Huang et al. "L1-medial skeleton of point cloud." In: *ACM Trans. Graph.* 32.4 (2013), pp. 65–1.

[46] Wenming Huang et al. "Algorithm for 3D point cloud denoising". In: *2009 Third International Conference on Genetic and Evolutionary Computing*. IEEE. 2009, pp. 574–577.

[47] Benjamin Huhle et al. "Robust non-local denoising of colored depth data". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2008, pp. 1–7.

[48] Muhammad Abeer Irfan and Enrico Magli. "3D POINT CLOUD DENOISING USING A JOINT GEOMETRY AND COLOR K-NN GRAPH". In: *accepted in Eusipco*. 2020.

[49] Muhammad Abeer Irfan and Enrico Magli. "3D Point Cloud Denoising Using a Joint Geometry and Color k-NN Graph". In: *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 585–589.

[50] Muhammad Abeer Irfan and Enrico Magli. "Exploiting color for graph-based 3D point cloud denoising". In: *submitted in Journal of Visual Communication & Image Representation* (2020).

[51] Muhammad Abeer Irfan and Enrico Magli. "Point cloud denoising using joint geometry/color graph wavelets". In: *accepted in SiPs*. IEEE. 2020.

[52] Philipp Jenke et al. "Bayesian point cloud reconstruction". In: *Computer Graphics Forum*. Vol. 25. 3. Wiley Online Library. 2006, pp. 379–388.

[53] Thouis R Jones, Fredo Durand, and Matthias Zwicker. "Normal improvement for point rendering". In: *IEEE Computer Graphics and Applications* 24.4 (2004), pp. 53–56.

[54] Thouis R Jones, Frédo Durand, and Mathieu Desbrun. "Non-iterative, feature-preserving mesh smoothing". In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 943–949.

[55] Lu Jun et al. "Point cloud registration algorithm based on NDT with variable size voxel". In: *2015 34th Chinese Control Conference (CCC)*. IEEE. 2015, pp. 3707–3712.

[56] Evangelos Kalogerakis et al. "Extracting lines of curvature from noisy point clouds". In: *Computer-Aided Design* 41.4 (2009), pp. 282–292.

[57] Khurram Kamal et al. "Performance assessment of Kinect as a sensor for pothole imaging and metrology". In: *International Journal of Pavement Engineering* 19.7 (2018), pp. 565–576.

[58] Jan Klein and Gabriel Zachmann. "Point cloud collision detection". In: *Computer Graphics Forum*. Vol. 23. 3. Wiley Online Library. 2004, pp. 567–576.

[59] Leif Kobbelt and Mario Botsch. "A survey of point-based techniques in computer graphics". In: *Computers & Graphics* 28.6 (2004), pp. 801–814.

[60] Ravikrishna Kolluri. "Provably good moving least squares". In: *ACM Transactions on Algorithms (TALG)* 4.2 (2008), pp. 1–25.

[61] Peter Lancaster and Kes Salkauskas. "Surfaces generated by moving least squares methods". In: *Mathematics of computation* 37.155 (1981), pp. 141–158.

[62] Jaromır Landa, David Procházka, Jiři Št'astn, et al. "Point cloud processing for smart systems". In: *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis* 61.7 (2013), pp. 2415–2421.

[63] Carsten Lange and Konrad Polthier. "Anisotropic smoothing of point sets". In: *Computer Aided Geometric Design* 22.7 (2005), pp. 680–692.

[64] Kai-Wah Lee and Wen-Ping Wang. "Feature-preserving mesh denoising via bilateral normal filtering". In: *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*. IEEE. 2005, 6–pp.

[65] Joseph Lee Rodgers and W Alan Nicewander. "Thirteen ways to look at the correlation coefficient". In: *The American Statistician* 42.1 (1988), pp. 59–66.

[66] David Levin. "Mesh-independent surface interpolation". In: *Geometric Modeling for Scientific Visualization*. Springer, 2004, pp. 37–49.

[67] David Levin. "The approximation power of moving least-squares". In: *Mathematics of computation* 67.224 (1998), pp. 1517–1531.

[68] Qiannan Li et al. "Classification of gait anomalies from Kinect". In: *The Visual Computer* 34.2 (2018), pp. 229–241.

[69] Bin Liao et al. "Efficient feature-preserving local projection operator for geometry reconstruction". In: *Computer-Aided Design* 45.5 (2013), pp. 861–874.

[70] Fagerman Technologies. LiDARUSA. URL: https://www.lidarusa.com/sample-data.html.

[71] Lars Linsen. *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek Technical Report, Faculty of Computer ..., 2001.

[72] Yaron Lipman, Daniel Cohen-Or, and David Levin. "Data-dependent MLS for faithful surface approximation". In: *Proceedings of the fifth Eurographics symposium on Geometry processing*. 2007, pp. 59–67.

[73] Yaron Lipman, Daniel Cohen-Or, and David Levin. "Error bounds and optimal neighborhoods for mls approximation". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. 2006, pp. 71–80.

[74] Yaron Lipman et al. "Parameterization-free projection for geometry reconstruction". In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), 22–es.

[75] Shengjun Liu, Kwan-Chung Chan, and Charlie CL Wang. "Iterative consolidation of unorganized point clouds". In: *IEEE computer graphics and applications* 32.3 (2011), pp. 70–83.

[76] Yong-Jin Liu et al. "3D model retrieval based on color + geometry signatures". In: *The Visual Computer* 28.1 (2012), pp. 75–86.

[77] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. "Partial difference operators on weighted graphs for image processing on surfaces and point clouds". In: *IEEE Transactions on Image Processing* 23.9 (2014), pp. 3896–3909.

[78] Shuang Ma et al. "Depth image denoising and key points extraction for manipulation plane detection". In: *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE. 2014, pp. 3315–3320.

[79] Eric Kenneth Matti and Stephan Nebiker. "Geometry and colour based classification of urban point cloud scenes using a supervised self-organizing map". In: *Photogrammetrie-Fernerkundung-Geoinformation* 2014 (2014), pp. 161–173.

[80] Boris Mederos, Luiz Velho, and Luiz Henrique de Figueiredo. "Robust smoothing of noisy point clouds". In: *Proc. SIAM Conference on Geometric Design and Computing*. Vol. 2004. 1. 2003, p. 2.

[81] Alex Miropolsky and Anath Fischer. "Reconstruction with 3D geometric bilateral filter". In: *Proceedings of the ninth ACM symposium on Solid modeling and applications*. 2004, pp. 225–229.

[82] Bradley Moorfield, Ralf Haeusler, and Reinhard Klette. "Bilateral filtering of 3D point clouds for refined 3D roadside reconstructions". In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2015, pp. 394–402.

[83] Philippos Mordohai and Gérard Medioni. "Tensor voting: a perceptual organization approach to computer vision and machine learning". In: *Synthesis Lectures on Image, Video, and Multimedia Processing* 2.1 (2006), pp. 1–136.

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

[85] Anass Nouri, Christophe Charrier, and Olivier Lézoray. "Technical report: Greyc 3D colored database". PhD thesis. Normandie Université, Unicaen, EnsiCaen, CNRS, GREYC UMR 6072, 2017.

[86] VE Oniga and C Chirilă. "Hausdorff distance for the differences calculation between 3D surfaces". In: *Journal of Geodesy and Cadastre RevCAD* 15 (2013), pp. 193–202.

[87] Stanley Osher, Zuoqiang Shi, and Wei Zhu. "Low dimensional manifold model for image processing". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1669–1690.

[88] Glenn I Ouchi et al. *Personal computers for scientists*. American chemical society, 1987.

[89] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. "Feature preserving point set surfaces based on non-linear kernel regression". In: *Computer Graphics Forum*. Vol. 28. Wiley Online Library. 2009, pp. 493–501.

[90] Jiahao Pang and Gene Cheung. "Graph Laplacian regularization for image denoising: Analysis in the continuous domain". In: *IEEE Transactions on Image Processing* 26.4 (2017), pp. 1770–1785.

[91] Sylvain Paris. "A gentle introduction to bilateral filtering and its applications". In: *ACM SIGGRAPH 2007 courses*. 2007, 3–es.

[92] Jaesik Park et al. "High quality depth map upsampling for 3d-tof cameras". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1623–1630.

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

[94] Mark Pauly, Leif Kobbelt, and Markus H Gross. "Multiresolution modeling of point-sampled geometry". In: *CS technical report* 378 (2002).

[95] N. Perraudin et al. "GSPBOX: A toolbox for signal processing on graphs". In: *arXiv preprint arXiv:1408.5781* (2014).

[96] N. Perraudin et al. "UNLocBoX: A MATLAB convex optimization toolbox for proximal-splitting methods". In: *arXiv preprint arXiv:1402.0779* (2014).

[97] George M Phillips. *Interpolation and approximation by polynomials*. Vol. 14. Springer Science & Business Media, 2003.

[98] Francesca Pistilli et al. "Learning Robust Graph-Convolutional Representations for Point Cloud Denoising". In: *IEEE Journal of Selected Topics in Signal Processing* (2020).

[99] Reinhold Preiner et al. "Continuous projection for fast L1 reconstruction." In: *ACM Trans. Graph.* 33.4 (2014), pp. 47–1.

[100] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[101] Marie-Julie Rakotosaona et al. "POINTCLEANNET: Learning to denoise and remove outliers from dense point clouds". In: *Computer Graphics Forum*. Wiley Online Library. 2019.

[102] Marie-Julie Rakotosaona et al. "Pointcleannet: Learning to denoise and remove outliers from dense point clouds". In: *Computer Graphics Forum*. Vol. 39. 1. Wiley Online Library. 2020, pp. 185–203.

[103] José Carlos Rangel et al. "Object recognition in noisy RGB-D data using GNG". In: *Pattern Analysis and Applications* 20.4 (2017), pp. 1061–1076.

[104] Nor Aziyatul Izni Mohd Rosli and Ahmad Ramli. "Mapping bootstrap error for bilateral smoothing on point set". In: *AIP Conference Proceedings*. Vol. 1605. 1. American Institute of Physics. 2014, pp. 149–154.

[105] Riccardo Roveri et al. "Pointpronets: Consolidation of point clouds with convolutional neural networks". In: *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 87–99.

[106] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point cloud library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1–4.

[107] Radu Bogdan Rusu et al. "Towards 3D object maps for autonomous household robots". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 3191–3198.

[108] Radu Bogdan Rusu et al. "Towards 3D point cloud based object maps for household environments". In: *Robotics and Autonomous Systems* 56.11 (2008), pp. 927–941.

[109] Marcelo Saval-Calvo et al. "A comparative study of downsampling techniques for non-rigid point set registration using color". In: *International Work-conference on the Interplay between Natural and Artificial Computation*. Springer. 2015, pp. 281–290.

[110] Oliver Schall, Alexander Belyaev, and H-P Seidel. "Robust filtering of noisy scattered point data". In: *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE. 2005, pp. 71–144.

[111] Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. "Adaptive feature-preserving non-local denoising of static and time-varying range data". In: *Computer-Aided Design* 40.6 (2008), pp. 701–707.

[112] 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)*. IEEE. 2015, pp. 1–4.

[113] Bao-Quan Shi, Jin Liang, and Qing Liu. "Adaptive simplification of point cloud using k-means clustering". In: *Computer-Aided Design* 43.8 (2011), pp. 910–922.

[114] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98.

[115] Francesco Luke Siena et al. "Utilising the Intel RealSense camera for measuring health outcomes in clinical research". In: *Journal of Medical Systems* 42.3 (2018), p. 53.

[116] David Slater and Glenn Healey. "Combining color and geometric information for the illumination invariant recognition of 3-D objects". In: *Proceedings of IEEE International Conference on Computer Vision*. IEEE. 1995, pp. 563–568.

[117] Christopher G Small. "A survey of multidimensional medians". In: *International Statistical Review/Revue Internationale de Statistique* (1990), pp. 263–277.

[118] Yujing Sun, Scott Schaefer, and Wenping Wang. "Denoising point sets via L0 minimization". In: *Computer Aided Geometric Design* 35 (2015), pp. 2–15.

[119] Richard Szeliski and David Tonnesen. "Surface modeling with oriented particle systems". In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, pp. 185–194.

[120] Vinh-Thong Ta, Abderrahim Elmoataz, and Olivier Lézoray. "Nonlocal PDEs-based morphology on weighted graphs for image and data processing". In: *IEEE transactions on Image Processing* 20.6 (2010), pp. 1504–1516.

[121] David Joseph Tan, Federico Tombari, and Nassir Navab. "Real-time accurate 3D head tracking and pose estimation with consumer RGB-D cameras". In: *International Journal of Computer Vision* 126.2-4 (2018), pp. 158–183.

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

[123] Gabriel Taubin. "Estimating the tensor of curvature of a surface from a polyhedral approximation". In: *Proceedings of IEEE International Conference on Computer Vision*. IEEE. 1995, pp. 902–907.

[124] Robert Tibshirani, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[125] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 839–846.

[126] F. Verdoja, D. Thomas, and A. Sugimoto. "Fast 3D point cloud segmentation using supervoxels with geometry and color for 3D scene understanding". In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2017, pp. 1285–1290.

[127] Michael Wand et al. "Processing and interactive editing of huge point clouds from 3D scanners". In: *Computers & Graphics* 32.2 (2008), pp. 204–220.

[128] Jianzhong Wang. *Geometric structure of high-dimensional data and dimensionality reduction*. Springer, 2012.

[129] Jun Wang et al. "Consolidation of low-quality point clouds from outdoor scenes". In: *Computer Graphics Forum*. Vol. 32. 5. Wiley Online Library. 2013, pp. 207–216.

[130] Qian Wang, Yi Tan, and Zhongya Mei. "Computational methods of acquisition and processing of 3D point cloud data for construction applications". In: *Archives of Computational Methods in Engineering* 27.2 (2020), pp. 479–499.

[131] Yue Wang et al. "Dynamic graph cnn for learning on point clouds". In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12.

[132] C Weihs. "Multivariate exploratory data analysis and graphics: A tutorial". In: *Journal of Chemometrics* 7.5 (1993), pp. 305–340.

[133] Chunxia Xiao et al. “A dynamic balanced flow for filtering point-sampled geometry”. In: *The Visual Computer* 22.3 (2006), pp. 210–219.

[134] Hui Xie, Kevin T McDonnell, and Hong Qin. “Surface reconstruction of noisy and defective data sets”. In: *IEEE Visualization 2004*. IEEE. 2004, pp. 259–266.

[135] Wenkai Xu et al. “MultiView-based hand posture recognition method based on point cloud”. In: *KSII Transactions on Internet and Information Systems (TIIS)* 9.7 (2015), pp. 2585–2598.

[136] Fu Yan and Zhai Jinlei. “Research on scattered points cloud denoising algorithm”. In: *2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE. 2015, pp. 1–5.

[137] Mao Ye et al. “Accurate 3d pose estimation from a single depth image”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 731–738.

[138] Faisal Zaman, Ya Ping Wong, and Boon Yian Ng. “Density-based denoising of point cloud”. In: *9th International Conference on Robotic, Vision, Signal Processing and Power Applications*. Springer. 2017, pp. 287–295.

[139] Jin Zeng, Gene Cheung, and Antonio Ortega. “Bipartite approximation for graph wavelet signal decomposition”. In: *IEEE Transactions on Signal Processing* 65.20 (2017), pp. 5466–5480.

[140] Jin Zeng et al. “3D point cloud denoising using graph laplacian regularization of a low dimensional manifold model”. In: *IEEE Transactions on Image Processing* (2019).

[141] Jin Zeng et al. “3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model”. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 3474–3489.

[142] Q. Zhan, Y. Liang, and Y. Xiao. “Color-based segmentation of point clouds”. In: *Laser Scanning* 38.3 (2009), pp. 155–161.

[143] Yinglong Zheng et al. “Guided point cloud denoising via sharp feature skeletons”. In: *The Visual Computer* 33.6-8 (2017), pp. 857–867.

[144] Michael Zollhöfer et al. “State of the Art on 3D Reconstruction with RGB-D Cameras”. In: *Computer Graphics Forum*. Vol. 37. Wiley Online Library. 2018, pp. 625–652.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.