symmetry

MDPI

Article

Three Dimensional Point Cloud Compression and Decompression Using Polynomials of Degree One

Ulfat Imdad [1], Muhammad Asif [1,*], Mirza Tahir Ahmad [2], Osama Sohaib [3], Muhammad Kashif Hanif [4] and Muhammad Hasanain Chaudary [5]

1 Department of Computer Science, National Textile University, Faisalabad 37600, Pakistan; ulfatimdad@gmail.com
2 Department of Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada; 81mta@queenus.ca
3 School of Information, Systems and Modeling,University of Technology, Sydney, NSW 2007, Australia; Osama.Sohaib@uts.edu.au
4 Department of Computer Science, Government College University, Faisalabad 38000, Pakistan; mkashifhanif@gcuf.edu.pk
5 Department of Computer Science, COMSATS University, Islamabad, Lahore Campus, Lahore 5400, Punjab, Pakistan; mhchaudary@cuilahore.edu.pk
* Correspondence: asif@ntu.edu.pk; Tel.: +92-333-644-4457

Received: 10 January 2019 ; Accepted: 6 February 2019 ; Published: 12 February 2019

Abstract: The availability of cheap depth range sensors has increased the use of an enormous amount of 3D information in hand-held and head-mounted devices. This has directed a large research community to optimize point cloud storage requirements by preserving the original structure of data with an acceptable attenuation rate. Point cloud compression algorithms were developed to occupy less storage space by focusing on features such as color, texture, and geometric information. In this work, we propose a novel lossy point cloud compression and decompression algorithm that optimizes storage space requirements by preserving geometric information of the scene. Segmentation is performed by using a region growing segmentation algorithm. The points under the boundary of the surfaces are discarded that can be recovered through the polynomial equations of degree one in the decompression phase. We have compared the proposed technique with existing techniques using publicly available datasets for indoor architectural scenes. The results show that the proposed novel technique outperformed all the techniques for compression rate and RMSE within an acceptable time scale.

Keywords: 3D point cloud; compression; decompression; polynomials

1. Introduction

There a lot of things that can be carried out when we talk about digital imagery. Ideally, in this multimedia world, it is important to ensure that you offer the best visualization of scenes. Simple digital cameras capture objects and their environmental information in such a way that each data point (also called a pixel) is represented in two dimensions x and y. An image provides assorted features about a scene like the shape of an object, its color and its texture. However, the most realistic type of data is three-dimensional (3D) data which is a Point Cloud, which contains another dimension of depth along-with length and width. Technically, a point cloud is a database or set of points in space containing data in 3D (x, y & z) coordinate system. We are living in, moving through and seeing the world in three dimensions and a point cloud simulates objects and their environment as we see them through our own eyes. The chief concern is that a point cloud is an accurate digital record of objects and scenes. Generally, point clouds are produced by 3D systems, like Microsoft Kinect, Light Detection and

Ranging (LiDAR) that actively acquire accurate and dense point clouds of the environment and provide useful data, which consists of 3D points with geometric and color information [1]. Microsoft Kinect hardware consists of an RGB-Camera and a depth sensor from which the prior detects red, green and blue colors of the object and then uses infrared invisible rays to measure the depth and create 3D point clouds. LiDAR systems used light waves from LASER to measure the depth of an object. It calculates how long it takes for the light to hit an object and reflect back. The distance is calculated using the velocity of light [2]. The images provide less information than point clouds because the former represent pixels in two dimensions x & y and the latter utilize three dimensions x, y & z for each point. This enormous information enriches point cloud usage in multiple fields such as robotics, 3D modeling of architecture, agriculture and industrial automation [3]. The use of point clouds is increasing day by day from 3D games to real-time robotic applications and it is difficult to process such types of massive information. A point cloud requires an additional storage space penalty to represent a scene as compared to images due to additional coordinate z. Therefore, the availability of a depth sensor in hand-held and head-mounted devices has increased the need for memory management for depth information on smart devices. This can also help to reduce the requirement of bandwidth for data transfer over the network.

In robotics real-time environment, bandwidth to transfer 3D data on network and storage space requirements demand to compress point clouds up to a maximum level and minimize memory requirements without disturbing the entire structure of objects or scenes. Various 3D compression techniques stand in the literature which can be classified as Lossy and Lossless. These are the characterizations used to explicate the distinction between two file compression formats. These terms can be elaborated on the basis of information lost. The lossless compression techniques do not involve the loss of information [4,5]. These are very applicable where we cannot tolerate the difference between original and reconstructed data. On the other hand, the lossy compression techniques are associated with minor loss of information. The file size is reduced in such techniques by permanently eliminating the certain information. The reconstructed data acquired from decompression phase is closely matching to the original one but not exactly. This compression results in loss of quality and reconstructed point cloud admit some errors [6,7]. So, the applications that utilize lossy compression techniques have to tolerate the difference between reconstructed and original data [8,9]. Several lossy compression techniques compress point clouds by using an efficient data structure like An octree [10,11]. Octree is a tree like data structure in which each internal node has exactly eight children. It divides a 3D point cloud by recursively subdividing it into eight octants. A fast variant of Gaussian Mixture Model and an Expectation-Maximization algorithm is used in [12] that replaces points with a set of Gaussian distribution. Delaunay triangulation for a given set of points P in a plane is a triangulation such that no point is inside the circumcircle of any triangle.

We propose a novel lossy point cloud compression and decompression technique in which a point cloud is first partitioned into a set of segments and planes are extracted from each segment. Then, each plane is represented by a polynomial equation of degree one that is also called a planer equation. This technique is very capable of reconstructing a surface by using polynomial equation coefficients and boundary conditions of a particular surface that is normally an α-concave hull. The proposed algorithm is evaluated based on three parameters compression ratio, Root-Mean-Square Error (RMSE) and time required for compression and decompression process on publicly available 3D point cloud datasets. The rest of the paper is organized as follows: Existing techniques are discussed in Section 2, methodology is expressed in Section 3, results are discussed in Sections 4 and 5. Section 6 concludes the work.

2. Related Work

Several point cloud compression approaches exist in the literature that optimize the storage space and bandwidth requirements. The lossy compression techniques [6,13–15] optimize storage requirements up to a significant level but reconstructed point cloud has structure, texture and color

errors. Lossless compression is expressed in [10] that utilizes double buffering Octree data structure to detect spatial and temporal changes in the stream of point clouds with the objective of optimizing memory requirements and processing latency to process real-time point cloud. The compression performance is measured based on two parameters, bits required to present each point and loss of quality is measured as the symmetric root-mean-square distance between the non-quantized and decompressed point cloud. The compression rate is increased in [10] by 10% as compared to state-of-the-art techniques. Another efficient implementation of Octree data structure is exercised in [11] to compress 3D point clouds data without any loss of information. An octree is the generalized form of binary trees and quad-trees, that store one and two-dimensional data respectively. Generally, each node in an octree represents the volume formed by a rectangular cuboid, but [11] enhanced that strategy and simplified to an axis aligned cube. In addition, they are focused on individual points compression, far from determining the geometry of surfaces. Normally, a Terrestrial laser scanner takes four bytes of memory to store each point in three dimensions and [11] used Octree in an efficient way and optimized memory requirement for each point up to two bytes. So, the whole point cloud is compressed up to 50%, which is a numerable contribution in lossless compression techniques.

A 3D mesh is the structural build of a 3D model consisting of polygons. 3D meshes use reference points in X, Y and Z axes to define shapes with height, width and depth. Polygons are a collection of vertices, edges and faces that define the shape of a polyhedral object. The faces usually consist of squares, rectangles (triangle mesh), quadrilateral or convex polygons. This small 3D meshes compression leads to point cloud memory optimization. A survey on mesh compression techniques is presented in [16] and another evaluation framework for 3D mesh compression is expressed in [17] in which authors claimed to have achieved a compression ratio 10 to 60%. A lossy compression model based on Hierarchical point clustering is exercised in [18] that focused on geometry of points. It generated Coarser Level of Detail (LOD) during point clustering phase. LOD hierarchy is further traversed in top-down order to update the geometry of each associated child. A real-time point cloud compression is described in [13] that decomposes the data chunks into voxels and then these are compressed individually. It supports both static and real-time point cloud compression in an online application. Voxel grid size can be controlled implicitly but compression ratio decreased with voxel size and it is difficult to choose the trade-off between compression ratio and locality.

Miguel Cazorla [12] contributions in the field of 3D point cloud lossy compression are prodigious. First, he proposed a data set composed of a set of 3D point clouds and also provided tools for comparing and evaluating compression methods based on compression level and distance parameters in [8]. These data sets are well organized and mainly categorized into two classes, real and synthetic types. Each category is further classified into three classes with respect to the structure of objects and texture available in the scene. Three classes are the high, medium and low structure in which high structured means more surfaces are planer and low means complex data structure. Furthermore, high texture means more text available in the scene and vice versa. The above discussed data sets [8] are published in a well-known journal and these are publicly available under the license and further utilized in Miguel Cazorla compression methods [7,12]. They mainly concentrated on geometric characteristics of objects and scenes to compress point clouds. First, it utilized the advantageous aspects of Delaunay triangles to compress 3D surfaces in [7]. The point cloud is first segmented into a set of planes and then points under each plane are computed. Further, these points are represented with Delaunay triangles that lead to compression of surfaces. That framework was tested on [8] datasets and performance was evaluated based on two parameters, Compression ratio and Root-Mean-Square-Error (RMSE) between original and reconstructed point cloud. Furthermore, it achieved 55%–85% compression ratio with 0.5 RMSE. The compression technique proposed in [7] is further extended in [12] by the same author in which it utilized Gaussian Mixture Models (GMMs) and an Expectation-Maximization algorithm to compress point cloud data. The points of interest are selected as a first step in [12]. These points are the best candidates to perform compression which are addressed by the planer patch extraction method as defined in [7]. Here GMMs are used to compress these points that hold the data as 3D

multivariate Gaussian distribution, which allows the reconstruction of the original data with a certain statistical reliability. The performance of [12] is evaluated by using [8] datasets and the compression ratio is achieved up to 85.25% with RMSE of 0.1.

We propose a novel lossy 3D point cloud compression method in which a point cloud is compressed in such a way that a set of points is represented by Polynomial equation of degree one that utilizes geometric information of scenes as highlighted by many researchers [19–23]. We used the framework illustrated in [24] for selection of points of interest which are the best candidates to represent a planer surface. Further, each plane is stored in the form of a polynomial equation of degree one coefficient instead of preserving points itself. We also preserve the boundary condition of each plane which helps us in the decompression phase. The proposed algorithm is tested on datasets described in [8] and these are utilized in previous techniques as a standard. Performance is evaluated based on three parameters, compression ratio, RMSE and compression and decompression time. The results are also compared with state-of-the-art lossy and lossless point cloud compression techniques. We achieved 89% compression ratio on average on highly structured data sets with lesser RMSE of 0.01.

3. Methodology

A raw point cloud may contain noise due to sensor limitations that result in unstable feature estimations. Therefore, reduction of noise becomes necessary, while it preserves features with its fine detail. A prior step to process a point cloud is to discard noisy points. These are normally referred to as Outliers. Therefore, noise removal techniques such as Voxel Grid filter, (Radius, Conditional & Statistical) Outlier Removal filters [25–27] are employed to reduce the effects of noise from a 3D point cloud. We use statistical outlier removal filter due to its lesser computational time and better results as compared to state-of-the-art filters [28]. In this filter, unwanted points are highlighted and discarded based on neighborhood information of each point. Mean distance and standard deviation of each point are computed to highlight noise and is tested for its being noisy or not. After detecting outliers, these points are trimmed out and filtered point cloud is provided for further processing.

3.1. Seeded Region Growing Segmentation

Segmentation is a process in which we divide a set of points into multiple segments. Here, the purpose of segmentation is to simplify the entire structure of a point cloud and to make it easier for further processing. In this process, a label is assigned to each point in such a way that every point of cloud belongs to any of its segments and this apportionment is done based on common attributes of points. As a result, a set of segments is received that covers all the points collectively. All the points in the same segment share abundant attributes like point normal, curvature value, angular distance, etc. Generally, segmentation is categorized into five classes: Edge-based, Model-based, Graph-based, Attribute-based and Region-based segmentation [29], as shown in the Figure 1.

The fundamental edge detection techniques are discussed in [30] and considerable advancements are obtained but the issues of sensitivity to noise and uneven density of point clouds may cause errors or inaccurate segmentation. Model-Based Segmentation techniques are efficient but the probability of inaccuracy arises when dealing with different point cloud sources [31].

Region growing based segmentation techniques use neighborhood information to merge those points in a region that hold similar properties and also isolate them from dissimilar properties regions. Region growing segmentation is further classified into two classes: unseeded and seeded region growing segmentation. The former is a top-down approach that considers the whole point cloud as a single region and makes segments by traversing it like a tree from top to bottom. The latter is a seeded region growing segmentation which is a bottom-up approach that selects a point as a seed and makes clusters of related points. This process iterates multiple times until all the points are to be the part of any segment. Two conceptions are involved in region growing process, intra and inter-segment

distance. The angular distance between points of the same segment is known as intra-segment distance and the distance between centroids of two different segments is known as inter-segment distance. In this research, the concept of lower intra-and higher inter-segment distance is populated to make surfaces more robust, as it is an indication of fruitful segmentation. The reason to use region growing segmentation in this research is that it partitions a point cloud based on the geometry of points and our proposed novel method is closely related to it, which also concentrates on the geometry of surfaces to present them in the form of polynomial equations.

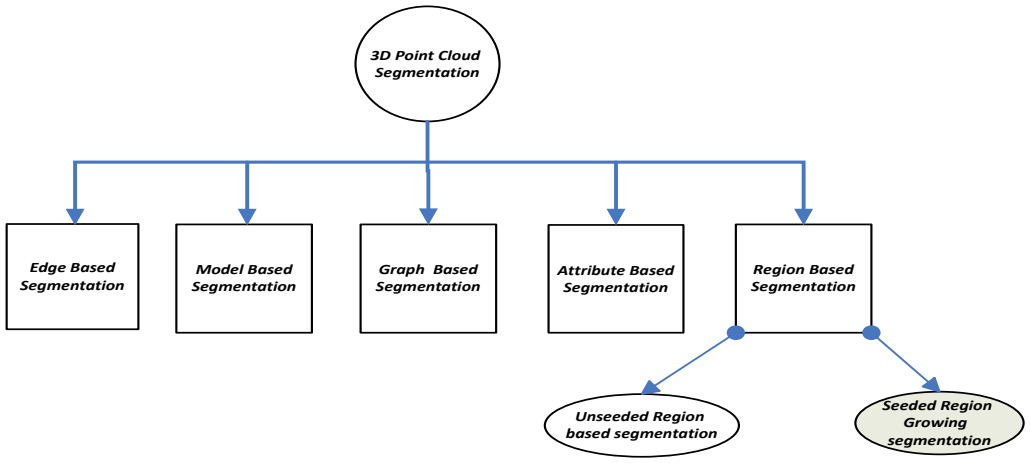


Figure 1. Segmentation types.

Here we demonstrate an overview of the seeded region growing process. The first step is the selection of a seed point which is considered as a base to form clusters. The region grows up on the selected seed point by testing and adding other neighborhood similar properties points that meet the defined criteria as shown in Figure 2. When the first region is matured considerably, another seed point is selected from the rest of the cloud and a new region is formed. This cluster formation process is repeated until all the points are included in any of this cloud segment. In the beginnings, the seed selection process was a manual activity, but nowadays it is automated, based on the curvature information of points. The geometric properties of point cloud i.e., curvature information and the normal vector of points is estimated with the help of Principal Component Analysis (PCA). The imperative step in curvature estimation is the computation of covariance matrix C which is estimated by Equation (1).

$$C = \frac{1}{k}\sum_{i=1}^{k}(P_i - \bar{P})^T.(P_i - \bar{P}) \quad (1)$$

$\bar{P}$ in Equation (1) can be calculated by Equation (2).

$$\overline{P} = \frac{1}{k}\sum_{i=1}^{k}(P_i) \quad (2)$$

$\bar{P}$ is the centroid (Arithmetic Mean) of the neighborhood points N_Pi of point P. Neighborhood points N_Pi of each point P are highlighted by k-d tree search algorithm. K-d tree algorithm is applied here due to several reasons illustrated below. It takes θ (log n) computation time that is the least time as compared to other search algorithms i.e., KNN that takes θ (nd) time. K-d tree data structure having the ability to represent points in k dimensions and it is broad into exercise to represent points in three-dimensional spaces for nearest neighbors search. Furthermore, at each level, all child nodes are split up against specific dimension and perpendicular hyperplane to the corresponding axis used. By this covariance matrix expressed in Equation (1), it is possible to compute eigenvector V (v_2, v_1 and v_0) and its corresponding eigenvalue λ (λ_0 , λ_1 and λ_2). Eigenvalues description is given below in Equation (3).

$$A \cdot v = \lambda \cdot v \tag{3}$$

In Equation (3), A is a 3×3 covariance matrix, v is a nonzero vector and λ is any scalar value.

$$\begin{aligned} A \cdot v - \lambda \cdot v &= 0 \\ (A - \lambda \cdot I)v &= 0 \\ |A - \lambda \cdot I| &= 0 \end{aligned} \tag{4}$$

Here V_o approximates the point normal for point P and the curvature is estimated by Equation (5)

$$\sigma(P) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{5}$$

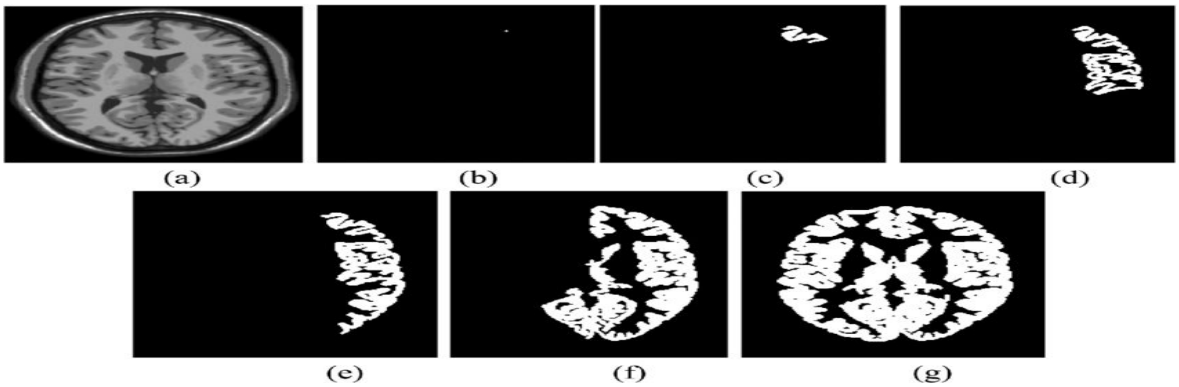


Figure 2. Seeded Region Growing Process. (**a**) sample point cloud; (**b**) one seed point selected; (**c**) 10 nearest neighbor points added; (**d**) 100 nearest neighbor points added; (**e**) 1000 nearest neighbor points added; (**f**) 2000 nearest neighbor points added; (**g**) all neighbor neighbor points added, region is complete.

A crucial parameter is the smoothness threshold that is the key element of the segmentation process. When the smoothness level is increased, spheres and other structures are considered as the part of planes and vice versa.

In the first stage, all points and their associated curvature γ values are added to a list, which is sorted by ascending order of γ. The point of minimum curvature value which therefore lies in the flattest region, is identified as the initial seed point, and is extracted from the point list.

The algorithm proceeds by finding a neighborhood support region for the seed, as follows:

1. k nearest neighbors of the seed point are identified from the point list.
2. Normal vectors of seed point and neighborhoods are compared. If a solid angle is found between the seed and neighbor point normal and the angle is less than a threshold value, then move to the next step.
3. Surface curvature of the seed and neighbor point are compared and the difference is noted. If the difference is less than a threshold value, then the neighbor point is removed from the point list and added to current seed region. This ensures that the curvature is consistent and relatively smooth within a region.
4. Repeat point 2 & 3 for all neighborhood points.

The above process iterates by extracting the next seed point with minimum γ value from the point list, and repeating steps 1 to 3 until the point list is empty. Finally, output of the algorithm is a set of regions.

3.2. Polynomial Equations of Degree One

Here, we unfold the novelty of our proposed technique and discuss in detail how the polynomial equations of degree one are used to compress a point cloud. The output of the segmentation process discussed in Section 3.1 is a set of regions and all points within each region are geometrically and closely related to each other. In the next step we extract planes from each region. A plane is a flat surface that extends infinitely far and it can be founded by a point and a vector perpendicular to the point. Various techniques are available to extract planes from a set of points. In this research, we have utilized Sample Consensus (SAC) method [1] which is a standard technique for plane extraction stand in Point Cloud Library (PCL). Here we illustrate the polynomial equation and its coefficients.

Suppose P_1 is a known point on a plane having $(x_1, y_1$ & $z_1)$ coordinates and $\vec{n}$ is a normal vector of the plane. P is another point on a surface having coordinates $(x, y$ & $z)$. A vector connecting from P to P_1 is described in Equation (6).

$$\vec{P} - \vec{P_1} = (x - x_1, y - y_1, z - z_1) \tag{6}$$

Vector $(\vec{P} - \vec{P_1})$ and $\vec{n}$ are perpendicular to each other and the dot product of two perpendicular vectors is always zero as shown in the Equation (7).

$$\vec{n}.(\vec{P} - \vec{P_1}) = 0 \tag{7}$$

Put the value of Equation (6) in (7) and calculate dot product:

$$\begin{aligned} \vec{n}.(x - x_1, y - y_1, z - z_1) &= 0 \\ (a, b, c).(x - x_1, y - y_1, z - z_1) &= 0 \end{aligned} \tag{8}$$

After solving Equation (8) we found:

$$ax_1 + by_1 + cz_1 + d = 0 \tag{9}$$

The resultant equation of the above discussed scenario Equation (9) is a polynomial equation of degree one that represents a planer surface. It is also called a planer equation in which x, y & z are the coordinates of points belonging to that planer surface and a, b, c & d are coefficients of equations. These coefficients are helpful to test any 3D point, either it belongs to this plane or not. A planer surface may have 100,000 points that can be represented with only four coordinates a, b, c & d of Equation (9) and this is the novelty of our technique. This novel approach increases compression rate dramatically and a Megabyte 3D point cloud can be stored with only a few bytes.

3.3. Compression

Our proposed technique is shown in the Algorithm 1 and Figure 3 that compresses a point cloud by converting it into a data structure which is composed of plane coefficients and boundary conditions of each planer surface. This data structure is significantly smaller than the original point cloud because each segment that may contain thousands of 3D points is represented by four plane coefficients and an α-concave hull of that surface. Boundary conditions are really helpful in the decompression phase to reconstruct the point cloud. α-Concave hull is generalization of convex hull [32], where α parameter regulates the smoothness level of the constructed hull on a data set. Furthermore, it is a concave hull with angular constrained under the minimum area condition. The illustration of the concave hull is depicted in Figure 4 in which the Figure 4a is a simple point cloud surface that requires 3250 KB of memory to store and the Figure 4b is an α-concave hull of that surface which requires only 62 KB of memory to store. In the second stage, we indicate and discard the smallest λ coordinates that further decreases the compressed file size. Finally, compressed data structure preserves only four coordinates

and two-dimensional boundary of each surface. This technique optimizes the memory requirements to store and bandwidth requirements to transfer a point cloud on the network.

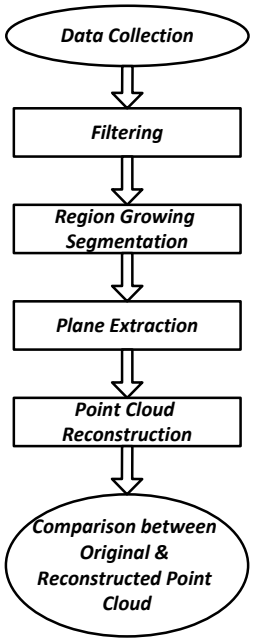
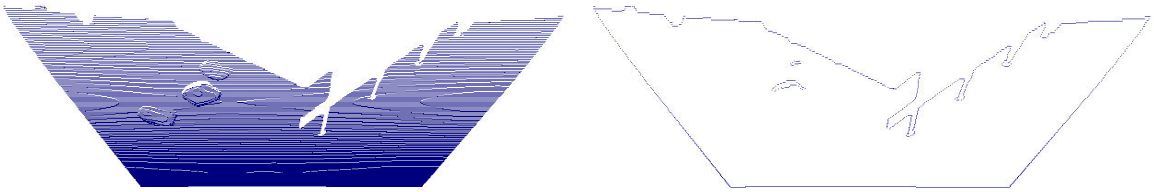


Figure 3. Compression and decompression methodology.



(**a**) (**b**)

Figure 4. A surface and its concave hull. (**a**) A simple planer surface; (**b**) Concave hull of a surface.

Algorithm 1 Point Cloud Compression

Require: Point Cloud: $\{PC\}$
Ensure: Final Planer file: $\{Planer\}$ is empty.
1: Filter PC by applying statistical outlier removal.
2: Compute Point Normals $\{N\}$
3: Apply Region Growing Segmentation that generate List of Regions $\{R\}$
4: **for each** Region r in R **do**
5: Apply SAC method and Extract Planes $\{P\}$
6: **for each** Plane p in P **do**
7: Compute Plane Coefficients.
8: Estimate boundary/Concave-hull of surface.
9: Discard minimum λ coordinate.
10: Preserve $2D$ boundary and coefficients.
11: **end for**
12: **end for**

3.4. Decompression

Decompression is the process of reconstructing a point cloud from the compressed file. In our proposed technique, decompression is a simple two-step process that uses a compressed data structure as input. The first step is to generate random points under the boundary conditions that were extracted in the compression phase as shown in Algorithm 2. We also care about the number of points generated under each surface and these are equal in numbers to the original surface points. These generated points have two coordinates, x and y. The third coordinate that shows the depth of each point is set to zero and polynomial equation of degree one (9) is used to compute the actual value of z coordinate. These reconstructed surfaces are merged to build a complete point cloud. After decompression, we compare original and reconstructed point cloud and measure the performance based on two parameters: compression ratio and RMSE. Compression ratio depicts how much point cloud is compressed, which is calculated by the Equation (10).

$$Ratio = \frac{(Original - Compressed)FileSize}{OriginalFileSize} * 100 \tag{10}$$

RMS Error is an important parameter to measure the performance of the proposed algorithm illustrated below. We compute RMS error on average from original to reconstructed point cloud and vice versa. The Squared Euclidean distance is computed from a point in the first cloud to the nearest neighbor point in the second cloud and this process is repeated for all points.

Algorithm 2 Point Cloud Decompression

Require: $\forall$ Planer files list: $\{Planer\}$
Require: Filtered Point Cloud $\{FPC\}$
Ensure: Reconstructed Point Cloud: $\{RPC\}$ is empty

```
for each Plane p in Planer do
    Generate {Points} under 2D Concave-hull.
    for each point in Points do
        Compute missing coordinate by Polynomial Equation of degree one.
    end for
    Write points to RPC.
end for
```

4. Results

Our proposed technique is tested on 3D point cloud datasets which are described in [8] and also used in state-of-the-art techniques [7,12] to measure their algorithms' performance. These are freely available under the license. These are well organized and categorical. Such properties make it favorable to evaluate our proposed algorithm and a fair comparison with other techniques. These data sets are tested in windows environment using Microsoft Visual studio 2010 framework on 2.5 Gigahertz quad core processor with 8 Gigabytes RAM. We write our code in C++ language and utilized Point Cloud Library (PCL) to process 3D point clouds. The Point Cloud Library is a large scale, open project [1] for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. It provides open access to utilize existing algorithms, and to modify and build our own algorithms to process point clouds.

Before directly jumping into results collection, we pay special attention to parameter tuning, because parameters are the core key to computing accurate results. Here, we discuss the most vital parameter smoothness threshold and its effects on the performance of the proposed algorithm. It controls the segmentation process and clusters formation, because points inclusion in a segment is

heavily dependent on a sold angle between seed and neighborhood point normal that is discussed in detail in Section 3.1. We start from a small threshold value and gradually increase it and summarize the effects of multiple data sets on average in the Figure 5. When the smoothness threshold is too much small, it makes 100 clusters represented as Total Clusters (TC) with a red color line in the Figure 5. The smoothness threshold is an entry point which allows a point to be part of a segment, if the neighborhood point normal deviation from seed point is less than the threshold value then the point is included in the current segment, otherwise it makes another segment of related points. A tight threshold causes a lower number of points per segment and makes a lot of segments but the planes per segment are very low near one because plane extraction also depends on the geometry of points. Numerous segments lead to a lower compression ratio but the error rate between original and reconstructed surface is very low as shown in the Figure 5. When the smoothness threshold increases, it increases the size of the segment and allows many points to be part of a segment, which leads to a lower number of total segments. The compression ratio is increased and the error rate is also increased; it is difficult to find a trade off between compression ratio and error rate. So, the smoothness threshold is flexible and varies according to the application's requirements; for analysis purposes, we set it to 3 Degree at which a reasonable compression ratio is achieved within an acceptable error rate.

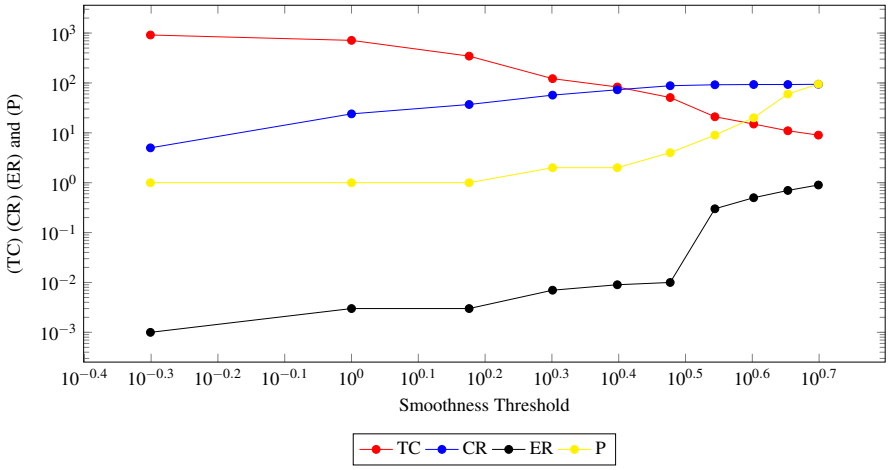
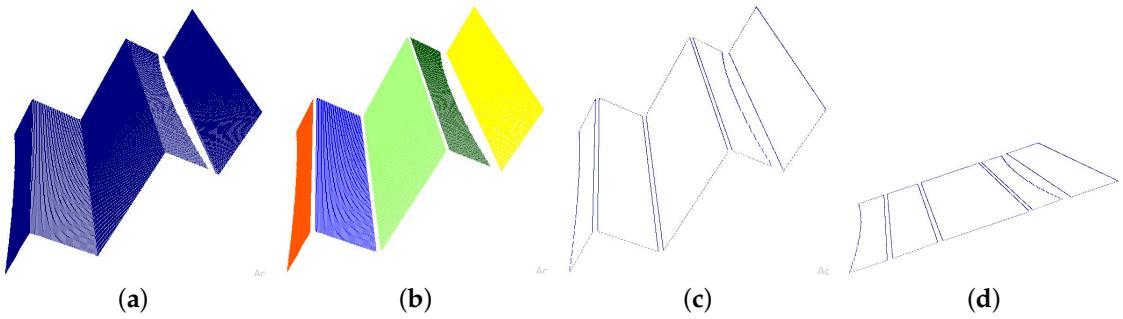


Figure 5. Total Clusters (TC), Compression Ratio (CR), Error Rate (ER) and Number of Planes per cluster (P).

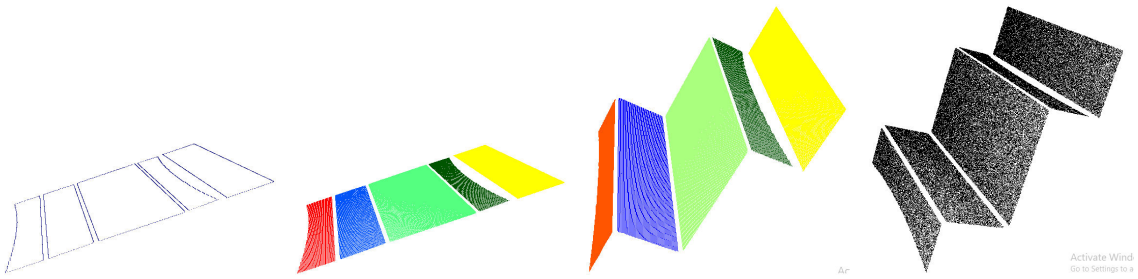
We test our proposed technique on different classes of data sets which are classified with respect to structural complexity and compare our results with state-of-the-art lossy and lossless techniques discussed in literature review Section 2. A very high structure point cloud compression and decompression process is depicted in Figures 6 and 7, results are shown in the Table 1 and the illustration is given below. A Stair steps point cloud is shown in Figure 6a that has following properties: 148,039 three-dimensional points and 2719.29 KB memory to store on disk in binary format. Unnecessary points are trimmed out and set of segments is presented with different colors in the Figure 6b that have 147,309 three-dimensional points and take 2680 KB of memory. A three-dimensional α-concave hull of segments is computed and presented in the Figure 6c that has only 4210 three-dimensional points and takes 170 KB of memory. We have further compressed it by discarding the minimum variance coordinate of each segment and preserving two-dimensional boundary conditions of each segment in a simple text file which contains 4210 two-dimensional points and takes 130.98 KB memory. The planer coefficients are also in our accountability and prepared in a separate file, that preserves four coefficients a, b, c and d and the total number of points under each plane for all planer surfaces that take only 2 KB of memory. A resultant compressed file that comprises the two-dimensional boundary of each segment and surface coefficients requires only 132.98 KB of memory and compression ratio is computed by Equation (10). We compress the above discussed

point cloud and optimize memory requirements up to 95.11% which is a major contribution to our proposed techniques in the field of lossy point cloud compression. The compressed data file can be traveled on the network and requires only 5% bandwidth as compared to the original. The promising capability of our proposed technique is that it can be decompressed anywhere, independent of the compression process. The decompression phase is depicted in the Figure 7 and the reconstructed point cloud resembles the original. The root-mean-square error between original and reconstructed point cloud is 0.0037, as shown in the Table 1.



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

Figure 6. Point cloud compression. (**a**) A 3D point cloud; (**b**) Point cloud segments; (**c**) 3D concave hull; (**d**) 2D concave hull.



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

Figure 7. Point cloud decompression. (**a**) 2D Boundary of a point cloud; (**b**) Reconstructed 2D segments; (**c**) Reconstructed 3D segments; (**d**) Reconstructed 3D point cloud.

Our proposed technique is tested on four classes of datasets which can be named as high, medium, low and complex structured point clouds. We test hundreds of datasets of each class and measure compression ratio, RMSE and processing time on average of each class. It is difficult to present all the properties of each point cloud. However, we present all properties of five point clouds of each class in Table 2. The efficiency of the algorithm on high structured point clouds is optimal in all three parameters i.e., compression ratio 89% achieved, RMSE 0.0031 and processing time 1.0812 milli seconds as shown in the Figure 8 and Table 2. We also compare our results with state-of-the-art compression techniques as shown in Figure 9 and Table 3. Several techniques are available to compress point clouds as discussed in the literature and some well-known methods are presented in Figure 9 listed below, Burrows 1994 [4], Octree-based point cloud compression 2006 [10], One billion points compression based on octree data structure 2013 [11], Morell 2014 with k = 5 and 1 [8], Miguel Cazorle 2016 that utilized Delaunay Triangles (DT2016) [7] and Miguel Cazorle 2018 that utilized Gaussian Mixture Model (GMMs2018) [12]. In the late 1990s, Burrows developed a lossless compression method [4] that can compress point cloud up to 30%. Many improvements are done in compression algorithms with the passage of time. Efficient data structures i.e., Octree, Delaunay Triangles and Gaussian Mixture Model are utilized and compress point clouds up to 85.25% with 0.1 RMSE [12]. Our proposed technique increases compression ratio by 3.75% on highly structured point clouds and compress up to 89.15%

with 0.0031 RMSE and 1.0812 ms processing time per point. This increased compression ratio leads to memory optimization and bandwidth requirements to transfer point cloud data sets on the network.

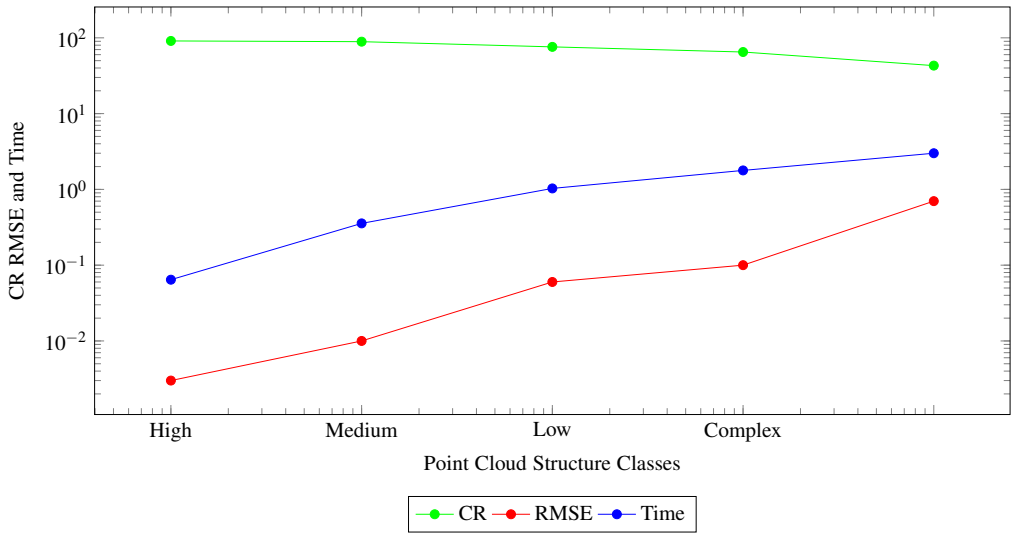


Figure 8. Compression ratio (CR), RMSE, processing time.

Table 1. A sample point cloud compression and decompression results.

Description	Points	Measure
Original Point Cloud Size Figure 6a	148,039	2719.29 KB
Segments Size Figure 6b	147,309	2680 KB
3D Concave Hull Size Figure 6c	4210	170 KB
2D Concave Hull Size Figure 6d	4210	130.98 KB
Planer Coefficients File Size	20	2 KB
Compressed File Size	4220	132.98 KB
Decompressed File Size	147,309	2680 KB
Compression Time Per Point		0.93 millisecond
Decompression Time Per Point		1.81 millisecond
Compression Ratio		95.11%
Root Mean Square Error		0.0037

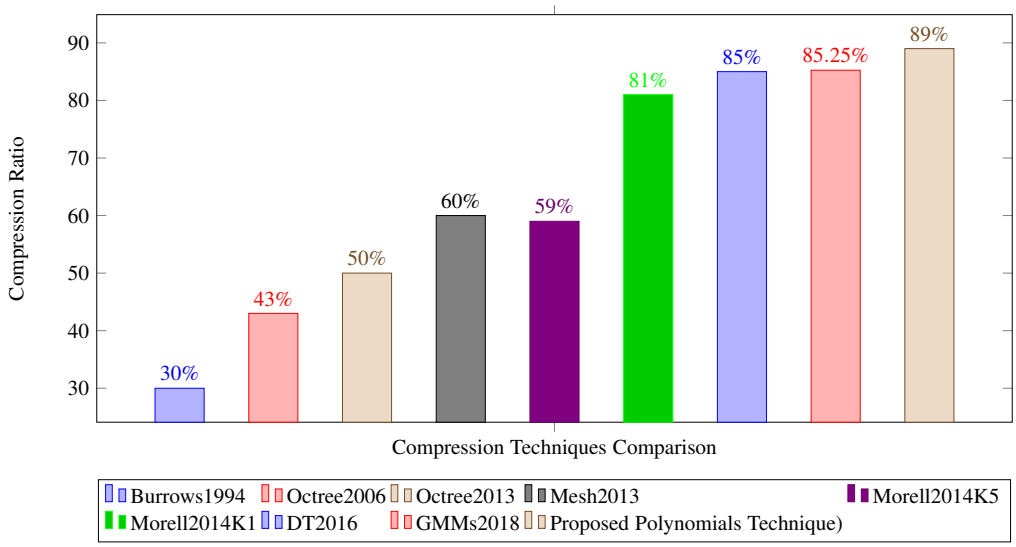


Figure 9. Comparison with state-of-the-art techniques.

Table 2. Performance measures of multiples classes of point clouds.

Category of Dataset	Name of Point Cloud	Size (KB)	Number of Points	Compressed File Size (KB)	Compression Ratio	Compression Time Per Point (ms)	Decompression Time Per Point (ms)	RMSE Per Point
High Structure	hs_1	4664	298,463	165.10	94.46%	0.5963	0.6423	0.0042
	hs_2	2971	190,073	264.71	91.09%	0.8534	2.8512	0.0011
	hs_3	3565	228,141	391.08	89.03%	1.2924	2.3501	0.0007
	hs_4	4	210	1.08	72.79%	1.9056	1.2543	0.0058
	hs_5	2720	174,023	94.64	96.41%	0.7594	4.1685	0.0037
	Average				**89.15%**	**1.0812 ms**	**2.1038 ms**	**0.0031**
Medium Structure	ms_1	1151	73,637	290.51	74.76%	1.8543	2.1576	0.2149
	ms_2	2376	152,042	325.03	86.32%	2.5532	3.7833	0.0423
	ms_3	2429	155,430	701.49	71.12%	1.3321	1.5432	0.3568
	ms_4	4801	307,200	771.04	83.94%	3.5368	3.2163	0.0762
	ms_5	1186	75,875	312.39	73.66%	4.6215	4.8933	0.2254
	Average				77.96%	2.7795 ms	3.1187 ms	0.1831
Low Structure	ls_1	4208	269,268	1037.69	75.34%	4.2364	4.2643	0.2013
	ls_2	1263	80,813	457.96	63.74%	3.1634	3.5276	1.0433
	ls_3	905	57,905	396.66	56.17%	2.8603	2.8954	0.3568
	ls_4	1367	87,455	419.12	69.34%	3.6953	3.5764	0.2762
	ls_5	1782	114,029	894.92	49.78%	3.1795	2.8654	0.0124
	Average				62.87%	3.4269 ms	3.4258 ms	0.3780

Table 3. Proposed technique comparison with state-of-the-art techniques.

Evaluation Parameter	Burrows [4]	Octree [10]	Octree [11]	Morell k5 [8]	Morell k1 [8]	Dalunay Triangles [7]	GMMs [12]	Proposed Method
Compression Ratio	30%	43%	50%	59%	81%	85%	85.25 %	89.15%

5. Discussions

Our proposed lossy three-dimensional point cloud compression technique achieves higher compression ratio with lower error rate as compared to state-of-the-art techniques on very high and high structured point cloud datasets as shown in Figure 8. However, when data set complexity increases, algorithm performance declines. This is because highly structured datasets contain large planer surfaces which are presented in a compressed file with only four coefficients of the polynomial equation of degree one and few two-dimensional boundary condition points. On the other hand, less structured or very complex datasets contain small planer surfaces with a few points of each; these are also presented with four coefficients and boundary condition points as large surfaces, but these small planer surfaces are very large in numbers and overhead cost of all these surfaces leads to reduction of compression ratio. These results were expected, but most of the time we dealt with structured point clouds i.e., open door streets, buildings or indoor object point clouds are well-structured and our proposed algorithm performs well on these point clouds; in rare cases we have to deal with very complex datasets, i.e., trees with leaves or embroidery cloths.

6. Conclusions

We proposed a lossy 3D point cloud compression and decompression technique. The proposed method compresses point clouds by representing implicit surfaces with polynomial equations of degree one. The compression phase consists of the segmentation process in which a point cloud is partitioned into smaller clusters based on geometric information of points. Furthermore, planes are extracted and boundary conditions are stored in a compressed file for each segment. The decompression phase reconstructs surfaces by generating two-dimensional points under boundary conditions and computes the missing third coordinate with planer equation. Our proposed algorithm is tested on multiple classes of datasets i.e., high, medium, low structured and very complex datasets and performance is evaluated based on compression ratio and root-mean-square error. We have compared the results with state-of-the-art lossless and lossy compression techniques.

Our proposed algorithm performs well on highly structured datasets and achieves compression ratio up to a significant level of 89% with 0.003 RMSE within the acceptable time scale of 0.0643 milliseconds processing time per point on average. However, the proposed method fails to deal with complex point clouds. So, it is recommended to use the proposed algorithm for highly structured datasets. The proposed technique does not deal with objects of color information. In the future, we will enhance the proposed technique to higher degree polynomials to cater to non-planer surfaces; dealing with color information is also a part of our future work.

Author Contributions: Conceptualization, U.I.; Formal analysis, U.I., M.T.A.; Methodology, U.I., M.A., M.T.A., O.S. and M.H.C.; Project administration, M.A.; Resources, M.T.A.; Software, U.I.; Supervision, M.A.; Validation, U.I.; Writing-original draft, U.I., M.T.A. and M.A.; Writing-review and editing, M.A., M.T.A., O.S., M.H.C. and M.K.H.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rusu, R.B.; Cousins, S. 3D is here: Point cloud library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 1–4.
2. Hämmerle, M.; Höfle, B.; Fuchs, J.; Schröder-Ritzrau, A.; Vollweiler, N.; Frank, N. Comparison of kinect and terrestrial lidar capturing natural karst cave 3-d objects. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 1896–1900.
3. Wang, X.V.; Wang, L.; Mohammed, A.; Givehchi, M. Ubiquitous manufacturing system based on Cloud: A robotics application. *Robot. Comput.-Integr. Manuf.* **2017**, *45*, 116–125.
4. Burrows, M.; Wheeler, D.J. *A Block-Sorting Lossless Data Compression Algorithm*; Systems Research Center: Palo Alto, CA, USA, 1994.

5. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **1977**, *23*, 337–343.
6. De Queiroz, R.L.; Chou, P.A. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Trans. Image Process.* **2016**, *25*, 3947–3956.
7. Navarrete, J.; Morell, V.; Cazorla, M.; Viejo, D.; García-Rodríguez, J.; Orts-Escolano, S. 3DCOMET: 3D compression methods test dataset. *Robot. Auton. Syst.* **2016**, *75*, 550–557.
8. Morell, V.; Orts, S.; Cazorla, M.; Garcia-Rodriguez, J. Geometric 3D point cloud compression. *Pattern Recognit. Lett.* **2014**, *50*, 55–62.
9. Zhang, C.; Florêncio, D.; Loop, C. Point cloud attribute compression with graph transform. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 2066–2070.
10. Schnabel, R.; Klein, R. Octree-based Point-Cloud Compression. *Spbg* **2006**, *6*, 111–120.
11. Elseberg, J.; Borrmann, D.; Nüchter, A. One billion points in the cloud–an octree for efficient processing of 3D laser scans. *ISPRS J. Photogramm. Remote Sens.* **2013**, *76*, 76–88.
12. Navarrete, J.; Viejo, D.; Cazorla, M. Compression and registration of 3D point clouds using GMMs. *Pattern Recognit. Lett.* **2018**, *110*, 8–15.
13. Golla, T.; Klein, R. Real-time point cloud compression. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5087–5092.
14. Ahn, J.K.; Lee, K.Y.; Sim, J.Y.; Kim, C.S. Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains. *IEEE J. Sel. Top. Signal Process.* **2015**, *9*, 422–434.
15. Klima, O.; Barina, D.; Kleparnik, P.; Zemcik, P.; Chromy, A.; Spanel, M. Lossy Compression of 3D Statistical Shape and Intensity Models of Femoral Bones Using JPEG 2000. *IFAC-PapersOnLine* **2016**, *49*, 115–120.
16. Peng, J.; Kim, C.S.; Kuo, C.C.J. Technologies for 3D mesh compression: A survey. *J. Vis. Commun. Image Represent.* **2005**, *16*, 688–733.
17. BerjóN, D.; MoráN, F.; Manjunatha, S. Objective and subjective evaluation of static 3D mesh compression. *Signal Process. Image Commun.* **2013**, *28*, 181–195.
18. Fan, Y.; Huang, Y.; Peng, J. Point cloud compression based on hierarchical point clustering. In Proceedings of the 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Kaohsiung, Taiwan, 29 October–1 November 2013, pp. 1–7.
19. Birdal, T.; Busam, B.; Navab, N.; Ilic, S.; Sturm, P. A Minimalist Approach to Type-Agnostic Detection of Quadrics in Point Clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3530–3540.
20. Suwajanakorn, S.; Snavely, N.; Tompson, J.; Norouzi, M. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *arXiv* **2018**, arXiv:1807.03146.
21. Zeng, A.; Song, S.; Nießner, M.; Fisher, M.; Xiao, J.; Funkhouser, T. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 199–208.
22. Li, Y.; Ma, Y.; Tao, Y.; Hou, Z. Innovative Methodology of On-Line Point Cloud Data Compression for Free-Form Surface Scanning Measurement. *Appl. Sci.* **2018**, *8*, 2556.
23. Zhou, X.; Lu, Y.; Yan, X.; Wang, Y.; Liu, L. Lossless and Efficient Polynomial-Based Secret Image Sharing with Reduced Shadow Size. *Symmetry* **2018**, *10*, 249.
24. Ahmed, M.T.; Mohamad, M.; Marshall, J.A.; Greenspan, M. Registration of noisy point clouds using virtual interest points. In Proceedings of the 2015 12th Conference on Computer and Robot Vision (CRV), Halifax, NS, Canada, 3–5 June 2015; pp. 31–38.
25. Whelan, T.; Kaess, M.; Fallon, M.; Johannsson, H.; Leonard, J.; McDonald, J. *Kintinuous: Spatially Extended Kinectfusion*; MIT Computer Science and Artificial Intelligence Lab: Cambridge, MA, USA, 2012.
26. Pirovano, M. Kinfu—An Open Source Implementation of Kinect Fusion + Case Study: Implementing a 3D Scanner with PCL. Available online: https://homes.di.unimi.it/borghese/Teaching/IntelligentSystems/Documents/PirovanoMichele-VisualReconstructionReport.pdf (accessed on 30 January 2019).
27. Hsieh, C.T. An efficient development of 3D surface registration by Point Cloud Library (PCL). In Proceedings of the 2012 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), Taipei, Taiwan, 4–7 November 2012; pp. 729–734.

28. Schall, O.; Belyaev, A.; Seidel, H.P. Robust filtering of noisy scattered point data. In Proceedings of the Eurographics/IEEE VGTC Symposium Point-Based Graphics, Stony Brook, NY, USA, 21–22 June 2005; pp. 71–144.
29. Nguyen, A.; Le, B. 3D point cloud segmentation: A survey. In Proceedings of the 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), Manila, Philippines, 12–15 November 2013; pp. 225–230.
30. Sappa, A.D.; Devy, M. Fast range image segmentation by an edge detection strategy. In Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling, Quebec City, QC, Canada, 28 May–1 June 2001; pp. 292–299.
31. Li, Y.; Wu, X.; Chrysathou, Y.; Sharf, A.; Cohen-Or, D.; Mitra, N.J. Globfit: Consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics (TOG)*; ACM: New York, NY, USA, 2011; Volume 30, p. 52.
32. Asaeedi, S.; Didehvar, F.; Mohades, A. A-Concave hull, a generalization of convex hull. *Theor. Comput. Sci.* **2017**, *702*, 48–59.