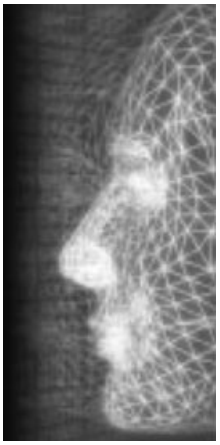
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2009; **20**: 343–354
Published online 11 June 2009 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.319

TFAN: A low complexity 3D mesh compression algorithm



By Khaled Mamou, Titus Zaharia and Françoise Prêteux*

This paper proposes a novel approach for mono-resolution 3D mesh compression, called TFAN (Triangle Fan-based compression). TFAN treats in a unified manner meshes of arbitrary topologies, i.e., manifold or not, oriented or not, while offering a linear computational complexity (with respect to the number of mesh vertices) for both encoding and decoding algorithms. In addition, the TFAN compressed representation is optimized for real-time decoding applications. In order to validate the proposed approach, two databases have been considered for experimentations. The first is the MPEG-4 test set, which includes over 3500 general purpose manifold meshes. The second, related to the French national project SEMANTIC-3D, includes over 4000 computer assisted design (CAD) meshes of highly irregular, non-manifold topologies. In both cases, the TFAN approach outperforms existing techniques such as MPEG-4/3DMC (3D Mesh Coding) or Touma and Gotsman, with decoding times lower by an order of magnitude at equivalent or even better levels of compression efficiency (±10% in bitrate). In addition, when applied to non-manifold 3D data, the compression performances are significantly enhanced (6–30% gain in bitrate). Due to its high compression performances the TFAN approach has been recently retained for ISO standardization, within the framework of the MPEG-4/AFX standard. Copyright © 2009 John Wiley & Sons, Ltd.

Received: 26 March 2009; Accepted: 27 March 2009

KEY WORDS: 3D mesh compression; real-time decoding; low complexity; MPEG-4 standard

Introduction

Today's multimedia applications are more and more featuring 3D content within various, both general public and professional industrial contexts. Computer assisted design (CAD), e-medicine, video gaming, CGI (computer generated imagery) films, special effects, and cartoons are some representative examples of industrial domains where the issue of 3D modeling and representation plays a central role.

Most of the time, 3D content is represented as 3D meshes, which offers the advantage of both generality and interoperability in multi-platform environments. However, the drawback of such a representation is related to the high bandwidth and storage requirements involved, since modeling complex, realistic 3D objects might need meshes with thousands of vertices. Elaborating efficient, dedicated 3D mesh compression algorithms then becomes a crucial challenge that needs to be treated and solved.

This issue has been extensively addressed in the literature, as testified by the rich and diverse set of methods proposed over the last two decades, which includes both mono and multi-resolution approaches. As in this paper we will solely consider the mono-resolution framework, let us first analyze the most representative mono-resolution 3D mesh compression techniques. For a more general overview of 3D mesh compression algorithms, the reader is invited to refer to Reference.[1]

Related Work

Historically, the first 3D triangular mesh compression algorithms aimed at reducing the transmission bandwidth between the CPU and the graphics card of a computer. The basic principle of these hardware-oriented compression methods consists of decomposing

*Correspondence to: T. Zaharia, ARTEMIS Department, Institut TELECOM, TELECOM SudParis, 9 Rue Charles Fourier, 91011 Evry Cedex, France. E-mail: titus.zaharia@it-sudparis.eu

the mesh into sequences of adjacent triangles, the so-called triangle strips (TS), which can be encoded with less redundancy (about one vertex index per triangle) than the uncompressed indexed face set representation used for instance in Reference[2] (which requires three-vertex indices per triangle).

In his pioneering work, Deering[3] introduces the generalized triangle mesh (GTM) representation which combines a generalized triangle strip with a 16-vertex buffer. Once a vertex is stored in this buffer, it can be accessed by a 4 bits index. Under the assumption of reusing each vertex from the buffer only once, Taubin and Rossignac[11] proved that GTM codes the mesh connectivity at about 11 bits per vertex (bpv).

However, Deering did not provide any decomposition scheme to convert an arbitrary mesh into the GTM representation. Subsequent work has been so dedicated to finding some optimal GTM. Chow[5] was the first to propose a GTM decomposition algorithm which optimizes the Deering's vertex Buffer use. The challenging issue of finding an optimal GTM decomposition was also studied in References[6,7] and some heuristic solutions were provided in References.[8,4] In Reference[9], the authors optimize the GTM representation by introducing the concept of vertex chains and by applying entropy encoding, which provides up to 32% gain in compression ratios.

The TS based techniques are generic (i.e., handle directly arbitrary topologies) and are adapted for real-time hardware decoding. However, they result in very low compression rates.

In Reference[10], the authors propose the shared vertex analysis (SVA) technique. Here, instead of decomposing the mesh into a set TS, the authors encode directly the facets indices by considering four configurations, which indicate the number of vertices shared between the current triangle and the last decoded one. Let us note that the SVA technique preserves the original order of the mesh triangles and vertices, which makes it less efficient that the GTM approach.

Taubin and Rossignac[11] introduced the topological surgery (TS) coding scheme. Inspired by the work of Turan,[12] authors propose to represent the connectivity of a manifold triangular mesh as two dual spanning trees: the vertex tree and the face tree. The two trees are scanned following a deep-traversal rule and run-length encoded. The geometry information is quantized, linearly predicted (each vertex from his ancestors in the vertex tree), and the residual errors entropy encoded. Experimentally, the TS algorithm compresses the mesh connectivity at 2.48–7.0 bpv.

Let us note that the TS method is not suited for hardware implementation since it requires a random access to the mesh vertices. Because of its simplicity, the TS compression scheme has been adopted within the MPEG-4 compression standard,[13] under the name of MPEG-4/3DMC (3D Mesh Coding).

In contrast with the spanning tree based representations, the ''Cut-Border Machine'' [14] (CBM) uses a triangle conquest approach to code the connectivity information. At each step, one triangle is conquered and the so-called op-code (operation code) is output. The connectivity information can be recovered from the op-code sequence which is entropy coded.

Gumhold[15] improves the CBM connectivity encoding performances (initially of 3.22–8.94 bpv) by using an adaptive arithmetic coder and optimizing the border encoding. The reported connectivity compression rates range from 0.3 to 2.7 bpv.

The CBM approach provides high compression ratios while enabling real-time decoding of 3D meshes. However, it handles solely manifold meshes, which limit its applicability.

As CBM, the valence-driven approach introduced by Touma and Gotsman[16] (TG) defines a mesh traversal based on a conquest approach. The innovative principle of the TG encoder consists of successively outputting the valences of the conquered vertices. The encoder proceeds by maintaining a list of vertices (called active list) defining a polygon separating the conquered vertices from the unconquered ones. At each step, one vertex of the active list is processed and its valence is output. Particular cases corresponding to self-intersections of the active list are handled by applying some specific ''split'' and ''merge'' operations, encoded by dedicated exception codes. The connectivity of the mesh is completely represented by the sequence of valences and exception codes.

Since the valence distribution is concentrated around a mean value of 6 (with a relatively low dispersion), the corresponding valence sequence can be efficiently compressed with an arithmetic encoder. The TG encoder achieves a compression ratio of 0.2 bpv for the connectivity of regular meshes and 2–3.5 bpv otherwise.

The TG approach optimizes the coding efficiency at the cost of a limited applicability (i.e., handles only oriented manifold 3D meshes). Moreover, decoding the TG stream requires maintaining multiples active lists, which requires expensive (in terms of execution times) random memory accesses.

Long time considered as state of the art for mono-resolution compression methods, the TG method has been further optimized in Reference.[17]

DOI: 10.1002/cav

The main drawback of existing methods is related to the lack of generality, since most of the methods strictly apply to manifold meshes. For dealing with generic, non-manifold meshes, some optimization/conversion procedures (such as those proposed in Reference[18]) are required prior to applying 3D mesh compression algorithms. However, such procedures are costly in computation time and irreversibly degrade the original 3D models, which is not acceptable in certain applications (e.g., CAD, computer animation etc.) where content creators need to preserve the original data.

In addition, with the explosive evolution of today's portable devices, additional requirements and functionalities should be considered and treated.

New Trends and Objectives

With the recent and massive proliferation of light devices (smart phones, palm computers, PDAs etc.), which are becoming more and more popular, and within the framework of convergence of technologies, 3D content should be today available on such low complexity, low power, and low cost terminals. This leads to additional requirements for 3D compression algorithms. In particular, the issue of real-time decoding and rendering of complex 3D data on portable devices becomes a crucial challenge for numerous industrial applications (e.g., online gaming). Let us note that the main constraints apply especially at the decoding level, since the encoding is performed most of the time off-line and on dedicated servers. On the contrary, fast decoding capabilities are critical for visualizing 3D data at interactive, real-time rates.

The challenge of elaborating fast decoding 3D compression algorithms has been recently considered by the MPEG-4/3DGC (3D Graphics Compression) group, within the framework of scalable complexity compression Core Experiment (CE),[19] which specifically aims at elaborating 3D compression algorithms dedicated and adapted to real-time applications on light terminals.

The TFAN (Triangle Fan-based compression) 3D mesh compression method proposed in this paper has been developed within the framework of this international effort of standardization. TFAN deals with 3D meshes of arbitrary topologies, while ensuring a low decoding complexity. The rest of the paper is organized as follows. The TFAN method is described in detail in Section 2, with both encoding and decoding algorithms. Section 3 proposes an experimental evaluation of the proposed TFAN approach, which is compared to both MPEG-4/3DMC and TG algorithms. Finally, Section 4 concludes the paper and opens some perspectives of future work.

The TFAN Approach

The TFAN approach exploits the concept of triangle fan (TF),[20] described below.

Triangle Fan: Definition and Properties

A TF of degree d is an ordered set of d triangles, denoted by $(t_j)_{j\backslash in\{0,\ldots,d-1\}}$ and defined by a sequence of $(d+2)$ vertices $(v_0,v_1,\ldots,v_{d+1})$ such that

$$\forall j \in \{0,\ldots,d-1\},\ t_j = \{v_0, v_{j+1}, v_{j+2}\}. \quad (1)$$

By definition, the triangles of a TF satisfy the following properties:

- (P1): each two successive triangles in the triangle are neighbors, in the sense that they share a common edge,
- (P2): all the triangles of a TF have the same orientation,
- (P3): all the triangles of a TF share a common vertex v_0, the so-called the center of the TF.

Let us recall that the *orientation* of a triangle is defined by the traversal order of its vertices. Two neighboring triangles have the same orientation if their common edge is traversed in opposite direction in the two triangles. A mesh is called *oriented* if the set of its triangle has a unique orientation.

Let us also note that the orientation of triangles involves a unique traversal order of the TF vertices. A TF is then completely determined by the ordered sequence of its vertices $(v_0, v_1, v_2,\ldots, v_{d+1})$, listed starting from the central vertex v_0.

The TF concept is particularly useful for deriving a compact representation of the mesh connectivity, as described in the following section.

Encoding of the Mesh Connectivity

The TFAN encoding algorithm is based on a traversal of the mesh vertices from neighbor to neighbor. At the beginning, the whole set of mesh vertices is considered as non-visited. A FIFO (first in first out) queue structure, denoted by F, is used here for successively storing the mesh vertices.

Initially, the queue F is fed by an arbitrary vertex. At each stage, a vertex is extracted from F. The set of its incident triangles is then partitioned into a set of TFs.

DOI: 10.1002/cav

The vertices of each TF are subsequently traversed, inserted into the queue F, and marked as visited. The process stops when the queue F becomes empty and all the mesh vertices marked as visited.

In addition, a counter, initialized to zero and incremented at each time a new vertex is extracted from the queue F, is considered. Its value is associated with the current vertex extracted from the queue and defines the traversal order (or index) of the considered vertex in the new representation. Let v_j be the jth vertex extracted from the queue. Its traversal order, denoted by $O(v_j)$, will be then equal to j.

Several different strategies may be considered for partitioning the set of triangles incident to a considered vertex v_j into a set of TFs. In our work, we have adopted an iterative approach, described below.

Partition into a Set of Triangle Fans

Let us consider the case of the jth vertex extracted from the queue F, denoted by v_j. At each stage n ($n \geq 1$), a triangle fan $\mathrm{TF}_n(j)$ is created starting from the triangle $t_0(j)$ incident to v_j having a minimum number of neighboring triangles (i.e., triangles sharing and edge with $t_0(j)$), non-visited and with the same orientation as $t_0(j)$.

The triangle $t_0(j)$ is then added to $\mathrm{TF}_n(j)$ and marked as visited. If $t_0(j)$ has non-visited neighbor triangles with the same orientation and incident to v_j, then a triangle $t_1(j)$ is randomly selected, added to $\mathrm{TF}_n(j)$ and marked as visited. The same procedure is subsequently applied to $t_1(j)$ and the process is reiterated until obtaining a triangle $t_1(j)$ with no neighbors of the same orientation and non-visited.

This partition procedure is inspired from the work of Akeley *et al*.[21] In practice, such a heuristics ensures most of the times the detection of a minimal number of TFs.

Let us denote by

- $(\mathrm{TF}_n(j))_{n\in n\{1\ldots,N(j)\}}$ the set of $N(j)$ TFs associated with the vertex v_j,
- $\{v_j, w^n{}_j(1), w^n{}_j(2), \ldots, w^n{}_j(d(j,n)+1)\}$ the ordered set of vertices of the TF $\mathrm{TF}_n(j)$, and $d(j,n)$ its degree.

Let us now detail the traversal procedure of the mesh vertices as well as the construction of the TFAN representation.

Construction of the TFAN Representation

Let us denote by $L(j)$ the set of vertices sharing with v_j at least a visited triangle and with a traversal order higher than $O(v_j)$. The set $L(j)$ is then ordered increasingly by considering the following order relation:

$$\forall\{w_1, w_2\} \in L(j), \quad w_1 \leq w_2 \Leftrightarrow \mathrm{O}(w_1) \leq \mathrm{O}(w_2). \quad (2)$$

Excepting the central vertex v_j, all the vertices of $\mathrm{TF}_n(j)$ are treated iteratively, by considering the natural traversal order of the TF. To each vertex $w^n{}_j(k)$, a binary value $s^n{}_j(k)$ is associated. The value $s^n{}_j(k)$ indicates if $w^n{}_j(k)$ has been already visited ($s^n{}_j(k)=0$) or not ($s^n{}_j(k)=1$). Let $S(j,n)$ denote the binary vector composed of the set of $s^n{}_j(k)$ values related to the considered triangle $\mathrm{TF}_n(j)$.

If the vertex $w^n{}_j(k)$ is non-visited, then it is marked as visited, inserted into the queue F, and added at the list $L(j)$. On the contrary, if the vertex $w^n{}_j(k)$ is already visited, two different cases can occur. If $w^n{}_j(k)$ belongs to $L(j)$ then the relative index of $w^n{}_j(k)$ in the ordered set $L(j)$ is stored into an additional structure $I(j,n)$. If not, the negative value $\mu^n{}_j(k) = O(v_j) - O(w^n{}_j(k))$ is stored into $I(j,n)$.

Let us note that it would be possible to code the indices of already visited vertices without using the ordered set $L(j)$, by directly specifying their values. However, such an approach would be by far less appropriate for compression purposes. By exploiting the ordered set $L(j)$, the indices of already visited vertices are expressed locally, by using relative values with respect to the central vertex. Thus, the number of vertices in $L(j)$ is always inferior to the valence of the v_j vertex. As the valence of a given vertex is generally negligible with respect to the total number of mesh vertices, the relative indices vary in a range which is much narrow than that of the absolute vertex indices. By using such relative values, we expect to obtain a more efficient compression of the TFAN representation.

Let us recall that, by definition of the ordered set $L(j)$, all its vertices have an index superior to $O(v_j)$. Consequently, the quantity $\mu^n{}_j(k)$ is always negative. This property will be exploited by the decoder in order to determine if the index of vertex should be or not searched in the list $L(j)$.

Once all the vertices of the TF $TF_n(j)$ treated as described above, the procedure is continued until all the mesh vertices are visited. If at any moment the queue F becomes empty and if there are non-visited vertices left, then the first non-visited vertex is inserted to F and the process is reiterated.

The TFAN representation associated with each TF $TF_n(j)$ is defined as its degree $d(j,n)$ and the two sets $S(j,n)$ and $I(j,n)$. In order to compress such a representation, a straightforward approach would consist in a direct

DOI: 10.1002/cav

arithmetic encoding of all these elements. However, in order to optimize the compression efficiency, we have adopted a different strategy, based on the encoding of a set of pre-defined configurations. The proposed approach is described in the next section.

Configuration-based TFAN Compression

The principle consists of identifying a set of TF configurations, which appear most frequently in practice, and to describe them by some binary compact representations. A number of nine configurations have been thus identified. In order to ensure completeness of the representation, a generic configuration is also introduced. It corresponds to the case where all the elements $d(j, n)$, $S(j, n)$ and $I(j, n)$ describing the TF are directly encoded.

Let $C(j, n)$ be the number of the configuration associated with the triangle fan $TF_n(j)$. Table 1 summarizes the ten proposed configurations. In the associated illustrations, the central vertex is represented in green, while non-visited and visited vertices are colored in red and blue, respectively. Analogously, the non-visited triangles are illustrated in red while the visited ones in blue.

The determined configurations are then encoded with the help of an arithmetic encoder,[22] in order to take into account their statistical properties.

Let us note that for manifold, closed and oriented meshes, solely configurations 8 and 9 are necessary for completely describing the connectivity information. In addition, on the MPEG-4 data set (*cf.* Experimental results), configuration 8 appears in 85% of cases. In addition, for configuration 8, solely the degree of the TF needs to be encoded. This makes it possible an efficient compression of the configuration sequence.

In a certain sense, the TFAN approach can be considered as an extension of the TG[16] compression method. Let us analyze the analogies between the TFAN and TG approaches. Configurations 1 and 8 correspond to the operation ADD, with the difference that here the degree of the TF is encoded instead of the vertex valence. The TFAN approach introduces two configurations in order to handle non-orientable meshes. Configurations 2 and 9 correspond to the TG operations of MERGE and SPLIT. As in, Reference[16] the encoding of some vertex indices associated with auto-intersections of the active list (i.e., the TG equivalent of the queue F) is required here. Let us note that, contrary to the TG approach, the TFAN method uniquely needs a single list, represented by the queue F. This makes it possible to significantly reduce the amount of memory necessary for the encoding process. In addition, such an approach avoids testing for intersections between multiples lists, which translates in a gain in time at the decoder side.

The TG approach uniquely applies to manifold, closed and oriented meshes. In order to deal with open surface, the introduction of an additional dummy vertex is needed. On the contrary, by defining additional configurations, the TFAN approach makes it possible to manage arbitrary topologies. Configurations 3–6 are dedicated to TFs including border vertices. Configuration 7 is exploited in the case of irregular vertices[1] (i.e., has no neighborhood homeomorphic to an open disk or to a half disk).

In the case of the SEMANTIC-3D corpus of non-manifold meshes (*cf.* Experimental results), configurations 1–9 have been applied in 87% of cases. The remaining 13% TF cases have been described by the generic configuration 10.

Connectivity Decoding Process

The TFAN decoder reconstructs the mesh connectivity by successively decoding the set of TFs transmitted. The mesh vertices are traversed in the same order as the one established during the encoding process.

Thus, at each stage j, the set of TFs $(TF_n(j))_{n\in n\{1\ldots,N(j)\}}$ associated with the current vertex j are reconstructed as follows. First, the ordered set $L(j)$ of vertices neighbors to j and with an index greater than j is determined. The number of TFs $N(j)$ is read from the bitstream. The TFs are then successively generated in the order they were encoded. More precisely, in order to reconstruct the TF $TF_n(j)$, the decoder recovers from the bitstream the following information:

- The degree $d(j, n)$ of the TF,
- The binary vector $S(j, n)$, indicating the set of already visited vertices,
- The relative index vector $I(j, n)$.

Initially, the TF $TF_n(j)$ contains uniquely the j vertex. The other vertices are successively added, in the order of their encoding. Let $w^n_j(k)$ be the kth vertex of the TF.

In order to determine if $w^n_j(k)$ is a novel vertex or an already decoded one, the TFAN decoder extracts from the binary vector $S(j, n)$, the corresponding bit $s^n_j(k)$. If $s^n_j(k) = 0$, then $w^n_j(k)$ is a novel vertex that is created. An index, equal to its traversal order, is then created and associated with $w^n_j(k)$. This index is then added to the TF

DOI: 10.1002/cav

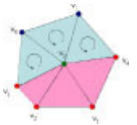
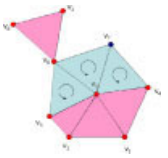
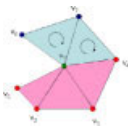
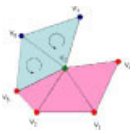
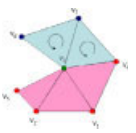
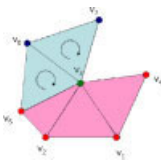
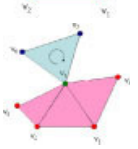
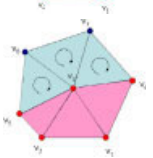
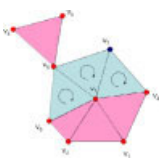
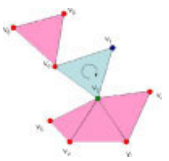
Configuration	Initial TFAN information	Encoded information	Illustration
$C(j,n)=1$	$d(j,n), S(j,n)=\underbrace{\{1,0,0,\ldots,0,0,1\}}_{1+d(j,n)}$ and $I(j,n)=\{1,2\}$	$C(j,n)=1$ and $d(j,n)$	
$C(j,n)=2$	$d(j,n), S(j,n)=\underbrace{\{1,X,X,\ldots,X,X,1\}}_{1+d(j,n)}$ and $I(j,n)=\{1,X,X,\ldots,X,X,2\}$	$C(j,n)=2, d(j,n),$ $S'(j,n)=\underbrace{\{X,X,\ldots,X,X\}}_{d(j,n)-1}$ and $I(j,n)=\{X,X,\ldots,X,X\}$	
$C(j,n)=3$	$d(j,n), S(j,n)=\underbrace{\{0,0,\ldots,0,0,1\}}_{1+d(j,n)}$ and $I(j,n)=\{1\}$	$C(j,n)=3$ and $d(j,n)$	
$C(j,n)=4$	$d(j,n), S(j,n)=\underbrace{\{0,0,\ldots,0,0,1\}}_{1+d(j,n)}$ and $I(j,n)=\{2\}$	$C(j,n)=4$ and $d(j,n)$	
$C(j,n)=5$	$d(j,n), S(j,n)=\underbrace{\{1,0,\ldots,0,0,0\}}_{1+d(j,n)}$ and $I(j,n)=\{1\}$	$C(j,n)=5$ and $d(j,n)$	
$C(j,n)=6$	$d(j,n), S(j,n)=\underbrace{\{1,0,\ldots,0,0,0\}}_{1+d(j,n)}$ and $I(j,n)=\{2\}$	$C(j,n)=6$ and $d(j,n)$	
$C(j,n)=7$	$d(j,n), S(j,n)=\underbrace{\{0,0,\ldots,0,0,0\}}_{1+d(j,n)}$ and $I(j,n)=\{\}$	$C(j,n)=7$ and $d(j,n)$	
$C(j,n)=8$	$d(j,n), S(j,n)=\underbrace{\{1,0,0,\ldots,0,0,1\}}_{1+d(j,n)}$ and $I(j,n)=\{2,1\}$	$C(j,n)=8$ and $d(j,n)$	
$C(j,n)=9$	$d(j,n), S(j,n)=\underbrace{\{1,X,X,\ldots,X,X,1\}}_{1+d(j,n)}$ and $I(j,n)=\{2,X,X,\ldots,X,X,1\}$	$C(j,n)=9, d(j,n),$ $S'(j,n)=\underbrace{\{X,X,\ldots,X,X\}}_{d(j,n)-1}$ and $I(j,n)=\{X,X,\ldots,X,X\}$	
$C(j,n)=10$	$d(j,n), S(j,n)=\underbrace{\{X,X,\ldots,X,X\}}_{1+d(j,n)}$ and $I(j,n)=\{X,X,\ldots,X,X\}$	$C(j,n)=10, d(j,n),$ $S(j,n)=\underbrace{\{X,X,\ldots,X,X\}}_{1+d(j,n)}$ and $I(j,n)=\{X,X,\ldots,X,X\}$	

Table I. The tenTFAN configurations retained (the '*X*' symbol represents an arbitrary value)

DOI: 10.1002/cav

$TF_n(j)$ as well as to the ordered set $L(j)$. The traversal order counter is finally incremented.

In the opposite case, where the bit $s^n_j(k)$ equals 1, the vertex $w^n_j(k)$ is identified as an already decoded vertex. In order to deduce its index, the decoder extracts the element $\mu^n_j(k)$ of $I(j, n)$. If $\mu^n_j(k)$ is strictly positive, the index of vertex $w^n_j(k)$ is obtained by accessing to the element $\mu^n_j(k)$ of $L(j)$. In the opposite case, (i.e., $\mu^n_j(k) < 0$), the index of $w^n_j(k)$ is assigned the value $(j - \mu^n_j(k))$. In both cases, the so-determined index is added to the TF $TF_n(j)$, which is thus updated.

Geometry Compression

The decomposition into TFs is equally exploited in order to set up an efficient prediction strategy of mesh geometry (i.e., 3D position of mesh vertices) and photometric information (e.g., texture, color and normal vectors).

As in Reference[4] for the geometry data, we exploit the so-called parallelogram prediction scheme, which is in our case applied to each pair of successive triangles in a TF. The residual prediction errors are finally arithmetically encoded with the approach described in Reference.[22]

A quantization procedure is applied prior to the prediction stage, in order to avoid cracking problems.[1] Here, the number of bits B used for quantifying the x, y, and z coordinates is used as a parameter to control the quality of the compressed representation. The quantization step is $(D/2^B)$, where D is the size of the object's bounding box.

Concerning the photometric information, the parallelogram prediction rule is replaced by a simple delta prediction, which shows in practice superior compression results.[11]

Experimental Results

In order to validate the proposed approach, we have considered two different test databases, described in the following section.

Test Databases and Evaluation Criteria

The first one corresponds to the MPEG-4 database, which includes some more than 3500 general purpose 3D meshes, with various sizes and shapes. All meshes in the MPEG-4 database are manifold and oriented.

The second corpus has been established within the framework of the French National project SEMANTIC-3D and includes more than 4000 CAD meshes corresponding to the elements of a Renault Laguna car model. As described in, Reference[23] the Laguna corpus includes highly irregular, non-manifold meshes, generally composed of multiple connected components. Because of such topological difficulties, classic 3D mesh compression approaches are ineffective or even completely not applicable to such data.

Figure 1(a–b) illustrates some 3D meshes from the two MPEG-4 and Laguna databases considered.

Concerning the objective performance criteria retained, in order to be able to evaluate compression performances for meshes with arbitrary numbers of vertices, the bitrates are expressed in bpv. As distortion measure, we have retained the RMSE error between initial and decoded mesh, computed with the freely available MESH software.[4]

In order to perform the benchmark, we have considered the MyMultimediaWorld platform (www.mymultimediaworld.com) described in Reference.[24] Specifically designed for benchmarking 3D mesh compression algorithms, MyMultimediaWorld is an online benchmarking platform, adopted for evaluation purposes by the MPEG-4/3DGC group. By offering easy to implement APIs as well as online MPEG-4 based visualization of 3D meshes, and access to 3D mesh databases, MyMultimediaWorld is completely open to the scientific community for testing and evaluating algorithms.

Compression Performances

Figure 1c presents the bitrates obtained upon the MPEG-4 database by using the following three compression approaches: TFAN, MPEG-4/3DMC Extension, and TG. Nine different quantization steps (parameter B between 6 and 14) have been applied for geometry quantization. No photometric attributes have been considered here.

With respect to MPEG-4/3DMC, the TFAN approach enhances the compression efficiency, with an average gain in bitrate of 20%. When compared to the TG approach, TFAN performs slightly worse, with a negative gain of −10%. This is the price to pay for the generality of the TFAN representation, which is specifically designed for dealing with generic, both manifold and non-manifold meshes. When applied to strictly manifold data, we can expect, without surprise, to a slight degradation of performances.

DOI: 10.1002/cav

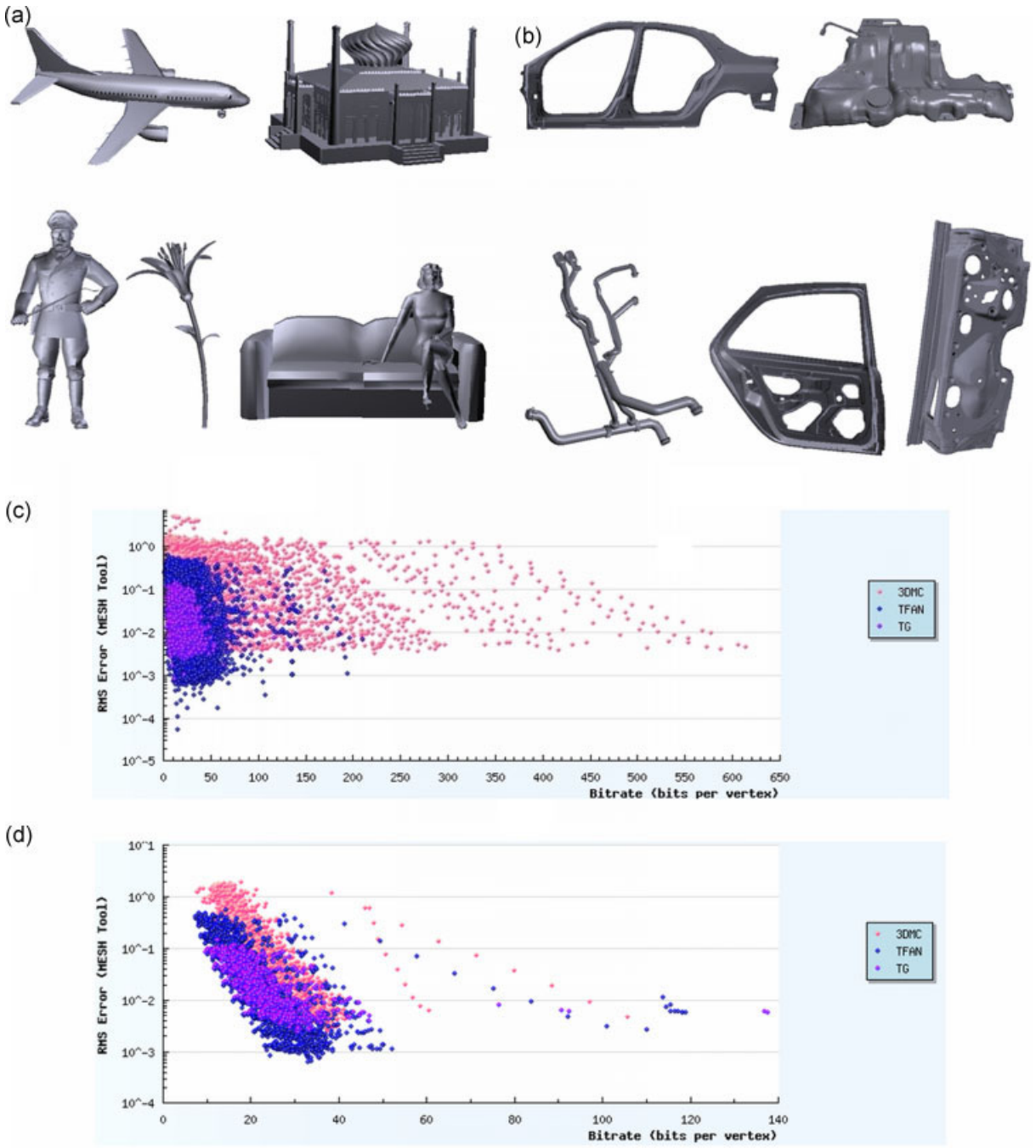


Figure 1. (a) Meshes from the MPEG-4 database; (b) meshes from the SEMANTIC-3D Laguna corpus; (c) compression performances for the MPEG-4 database; (d) compression performances for the SEMANTIC-3D Laguna corpus.

Let us now examine the results obtained on the non-manifold meshes of the Laguna corpus. Under the same experimental conditions as described above, Figure 1d presents the bitrates corresponding to different RMSE errors.

Let us note that the TG approach cannot be applied directly to non-manifold and non-oriented meshes. The algorithm described in Reference[18] has been applied here to the original models, prior to TG compression, in order to convert them into manifold, oriented meshes.

DOI: 10.1002/cav

On this difficult database, the average gains in bitrate obtained by the TFAN approach with respect to the MPEG-4/3DMC technique are of 30%. These gains are explained by the effectiveness of the TFAN representation, able to represent in a compact and generic manner the connectivity of 3D meshes with arbitrary topologies. On the contrary, in the case of non-manifold meshes, the MPEG-4/3DMC technique needs to incorporate into the bitstream some auxiliary stitching information which severely penalizes the compression efficiency.

At the same time, the TFAN approach leads to slightly better compression performances with respect to the TG method, with an average gain in bitrate of 6%. This is due to the conversion of the initial meshes into manifold

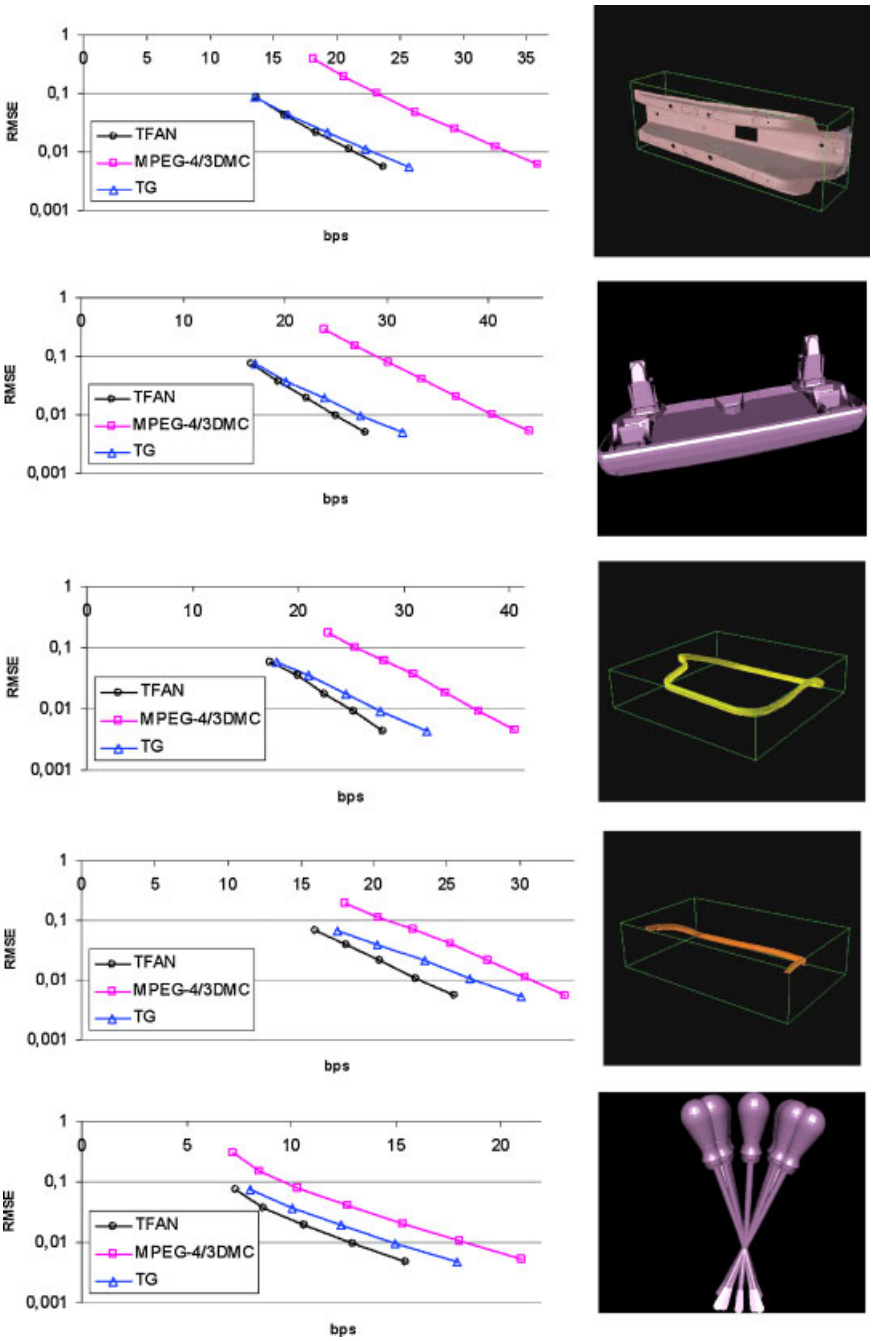


Figure 2. Rate distortion curves for several 3D meshes from the Laguna corpus.

DOI: 10.1002/cav

topologies, which augments the number of vertices by 12% in average. Here, again, dealing directly with arbitrary topologies makes it possible to avoid such problems.

Figure 2 presents the rate–distortion curves obtained on several models of the Laguna corpus. In all cases, the TFAN approach performs better over the whole range of bitrates considered.

Let us now analyze the performances of the proposed compression technique in terms of decoding complexity.

Decoding Complexity

In terms of algorithmic complexity, all the considered methods perform linearly with the number of vertices. In order to objectively analyze and compare the decoding times, we have run all the algorithms under identical conditions, on the same machine (which corresponds to the MyMultimediaWorld server), and with a unique process at the time. The server is an Intel Core2 Duo processor at 2.3 Ghz, with 2Go RAM, running under Linux Fedora Core 9 (Kernel 2.6.25.14-108.fc9.686) operating system.

The decoding times reported in this paper relate uniquely to the decoding procedure and discard input/output operations (e.g., reading of the bitstream from the hard disk).

Table 2 summarizes the obtained average decoding times (expressed in seconds) for the TFAN and MPEG-4/3DMC approaches on both MPEG-4 and Laguna databases.

The obtained results show that TFAN performs about 4–5 times faster than MPEG-4/3DMC, whatever the size of the mesh (in number of vertices), for a superior (up to 30%) compression efficiency.

Unfortunately, since the source code of the TG method is not freely available, we have not been able to integrate the TG approach in the MyMultimediaWorld platform. However, in order to dispose of at least a rough estimation of the relative decoding times, we have run the TG and TFAN decoder executables (on the same MPEG-4 and Laguna test sets) and analyzed the global decoding times (including the reading of the bitstream and the writing of the decoded VRML model). The TG executables are those provided by the authors of the TG method. Under these conditions, the TFAN decoder runs six times faster than the TG decoder. As the input/output management is highly time consuming (more than a half of the total decoding time in the case of TFAN method), we can expect a speed-up factor of at least 10 times with respect to the TG approach.

Finally, let us note that the TFAN average decoding speed is of about 10^6 vertices per second, which confirms the pertinence of the proposed approach for real-time applications requiring low complexity decoding.

Conclusion and Future Work

In this paper, we have proposed a novel low complexity 3D mesh compression scheme, the so-called TFAN. Based on the decomposition of the first order neighborhood of each vertex into a set of TFs, the method offers the advantage of generality, being able to process within a unified manner arbitrary topologies, manifold or not, oriented or not. Evaluation experiments have been performed on both general public (MPEG-4) and CAD (Laguna) databases. In terms of compression efficiency, TFAN offers equivalent or even better performances w.r.t. TG and MPEG-4/3DMC algorithms when dealing with manifold meshes. In the case of non-manifold meshes, TFAN leads to significant (30%) gains in bitrate w.r.t. MPEG-4/3DMC, while performing slightly better (6% gain in bitrate) than the TG compression method. In all cases, the TFAN approach ensures a very low decoding time (one order of magnitude lower than TG and about 4–5 times faster than MPEG-4/3DMC), adapted for real time applications. Due to its performances, the TFAN approach has been recently accepted for standardization by the MPEG-4/3DGC group.

	MPEG-4 database		Laguna database	
	3DMC	TFAN	3DMC	TFAN
0–1000 vertices	0.0019	0.0005	0.0035	0.006
1000–10 000 vertices	0.0169	0.0040	0.0214	0.0038
10 000–100 000 vertices	0.1246	0.0254	0.1484	0.0214

Table 2. Decoding times (in seconds) of the TFAN and MPEG-4/3DMC methods on the MPEG-4 and Laguna databases

DOI: 10.1002/cav

At short term, our future work concerns the complete validation of the TFAN method within the MPEG-4 CE on scalable complexity compression. Specifically, the issue of testing TFAN performances when applied to compression of photometric attributes (i.e., normal vectors, texture and color coordinates) needs here to be completed. Another topic of interest concerns a future implementation of the TFAN method on smart phones. In particular, it would be useful to investigate how the TFAN approach could support the transmission of animated 3D meshes in key frame representations on such low complexity devices.

At a longer term, and from a more methodological point of view, our research will focus on deriving a multi-resolution extension of TFAN, in order to provide additional functionalities such as scalability and progressive transmission.

References

1. Mamou K. Compression de maillages 3D statiques et dynamiques. *PhD. Dissertation*, Université Paris V - René Descartes, France. 2008.
2. International standard ISO/IEC 14772-1: 1997. The Virtual Reality Modeling Language (VRML) 1997.
3. Deering M. Geometry compression. *ACM SIGGRAPH* 1995; 13–20. ISBN:0-89791-701-4
4. Aspert N, Santa-Cruz D, Ebrahimi T. MESH: measuring errors between surfaces using the Hausdorff distance. *Proceedings IEEE ICME 2002*, 2002; **1**: 705–708.
5. Chow M. Optimized geometry, compression for real-time rendering. *Proceedings of the 8th IEEE Conference on Visualization 1997*, 1997; 347–354.
6. Evans F, Skiena S, Varshney A. Completing Sequential Triangulations is Hard. *Technical Report*, Department of Computer Science, University of New York. 1996.
7. Evans F, Skiena SS, Varshney A. Optimizing triangle strips for fast rendering. *Proceedings of the 7th IEEE Conference on Visualization 1996*, 1996; 319–326.
8. Xiang X, Held M, Mitchell J. Fast and efficient stripification of polygonal surface models. *Symposium on Interactive 3D Graphics*, 1999; 71–78.
9. Park DG, Kim YS, Cho HG. Triangle Mesh Compression for Fast Rendering. *International Conference on Information Visualization*, 1999; 280–285.
10. Son G, Min B, Kim D, Kim H, Jang ES. Simple and Fast Compression of 3D Meshes. *International Conference on Convergence Information Technology*, 2007; 2175–2180.
11. Taubin G, Rossignac J. Geometric compression through topological surgery. *Transactions on Graphics* 1998; **17**(2): 84–115.
12. Turan G. On the succinct representations of graphs. *Discrete Applied Mathematics* 1984; **8**: 183–198.
13. MPEG-4Visual. 2001; International standard ISO/IEC/JTC1/SC29/WG11 14496–2:2001. *International Organization for Standardization*, Switzerland.
14. Gumhold S, Straßer W. Real time, compression of triangle mesh connectivity. *ACM SIGGRAPH* 1998; 133–140. ISBN: 0-89791-999-8
15. Gumhold S. C Improved Cut-Border Machine For Triangle Mesh Compression. OT *Erlangen Workshop on Vision, Modeling and Visualization*, 1999; 261–268.
16. Touma C, Gotsman C. Triangle mesh compression. *Proceedings of Graphics Interface* Vancouver, Canada, 1998; 24–34.
17. Alliez P, Desbrun M. Valence-driven Connectivity Encoding of 3D Meshes. *Eurographics Conference*, 2001; 480–489.
18. Lazarus F, Guéziec A, Taubin G, Horn B. Cutting and stitching: converting sets of polygons to manifold surfaces. *IEEE Transmission on Visualization and Computer Graphics* 2001; **7**(2): 136–151.
19. CE Report on the SC3DMC, ISO/IEC JTC1/SC29/ WG11 M15667 2008, Hannover (Germany).
20. Galin E, Akkouche S. C Fast, processing of triangle meshes using triangle fans. *Proceedings of the International Conference on Shape Modeling and Applications*, 2005; 328–333.
21. Akeley K, Haeberli P, Burns D, Tomesh C. C Program on SGI developer's toolbox CD 1990.
22. Moffat A, Neal R, Witten I. Arithmetic coding revisited. *ACM Transactions on Information Systems* 1998; **16**(3): 256–294.
23. Mamou K, Zaharia T, Prêteux F. Evaluation des approches de compression 3D pour les maillages de type CAO. *Actes 4ème Ateliers de Travail sur l'Analyse d'Images, Méthodes et Applications (TAIMA'2005)* Hammamet, Tunisia, September 2005; 381–388.
24. Le Bonhomme B, Preda M, Prêteux F. Mymultimediaworld: a benchmarking platform for 3D compression algorithms. *Proceedings of the 15th IEEE International conference on Image Processing (ICIP 2008)*, San Diego, CA, USA, October 2008; 2700–2703.
25. Lee SW, Kim B, Chen M, Preda M. CE on scalable complexity 3D mesh coding 3DGC. ISO/IEC JTC1/SC29/WG11 N9888 Archamps (France). 2008.

Authors' biographies:

Khaled Mamou received an Engineering Degree in Computer Science from the Tunisia Polytechnic School (Tunis, Tunisia) in 2004, and a PhD. Degree in Applied Mathematics and Computer Science from the University Paris V—René Descartes (Paris, France) in 2008. He has been actively involved in the ISO/IEC Moving Pictures Experts Group (MPEG) since 2005, especially focusing on 3D Graphics Compression. He significantly contributed to the standardization of the SC3DMC (Scalable Compression 3D mesh Compression) and FAMC (Frame-based Animated Mesh Compression) techniques for static and animated 3D mesh compression.

DOI: 10.1002/cav

Titus Zaharia received an engineer degree in Electronics and the Masters degree in Electronics from University POLITEHNICA (Bucharest, Romania) in 1995 and 1996, respectively. In 2001, he received a PhD. in Mathematics and Computer Science from University Paris V—René Descartes (Paris, France). He then joined the ARTEMIS Department at the Institut TELECOM, TELECOM Sud-Paris as a research engineer, and became an associate professor in 2002. His research interest includes static and animated 3D mesh compression, visual content indexing and retrieval and 2D/3D reconstruction methodologies.

Françoise Prêteux is an international expert in the field of digital image sciences and technologies and is the official representative of France with the ISO multimedia standardization committees (SC 29, WG 11 MPEG). Graduate from the Ecole Nationale Supérieure des Mines de Paris as a Civil Engineer, Professor Françoise Preteux holds a Docteur d'Etat ès Sciences Mathématiques from the Université Paris VI. In 2002 she became a doctor Honoris Causa at the Politehnica University of Bucharest. She is a Professor at the Institut TELECOM, TELECOM SudParis and heads the ARTEMIS Department. Her research interests include 3D graphics, digital image coding, indexing techniques, and protection of multimedia content.

DOI: 10.1002/cav