A novel 3D mesh compression using mesh segmentation with multiple principal plane analysis

Shyi-Chyi Cheng\textsuperscript{a,∗}, Chen-Tsung Kuo\textsuperscript{b,c}, Da-Chun Wu\textsuperscript{b}

\textsuperscript{a}Department of Computer Science and Engineering, National Taiwan Ocean University, 2 Pei-Ning Road, Keelung 202, Taiwan
\textsuperscript{b}Institute of Engineering Science and Technology, National Kaohsiung First University of Science and Technology, 1 University Road, Yenchao, Kaohsiung 824, Taiwan
\textsuperscript{c}Department of Information Management, Longcyuan Veterans Hospital, VAC, Executive Yuan, 1 Anping Lane 1 Jiaosheng Road, Pingtung 912, Taiwan

\textbf{A R T I C L E I N F O}

Article history:
Received 24 July 2007
Received in revised form 20 May 2009
Accepted 26 May 2009

Keywords:
3D mesh
Compression
Segmentation
Principle plane analysis
\textit{k}-means clustering

\textbf{A B S T R A C T}

This paper proposes a novel scheme for 3D model compression based on mesh segmentation using multiple principal plane analysis. This algorithm first performs a mesh segmentation scheme, based on fusion of the well-known \textit{k}-means clustering and the proposed principal plane analysis to separate the input 3D mesh into a set of disjointed polygonal regions. The boundary indexing scheme for the whole object is created by assembling local regions. Finally, the current work proposes a triangle traversal scheme to encode the connectivity and geometry information simultaneously for every patch under the guidance of the boundary indexing scheme. Simulation results demonstrate that the proposed algorithm obtains good performance in terms of compression rate and reconstruction quality.

© 2009 Elsevier Ltd. All rights reserved.

\section{1. Introduction}

In recent years, three-dimensional (3D) mesh models have been widely used in geographical databases, manufacturing assemblies, virtual reality for entertainment applications, or other interactive applications. These models are often represented as complex triangular meshes, which may have thousands or even millions of vertices and polygons. All of these applications require large storage, computer power and access over bandwidth-limited links. Thus, it is essential to compress the 3D models efficiently. Since Deering [1] first introduced the concept of generalized triangular mesh compression, many algorithms categorized as lossless or lossy, single-rate or progressive, and single resolution or multi-resolution, have been proposed to compress 3D meshes [1-18]. A good review on 3D mesh compression technology can be found in [12,13].

Compared with the lossy 3D mesh coding, a lossless technique reconstructs mesh data identical to the original. Many applications use lossless data compression, such as in executable code, word processing files, etc. To compare single-rate coding with progressive coding, single-rate coding focuses on saving the bandwidth between CPU and the graphic card; however, progressive compression of 3D meshes provides multiple resolutions for transmitting complex meshes over networks with limited bandwidth. Early research conducted single-rate coding on 3D mesh compression. Chow [2] presented an algorithm to efficiently produce generalized triangle meshes. His meshifying algorithms and variable compression method achieved compression ratios higher than Deering’s method [1]. Touma et al. [3] proposed the valence-driven approach that records the degree of each vertex along a spiraling vertex tree. Progressive mesh compression has been also intensively researched, since it enables the decoder to reconstruct 3D models continuously from coarse to fine levels-of-details (LODs) [4-8]. The Edgebreaker method proposed by Rossignac [9] can handle manifold meshes with multiple boundary loops and handles. Gumhold et al. [10] improved this connectivity upper bound to 3.522 bpv.

Triangle mesh compression has been the focus of much study. This representation contains two kinds of information: geometry and connectivity. Geometry coding describes vertex coordinates in the 3D space, and connectivity coding describes how to connect these positions. The connectivity compression problem has been well studied, with many existing methods [12,13] achieving bit-rates of less than two bits per triangle for the connectivity portion of a mesh [11]. Less effort has gone into geometry compression, often simply performed by prediction coding and quantization [14]. Recently, researchers have proposed other geometry compression techniques, including wavelet transform [15], spectral compression [16], and \textit{k}-d tree or octree decomposition [17]. Although the performance of these 3D mesh compression techniques achieving bit-rates between 1 and 2 bytes per vertex is good, the output bit stream remains large for complex objects with high numbers of vertices and triangles. Thus, developing effective compression techniques for the vertex data to

\textsuperscript{∗}Corresponding author.
E-mail addresses: csc@mail.ntou.edu.tw (S.-C. Cheng), dcvu@ccms.nkfust.edu.tw (C.-T. Kuo), jmskuo@mail.nlch.gov.tw (D.-C. Wu).
transmit 3D mesh models in a bandwidth-limited network is important. Sim et al. [18] proposed an algorithm to encode connectivity data and geometry data using a triangle fan structure. Most previous works typically discuss connectivity compression and geometry compression issues individually. This work proposes a segmentation-based 3D mesh compression method that simultaneously considers connectivity and geometry compression problems.

Mesh segmentation is the first step towards region-based 3D mesh compression [19]. Segmentation is more common in the image processing area, and has been recently introduced into the 3D mesh area [20–25]. Among existing algorithms, the watershed-based approach has received more attention. This approach uses discrete curvature at each mesh vertex as the height field that drives watershed segmentation. The curvature estimation for 3D meshes is computationally expensive [20–22], as it is mathematically defined for a smooth surface only. Feature sensitive mesh segmentation that maintains salient features is important for many computer graphics and geometric modeling applications [23]. Cheng et al. [24] proposed a patch-growing approach using a shortest-path labeling technique to segment the model into multiple regions. Zhang [25] proposed a simple segmentation algorithm using Gaussian curvature analysis, efficient for certain 3D models. However, similar to other existing methods, this algorithm is defective when processing a high-resolution 3D model, as the geometric characteristics of adjacent polygons in such a model are too close to be differentiated. Therefore, an efficient and robust algorithm is needed for 3D mesh segmentation.

The current work proposes an efficient segmentation-based 3D mesh compression system, which provides progressive transmission for 3D models over a bandwidth-limited network. This algorithm first performs a mesh segmentation scheme, based on fusion of the well-known k-means clustering and the proposed multiple principal plane analysis to separate the input 3D mesh into a set of disjointed polygonal regions. The boundary indexing scheme for the whole object is created by assembling local regions. Finally, a proposed triangle traversal scheme encodes connectivity and geometry information simultaneously for every region under the guidance of boundary indexing. At the decoder, based on boundary indexing, this work obtains each individual region and reconstructs the whole object using connectivity and geometry coding information. Simulation results demonstrate that the proposed algorithm obtains good compression performance.

The remainder of this paper is organized as follows. Section 2 presents the method to separate the input triangle mesh into multiple uniform regions using a k-means clustering with the principal plane analysis technique. The current approach to 3D mesh model compression is described in detail in Section 3. Section 4 presents experimental results obtained from applying the proposed method to test 3D models. Section 5 outlines a brief conclusion.

2. Clustering-based 3D mesh segmentation

This section presents a clustering-based 3D mesh segmentation method based on k-means clustering [25–27] with the proposed principal plane analysis [28]. To clearly introduce the proposed segmentation scheme, the k-means method and principal plane analysis are first described.

2.1. k-means clustering

In 1967, MacQueen proposed the k-means [26] algorithm as one of the simplest unregulated learning algorithms to solve the clustering problem. The procedure follows a simple and easy method to classify a given data set into k of groups. The main idea is to define k centroids, one for each group. These centroids should be properly placed because different locations cause different results. Choosing them as far away as possible from each other is the better choice. The next step takes each point belonging to a given data set and associates it to the nearest centroid. When no point is pending, the first step is accomplished with an early grouping. At this stage, k new centroids of groups resulting from the previous step need recalculating. After obtaining k new centroids, a new binding must take place between a point in the same data set and the nearest new centroid, generating a loop. This loop may cause k centroids to change their location step by step until no more changes are accomplished. In other words, centroids do not change anymore. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

\[ J = \sum_{i=1}^{k} \sum_{j=1}^{n} ||x_{ij} - u_i||^2 \]  

(1)

where \( ||x_{ij} - u_i||^2 \) is a chosen distance measure between a data point \( x_{ij} \) and the group centroid \( u_i \), which is an indicator of the distance of n data points from their respective group centers.

The algorithm is composed of the following steps:

1. Data points are assigned at random to the k groups. The centroid is computed for each group.
2. Each point is assigned to the group with the closest centroid.
3. When all points have been assigned, recalculate the positions of the k centroids.
4. Repeat Steps (2) and (3) until centroids stop changing. This produces a separation of the points into k groups from which the metric to be minimized can be calculated.

k-means is a simple algorithm that has been adapted to many problem domains. As the following shows, it is a good candidate for extending work with 3D mesh segmentation.

2.2. Principal plane analysis for 3D models

In [29], this work proposes a 3D model retrieval method using principal plane analysis that defines the principle plane as a skeleton representation corresponding to the symmetric surface for a 3D object.

The principal plane can be conveniently represented in terms of moments. In the case of 3D feature space S^3, the principal plane H can be represented as

\[ Ax + By + Cz = D \]  

(2)

where A, B, and C are the directional numbers of H that satisfy the following relationship:

\[ A^2 + B^2 + C^2 = 1. \]  

(3)

The distance from a 3D vector \( \vec{c} = (x,y,z) \) to H is given by

\[ f(x,y,z) = Ax + By + Cz - D. \]  

(4)

Let the centroid \( \vec{c} \) of the 3D space be defined as

\[ (\bar{x}, \bar{y}, \bar{z}) = \left( \frac{1}{N} \sum_{i=1}^{N} x_i, \frac{1}{N} \sum_{i=1}^{N} y_i, \frac{1}{N} \sum_{i=1}^{N} z_i \right) \]  

(5)

where \( x_i, y_i, \) and \( z_i \) are the tri-stimulus values of the ith vector. Obviously, the distance from the origin (0,0,0) of S^3 to H is D. For processing convenience, we subtract the centroid from all the 3D points in order to translate the origin of the 3D space to the centroid. In this case, the value of D in (2) is zero.
To find the direction numbers of the 3D principal plane $H$, take the origin at the centroid; then the inertia moment of the points in $S^3$ about the plane $H$ is

$$I(A, B, C) = \sum_{(x,y,z) \in S^3} (Ax + By + Cz)^2.$$  

(6)

Differentiates with respect to $A, B$ and $C$, and equating to zero gives

$$2 \sum_{(x,y,z) \in S^3} x(Ax + By + Cz) = 0$$

$$2 \sum_{(x,y,z) \in S^3} y(Ax + By + Cz) = 0$$

$$2 \sum_{(x,y,z) \in S^3} z(Ax + By + Cz) = 0.$$  

Hence,

$$m_{2,0,0}A + m_{0,1,1}B + m_{1,0,1}C = 0$$

$$m_{1,1,0}A + m_{0,2,0}B + m_{1,1,0}C = 0$$

$$m_{1,0,1}A + m_{0,1,1}B + m_{0,0,2}C = 0$$  

(7)

where $m_{i,j,k}$ is a 3D moment given by

$$m_{i,j,k} = \sum_{(x,y,z) \in S^3} x^i y^j z^k.$$  

(8)

To solve the set of linear system in (6), we get

$$A = \frac{m_{0,2,0}m_{1,0,1} - m_{1,1,0}m_{0,1,1}}{m_{2,0,0}m_{1,0,1} - m_{1,1,0}m_{1,0,1}} = k_1$$

$$B = \frac{m_{0,2,0}m_{1,0,1} - m_{0,1,1}m_{1,1,0}}{m_{0,2,0}m_{1,1,0} - m_{0,1,1}m_{1,0,1}} = k_2.$$  

(9)

The values of $k_1$ and $k_2$ can be obtained directly from the values of 3D moments which can be computed in advance according to (8). Combining Eqs. (3), (9), and (10), it is simple to obtain

$$(A, B, C) = \left( \frac{k_1}{\sqrt{1 + k_1^2 + k_2^2}}, \frac{1}{\sqrt{1 + k_1^2 + k_2^2}}, \frac{k_2}{\sqrt{1 + k_1^2 + k_2^2}} \right).$$  

(11)

Once the principal plane $H$ is obtained, for each 3D vector $c = (x, y, z)$ we can compute the depth from $c$ to $H$ by

$$d_H(c) = Ax + By + Cz.$$  

(12)

3. The proposed 3D mesh segmentation

Clustering-based segmentation uses iterative clustering as a tool to separate the input mesh into multiple regions according to local properties of vertices. For example, Shlafman et al. [30] used $k$-means clustering to provide a meaningful segmentation. However, the produced regions have jagged boundaries, shown in Fig. 1(a). The problems with applying $k$-means clustering to segment 3D mesh models are threefold: (1) $k$-means clustering does not guarantee generating contiguous clusters and thus might result in several small uniform regions. (2) The method selects a number of seed vertices and then assigns each triangle to the cluster of the nearest seed vertex. The resulting regions are dependent on the initial set of seed vertices. Moreover, it is not adaptive to the shapes of mesh surfaces. (3) The number of clusters, i.e., the value of $k$ is in generally unknown. This work presents a segmentation framework incorporating $k$-means clustering and principal plane analysis to solve the above problems. Fig. 1(b) shows a segmentation example using the proposed method. This paper introduces a multispace generalization of the multiple principal plane analysis (which we call MPPA), where more subspaces are created to approximate the different uniform regions of the input mesh model. The MPPA can be used to generate a compact representation of the original mesh model by mapping each vertex only into the best-suited subspace, shown in Fig. 2. Then, all the subspaces are simultaneously used to encode each connectivity and vertex, thus providing multiple points of view to the input mesh.

Let $V = \{v_i \in R^3 | i = 1, ..., n\}$ be a set of $n$ 3D vertices of an input mesh $M$, then for a given partition $\wp = \{P_1, P_2, ..., P_k\}$ of $V$ such that

$$\bigcup_{i=1}^{k} P_i = V, \quad P_i \cap P_j = \emptyset \quad i, j = 1, ..., k, \quad i \neq j$$

the MPPA segmentation is defined by the set of subspaces $S = \{S_1 | S_i = \bar{S}_i \cup \bar{S}_i^k, \bar{i} = 1, ..., k\}$, where $\bar{c}_i = 1/|S_i| \sum_{v \in S_i} \bar{v}$ is the centroid coordinate of $S_i$ and $H_i$ is the principal plane of $S_i$ defined in (2). Each subspace determines a principal plane, and the set of principal planes provides a compact representation of the original mesh model.

A huge number of MPPA segmentations may be derived from the same input mesh model by varying $k$ and $\wp$. The approach to obtain better $k$ and $\wp$ aims at minimizing the mean-square average reconstruction error on $\wp$, defined as a weighted sum of reconstruction errors related to the subspace $S_i$ approximated by $H_i$

$$\psi(k, \wp) = \frac{1}{n} \sum_{i=1}^{k} \sum_{v \in P_i} |d_H(\bar{v})|$$

(13)

where $n$ is the cardinality of $P_i$ and $d_H(\bar{v})$ is the depth value from $\bar{v}$ to $H_i$ defined in (12). $\psi(k, \wp)$ is then a cost function representing the input mesh model $M$ and used as a merit function for choosing $k$ and $\wp$.

Obviously, the larger value of $k$ leads to a smaller value of $\psi(k, \wp)$ using the unconstrained minimization process. A limit case is when each triangle constructs a region and enables a zero-error solution, i.e., $\psi(k, \wp) = 0$. On the other hand, employing a few elements for creating a region would not achieve high compression rate. Thus, this work proposes a practical strategy to determine an optimal MPPA segmentation using $k$-means clustering. Let $\psi_{\max}$ be the maximum error chosen for $\psi(k, \wp)$. The algorithm proceeds by increasing $k$ until finding a solution $(k, \wp)$ such that $\psi(k, \wp) \leq \psi_{\max}$ or the maximum allowed number of regions $k_{\max}$ is reached. Let $\psi^\star_{\wp}$ be the optimal segmentation for a partition $\wp$. The MPPA segmentation algorithm is separated into three main steps:

MPPA ($V, \psi_{\max}, k_{\max}$)

$$k = 1;$$

$\psi^\star = \infty; //$ sets the optimal reconstruction error founding to be a very large number

do {

$$k = k+1;$$

$$\wp_k = \wp - \text{Generate}(k, V); //$$ generates initial segmentation for $k$

$$\psi^\star_k = \psi - \text{Optimize}(k, \wp_k); //$$ optimizes the partition $\wp_k$

if ($\psi^\star > \psi_k$)$\psi^\star = \psi_k$; $//$ updates the optimal reconstruction error founding

$$t = k; //\psi^\star_k$$ is the best segmentation founding

} while ($\psi^\star > \psi_{\max}$ \&\& $k < k_{\max}$)

return ($t, \wp^\star$);

}

Obviously, requiring small reconstruction errors, i.e., small values of $\psi_{\max}$ allows the regions to better fit the input mesh model, but at the same time, determines the creation of a larger number of regions. The following discusses the optimization procedure ($\wp$-Optimize) and the initialization procedure ($\wp$-Generate) in detail.
The vertices in the input mesh are placed in the set \( v \) if the reconstruction error is included in the set \( H_1 \) and \( H_2 \), respectively. At each step, principal planes are derived to approximate the shapes of corresponding subspaces and each vertex is reassigned to the nearest subspace according to the depth information. Fig. 3 shows an example of separating the input mesh into two regions, each approximated by its principal plane. The \( \varphi \)-Optimize procedure is summarized as follows:

\[
\begin{align*}
\varphi \text{-Optimize} & \left( k, \varphi = \{ P_1, P_2, \ldots, P_k \} \right) \\
& \text{repeat} \\
& \quad t = 0; // t is the iteration counter \\
& \quad \text{repeat} \\
& \quad \quad \forall i = 1, \ldots, k; H_i = \text{PPA}(P_i); // \text{PPA computes the principal plane } H_i \\
& \quad \quad \forall \vec{v} \in V \{
\begin{cases}
\vec{v} \text{ is reassigned to the nearest subspace according to the depth in-} \\
\text{formation.}
\end{cases}
\}
\end{align*}
\]

3.2. \( \varphi \)-Generate

Analogous to the k-means algorithm, the initial clusters can be randomly generated. However, bad initial conditions often result in a bad partition for the k-means like algorithms. The current work uses a more effective approach to generate better initial solutions. This method is based on a cluster separating scheme by splitting a cluster with a large reconstruction error into two sets. Initially, all the vertices in the input mesh are placed in the set \( P_1 \). In the first time call of the \( \varphi \)-Generate procedure, the principal plane \( H_1 \) of \( P_1 \) is obtained by performing the PAA transformation. For each vertex \( \vec{v} \) in \( P_1 \), we compute its depth value \( d_{H_i}(\vec{v}) \) using (12). Then, \( P_1 \) is separated into two sets according to the depth values of vertices

\[
\vec{v}, \begin{cases}
P_{11}, & \text{if } \left( d_{H_1}(\vec{v}) \leq \bar{d}_{H_1} \right) \\
P_{12}, & \text{otherwise}
\end{cases}
\]

If it is the 1st time to use the \( \varphi \)-Generate procedure, the optimization partition \( P_1 \) obtained from the \( \varphi \)-Optimize procedure in the previous iteration of the MPPA segmentation algorithm is used to generate the candidate partition \( \varphi_k \) for further optimization processing. The \( k-1 \) vertex sets, i.e., \( P_1, P_2, \ldots, P_{k-1} \), in \( \varphi_{k-1} \), and their principal planes are given. For each vertex set \( P \) in \( \varphi_{k-1} \), we first calculate its weighted reconstruction error as follows:

\[
\varepsilon(P) = \frac{m}{n} \sum_{\vec{v} \in P} |d_{H_1}(\vec{v})|
\]

where \( m \) and \( n \) are the numbers of vertices belonging to \( P \) and \( \varphi_{k-1} \), respectively. Accordingly, the vertex set with larger size and reconstruction error has a larger value of \( \varepsilon(P) \). The set with the largest value of \( \varepsilon(P) \) is selected from \( \varphi_{k-1} \), for splitting using the rule defined in (13) to provide an additional vertex set for \( \varphi_k \). The \( \varphi \)-Generate procedure is summarized as follows:

\[
\begin{align*}
\varphi \text{-Generate} & \left( k, V \right) \\
& \text{if } (k = 2) \quad \begin{cases}
P_1 = V; & \text{The first time using the procedure} \\
H_1 = \text{PPA}(P_1); & // \text{PPA computes the principal plane } H_1 \text{ for } P_1 \\
P_2 = \{ \vec{v} \mid d_{H_1}(\vec{v}) > \bar{d}_{H_1} \ \& \ \vec{v} \in P_1 \}; & \text{for further optimization processing.}
\end{cases}
\end{align*}
\]
Fig. 3. An example of rearranging vertices in the input mesh to the nearest neighbor regions using multiple principle plane analysis.

\[ P_1 = P_1 - P_2; \]
\[ \}
\[ \text{else} \]
\[ \text{for each vertex set } P \text{ in } \varphi_{k-1}, \text{computes } z(P) \text{ using (14);} \]
\[ z = \arg \max_{i=1,...,k-1}(z(P_i)); \]
\[ \text{// choose the set with the largest reconstruction error} \]
\[ P_k = \{ \bar{v} | d_H(\bar{v}) > d_{H_k} \& \bar{v} \in P_2 \}; \]
\[ \text{// split } P_2 \text{ into two sets} \]
\[ P_2 = P_2 - P_k; \]
\[ \}
\[ \text{return } (\varphi_k = \{ P_1, P_2, ..., P_k \}); \]

4. Segmentation-based 3D mesh model compression

Performing the segmentation algorithm mentioned above, the input 3D mesh model is separated into multiple meaningful regions, which are then separately compressed and transmitted to the decoder. For each region, we have two compression jobs to do: geometry coding and connectivity coding, often treated as two independent compression phases in existing researches [12,13]. However, the connectivity and geometry information for a 3D mesh should have high correlation. Thus, combining geometry coding and connectivity coding is essential to achieve high rate 3D mesh compression.

Fig. 4 shows the block diagram of the proposed 3D mesh compression system. Basically, the proposed region-based 3D mesh compression consists of two phases: boundary coding and internal connectivity–geometry coding. This hierarchical representation scheme provides a flexible way to implement a progressive 3D mesh: in the former stage, the boundaries of each region corresponding to the compact shape approximation of the input mesh are transmitted. Then, the internal connectivity and geometry information for each region is transmitted to fine tune the topology structure of the region.

4.1. Boundary coding

A 3D mesh can be represented as one or more connected components, each consisting of a set of regions with every pair of regions reachable with each other. Given a connected component \( C \), we construct a graph \( G = (N,E) \) to model, where \( N \) is the set of regions in \( C \) and \( E \) is the set of edges, each of them links every pair of adjacent regions. Constructing a spanning tree from \( G \) is easy, and the mesh regions encode one by one by following the tree edges using a specific tree traversal algorithm.

Remember that a principal plane is used to approximate a region mesh. Let \( \vec{v}_i, i = 1, ..., m \) be a set of vertices collected from projecting the vertices \( \vec{v}_i, i = 1, ..., m \) of a region \( P \) onto its principal plane \( H \). The coordinates of \( \vec{v}'_i \), shown in Fig. 6(a), can be obtained from

\[ \vec{v}'_i = \vec{v}_i - d_H(\vec{v})n_H, \quad i = 1, ..., m \]  \hspace{1cm} (16)

where \( n_H \) is the unit normal vector of the principal plane \( H \). All the projected vertices \( \vec{v}'_i, i = 1, ..., m \) of \( P \) are on the common plane \( H \), shown in Fig. 6(b).

The proposed boundary coding algorithm starts encoding the boundary \( B \) of a region by projecting the region onto its principal plane \( H \). The projected boundary \( B' \) of the region forms a closed loop on \( H \) by linking up the projected boundary edges. The boundary may have an orientation, in which case, every boundary edge is a directed edge, with a start and end vertex. Let \( \vec{v}_i \) be a vertex of \( B \), and \( \vec{v}'_i \) be...
the absolute value of clockwise fashion. During this scan, boundary coding records the relationship between the original vertices \( \bar{v}_i \), \( i = 1, \ldots, m \) and their corresponding vertices \( \bar{v}'_i \), \( i = 1, \ldots, m \) on \( H \); (b) \( H \) is the common plane for the projected vertices \( \bar{v}'_i \), \( i = 1, \ldots, m \).

![Fig. 6. Projecting vertices of a region \( P \) onto its principal plane \( H \): (a) the relationship between the original vertices \( \bar{v}_i \), \( i = 1, \ldots, m \) and their corresponding vertices \( \bar{v}'_i \), \( i = 1, \ldots, m \) on \( H \); (b) \( H \) is the common plane for the projected vertices \( \bar{v}'_i \), \( i = 1, \ldots, m \).](image)

Fig. 7. Boundary description.

The VCC to encode the projected boundary \( B' \) can be further transformed into the DifferentialVertexChainCode (DVCC) by subtracting the polar coordinates of \( \bar{v}'_i \) from those of its previous vertex \( \bar{v}'_{i-1} \) to remove redundant VCC information. In this case, the boundary description code for \( B' \) is changed into

\[
(\bar{u}_0)(r'_1, \theta'_1)(\text{DVCC}_B)
\]

where \( \text{DVCC}_B \) is defined as

\[
\text{DVCC}_B = ((\Delta r'_i, \Delta \theta'_i)|\Delta r'_i = r'_i - r'_{i-1},
\]

\[
\Delta \theta'_i = \theta'_i - \theta'_{i-1}, i = 2, \ldots, |B'|.
\]

The values of \( \Delta r'_i \) and \( \Delta \theta'_i \) in \( \text{DVCC}_B \) are relatively small compared with \( r'_i \) and \( \theta'_i \) in VCC, and thus representing \( B' \) in terms of \( \text{DVCC}_B \) has better compression result. This work encodes DVCC for each projected boundary using the well-known variable length arithmetic coding [32] to further improve compression rate.

To complete the discussion of the proposed boundary coding, for each region, the total information to record includes (1) the normal vector \( \bar{n}_H \) of the principal plane, (2) the depth value sequence for each vertex, (3) and the boundary description code defined in (19).

In order to zip the joint boundaries, the shared edges and vertices are indexed at both the encoder and decoder ends. A boundary separating two regions is also encoded twice. The current work reconstructs the joint boundary using the same rule, so that the decoder zips the two regions seamlessly without additional information.

4.2. Internal connectivity–geometry coding

For each region, the internal connectivity–geometry code describes the topology structure within the region in detail. The procedure to encode the internal connectivity and geometry information simultaneously for a region is similar to that used to encode the boundary. Once again, the principal plane of each region can be used to project every vertex of the region onto the same plane. The vertex scanning technique mentioned above can be applied to visit and encode all the internal vertices and edges.

This research chooses the simple triangle mesh of Fig. 8 to illustrate how the proposed internal connectivity–geometry algorithm spans the mesh with successive circle scanning. The mesh shown in Fig. 8 is the version of projecting vertices as well as their edges on the principal plane of a region. This work conducts the encoding scheme on this plane.

Initially, the algorithm supposes all mesh boundary vertices (1–13) are encoded in the boundary encoding stage (cf. Fig. 8(a)). To conduct the circular canning procedure for encoding internal connectivity and geometry information, we need to select an internal vertex as the starting reference vertex consisting of two vertices—one of which is internal. Let vertices (15,13) constitute the starting reference vertices (cf. Fig. 8(b)). Starting with this reference vertex, the internal connectivity–geometry coding algorithm then recursively and circularly visits the internal vertices. The seed vertex 15 visits the vertices (13,19,17,16,14) in clockwise direction (Fig. 8(b)). After this step, the algorithm outputs the internal connectivity–geometry description code

\[
(\bar{v}_{15})(\bar{v}_{13})(T, \Delta \bar{r}_{13}^{19-15}, \Delta \bar{r}_{13}^{19-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{19-15}, \Delta \bar{r}_{15}^{19-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{19-15}, \Delta \bar{r}_{15}^{19-15})
\]

(20)

\[
(\bar{v}_{15})(\bar{v}_{13})(T, \Delta \bar{r}_{13}^{19-15}, \Delta \bar{r}_{13}^{19-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{17-15}, \Delta \bar{r}_{15}^{19-15}, \Delta \bar{r}_{15}^{19-15})
\]

(20)
where $\vec{v}_i$ is the coordinate of vertex $i$, $\Delta r_{i-s} = \|\vec{v}_i - \vec{v}_s\| - \|\vec{v}_j - \vec{v}_s\|$ is the norm difference between vectors $\vec{v}_i - \vec{v}_s$, $\vec{v}_j - \vec{v}_s$, and $\theta_{i-s} = \cos^{-1}(\langle\vec{v}_i - \vec{v}_s, \vec{v}_j - \vec{v}_s\rangle) / \|\vec{v}_i - \vec{v}_s\| \|\vec{v}_j - \vec{v}_s\|$ is the angle between vectors $\vec{v}_i - \vec{v}_s$ and $\vec{v}_j - \vec{v}_s$, and $T=0$ indicates that the accompanying information for each braced part is used for reconstructing the vertex coordinates. Note that every visited vertex $i$ has an edge connecting to the seed vertex $s$ and every pair of consecutive visited vertices has an edge. This study maintains a vertex ordered set $S$ consisting of internal vertices visited in the previous circular scanning processes. After the first run, $S = (13, 19, 17, 16, 14)$. The vertex in $S$ is one of the seed vertex candidates to start a circular scan.

Next, we select the first internal vertex visited by the previous scan process as the next seed vertex. In this case, 13 will be the seed vertex.
vertex for the next run to scan unvisited vertices. In S, vertex 13 has two neighboring vertices 14 and 19. Again, we scan the vertices (4,3,2,1) from the edge between 14 and 13 to the edge between 19 and 13 (Fig. 8(c)) and output the internal connectivity-geometry description code
\[
(T, 4) (T, 3) (T, 2) (T, 1)
\]
where \( T = 1 \) indicates that the accompanying information for each braced part is a vertex index whose coordinates have been coded. In this case, (4,3,2,1) are boundary vertices coded in the above boundary coding procedure. After the step, vertex 13 will be removed from S and vertex 14 is the next seed. Further steps in the conquest are shown in Fig. 8(d)–(i).

To reconstruct the input mesh, the reconstruction algorithm initially restores the boundary (0–12) using the boundary description code (Fig. 9(a)). Now from the first internal connectivity-geometry description code defined in (19), the algorithm restores the internal nodes (13,19,16,17,14) (Fig. 9(b)), resulting in setting the vertex 13 as the new seed node and the edges from 14 to 13 and from 19 to 13 form the two sides of the next restoration run. The successive reconstruction of fans continues until the entire mesh is reconstructed as shown in Fig. 9(c)–(i).

The internal connectivity-geometry coding algorithm visits the input region vertices on its principal plane. For each vertex, the depth value should be added into the coding stream to restore its original coordinates. The projected region is on a 2D space if we rotate the region by aligning the axis \( z \) with the normal vector of the principal plane as shown in Fig. 10.

5. Experimental results

To evaluate the effectiveness of the proposed segmentation-based compression, the current study implements a 3D mesh model compression system for geometric and connective encoding/reconstruction of meshes on a PC with Pentium IV 1.8 GHz CPU running the Windows XP operating system. The experiments used a set of 3D mesh models (Fig. 11) for performance measurements that are freely available on the Internet.

This investigation first tested effectiveness of the proposed mesh segmentation algorithm using the hybrid of \( k \)-means clustering and the multiple principal planes analysis. Fig. 12 shows the segmentation result of this approach applied to a noisy simple “triceratops” model with 90% vertices moved along the surface normals. Given the segmentation result of the original mesh model \( M \) as the ground truth, we might derive a simple criterion \( SP \) to verify robustness of the proposed mesh segmentation algorithm. \( SP \) is defined as
\[
SP = \frac{1}{|M|} \sum_{i=1}^{k} |R_i \cap \tilde{R}_i|
\]
where \( |M| \) is the number of vertices in \( M \) and \( |R_i \cap \tilde{R}_i| \) is the number of vertices which are both within the segmented region \( R_i \) for the original mesh and \( \tilde{R}_i \) for the noise-added mesh model. The value of \( SP \) is high when the segmented regions of a noise-added mesh are almost the same as those of the original mesh. In the case of Fig. 12, the value of \( SP \) achieves 99%. Thus, the proposed segmentation algorithm is robust in resisting noise.

The current segmentation procedure produces the correct segmentation of the model. The number of regions segmented from a complex mesh model affects precise compact representation. Fig. 13 shows that varying the number of segments \( k_{max} \) used in the segmentation procedure leads to adaptive mesh smoothing with simultaneous sharpening of surface creases at different geometric scales. Fig. 14 shows distributing reconstruction errors of Fig. 13(a) using segmented regions to approximate the original input mesh. Accordingly, the proposed mesh segmentation procedure provides accurate compact representation to approximate the input mesh. According to (15), we can calculate the reconstruction error for an input mesh. Fig. 15 shows other segmentation results for different mesh models, and Table 1 lists reconstruction errors using the segmented regions to approximate the input models.
Second, we compress the input mesh models by encoding the segmented regions one by one with the proposed compression algorithm. Fig. 16 shows the compression results consisting of a series of region encoding steps for “Nefertiti”, and Fig. 17 shows the corresponding reconstruction steps. Fig. 18 shows a simulation of the progressive region-by-region transmission. The user can choose to resume a particularly interested region (Fig. 18(a)) or restore the whole object using an interactive loop until reconstructing the whole object (Fig. 18(b)). The proposed compression scheme uses variable length arithmetic coding for further compressing the geometry codes and both the lossy and lossless geometry coding modes are offered in the system. The relative demonstrations are shown in Fig. 19.

Finally, in order to further verify effectiveness of the proposed method, this work also simulates the Dual Parallelogram Prediction (DPP) method [18] and Asymptotic Closed Loop Vector Quantization (ACL VQ) scheme [14] for performance comparison. Either one focuses on the design of geometry coding of 3D meshes. Table 2 lists the results of compression rate and running time during coding and decoding for each test model using different methods. The bpvs listed in Table 2 encodes both the connectivity and geometry information for the proposed method, but the bpvs for the DPP method and the ACL VQ method include only geometry information with vertex positions being pre-quantized with 10 bits per coordinate (bpc). The connectivity information contributes an additional 1–2 bpvs to
Fig. 15. The segmentation examples: (a) “body” segmented into two regions; (b) “nefertiti” segmented into three regions; (c) “triceratops” segmented into five regions; (d) “tiger” segmented into six regions.

Fig. 16. Region-by-region encoding stages of the “nefertiti” model: (a)–(c) are a series of circular scan processes to encode the first region and (d)–(h) are the connectivity–geometry encoding step to compress regions 2 and 3.

Table 1
Reconstruction errors using the segmented regions to approximate the input models.

<table>
<thead>
<tr>
<th>Models</th>
<th>$k_{max}$</th>
<th>Construction errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nefertiti</td>
<td>4</td>
<td>0.1320</td>
</tr>
<tr>
<td>Body</td>
<td>4</td>
<td>0.0968</td>
</tr>
<tr>
<td>Tiger</td>
<td>13</td>
<td>0.0179</td>
</tr>
<tr>
<td>Bunny</td>
<td>19</td>
<td>0.0228</td>
</tr>
<tr>
<td>Triceratops</td>
<td>9</td>
<td>0.2148</td>
</tr>
</tbody>
</table>

Fig. 20 shows compression performance comparison for different methods using the test models in terms of compression rate and distortion error. Fig. 21 shows reconstruction quality comparison for the “Tiger” model using different methods with nearly the same compression rate.

The proposed circular scanning process for visiting vertices is similar to the fan scanning process in the Advancing Fan-Front (AFF) method proposed by Mudur et al. [11]. The AFF traverses the fans of a triangle mesh using a boundary vertex and two boundary edges as
Fig. 17. Region-by-region reconstruction stage of the “nefertiti” model: (a)–(c) are the reconstruction steps to restore region 1 and (d)–(h) are the reconstruction steps to restore regions 2 and 3.

Fig. 18. Simulation of the progressive region-by-region transmission: (a) particularly interesting region selected by user and (b) whole object step by step.

Fig. 19. Demonstration of lossy and lossless geometry coding: (a) original model; (b) lossless coding and (c)–(e) lossy coding under 16, 12, 8 bits per vertex.

The initial seed and starting gates, respectively. The proposed scheme chooses an interior vertex as the initial seed for traversing the fans of the triangle mesh. The interior vertex as the initial seed reduces the number of fans in both the encoding and reconstruction stages. The proposed algorithm also encodes both the connective and geometry information simultaneously compared with AFF which only
Table 2
Performance comparison for mesh compression using the proposed method A, DPP method B [18], and ACL VQ method C [14].

<table>
<thead>
<tr>
<th>Model</th>
<th>Vertices</th>
<th>Faces</th>
<th>No. Fans in A</th>
<th>Method A</th>
<th>Method B</th>
<th>Method C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GC+IC</td>
<td>T</td>
<td>GC</td>
</tr>
<tr>
<td>Nefertiti</td>
<td>299</td>
<td>562</td>
<td>145</td>
<td>15.04</td>
<td>0.012</td>
<td>15.61</td>
</tr>
<tr>
<td>Body</td>
<td>711</td>
<td>1796</td>
<td>354</td>
<td>12.22</td>
<td>0.053</td>
<td>13.12</td>
</tr>
<tr>
<td>Tiger</td>
<td>5169</td>
<td>10,061</td>
<td>2223</td>
<td>7.78</td>
<td>0.452</td>
<td>9.23</td>
</tr>
<tr>
<td>Bunny</td>
<td>34,834</td>
<td>69,451</td>
<td>16,098</td>
<td>7.23</td>
<td>3.321</td>
<td>7.47</td>
</tr>
<tr>
<td>Triceratops</td>
<td>3,124</td>
<td>5,680</td>
<td>1,254</td>
<td>10.56</td>
<td>0.245</td>
<td>11.32</td>
</tr>
</tbody>
</table>

GC is the geometry coding in terms of bits per vertex (bpv), IC is the connectivity coding, and T is the average execution time in seconds to coding and decoding a model.

6. Conclusion

This paper proposes a segmentation-based 3D mesh compression. Our contributions lie in three aspects. Firstly, fusing the well-known k-means clustering and the proposed principal plane analysis to separate the input 3D mesh into a set of disjointed polygonal regions is efficient and robust. Secondly, we compress the mesh object into independent regions and a boundary, so the decoder obtains each individual part and restores the original object. This is especially useful in some interactive applications such as Internet gaming and remote training in a virtual environment. The user can restore the particularly interested part(s) or the whole object step by step. Finally, the proposed circular scanning procedure provides a new method to encode connectivity and geometry information of the input mesh simultaneously. Traditional methods belonging to either connectivity coding or geometry coding do not offer a total solution to the problem of 3D mesh compression.

Experimental results show that our scheme provides better compression than earlier known techniques. Our future study will focus on content-based 3D model retrieval in the compression domain and error resilience during triangular mesh reconstruction.
References


About the Author—SHIYI-CHYI CHENG received the B.S. degree from National Tsing Hua University, Hsinchu, Taiwan in 1986, and the M.S. and Ph.D. degrees in Electronics Engineering and Computer Science and Information Engineering in 1988 and 1992, respectively, both from National Chiao Tung University, Hsinchu, Taiwan. From 1992 to 1998, he was a Technical Staff at the Chunghwa Telecom Laboratories, Taoyuan, Taiwan. He joined the faculty of Department of Computer and Communication Engineering, National Taiwan Ocean University, Keelung, Taiwan. His research interests include multimedia databases, image/video compression and communications, and intelligent multimedia systems.

About the Author—CHEN-TSUNG KUO received the Ph.D. degree in Computer and Communication Engineering from National Kaohsiung First University of Science and Technology, Taiwan in 2007. Since 1995, he has served as a Staff at the Computer Center of the Longchuan Veterans Hospital, Pingtung, Taiwan. His research interests include image processing, computer graphics, data mining and decision support system.

About the Author—DA-CHUN WU received the Ph.D. degree in Computer Science from National Chiao Tung University, Taiwan in 1996. He is currently an Associate Professor with the Department of Computer and Communication Engineering, National Kaohsiung First University, Kaohsiung, Taiwan. His research interests include multimedia databases, image/video watermarking, and digital right management systems.