A Linear Programming Approach for 3D Point Cloud Simplification

Nallig Leal, Esmeide Leal, and Sanchez-Torres German

***Abstract*— Advances in data acquisition technology have made it possible to obtain high-density samples from the surface of an object. Such samples produce thousands or even millions of data points. Processing such large amounts of data is computationally expensive. This study presents a novel method for point cloud simplification using an estimated local density of the point cloud. The proposed approach is robust to noise and outliers. The method is comprised of three stages. The first stage uses the expectation maximization algorithm to cluster the point cloud according to the local distribution of the points. The second stage identifies the points with a high curvature. These are feature points that will not be removed. In the final stage, a linear programming model is applied to reduce the cloud. Each cluster is a graph where the nodes have a cost defined by the inverse of its distance to the centroid. The results show that the reduced cloud is a good approximation of the original.**

***Index Terms*— Expectation maximization, Linear programming, Point cloud simplification, 3D objects.**

I. Introduction

Three-dimensional (3D) reconstruction constitutes an active research area due to its wide range of computational applications including robotic vision [1], cultural heritage restoration, archeology, medical imaging processing, reverse engineering and graphical computation. 3D object reconstruction is based on a digital computational model obtained from a data set of discrete samples that hold their geometric characteristics, such as volume and shape. It is a non-trivial task that includes five stages, namely, 3D image acquisition, registration, integration, segmentation and surface fitting.

3D image acquisition allows us to obtain partial spatial information from real objects. By merging the different images, we can form a cloud of points that describe the geometry of the object surface.

Due to the technological progress in laser scanning devices, it is possible to densely sample real objects. Such dense samples may contain thousands or even millions of 3D points. Point set simplification, without accuracy loss, is of interest to the scientific community because of the computational cost and memory requirements to process large data sets. The reduction of samples is an important aspect of the design of scalable algorithms for visualization and modeling [2].

Manuscript received June 20, 2016; revised November 10, 2016.

N. Leal and E. Leal are with the Faculty of Engineering, Universidad Autónoma del Caribe, Barranquilla, Colombia.

Sanchez-Torres German is with the Faculty of Engineering, Universidad del Magdalena, Carrera 32 No. 22-08, Santa Marta, Colombia. Email: gsanchez@unimagdalena.edu.co.

Corresponding author is N. Leal, Email: nleal@uac.edu.co.

The 3D laser scanning process can generate redundant information mainly in planar regions. Therefore, point simplification or surface simplification allows us to produce a mathematical approximation of a given surface using a reduced data set. Such approximations should be as near as possible to the original surface [2].

Several different approaches have been proposed for point cloud simplification [2]–[7]. The typical disadvantages of these methods are:

- The local and global distributions of the original data set are altered.
- Parameter dependency.
- The need for additional procedures such as polygonal mesh estimation.

This work describes a new method for point cloud simplification. In contrast to previous methods, it requires no additional mesh estimation procedure and is robust to the presence of noise and outliers.

The proposed methodology has three stages. The first stage subdivides or clusters the point cloud according to local distribution and it consecutively estimates a curvature approximation for each one of the points. Thus, the selection of feature points uses the curvature measurements, selecting those that lie on edges or corners. This allows sharp geometrical characteristics to be maintained. Therefore, the reduction stage does not consider feature points. Finally, a linear programming model is established. The objective function maintains the proportion of the density of the original points and reduces the data set.

Section II describes the related work on point cloud simplification. Section III explains the proposed simplification method. In Section IV, we give the results and discussion. Finally, we conclude our study in Section V.

II. Related Work

After the acquisition procedure, thousands or millions of data points can be obtained. Increasingly complex models are becoming more frequent because of the ease with which modern acquisition systems, such as range scanners, acquire these models. However, because the amount of data often exceeds the hardware performance, it is necessary to apply some procedure to reduce the original data that holds the accurate measurements. These approaches depend on the data representation.

The capturing of partial information from a fixed point of view is the basis of the traditional acquisition process. This type of representation is called range imaging. Multiple range images have to be acquired from views from different angles to obtain a complete object representation [8]. If the local coordinate systems of all images from an object are aligned

with a global coordinate system [9], [10], and the data is unified, we formed a point cloud. Due to the disadvantages of points used for visualization, they are transformed into an intermediate representation using polygons known as polygonal meshes. These meshes constitute the industry standard for visualization. Triangular and tetrahedral meshes are the most popular representations. For triangular meshes, there are many graphics library resources available, which is why triangles meshes are the most popular drawing primitive [11]. Consequently, data reduction procedures can be grouped into two main categories, mesh-based simplification and point-based simplification.

Mesh-based simplification methods have been widely studied and can be classified into four approaches, namely, sampling, adaptive subdivision, decimation and vertex merging [12]. A more extended taxonomy was previously reported [11] that includes methods based on energy function optimization [13], [14] and sub-classification of the four methods mentioned previously. In many cases, a defined metric was used to regulate the simplification process. Simplification metrics can be divided into two classes, local or global, according to the feature that is measured. However, local properties are the basis of most of the proposed metrics, which guarantee the preservation of local features [15].

Regardless of the types of methods used in mesh simplification, a mesh must be initially constructed. The mesh represents the connectivity of points through vertices and edges. The main advantage of meshes is that they can be used to easily estimate point neighborhoods and to define local continuous surface patches. In contrast, mesh estimation constitutes their main disadvantage due to the computational cost of mesh generation [16].

Point simplification methods avoid this computational cost. In [17], the point reduction assumes that the original point set is redundant. A decimation approach applied to the point set minimizes the redundancy. The point selection for reduction takes into account the point contribution to a moving least squares (MLS) surface representation. The contribution of a point is dictated by the definition of the shape.

In [6] presented different strategies for surface simplification from an unstructured point cloud. The implemented techniques cover incremental and hierarchical clustering, iterative simplification and particle simulation algorithms. The methods used local variation estimation and quadric error metrics to guide the simplification process, concentrating more points in regions of high curvature. The quadric error based technique and particle simulation methods generate approximations with low average errors but at high computational costs. The hierarchical clustering algorithm and uniform incremental clustering are computationally efficient but have higher average errors.

In [2] presented a method for point simplification using intrinsic farthest point sampling. This constitutes an approximation of geodesic Voronoi diagrams. The method uses the extended fast marching concept for discrete Voronoi diagram estimation. The sampling from farthest point places the next sample in the middle of the least-known area of the sample domain. The method permits sampling with user defined density levels to maintain geometrical features.

In [7] reported point simplification based on sample points using global distribution lines. The method uses star discrepancy as a sample quality measure. The reduced data set holds the spatial distribution of the original data set, but it needs an intermediate mesh representation.

In [18] showed an approach for minimizing the standard deviation of original and reduced data sets. This reduction is based on local analyzes of the clusterization of the initial data set and the Voronoi diagram concept. The amount of reduction is controlled by a user specified data reduction ratio. Other approaches that used some user parameter specification include [2], [19].

In [9] proposed an adaptive recursive subdivision scheme for point cloud simplification. It uses k-means clustering using the maximum normal vector deviation as a scatter measurement. Each one of the formed clusters is recursively decomposed into smaller sub-clusters until a similarity threshold is reached. Then, the centroid is selected. The method initially applies a boundary cluster identification procedure to avoid boundary shrinkage and a loss in accuracy.

In [20] introduced an algorithm for incremental planar simplification of dense cloud maps. The data used was a dense open scene point cloud from LIDAR (Light Detection and Ranging) scanners. The curvature estimation is the basis of the simplification procedure. Curvature estimation allows us to distinguish planar and non-planar surfaces. For a planar surface, it applies a region-growing scheme, fusing points with a similar tangent plane. Despite the fact that curvature based simplification methods have been widely studied, the methods are only suitable for real-time online operation. The reduction rate achieved is near to 90% of the planar input points.

In [16] presented an algorithm for point cloud simplification with edge preservation. The local surface topology is approximated using an octree data structure. Using the octree structure, a point neighborhood is selected and a projection plane is estimated. It permits differentiation between the edge and non-edge points. The procedure does not remove edge points. For the remaining points, it calculates an importance measure based on the normal vector.

III. Point Cloud Simplification Method

Our method simplifies point cloud simplification into three stages. The first stage performs cloud clusterization of the points based on their local distribution using the expectation maximization (EM) algorithm. The second phase identifies points in high curvature areas and marks them as feature points. Finally, to reduce the cloud of points, we apply a linear programming model adapting points in each cluster to node net (graph). Figure 1 shows the proposed methodology.

A. Local distribution based clustering

Cluster approaches aim to determine the set of points in small uniform regions. The main idea for determining the homogeneous cluster is to remove points without a significant loss of geometrical information. The clustering process uses the EM algorithms that consider the local distributions of the point set.

The EM algorithm is a general iterative technique for computing the maximum likelihood or maximum posterior parameter estimation of an incomplete data problem. The EM algorithm is similar to k-mean approaches, which are relatively straightforward. For a given fixed number of k groups, they assign samples to the groups in a way to be as different as possible. The EM algorithm extends this basic

approach. Thus, despite assigning samples to clusters for maximizing the mean differences, it calculates the ownership probabilities based on one or more probability distributions. Therefore, the objective of the procedure is to maximize the overall likelihood of data given in the final set.

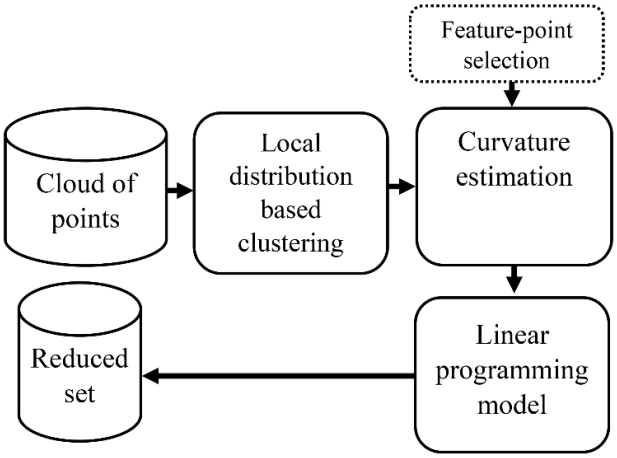


Fig. 1. Block diagram of proposed method.

In general, EM algorithms have two steps in each iteration, expectation and maximization. In the expectation step, the algorithm estimates the expectation of the likelihood, assuming that the latent variables were observed. Then, the maximization step computes the maximum likelihood, maximizing the expected likelihood previously estimated. New parameter values are used in subsequent interactions until convergence is reached [21].

Given an observed variable data set, $\{x_1, x_2, \dots, x_n\}$, and initial parameter values for a mixture model in iteration 0, $w_c^0, \mu^{0,c}$ and $\Sigma^{0,c}$, where c is each of k initial cluster, the basic iterative grouping algorithms proceed via a few steps:

1. Compute the membership probability of each point in each cluster in iteration j.
2. Update the mixture model parameters.
3. Verify if convergence is reached, if not continue to iteration j+1 in step 1.

For iteration j, for a point, x_i, the membership probability in cluster c is given by [22]:

$$P(c/x_i) = \frac{w_c^j p^j(x_i/c)}{p^j(x_i)} \tag{1}$$

The new parameter values are thus estimated using:

$$w_c^{j+1} = \frac{1}{n}\sum_{i=1}^{n} p(c/x_i) \tag{2}$$

$$\mu^{j+1,c} = \frac{\sum_{i=1}^{n} x_i \cdot p(c/x_i)}{\sum_{i=1}^{n} p(c/x_i)} \tag{3}$$

For convergence verification, it uses the difference between the log likelihood of the mixture model at iterations j and j+1. The log likelihood in iteration j is given by:

$$E^j = \sum_{i=1}^{n} \log\left(p^j(x_i)\right) = \sum_{i=1}^{n} \log\left(\sum_{c=1}^{k} w_c^{j+1} \cdot p^j(x_i/c)\right) \tag{5}$$

Therefore, if $\left|E^j - E^{j+1}\right| \leq \varepsilon$, convergence is reached, otherwise a new iteration is computed.

Due to the grouping of point clouds with data in 3D space, each x_i corresponds to the distinguishing features that consider only the spatial locations along the three axes.

B. *Curvature estimation*

The curvature is an invariant measure of surfaces, it indicates how much they differ from being planar. Geometrically, it is the rate at which the tangent vector changes. Curvature estimation is called invariant because it remains constant under geometrical transformations, such as translations and rotations. This measurement is estimated at a particular point.

Formally, the curvature can be defined by considering $S(r,t)$, a regular C^2 continuous parametric surface in $\mathbb{R}^3$ and the principal curvatures, $k_1(r_0,t_0)$ and $k_2(r_0,t_0)$, of S in $S(r_0,t_0)$, which are defined as maxima and minimal normal curvatures, respectively. According to Euler's theorem, the normal curvature of surface S in the tangent plane direction is (do Carmo, 1976):

$$k_n(\theta) = k_1 \cos^2\theta + k_2 \operatorname{sen}^2\theta \tag{6}$$

where θ is the angle between the first principal direction and the tangent vector.

From the k_1 and k_2 curves, we can derive two widely used curvature definitions known as the Gaussian and average curvatures. They are denoted as K and H, respectively, and are defined as [23]:

$$\frac{1}{2\pi}\int_0^{2\pi} k_n(\theta)d\theta = \frac{k_1 + k_2}{2} = H \tag{7}$$

$$\frac{1}{2\pi}\int_0^{2\pi} k_n(\theta)^2 d\theta = \frac{3}{2}H^2 - \frac{3}{2}K \tag{8}$$

$$H = \frac{1}{2}(k_1 + k_2) \tag{9}$$

$$K = k_1 \cdot k_2 \tag{10}$$

The main limitation of the Gaussian curvature is that it equals zero when any of the principal curvature is zero. The problem with the average curvature is that the minimal surface always has H = 0. Another advantageous measure is the absolute curvature, A, given by:

$$A = |k_1| + |k_2| \tag{11}$$

Regardless of the curvature type, to estimate it we can use any of the numerical algorithms for surface fitting. Fitting the surface at a given point permits us to calculate the curvatures K, H and A. We use the least-squares regression algorithm for local surface fitting.

$$\Sigma^{0,c} = \frac{\sum_{i=1}^{n} p(c/x_i)(x_i - \mu^{j+1,c})(x_i - \mu^{j+1,c})^T}{\sum_{i=1}^{n} p(c/x_i)} \tag{4}$$

The curvature is an intrinsic surface feature that can describe the local geometrical variations. Therefore, to avoid points elimination which representing sharp geometrical variation, we estimate the curvature over each one of the points in the cloud and mark those which the estimated curvature exceeding a fixed threshold. For the threshold set, we calculate the local average mean curvature inside the cluster to which that point belongs. Specifically, we estimate the mean curvature, A_i, for all points in a given cluster. The average mean curvature, $\bar{A}_i$, is computed and the following rules are applied:

a. If $A_i > \bar{A}_i$, the curvature indicates that a significant change of the surface has occurred at point *i*. This point is marked as a feature point.
b. If $A_i < \bar{A}_i$, point *i* corresponds to a smooth area without sharp features. This point can be reduced in the next stage.

C. Linear programming model

For each of the clusters, we adopt a linear programming model that selects the points by minimizing a cost function. The general procedure consists of two stages, density assessment cost function estimation and linear programming model solution.

Cluster density estimation

The density of the cluster uses the average of the inverse distance. The distance is measured from each of points to its centroid:

$$D_k = \frac{1}{\sum_{i=1}^{n} \frac{1}{d_i}} \tag{12}$$

where D_k is the cluster *k* density, *n* is the number of points of the cluster and d_i is the distance from the *i*th point to the cluster centroid.

Cost function calculation

The inverse distance from the points to the centroid is the basis of the cost function (see Figure 2). Therefore, the cost value of the *i*th point is given by:

$$C_i = \frac{1}{d_i} \tag{13}$$

where C_i is the cost value of the *i*th node and d_i is the distance from node to centroid.

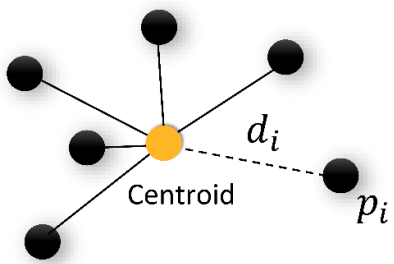


Fig. 2. Cluster representation as a graph where the node cost is the inverse distance to the centroid.

Linear programming model solution

The main objective of the linear programming model is to select a reduced set, which has a density equivalent to the original data set. Mathematically, it is impossible to obtain the same density measure if the points of reduced set are scattered over the same area of the original data set. As a result, we search for an equivalent or proportional density. Therefore, $D_c = kD_{cr}$, where D_c is the original cluster density, D_{cr} is the reduced cluster density and k is a proportionality constant given by $\frac{1}{PR}$, where PR is the simplification percentage. Therefore, the objective function is defined as:

$$Z = |kD_{cr} - D_c| \tag{14}$$

where,

$$D_{cr} = \frac{1}{nr} \sum_{i=1}^{nr} \frac{1}{d_i} \tag{15}$$

where nr is the amount of points of the reduced cluster, defined as $nr = [\![n \cdot PR]\!]$, and n is the number of points of the original cluster. Due to the nonlinearity of the absolute value function, we transform it as:

$$Min\ Z = |\mathrm{C^T x}| \tag{16}$$

Subject to:

$$\mathrm{Ax = b} \tag{17}$$

Therefore,

$$Min\ Z = y \tag{18}$$

Subject to:

$$\begin{gathered} \mathrm{Ax = b} \\ C^{\mathrm{T}}\mathrm{x} \le y \\ -C^{\mathrm{T}}\mathrm{x} \le y \\ y \ge 0 \end{gathered} \tag{19}$$

The complete model with all variables is established as:

$$Z = \left| \frac{kC_1}{nr} x_1 + \frac{kC_2}{nr} x_2 + \cdots + \frac{kC_{nc}}{nr} x_{nc} - D_c x_{nc+1} \right| \tag{20}$$

Substitute,

$$Min\ Z = \mathrm{y} \tag{21}$$

Subject to:

$$x_1 + x_2 + x_3 + x_4 + \cdots + x_{nc} = n_r \tag{22}$$

$$x_{nc+1} = 1 \tag{23}$$

$$\frac{kC_1}{nr} x_1 + \frac{kC_2}{nr} x_2 + \cdots + \frac{kC_{nc}}{nr} x_{nc} - D_c x_{nc+1} \le y \tag{24}$$

$$-\frac{kC_1}{nr} x_1 - \frac{kC_2}{nr} x_2 - \cdots - \frac{kC_{nc}}{nr} x_{nc} + D_c x_{nc+1} \le y \tag{25}$$

$$x_i \in \{0,1\} \tag{26}$$

Eq. 22 limits the number of points in the cluster to nr according to a simplification percentage. The assignment, $x_{nc+1} = 1$ (Eq. 23), guarantees that the minimization procedure selects the point set for minimizing Eq. 21. Eqs. 24 and 25 are used to linearize the absolute function. Finally, Eq. 26 ensures that the x_i decision variables are binaries.

Error measurement

Frequently, error measurement on reduced surfaces is based on the visual evaluation of the surface generated from reduced point clouds. However, in order to improve the error analysis, we consider both visual and numeric error estimation. In a similar way to [6], the numeric error estimation is based on the measured distance between original and reduced point cloud.

We measure the average of Euclidean distances between the original point cloud, P, and the surface generated by the reduced point cloud, P'. Since our method is mesh-free, we estimated the distance from each point in $P$to the closest local planar approximation in P', i.e. we compute the distance $d(p_i, S')$, where S' the local planar approximation in P' closest to p_i.

Given a point $p_i \in P$, in order to get its corresponding S', we select the set of points NH_i (neighborhood) in P' closest to p_i. The NH_i point set is selected using a kd-tree data structure and is used for estimating the least squares plane L_{NH_i} using Principal Analysis Component (PCA) [24], which represents the local approximation S' (see Fig. 3).

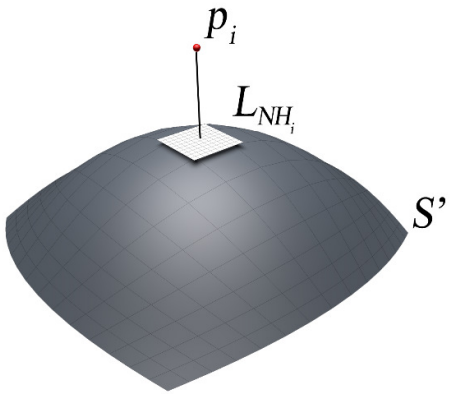


Fig. 3. Error computed as distance from p_i to L_{NH_i}.

Thus, we compute the difference between original point cloud and reduced point cloud, as the mean of distances between each point p_i to its closest plane L_{NH_i}.

$$E = \frac{1}{N}\sum_{i=1}^{N} d(p_i, L^i_{s'}) \tag{27}$$

IV. Results and Discussion

The proposed method has been implemented using Matlab R2014a software on an 8 GB memory PC with an Intel Core i7 processor running at 2.2 GHz.

A. Clustering

Figure 4 show the results of the clustering stage applied to the Stanford bunny and Max Planck models using 20 clusters.

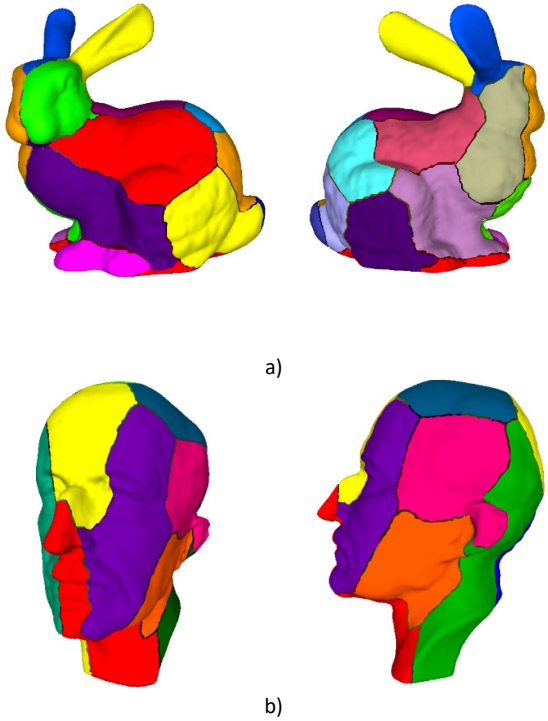


Fig. 4. Clusters formed by the EM algorithm using the a) Stanford bunny and b) Max Planck models.

B. Feature point selection

We test the curvature-based approach on the selection of feature points in different models. Figure 5 shows the original model and its estimated curvature maps. Points not in green points correspond to feature points. Conserving these points in the reduced version allows for the preservation of boundary and geometrical characteristics. Thus, the next stage applies the reduction procedure using only the non-feature points.

C. Linear programming model reduction

Figures 6 and 7 show the results obtained using the proposed method. In Fig. 6, we used the Stanford bunny model with 35.947 points and applied the method to reduce it by 70%, 80% and 90%, respectively. The model reduced versions had 20.130, 10.784, 7.189 and 3.594 points, respectively. After the reduction stage, we join the feature point set and the reduced data set to form the resulting reduced point cloud. For graphical analysis, we triangulate and render it. The reduced model preserves the geometrical characteristics of the original models, even for the 90% reduced models.

The Fig. 7 shows the results obtained using the Max Planck model. The original data set has 49.132 points and the reduced models have 24.566, 9.826 and 4.913 points, corresponding to reduction rates of 70%, 80% and 90%, respectively. In the same way as the above model, the reduced version held the geometrical features from the original model. For high reduction rates of 80% and 90%, some surface discontinuities appear. These discontinuities or holes are over planar regions.

D. Computational Effort

The Fig. 8 shows average computation times for different point cloud simplification executions setting the number of clusters to 15, 20, 25 and two 3D model. Due to initially the point simplification algorithm use only 3d points information, curvature calculation depends on the correct normal estimation. We report the averages times for the normal estimation, clustering and simplification stage on Table I. Consider the curvature calculation time as a factor of the normal estimation depending on point cloud size. Note that for clustering the time increases in a no linear way with the cluster size, in fact, clustering constitutes the most expensive stage. The simplification step is the least expensive stage with a variance of approximately 0,2 seconds for the multiples configurations.

The Fig. 9 shows the behavior of the error varying the number of the clusters from 10 to 35 with reduction rate of 70%, 80% and 90%. The error tends to decrease when the number of clusters increased and the reduction rate decreased. That behavior is predictable because there are more points when less reduction rate is applied. Using more clusters allow a best geometric representation of the surface. However, the more increase the number of clusters more expensive the computational cost of the process.

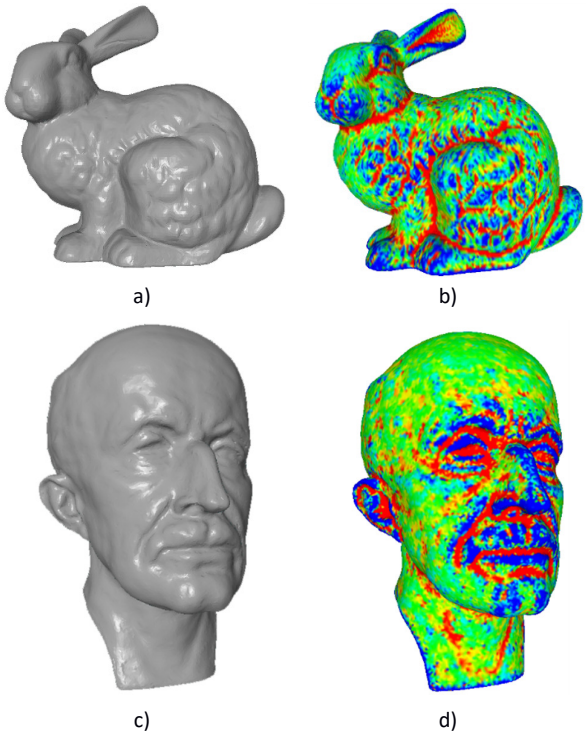
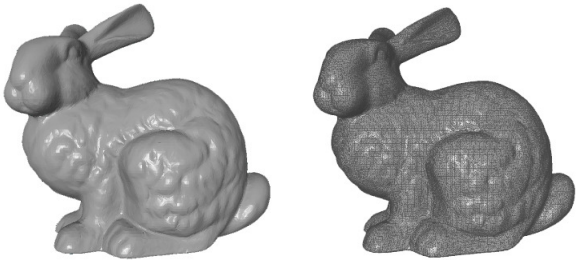


Fig. 5. Curvature-based selection of feature and non-feature points by curvature estimation.



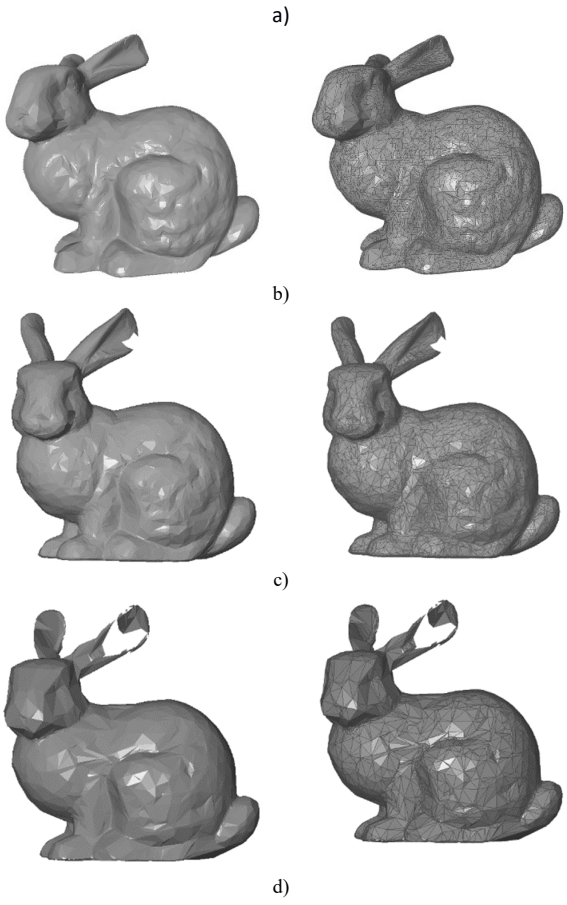
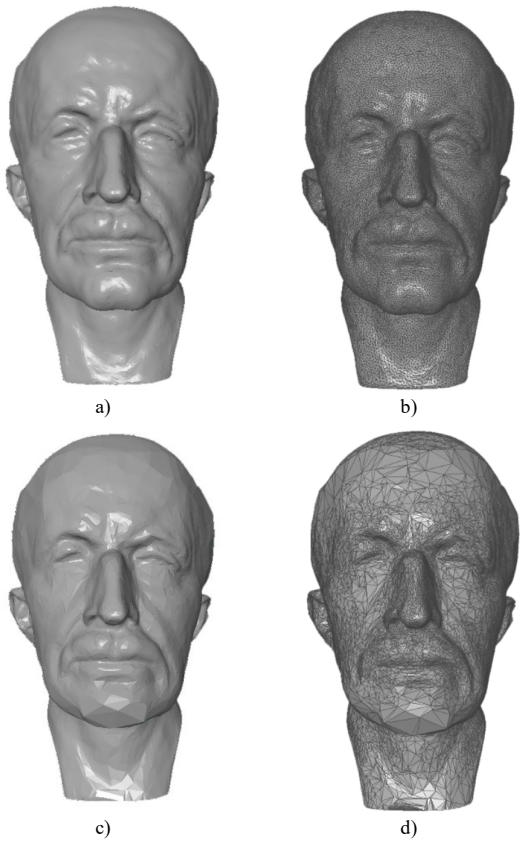


Fig. 6. Stanford model simplification, surface and mesh structure. a) Original model, reduced model using reduction rates of b) 70%, c) 80% and d) 90%.



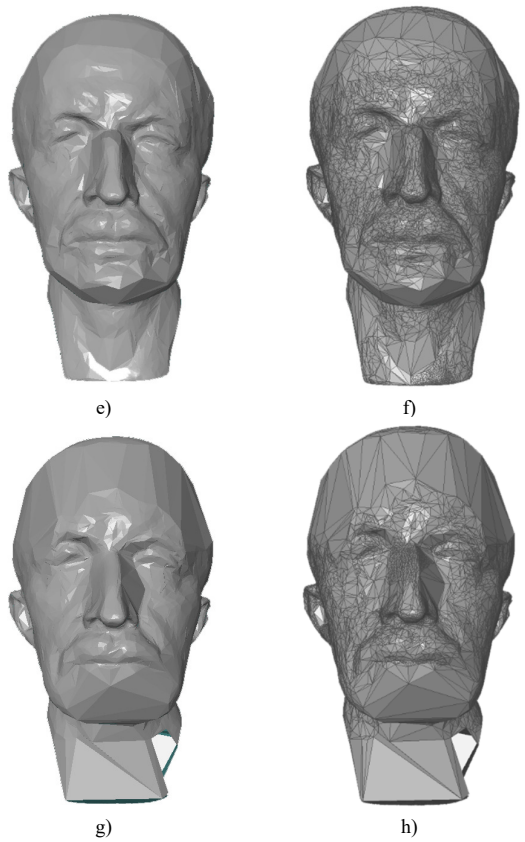


Fig. 7. Max Planck model simplification, solid and wireframe images. a-b) Original model, b-h) reduced model using 70%, 80% and 90% reduction rate, respectively.

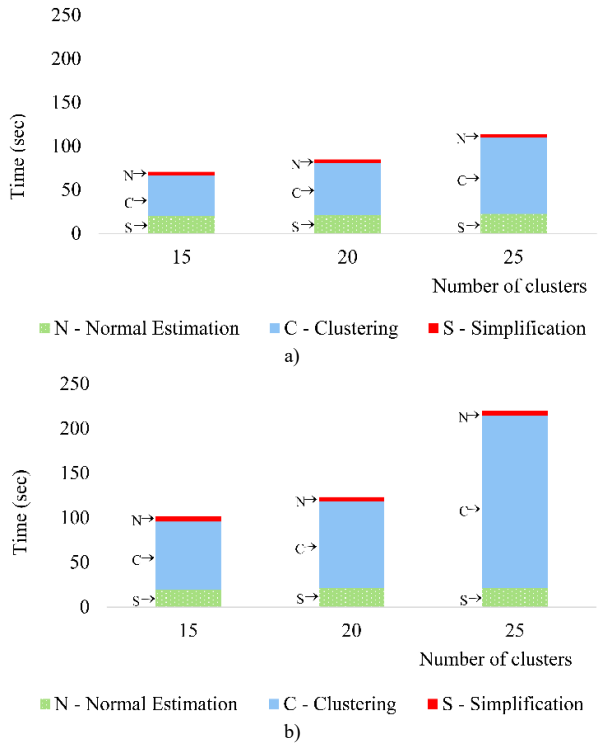


Fig. 8. Average time of normal estimation, clustering and simplification for used model varying the number of clusters from 15 to 25 clusters, a) bunny model and b) Max Planck model.

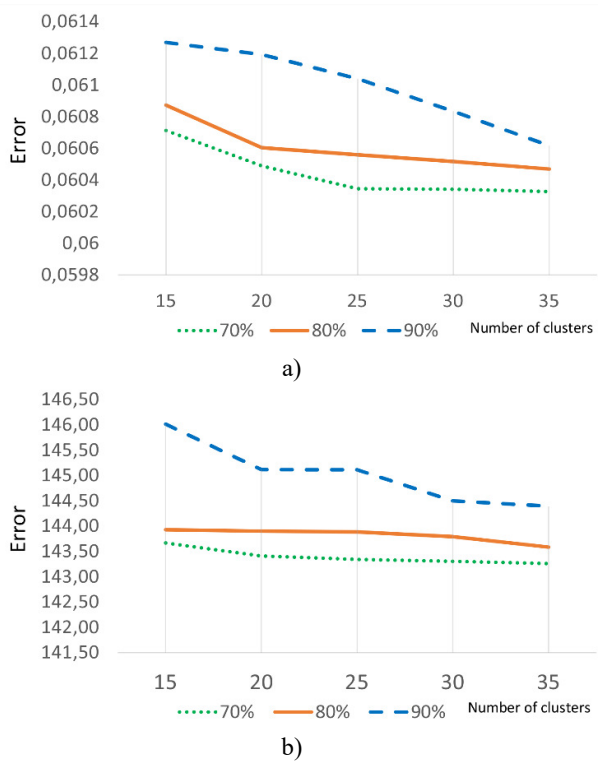


Fig. 9. Error behavior depending on the number of clusters, varying from 15 to 35 for reduction rates of 70%, 80% and 90% using the bunny model (a) and the Plank model (b).

V. Conclusions

We have presented a new approach for point cloud simplification. The method removes points that hold the local and global geometric features.

The principal advantage of using an EM based method for clustering of the point cloud is that it allows for more homogeneous cluster generation. However, the disadvantage is that increasing the amount of clusters generates poorly dense clusters when the original data set has a low density.

The mean curvature is used for feature point selection and it allows the main geometrical characteristics of the models to be retained. Even for high reduction rates of 80% and 90%, the models preserved the local and global geometric features. The method guarantees geometric feature preservation because the reduction algorithm does not remove high curvature points.

A linear programming model was applied to the point cloud reduction, addressing the problem from an optimization perspective.

Future works must address the automatic estimation of the amount of clusters. In the actual state, it is a user-defined parameter.

We believe, however, that using a faster technique for clusters generation made more suitable the whole algorithm.

The linear programming method should include multiple objectives to avoid hole generation. Thus, an additional objective in the optimization problem should guarantee uniformly distributed cluster generation.

TABLE I
POINTS REDUCTIONS AND COMPUTATIONAL TIME AVERAGE.

		Reduction rate and number of clusters								
		90%			80%			70%		
Model	Stage	15	20	25	15	20	25	15	20	25
Planck	Normal Estimation	15,656	18,817	19,780	21,728	24,266	21,855	21,028	21,111	21,965
	Clustering	44,690	84,424	179,546	114,362	127,193	172,894	69,542	77,910	227,683
	Simplification	6,028	4,946	5,324	6,464	5,441	5,620	6,429	5,748	5,337
Bunny	Normal Estimation	16,494	19,963	22,083	21,925	22,067	23,118	21,056	21,984	21,890
	Clustering	68,171	49,935	68,561	39,288	91,916	97,635	32,665	36,966	96,907
	Simplification	3,521	3,593	3,578	4,853	4,111	3,568	3,634	3,797	3,826

REFERENCES

[1] F. Toyomi and Y. Toshinori, "3D Terrain Sensing by Laser Range Finder with 4-DOF Sensor Movable Unit based on Frontier-Based Strategies," *Engineering Letters*, vol. 24, no. 2, pp. 164–171, 2016.

[2] C. Moenning and N. A. Dodgson, "A New Point Cloud Simplification Algorithm," in *In Proceedings 3rd IASTED Conference on Visualization, Imaging and Image Processing*, 2003, pp. 1027–1033.

[3] T. K. Dey, J. Giesen, and J. Hudson, "Decimating Samples for Mesh Simplification," in *Proc. 13th Canadian Conf. Comput. Geom*, 2001, pp. 85–88.

[4] S. Ferrari, I. Frosio, V. Piuri, and N. A. Borghese, "Enhanced vector quantization for data reduction and filtering," in *2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings*, 2004, pp. 470–477.

[5] L. Linsen, "Point Cloud Representation," Universitat Karlsruhe, Germany, CS Technical Report 2001-3, 2001.

[6] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization, 2002. VIS 2002*, 2002, pp. 163–170.

[7] J. Rovira, P. Wonka, F. Castro, and M. Sbert, "Point sampling with uniformly distributed lines," in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, 2005, pp. 109–118.

[8] L. Dan, J. Sun, Q. Li, and Q. Wang, "Fully automatic registration of structured objects based on laser radar range images," *Optik - International Journal for Light and Electron Optics*, vol. 126, no. 23, pp. 4698–4703, Dec. 2015.

[9] B.-Q. Shi and J. Liang, "An integration method for scanned multi-view range images (MRIs) based on local weighted least squares (LWLS) surface fitting," *Optics and Lasers in Engineering*, vol. 77, pp. 64–78, Feb. 2016.

[10] J. Zhu, Z. Li, S. Du, L. Ma, and T. Zhang, "Surface reconstruction via efficient and accurate registration of multiview range scans," *Optical Engineering*, vol. 53, no. 10, p. 102104, Apr. 2014.

[11] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, no. 1, pp. 37–54, Feb. 1998.

[12] D. P. Luebke, "A Developer's Survey of Polygonal Simplification Algorithms," *IEEE Comput. Graph. Appl.*, vol. 21, no. 3, pp. 24–35, May 2001.

[13] H. Hoppe, "Progressive Meshes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1996, pp. 99–108.

[14] R. Klein, G. Liebich, and W. Straßer, "Mesh Reduction with Error Control," in *Proceedings of the 7th Conference on Visualization '96*, Los Alamitos, CA, USA, 1996, pp. 311–318.

[15] H. Tang, H. Z. Shu, J. L. Dillenseger, X. D. Bao, and L. M. Luo, "Technical Section: Moment-based Metrics for Mesh Simplification," *Comput. Graph.*, vol. 31, no. 5, pp. 710–718, Oct. 2007.

[16] H. Han, X. Han, F. Sun, and C. Huang, "Point cloud simplification with preserved edge based on normal vector," *Optik - International Journal for Light and Electron Optics*, vol. 126, no. 19, pp. 2157–2162, Oct. 2015.

[17] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point Set Surfaces," in *Proceedings of the Conference on Visualization '01*, Washington, DC, USA, 2001, pp. 21–28.

[18] H. Song and H.-Y. Feng, "A progressive point cloud simplification algorithm with preserved sharp edge data," *Int J Adv Manuf Technol*, vol. 45, no. 5–6, pp. 583–592, Mar. 2009.

[19] A. Kalaiah and A. Varshney, "Modeling and Rendering of Points with Local Geometry," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 30–42, Jan. 2003.

[20] T. Whelan, L. Ma, E. Bondarev, P. H. N. de With, and J. McDonald, "Incremental and batch planar simplification of dense point cloud maps," *Robotics and Autonomous Systems*, vol. 69, pp. 3–14, Jul. 2015.

[21] H. Lan, X. Wang, Q. Pan, F. Yang, Z. Wang, and Y. Liang, "A Survey on Joint Tracking Using Expectation-maximization Based Techniques," *Inf. Fusion*, vol. 30, no. C, pp. 52–68, Jul. 2016.

[22] P. S. Bradley, U. M. Fayyad, and C. A. Reina, "Scaling EM (Expectation Maximization) Clustering to Large Databases - Microsoft Research," Microsoft, 1998.

[23] K. Watanabe and A. G. Belyaev, "Detection of Salient Curvature Features on Polygonal Surfaces," *Computer Graphics Forum*, vol. 20, no. 3, pp. 385–392, Sep. 2001.

[24] E. Leal, N. Leal, and G. Sánchez, "Estimación de Normales y Reducción de Datos Atípicos en Nubes de Puntos Tridimensionales," *Información tecnológica*, vol. 25, no. 2, pp. 39–46, 2014.