TEST Engineering & Management

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

3D Point Cloud Compression using Invariant Featured Map

Abdallah El Chakik, Beirut Arab University Tripoli, Lebanon
Abdul Rahman El Sayed, Normandie Univ, UNIHAVRE Le Havre, France
Amer Bakkach, Beirut International University Beirut, Lebanon

Article Info
Volume 82
Page Number: 12695 - 12700
Publication Issue:
January-February 2020

Article History
Article Received*: 18 May 2019*
Revised*: 14 July 2019*
Accepted*: 22 December 2019*
Publication*: 24 February 2020*

Abstract:
3D point clouds data obtained from scanning devices require a huge large storage and long post- processing time due to the number of points of the resulting point cloud. To reduce the processing time of these point clouds and to use them in many computer science application, the simplification is a very important phase. In our method, we build a graph from the given point cloud then we compute the similarity function using the Menger curvature. We define then a set of patches over the graph and we assign a degree for each vertex depending on its neighbors. We compute next the feature value of each vertex by projecting the point cloud onto different scale spaces and subtract the calculated degree at each scale. Furthermore, we remove iteratively the vertices based on an importance value calculated for each vertex based on the feature value of the vertex and its neighbors. Finally, we present the stability and robustness and of our approach on different 3D point clouds.

Keywords: *3D point, Cloud Data, Curvature, Vertex.*

I. INTRODUCTION

The recent advance in 3D digital scanning devices has been enabled 3D objects scanning as a practical reality [1] in the field of computer vision, scientific visualization, medical imaging and so on. 3D scanners produce meshes or point clouds. The last one consists of non-oriented and unorganized 3D points sets given by their x, y, z coordinates. These 3D points will be then post-processed to be finally converted into a mesh. One of these post-processing phases is the method of removing data redundancy from the 3D point clouds without affecting the initial shape [2] and known as 3D point cloud simplification. Algorithms for 3D point cloud simplification include the simplification based on curvature, clustering, iterative methods, and coarse-to-fine. Lee et al.[3] presented a 3D point cloud simplification method using 3D grids. Miao et al.[4] proposed an approach based on curvature aware re-sampling to simplify point clouds using the mean-shift adaptive clustering scheme. Moenning and Dodgson[5, 6] proposed a coarse-to-fine uniform simplification method using the idea of sampling of a point clouds surface by progressive intrinsic farthest point. In[7], authors have proposed a new simplification method for unstructured point clouds based on feature extraction. Song et al.[8] proposed a clustering point cloud simplification approach by defining the subset of the original data set according to a specified reduction ratio for data.

In this paper, we present a robust simplification method by defining a set of feature points and assign then an importance value for each point and remove iteratively least important points.. We firstly construct a weighted graph from the initial 3D point cloud. Next, we compute the similarity function between vertices based on Menger curvature. We compute then the degree of each vertex depending on its patch. Furthermore, we define a scaling space using a discrete convolution function. The feature value of each vertex is then calculated by subtracting vertices degrees calculated at different scales. Finally, we define an importance function for vertices to remove least important vertices recursively.

The novelty in this paper is shown in some key points. Firstly we construct a graph from the point

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

cloud and we compute the similarity function between vertices using the robust Menger curvature to ensure that the shape information is taken into consideration. Then, we compute a multi-scale vertices descriptors map using a convolution function. This map is calculated based on discrete gradient and vertices patches to will ensure that this map will be invariant to scaling and noise. Finally, we remove least featured vertices iteratively to preserve the initial point cloud shape.

The rest of this paper is organized as follow. Section 2 present the normal estimation and surface modeling. In section 3 we present our method. In section 4, we show the robustness of our approach by showing some results. Finally section 5 concludes our work.

II. COMPRESSION METHOD

A. Graph construction and normal estimation

We start our method by modelling the 3D point cloud by a weighted undirected graph G = (V, E, W) consisting of a finite set of vertices V, a finite set of edges E⊂ V × V, and a weight function: W: V X V→ [0, 1]. Consider a point cloud P as a set of points $\{p_1,\ldots,p_n\}$. Each point $\mathbf{p_i}$ will be associated to a graph vertex $\mathbf{v_i}$. Then, to define the set of $\mathbf{v_i}$'s neighbours $\boldsymbol{D_n}(\mathbf{v_i})$, we define a circumscribed spheres centred on pi with a radius r as $S(\mathbf{v_i}, \mathbf{r})=\{\mathbf{p_i} / \|\mathbf{p_i} - \mathbf{p_j}\|_2 \leq \mathbf{r}\}$. A vertex $v_j \in \boldsymbol{D_n}(\mathbf{v_i})$ if $\mathbf{p_j} \in S(\mathbf{v_i}, \mathbf{r})$ as shown in figure 1.

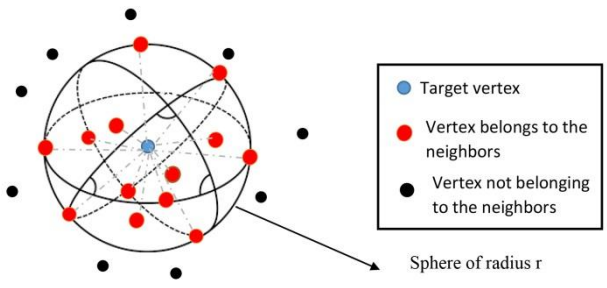


Figure 1 Set of neighbors for a given vertex

The normal $N(v_i)$ of a vertex v_i and 2-directional vectors following x- and y-axes are calculated as the eigenvalues of covariance matrix$Cov(v_i)$ defined as [9]:

$$Cov(v_i)= \sum_{j \in D_n(v_i)}(v_j - \check{v}_i)(v_j - \check{v}_i)^T \in \mathbb{R}3x3 \quad (1)$$

Where ŏi represents the gravity center and defined as:

$$\check{v}i = \frac{1}{Card\,(D_n(v_i))} \sum_{j \in D_n(v_i)} v_j \quad (2)$$

The normal of the vertex will be used to find the deviation factor of the vertex that will be a factor in the weight function.

B. Similarityfunction

In order to compute the similarity function between vertices(weight function), we propose to compute firstly the Menger curvature $MC(v_i, v_j, v_k)$which is the multiplicative inverse of the radius of the circle that passes through the three vertices $v_i, v_j, and\ v_k$. The Menger curvature can be seen as a theoretic formulation measure of curvature and mathematically related to the mean curvature. This curvature $MC(v_i, v_j, v_k)$ is defined as:

$$MC(v_i, v_j, v_k) = {}^{4 * A(v_i, v_j, v_k)}\Big/_{d(v_i, v_j) * d(v_j, v_k) * d(v_i, v_k)} \quad (3)$$

Then, the Menger curvature at a vertex v $MCP(v)$ is computed as:

$$MCP(v) = \begin{cases} MC(v_i, v_j, v_k) \\ where\ v_i, v_j, v_k \in D_n(v)\ and\ the\ furthest\ from\ v \\ 1 \qquad if\ |D_n(v)| \leq 1 \end{cases} \quad (4)$$

The precedent curvature measurement will be used to define a robust similarity measure between two vertices as:

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

$$\omega(v_i, v_j) = (\alpha(v_i, v_j)/(\sigma(v_i) \times \sigma(v_i))) \times e^{\left(-\frac{1}{MCP(v_j)}\right)} \quad (5)$$

where $\alpha(v_i, v_j)$ and $\sigma(v_i)$represent respectively the deviation factor between two vertices and the scale parameter and are defined as:

$$\propto(v_i, v_j) = arcos\left(\frac{\left(N(v_i).N(v_j)\right)}{\sqrt{(\sum_{k=1}^{3} N(v_i)(k)*N(v_i)(k)) * \sum_{m=1}^{3} N(v_j)(m)*N(v_j)(m)}}\right) \quad (6)$$

$$\sigma(v_i) = \max_{v_k \in D_n(v_i)} (||v_k - v_i||_2) \quad (7)$$

This weight function ensures that the similarity measurement takes into consideration the shape information which lead to a robust weight function.

C. Patch construction

To describe well the surface, we construct patches that represent sets of neighbouring vertices. These patches will be considered during the feature points detection.

To do so, given a vertex v_i, we construct a sphere around v_i

on its tangent plan as shown in figure 2. The patch radius is defined as $R(vi) = \max_{v_j \in D_n(v_i)} (||v_j - v_i||_2^2)$. Then, we decompose the patch into cells with side length $R(v_i)/n$, where n is a constant that depends on applications. Each cell of the patch is filled then by the average of Menger curvature at a vertex v_j $MCP(v_j)$where the patch cell contains v_j.

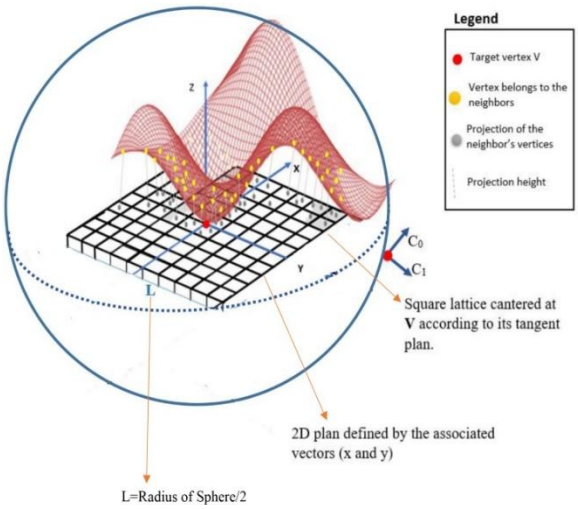


Figure 1 Patch construction

D. Vertex descriptor

To detect feature point, we define a scale space representation of mesh vertices using a convolution operation. Then, at each scale, we compute the discrete gradient of each vertex. We define then a patch weight for each patch depending on the set of vertices belonging to it. The feature points are selected as the maxima of the scale space across all scales.

The discrete gradient [10] of a vertex is defined as:

$$(\nabla f)(v) = \sum_{u \in D_{n(v)}} w(v,u)(f(v) - f(u)), \quad (8)$$

Where f :V ->**R** assigns a value f(v) to each vertex. The feature degree of a given vertex v deg(v) is then calculated based on the gradient and its patch weight pw(v). This patch weight is defined as:

$$pw(v) = \frac{\sum_{u \in |Vp|}(\boldsymbol{\omega}(v_i, v_j))}{|Vp|}, \quad (9)$$

Where |Vp| is the cardinality of the patch vertices Vp.

Then, the degree of a vertex v is computed as:

$$\deg(v) = (\nabla f)(v)/\, pw(v) \quad (10)$$

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

The convolution of scalar function f defined on the mesh with a kernel k is defined as:

$$(f * k)(v) = 1/|D_{n(v)}| \sum_{u \in D_{n(v)}} k \, ||f(v) - f(u)|| \, f(u) \quad (11)$$

The scale space is built by using the previously defined convolution operation as :

$$f^0 = f, f^1 = f^0 * k, f^2 = f^1 * k \quad (12)$$

and k is the weight function.

At each scale, we compute the degree of vertices using equation 9. Then for a given vertex, the degrees at all scales are subtracted to produce finally the vertex descriptor value DV(v). This multi scale operation results a robust approach that deals with different translation factors and noise.

Finally, the point cloud is compressed by using a threshold ε based on vertices descriptor values and the remaining vertices will define the compressed point cloud.

E. Compression

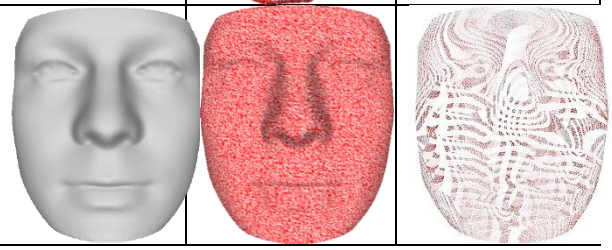
To compress the point cloud after assigning a description value for each vertex, we propose to iteratively remove the least important vertices until we reach the specified number. The importance of a given vertex is calculated as:

$$I(v) = 1/|\mathrm{Vp}| \sum_{u \in D_{n(v)}} ||DV(v) - DV(u)|| \quad (13)$$

The precedent equation calculated geometrically the Euclidian distance from v to the tangent plane of u. Then, by using equation 13, we calculate all vertices importance values and we start by removing the least important vertex in each patch. After removing a vertex, some of this vertex neighbors will update their importance values. So, we update the importance values of vertices and we remove the new least importance vertex. We keep doing that operation until we obtain the desired vertices threshold ε where $I(v) < \varepsilon$ for any vertex v or no new vertex can be removed. By doing this, we ensure that the set of vertices to be eliminated will not affect the final result shape as we remove only vertices with low importance values.

III. EXPERIMENTAL RESULTS

In this section, we present the efficiency of our approach by testing it on many 3D point clouds. Figure 3 the result of many 3D point clouds with different threshold ε. One can see that our method gives robust compressed point clouds while preserving the shape. Table 1 presents the number of simplified point clouds with percentages of simplification rate. To show the robustness of our approach, we apply it on noisy models as shown in figure 4. The results in that figure prove that our approach deals with noise and can compress noisy data to simplify the point cloud and preservers the initial shape.

	Original	**Simplified with ε =1**	**Simplified with ε =0.5**
M1			
M2			

TEST Engineering & Management

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

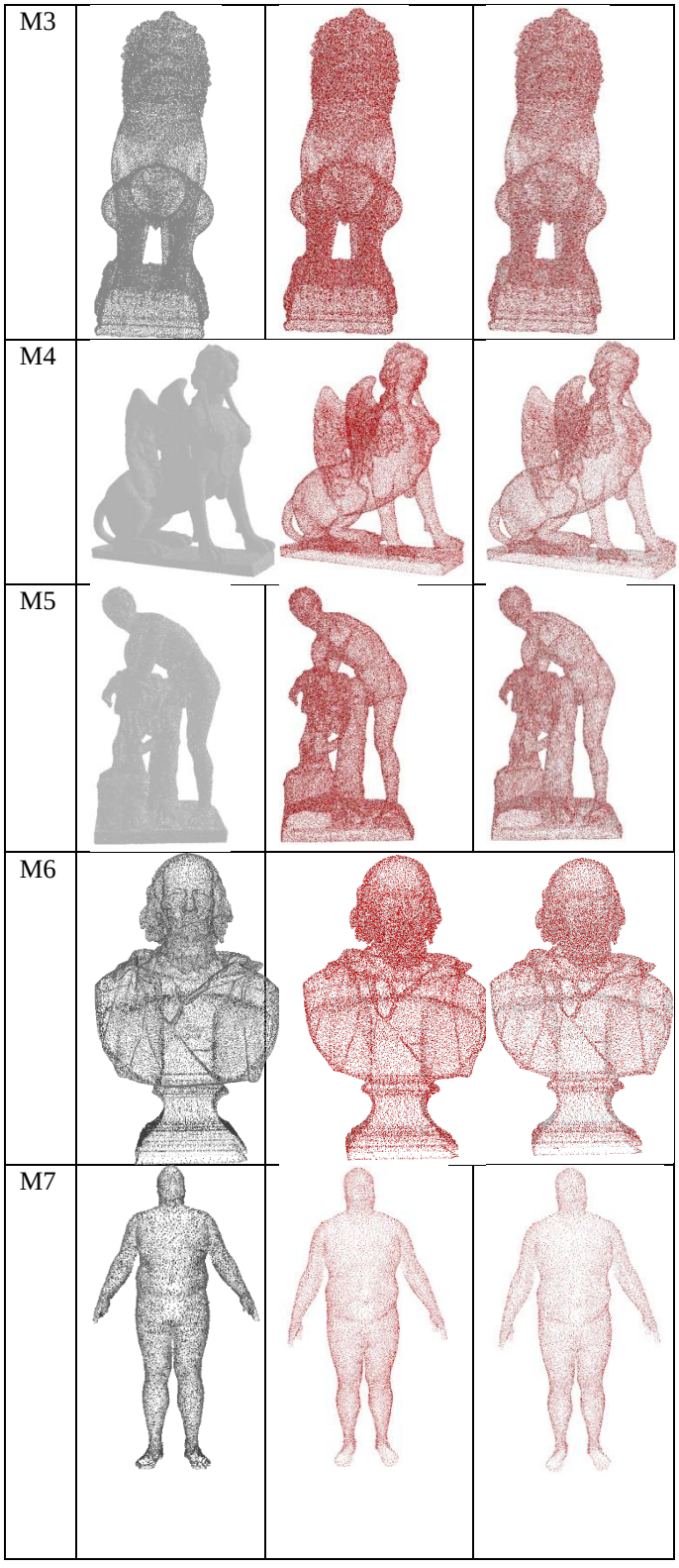


Figure 3 Results of our method on different 3D point clouds with different threshold values

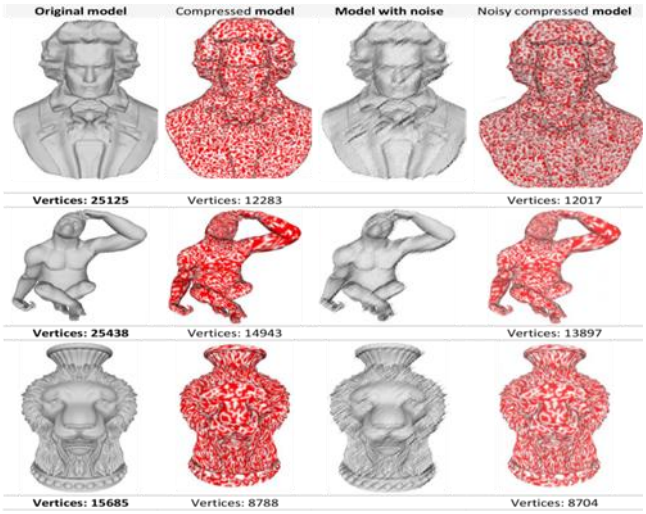


Figure 4 Results of noisy models

		Number of simplified points			
		ε =1		ε =0.5	
M1	262909	190572	27.51%	15406	94.14%
M2	62478	30552	51.10%	13731	78.02%
M3	100658	49241	51.08%	33361	66.86%
M4	83739	50013	40.28%	30867	63.14%
M5	107353	52414	51.18%	34331	68.02%
M6	73739	36118	51.02%	22548	69.42%
M7	13703	10791	21.25%	7011	48.84%

Table 1 Number of simplified points with percentages

F. Comparision with other methods

In this subsection, we compare our proposed method with some other existing methods in literature. In some cases where the simplification rate is very high some of sharp region points that conserve the characteristics of the surface shape will be removed in [10] resulting holes in the simplified 3D point clouds. In contrast, our method preserve the characteristics of the surface shape with a minimal configuration (Epsilon =0,$Order = 0$ and Level =1) due to the technique we use where we maintain internal and boundary feature points. In [11], authors mentioned that their proposed method works only with models whose shape is symmetrical or spherical and may produces holes in others. In contrast, figure 3 and 4 shows that our method produce better results with models of different shapes (symmetric and no symmetric). By comparing our method with [12],

January-February 2020
ISSN: 0193-4120 Page No. 12695 - 12700

our method is better in terms of geometric error. As for the bunny point cloud for example, the geometric error in our method is 0.000265 and in [12] is 0.000473.

IV. CONCLUSION

We presented in this paper a robust approach to compress point clouds by using invariant features map. We modeled the point cloud by a weighted graph and we calculated the similarity function between vertices by using the deviation factor and Menger curvature. We constructed then patches in order to assign a feature degree for each vertex based on multi scale approach. We finally remove vertices having least values iteratively depending on a threshold. Finally, we proved the efficiency of our method throw presenting many experimental results with different forms.

REFERENCES

1. J. Wu, X. Shen, W. Zhu, L. Liu, “Mesh saliency with global Rarity”, Graphical Models, Vol. 75, 2013, pp. 255–264.
2. A. Nouri, C. Charrier, O. Lézoray, “Multi-scale mesh saliency with local adaptive patches for view point selection”, Signal Processing: Image Communication, Vol. 38, 2015, pp.151–166
3. Y. Zhao, Y. Liu, R. Song, M. Zhang, A saliency detection based method for 3d surface simplification, In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Kyoto, Japan, (2012), pp.889–892.
4. A. El sayed, A. El chakik, H. alabboud, A. Yassine, “An efficient 3D mesh salient region detection using local homogeneity measure”, IET image processing, Volume 12, Issue 9, September 2018, p. 1663 – 1672
5. A. E. Johnson, M. Hebert, “Using spin images for efficient object recognition in cluttered 3D scenes”, IEEE Trans. Pattern Anal. Mach. Intell, Vol. 21, 1999, pp. 433–449
6. R. Song ,Y. Liu, Y. Zhao, R. R. Martin, P. L. Rosin, Conditional random field-based mesh saliency, in: 19th IEEE International Conference on Image Processing, (2012), pp. 637–640.
7. Chen, X., Saparov, A., Pang, B., et al. : 'Schelling points on 3d surface meshes', ACM Trans. Graph, 2012, 31, (4), pp 29.
8. P. Shilane, T. Funkhouser, “Distinctive regions of 3D surfaces”, ACM Trans. Graph. Vol. 26, 2007, pp. 7.
9. F. Lozes, A. Elmoataz, O. Lézoray, “Non-local processing of 3D colored point clouds”. In: 21st International Conference on Pattern Recognition, 2012, pp.1968–1971.
10. El Chakik A., Elmoataz A., DESQUESNES X. : Mean Curvature Flow on Graphs for Image and Manifold Restoration and Enhancement. Journal of signal processing 2014
11. Y. Yoshida, K. Konno, Y. Tokuyama, A Distributed Simplification Method with PC Cluster, The Journal of the Society for Art and Science,. 7(3), (2008), pp. 113-123.
12. C. Liao, X. Niu, M. Wang. “Simplification of 3D Point Cloud Data Based on Ray Theory”, COMPUTER MODELLING & NEW TECHNOLOGIES, (2014), 18, pp. 273-278.
13. Y. Miao, R. Pajarolac, J. Feng. “Curvature-aware adaptive re-sampling for point-sampled geometry”, Computer-Aided Design, 41(6), (2009), pp. 395-403.