Compression Techniques for Distributed Use of 3D Data: An Emerging Media Type on the Internet

Dinesh Shikhare S. Venkata Babji S. P. Mudur

Graphics & CAD Division,
National Centre for Software Technology,
Juhu, Mumbai 400049, India.
{*dinesh, babji*}*@ncst.ernet.in*

Department of Computer Science,
Concordia University,
Montreal, Quebec, Canada.
mudur@cs.concordia.ca

Abstract

3D data is being processed in a number of application domains such as, engineering design, manufacture, architecture, bio-informatics, medicine, entertainment, commerce, science, defense, heritage, etc. The volume of 3D data that is being circulated on the Internet is increasing very rapidly. Adding to this trend further are new developments like availability of inexpensive 3D scanners and 3D graphics accelerators, sophisticated 3D gaming devices, and increasing use of 3D imaging in distributed collaborative work in virtually all domains like engineering, scientific research, education, entertainment, commerce, etc. Clearly like text, images, audio and video, 3D data is emerging as yet another media type that has to be processed and visualized. Transmission issues concerning 3D data are instrumental for the distributed use of 3D information. In this paper we identify the major requirements for remote interactive access to locationally distributed 3D data, and then briefly describe two new compressed 3D data transmission techniques, developed by the authors. The first technique is for triangulated models representing individual, complex 3D sculptured objects, while the second is for large architectural and engineering class of models, typically large assemblies of a few types of components. Both techniques have the properties of high compaction ratios, fast decompression and low decompression latency, making them very well suited for rapid distribution and retrieval of 3D data on the Internet.

1 3D Data on the Internet

A large number of applications in varied domains have started deploying 3D data as an integral medium of communicating ideas and designs with immense benefits. The domains such as machine design, architecture, bio-informatics, heritage, etc. have information that is inherently defined and modeled in three dimensions. The industrial revolution introduced the use of 2D projections and drawings of these models for communication, often with extensive textual annotations and notes to describe just the visual appearances of the physical 3D models. Such add-on annotations would often be ambiguous and hard to update with changes in the evolving designs. These limitations are now overcome by computer aided engineering design systems that are invariably deployed by virtually all of the engineering and manufacturing industry. With ubiquitous access to the Internet combined with substantial increases in available bandwidth, real-time interactive exchange and remote visualization of 3D models are fast becoming necessities for computer supported cooperative work. These developments are further spurred by the availability of improved design and 3D shape acquisition tools, like 3D scanners, the need for increasing detail in 3D representation, availability of inexpensive display hardware supporting 3D rendering acceleration and the growing acceptance of 3D graphics technology in virtually all sectors, engineering, education, scientific research, governance, medicine, e-commerce etc. Internet-based access to such large models has thus become of utmost importance.

1.1 Typical Usage Scenarios on the Web

Applications such as collaborative 3D CAD (for example, CoCreate [10]), that work over a set of networked workstations, provide a shared 3D environment. These systems enable many remotely located design engineers to interactively view the same 3D model on their personal desktops. They allow editing, annotation and manipulation of the 3D data at interactive speeds. During such collaborative sessions, all the participants get a real-time feedback.

Commonwealth Business Council hosted a Virtual 3D Trade Fair (CVTF) on the Web for several engineering and technology companies (see *www.cvtf.org.in*). In this trade fair, exhibitors could contribute their promotional material as dynamic 3D content consisting of geometric models of their products, videos, sound and other documentation, together forming a virtual stall. The three dimensional content was modeled using Virtual Reality Modeling Language (VRML) [11] and rendered on Internet browsers using VRML plug-ins. The visitors to CVTF could interactively visualize the 3D models of the products, access other information hosted on the site and also use hyperlinks in the virtual stall for initiating e-mail and optionally video conferencing communication with real representatives of the exhibitors.

Virtual museums, digital recreations of heritage sites and historic objects are another area where large 3D models are accessed interactively over the Internet [14, 23]. The virtual 3D walkthroughs of Fatehpur Sikri fort city (currently accessible at *http://rohini.ncst.ernet.in/fatehpur*) and Notre-Dame Cathedral (*http://www.vrndproject.com/*) are two significant examples of such large 3D models and their interactive walkthroughs.

While today, e-commerce is largely restricted to the use of text or 2D images, it is increasingly being recognized that the true potential of e-commerce, particularly in the B2C (business to consumer) mode, can only be realized when the buyer can access the product/service of interest by picking it up, walking around it and generally interacting with it as if in the traditional market place. This again implies use and accessing of 3D models [44, 48].

Medicine, genomics, bio-informatics, scientific R&D collaboration are other domains where 3D data is in extensive use and where 3D data has to be exchanged over the Internet [18].

It is clear that 3D data emerging as a commonly used media-type is imminent in near future. User operations like fast access to remote 3D data for viewing, interactions such as cut-copy-paste and drag-and-drop on 3D models will naturally follow. Integrating 3D data in such a fashion would then extend their use in personal productivity tools by embedding them as objects in composite documents.

1.2 Major Issues in using 3D data on the Web?

The scenarios described above are far from revolutionary and at first glance appear easy to implement using the existing technologies, such as VRML. Furthermore, prototype implementations of some of these concepts have already been demonstrated. Yet, none of these possibilities have been deployed with a widespread impact on productivity, except in some specialized examples mentioned above. There are a number of major issues and problems that need to be tackled before 3D data attains the kind of popularity that is today enjoyed by other established media type such as text, images, sound etc.

Foremost of these issues is the fact that complex 3D data requires a large storage space and long transmission times. Typical large models consist of several hundred thousand triangles, occupy many mega-bytes of storage space and require several minutes of transmission time on popular bandwidth connections.

An effective solution to the problems of large storage requirements and long transmission times is to use 3D compression. Already, several 3D compression tools have begun to be commercially available (for example, the compressed geometry node of Java3d [35] and a plug-in for Web browsers to uncompress and view 3D models – developed by Virtue3D (`www.virtue.co.il`)) and standards for compressed representations are also being proposed [19, 29, 9].

In this paper we primarily address compression and transmission of 3D data. We present two new techniques. The first technique is for individual components that are geometrically large and complex. The second is for large and complex assemblies of components and incorporates the first technique for individual components. While we do not address the other major issues involved in enabling the distributed use of 3D data, for completeness, we list them below.

(a) *Extensive Pre-processing Requirements.* Adapting the acquired digital data of 3D models requires extensive pre-processing for their effective utilization. Manually modeled or semi-automatically acquired 3D model data can rarely be directly utilized in end-applications. Almost always, there is a mismatch between the condition in which a model is produced by a designer using the modeling software and the exact requirements of the application that deploys the model. This mismatch is seen in many different forms: (a) difference in data-formats, (b) lack of desired structure in geometry data, (c) replicated geometry, (d) excess detail, (e) gaps, holes, T-junctions, (f) overlapping faces, edges, etc. Such errors and artifacts can be manually corrected in small models; but automated techniques are essential for detecting and correcting all such problems in large models within reasonable time.

Most software systems dealing with large models carry out a pre-processing phase that "heals" the geometry in the model. As already mentioned, healing very large models manually is extremely laborious and difficult. Hence specialized automatic tools have been developed for this pre-processing task (see [5, 4, 3]). Rossignac's work on *Matchmaker* algorithm [43] highlights some of the artifacts in polygonal models and presents techniques to transform meshes having various non-manifold conditions into a manifold meshes. The previous work in healing is also referred to as CAD-data repair [7, 16, 33]. All these efforts focus on automatic detection of gaps, T-junctions, overlapping geometry, and their semi-automatic correction.

(b) *Incompatibility of Representation Formats.* File formats for the representation of 3D model are not compatible, although they describe similar information. Although the triangle meshes used for graphics applications have a very simple structure, a surprisingly large number of representation schemes have been developed for storing and processing them. Different schemes cater to the requirements of specific applications. For example, 3D data prepared primarily for visual inspection of some engineering model may not adequate for the requirements of programs that carry out engineering analyses. Capturing all the information for about the model in a single comprehensive representation will require the storage of a large amount of data.

The solution to this problem calls for a clever encoding scheme that stores different specific information distributed across multiple units of storage with a suitable relational inter-linking mechanism. For example, one storage unit may represent only the geometric data, while other related unit may capture visual attributes, and so on.

(c) *Problems in 3D Navigation.* Casual users cannot easily navigate in 3D to interactively inspect the data. This is purely a user interface issue, but a very significant one. While there are a number of applications that allow 3D interactions for real-time visualization of 3D models, casual users are still put off by most of the applications by the use of terms such as zoom, pitch, roll, pan, etc.

The solution should provide a view control mechanism so trivial to understand that a first time user should be comfortable with it in a few seconds, thereby immediately making the user more productive with the availability 3D information.

(d) *Slow Rendering Speeds.* A complex model takes too long to display on the screen. As of today, a typical polygon rendering hardware can render, on an average, 200K triangles per second. This speed is dependent on the bandwidth of system bus carrying the data transmitted from main memory to the graphics processor. Ideally, the rendering speeds of at least 10 to 12 frames per second are required to avoid confusing response from interactive visualization tools. A model of a modest size consisting of 100K triangles can be rendered by typical 3D accelerated hardware at only 2 frames per second.

Various software-based solutions utilize the basic strategy of avoiding unnecessary work. These solutions avoid rendering of hidden geometry, trade visual precision for interactive performance and also pre-compute information for run-time acceleration of visualization.

The rest of the paper is organized as follows. In the next section, we describe the nature of large 3D models and the specific requirements of networked 3D applications. In Section 3, we describe the state of the art in 3D compression and also briefly describe the two new compression algorithms mentioned above along with some implementation results. We conclude the paper in Section 4.

2 Large 3D Models – Representation and Requirements for Distributed Use

Polygon mesh is the most popular representation for 3D models. In this, the surface of any 3D object is represented by mosaicing together polygonal faces best approximating the surface. More curved the surface, larger is the number of polygons needed for an accurate approximation. Most engineering products and architectural designs are now modeled using sophisticated computer-aided design (CAD) systems. Literature abounds with a number of more sophisticated and less verbose representation schemes for 3D modeling such as quadrics, bi-polynomials, NURBS [17, 39], sweeps, Boolean operations [36] etc. These are supported by most of the commercially available CAD/CAM/CAE systems. However, even if these relatively more compact representations are used during the initial creation of the 3D models, the final evaluated and stored representation is usually in the form of polygon meshes, with triangle being the most common form of the polygon. Polygon mesh models are also acquired by semi-automatic and manual (computer-assisted) processes using devices such as laser range scanners and medical imaging systems. These can capture very fine details in curvature and thus produce extremely detailed polygonal models.

SOME DEFINITIONS: A polygon mesh model is typically defined in terms of (i) *geometry* – coordinate values of vertices making up the model, (ii) *connectivity* – adjacency relationship among the vertices which defines the polygonal faces of the model (also called as *topology*), and (iii) *non-geometric attributes* – vertex/polygon colour, vertex normals, texture, material properties, etc.

A *polygon mesh model* O consists of a set S of polygon meshes and associated non-geometric attributes. A *polygon mesh* consists of a set V of vertices, a set E of edges and a set P of polygons. Each vertex is associated with a geometric point position in the 3D Euclidean space, say, x_i, $\{x_i \in \mathbf{R}^3\}$. An edge is represented as a pair (v_1, v_2) and a polygon as a sequence $(v_1, v_2, ..., v_k)$ of vertices. A *triangle mesh* is a special case with all polygons being triangles.

A mesh having f faces, e edges and v vertices satisfies the equation, $f - e + v = 2$ (Euler's relation[36]). When all the faces have at least 3 sides, we can show that $f \leq 2v - 4$ and $e \leq 3v - 6$, with equality if and only if all faces are triangles. For meshes with g handles (genus g) the relation becomes $f - e + v = 2 - 2g$ and the bounds on the number of faces and edges increase correspondingly.

In a mesh model O, we call two polygons as *adjacent polygons* if they share an edge. There exists a *path* between polygons p_i and p_j if there is a sequence of adjacent polygons $p_i, p_1, p_2, ..., p_j$. A maximal subset O_c of the mesh model O is called a *connected component* if there exists a path between any two polygons in O_c. Note that a given mesh model may have multiple connected components. A mesh can be trivially decomposed into its connected components using a simple labeling algorithm based on breadth-first or depth-first traversal of complexity $O(n)$.

In a mesh representing a manifold, each edge is either shared by two polygons, called as an *interior edge*, or belongs to a single polygon, called a *boundary edge*. A closed loop formed by linking up such boundary edges forms a *boundary* of the mesh. Note that a mesh may have multiple boundaries.

We call two polygons as *adjacent polygons* if they share an edge. There exists a *path* between polygons p_i and p_j if there is a sequence of adjacent polygons $p_i, p_1, p_2, ..., p_j$. A maximal subset O_c of the mesh model O is called a *connected component* if there exists a path between any two polygons in O_c. Note that a given mesh model may have multiple connected components.

2.1 Storage and Transmission Requirements of 3D Models

A common scheme for representing and storing polygon meshes is to use a list of vertex geometry coordinates to store the geometry and a list of vertex indices for each face to store mesh connectivity. Edges are implied and not explicitly stored. For a triangle mesh having v vertices and t triangles, this requires approximately $3v$ space to store the vertex coordinates and $3t \log_2 v$ space to store connectivity among the vertices, where $t \approx 2v$ for most triangle meshes. The storage and transmission costs increase non-linearly as the number of vertices increase.

Representation of each of the n 3D vertices requires 3 floating point numbers (adding up to $n \times 3 \times 4$ bytes) and each of the m triangles requires 3 integer references for pointing into the shared list of vertices (adding up to $m \times 3 \times 4$ bytes). Given the common download speeds of 64kbps, even the smallest model in the Table 1 will need at least 154 seconds for transmission. These models are illustrated in Figures 1 and 2.

Large data sets are encountered in other forms as well – text, images, speech, music, video, etc. Data compression / decompression is the technique used to alleviate problems of storage, transmission and interactive rendering [37, 54]. While data compression has been and continues to be a topic of wide interest, the general purpose algorithms do not provide good enough results for all forms of data. Hence special compression techniques are being explored for different media types like images [53, 40], video (e.g., MPEG-1 and MPEG-2[1]), audio (e.g., RealAudio, MP3). These techniques achieve orders of magnitude more compression by use of the knowledge of specific structure inherent to the data, the mode of use of the data and fidelity requirements of an application using the data.

The data representing 3D polygonal models also do not compress well enough if general purpose data compression techniques are used. They too have specific structure, which is very different from that in other forms of data. Their mode of use is very different – for example, unlike audio or video, at least some approximation representing the entire 3D model is usually required by the applications. There is no sequential or temporal order that can be imposed on 3D polygonal models in applications. Fidelity requirements are largely application specific, but again, they need to be addressed differently from

[1]see *http://mpeg.telecomitalialab.com*

(a) Bunny

(b) Dragon

(c) Dinosaur

(d) Igea

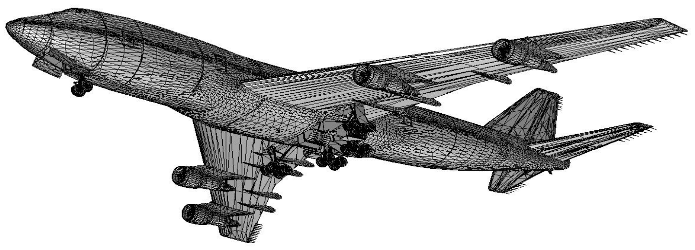

(e) Boeing 747

Figure 1. Sample 3D mesh models

image or video data. As a result in the recent years, 3D compression has emerged as a new branch of the general field of data compression aimed at the use of large 3D models in computer graphics, CAD/CAM, virtual reality, games, education, and many other scientific applications. A large number of researchers around the world are addressing this problem [21, 24, 42, 49, 50, 28, 20]. Techniques being evolved make use of concepts developed earlier for compression of other forms of data and also use completely different approaches based on very intimate knowledge of 3D data, structure of polygonal models, and so on.

Model	# **Vertices**	# **Triangles**	**File Size** (bytes)
Stanford Bunny	35,947	69,451	1,264,776
Dragon	437,645	871,414	15,708,708
Dinosaur	56,194	112,384	2,022,936
Igea Artifact	134,345	268,686	4,836,372
Capitol Building	52,606	87,258	1,678,368
Colosseum	69,828	135,159	2,459,844
Helicopter	105,079	187,929	3,516,096
Boeing 747	56,364	88,737	1,741,212
Taj Mahal	65,323	126,453	2,301,312

Table 1. Sizes of some large 3D polygon mesh models. The first four models are obtained using laser scanner, and the remaining models are constructed using CAD tools.

In fact, different classes of large 3D models are best compressed using specialized algorithms that exploit the characteristics models. In the following discussion, we describe three major classes of large models that we have experimented with.

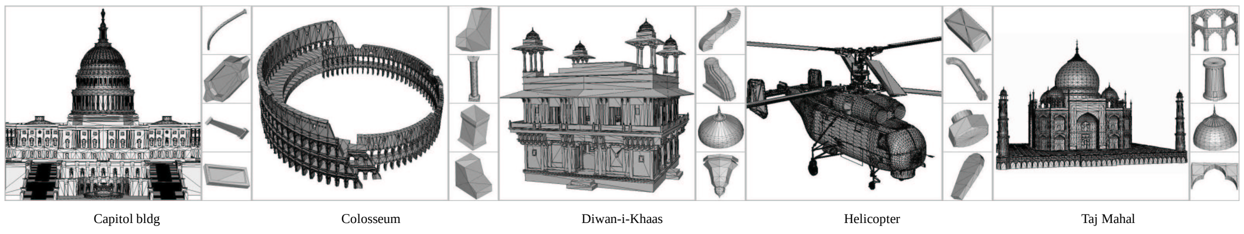


Figure 2. Architectural and engineering models with some examples of most frequently repeating features.

2.2 Classes of 3D data

2.2.1 Natural and Sculptured Objects

The digital models of natural shapes such as terrains, anatomies, sculptures, etc. are typically acquired using semi-automatic techniques like 3D scanning [51, 12, 30], satellite imagery, stereogrammetry or by construction of iso-surface boundary in a volumetric image [32]. Such techniques enable us to accurately capture the complex curvatures in such natural shapes. Many of these techniques carry out the construction of the surface from uniformly sampled points on the surface of the actual objects. It is common to see that such models have a small number of connected components (often a single component) having a large number of densely distributed polygons. Figure 1(a–d) shows examples of dense triangle meshed forming single component models of sculpture and clay models. A common feature of such models is that they have large patches of smooth regions, with few discontinuities.

Terrain models represent surfaces of the form $z = f(x, y)$. Usually this data is obtained in the form of a matrix of elevation values for a regularly sampled spatial region. This simple graph surface structure has been exploited by many special techniques developed for their simplification and rendering [6, 31].

2.2.2 Architectural/Heritage Designs

Three-dimensional documentation of architectural designs and heritage models are also extremely detailed and rich in content. These models are characterized by their much higher combinatorial complexity in terms of number of connected components. Often, most of the connected components are simple geometric shapes and have a sparse distribution of points and polygons.

In addition to the geometric description, these models always have texture maps and material properties associated with the meshes. To capture the texture maps, each vertex in the mesh has an associated pair of texture coordinates. Very often many vertices are repeated in their geometric positions to capture different texture coordinates with respect to the faces meeting at that position. Due to these repeating vertices, many polygons are adjacent only geometrically but not topologically. The vertices also have associated normals used while computing local and global illumination [18, 55]. Figure 2 illustrates examples of architectural and heritage models (Capitol building, Colosseum, Diwan-i-Khaas and Taj Mahal) along with a few of their component shapes.

2.2.3 Engineering Designs

Detailed engineering models such as power plants, aircraft designs, automobile assemblies, etc. are also examples of large models, most significant for industrial applications such as virtual mock-ups, collaborative CAD software [41, 10], interactive inspection of 3D models, etc. While we seldom see engineering models with texture maps, they have attributes like material properties associated with meshes. Sometimes, these meshes also have colour values associated with the vertices of the model.

Figure 2 shows an example of engineering model (Helicopter) and some of its component shapes. In addition to geometric and other visual content, these models often have non-visual attributes which are collectively called the product data. Data representing these models is often organized to group the components of the design based on their function or material. Such groups usually form meshes that are assigned a common material or other such non-visual attributes.

2.2.4 Distinctive Characteristics of Architectural and Engineering Models

Large 3D models of architectural/heritage and engineering designs have some distinctive properties:

- *High combinatorial complexity*: They have a large number of small to medium sized connected components – each having up to a few hundred vertices – forming combinatorially complex structures. The organization of the geometry is in terms of meshes, where each mesh is often a collection of all polygons that share the same material property or texture map. Consequently, many meshes in such models consist of multiple connected components.
- *Repeating features at different granularity levels and multiplicity of descriptions*: Shape features occurring at the granularity of component- and sub-component levels repeat many times across the model. The repeating geometric shapes are repeatedly described in the representation of the model. This is largely due to the difficulty in forcing the discipline of manually identifying repeating shapes. Moreover, no representation scheme provides a facility to express sub-component level repetition of shape features (for example, all teeth of a gear have the same geometric shape in a single mesh component of a mechanical assembly model). Figure 2 shows some models of this class along with some of the most commonly occurring component shapes in those models.
- *Sharp discontinuities in geometric features*: Most of the geometric features have sharp edges, corners and other discontinuities.
- *Modeling errors*: Almost always, these models are manually created using some 3D modeling tools, say, 3D Studio MAX [15]. Many of these large data sets often have errors like erroneous replication of geometry, invisible geometric shapes, etc. due to the manual process used for their creation.

2.3 Desirable Qualities of Compression Techniques

Any 3D compression technique intended for use in networked applications must include the following features:

1. *Wide applicability*: Must compress 3D models for the most popular representation schemes.
2. *High compression ratio*: This is possible by effective exploitation of data redundancy exhibited by large 3D models.
3. *Control over lossiness*: Compression may be lossy or lossless. In case of lossy compression, the amount of loss that is acceptable will depend very much on the application on hand. Hence user control over this is very important.
4. *Ubiquitous access*: simple and fast decompression, causing minimum overhead during runtime reconstruction so that 3D models can be accessed from the desktops and in future even from mobile computing devices.
5. *Minimum latency in reconstruction*: Since users are geographically distributed, 3D models must be visualisable with minimum latency. It should not be required that a large part of the compressed model be available before the decompression process can commence.
6. *Multi-resolution representation*: is highly desirable. This offers the basis for LoD (Level of Detail) processing of compressed data.
7. *Selective component-wise compression*: In many such applications, it is more effective to selectively compress a dataset component-wise rather than the entire dataset in totality. It is very desirable that a compression scheme includes this selective compression capability in its encoding algorithm for better compression.

3 Compression of 3D Models

3.1 Previous Work

Due to the growing significance of 3D polygon mesh models, many researchers have addressed the problem of geometry compression of polygon mesh models. Compression strategies for 3D polygon mesh models have three components: (a) compression of connectivity information, (b) compression of geometric data and (c) compression of attributes.

CONNECTIVITY COMPRESSION: In a triangle mesh, the number of triangles in a mesh is roughly twice the number of vertices and each vertex is referenced in 5 to 7 triangles, which indicates the scope for reducing redundancies in the description of connectivity among the vertices. Representation schemes that minimize repeated references to vertices result in compression. Connectivity compression techniques [21, 50, 49, 42, 24, 1] encode triangle/polygon connectivity in a lossless manner. Recent techniques [50, 21] construct spiraling triangle spanning trees in which vertices are labeled in the order in which they are first encountered in the triangle tree. A small set of operators is used to encode the traversal over the triangles. High compression ratios have been achieved for connectivity encoding, typically a few bits per vertex [42].
GEOMETRIC DATA COMPRESSION: Deering [13] used quantization of coordinate values from 32 bits to 16 bits, a 18-bit index into a global list of values for vertex normals and a 15-bit representation of color values.

Quantization of coordinates and color values leads to an indiscriminate truncation of the accuracy regardless of the features in the geometric models. Predictive encoders perform much better by taking advantage of the coherence in the data and known connectivity structure to predict the value of the next element in the sequence [49, 50, 38].

Signal processing based techniques which have been in use for surface fairing [47] and multi-resolution analysis [22] are based on the construction of a set of basis functions for decomposition of a triangle mesh geometry into signals. The low frequency components in the signals correspond to smooth features and high frequency components correspond to discontinuities such as creases, folds and corners. Retaining just a few frequency components suffices to capture the overall perceptual shape of the model. The *spectral compression* effort of Karni and Gotsman [26] and wavelet based technique of Khodakovsky et al [27] are examples of this approach.
COMPRESSION OF ATTRIBUTES: Attributes such as colour, normals, texture mapping coordinates and material properties defined at vertices, faces, or corners of faces are compressed using specialized encoding schemes [21, 48, 25]. The compression of this information involves compact encoding of two kinds of information: the *property values* and *property mapping*. Such separation enables the introduction of attribute compression techniques into almost any given connectivity compression technique.

3.1.1 Observations

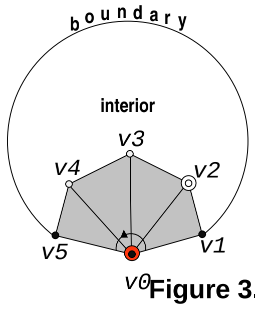
Earlier connectivity compression schemes can be broadly categorized into three classes[8]: face-based, edge-based and vertex-based traversals. Rossignac's Edgebreaker[42] and the Cut-border machine of Gumhold and Strasser[21] are face-based, the FaceFixer algorithm of Isenburg and Snoeyink[24] is edge-based and mesh compression technique of Touma and Gotsman[50] and valence-based technique proposed by Alliez and Desbrun[1] are vertex-based. Face-based traversals are encoded using one operator per face of the mesh. Similarly, edge-based traversals are encoded using e operators and vertex-based traversals are encoded using v operators. There could be some additional operators to take care of exceptional situations during the course of the traversal, which may require explicit vertex references. Alliez and Desbrun[1] have argued that such exceptions are sub-linear in order. They also show that a valence-based encoding scheme gets closest to the theoretical limit of 3.245 bits per vertex proved by Tutte[52] and outperforms face-based and edge-based techniques. This is explained by noting that the size of the compressed representation is proportional to the number of operators encoded and the fact that for polygon meshes representing closed manifold surfaces the relation $v < f < e$ is true.

We must also note here, that both predictive encoding and the signal processing approach work best for smooth surfaces with dense meshes of small triangles. Predictive encoders do not take into account that large triangles can be adjacent to tiny ones. Signal processing techniques rely heavily on the specific connectivity of the mesh for deriving the basis functions.

This leads us to believe that if we could indeed traverse the connectivity mesh in units which are larger than the above, say groups of vertices or faces or edges, then it may be possible to get better compression. The advancing fan front algorithm developed by the authors is built on this premise. It uses a group of triangles, a fan, as the unit of encoding. And the repeating feature detection based compression technique takes this even further, by using components and even groups of components as the units of encoding.

3.2 Advancing Fan Front: Connectivity Compression

The Advancing Fan-front (AFF) algorithm [2] described here proposes and uses a *fan-based* traversal of triangle meshes. In almost all the triangle meshes, the number of fans is approximately half the number of vertices and one-fourth the number of triangles. The traversal of the mesh is then represented as a sequence of operators describing, on the average, $v/2$ or $t/4$ fans. While the number of distinct fan configuration codes depends on the variation in the degree of fans, only a few of these fan configurations occur with high frequency, enabling a very compact representation using range encoding.



Seed: `v0`
Fan-front: `(v1,v2,v3,v4,v5)`
Seed bit pattern: <`01000`>
Visited bit pattern: <`10001`>

Figure 3. Fan description.

SOME DEFINITIONS: A *fan* is a polygon that admits a triangulation in which all triangles have a common vertex, called as *fan center*. The other vertices of the polygon are called as *front-vertices*. The front-vertices and the edges between them form a *fan-front*. The number of front-vertices is the *degree* of the fan. Order of the front vertices denotes the orientation of the fan. A *seed* is a vertex on fan-front which becomes a fan center.

The following notation is used in the illustrations while describing the AFF algorithm: We denote an unvisited vertex by a small hollow circle, a visited vertex by a small filled circle. Each identified seed on fan front is denoted by a concentric circle and the current seed by shaded concentric circles. The orientation of the fan is shown with a curved arrow around the seed.

OVERVIEW OF THE ALGORITHM: Consider the simple input of a consistently oriented triangle mesh forming a single connected component that represents a manifold with at most one boundary. Multiple boundaries can be handled with a little additional effort, as described later, and closed meshes may be trivially handled by removing one initial triangle to create a boundary. Although the AFF approach can be extended to meshes consisting of arbitrary polygons, for ease of explanation and understanding, we describe the algorithm for triangle meshes.

The algorithm starts the conquest of the mesh by arbitrarily choosing a vertex on the mesh boundary as the initial seed. A fan is formed with the selected seed. Each newly constructed fan has its two extreme fan-front vertices on the boundary and includes all the triangles incident on the seed vertex. The conquest of the fan removes the fan triangles from the mesh, identifies the seed on this fan-front and modifies the mesh boundary by inserting the non-boundary edges on the fan-front. Thus the fan-front is *advanced* into the mesh. This process is recursively continued until the entire mesh is conquered.

During this conquest, the following information is recorded for subsequent reconstruction of the mesh connectivity: (a) vertex coordinate list re-ordered to implicitly reflect the sequence of fan-based conquest, (b) number of vertices on the mesh boundary at start, (c) the start seed vertex on the boundary, and (d) a sequence of fan descriptions, each including the fan degree, a bit pattern indicating the next seed and a bit pattern indicating which of the fan-front vertices have already been visited.

A seed is identified on the fan-front using the following condition: *a vertex s on fan-front is a "seed" if there exists a mesh-boundary vertex v on fan-front such that (v, s) is an interior edge with the same orientation as of the fan.*

A fan is described in following form:

<degree><VisitBitPattern><SeedBitPattern>

where, *SeedBitPattern* is the sequence of binary flags, one for each front-vertex – the flag is 1 if the vertex is identified as the seed on the fan front for advancing the conquest, 0 otherwise. *VisitBitPattern* is the sequence of binary flags, one for each of the fronts – the flag is 1 if the vertex is an already visited, 0 otherwise (see Figure 3).

Reconstruction of the mesh connectivity proceeds in the same sequence as the conquest of fans in the mesh connectivity.

A SIMPLE EXAMPLE: We have chosen the simple triangle mesh of Figure 4(a)(i) to illustrate how AFF spans the mesh with successive conquest of fans. Throughout the illustration (Figure 4) we assume that the mesh boundary orientation is counter clockwise and the same is the orientation of the fans.

Encoding: Initially, AFF declares all the vertices on mesh boundary (*a, b, e, i, j, k, f, c*) as visited (see Figure 4(a)(i)) and as the first seed it chooses a vertex *a* on the mesh boundary (see Figure 4(a)(ii)). Starting with this boundary and the seed, AFF then recursively decomposes the mesh into fans. The fan with the seed *a* and fan-front *b, d, c* is spanned in a counter-clockwise direction (Figure 4(a)(iii)). With the earlier mentioned condition for a seed, vertex *d* is identified as the next seed. After this step, the output consists of fan description (<3><101> <010>) and vertex *d* is appended to the vertex stream. Next we advance the fan-front (*b, d, c*) and the new mesh-boundary now becomes (*b, e, i, j, k, f, c, d*). The next fan is conquered using this seed vertex (*d*). Further steps in the conquest are illustrated in Figure 4(a).

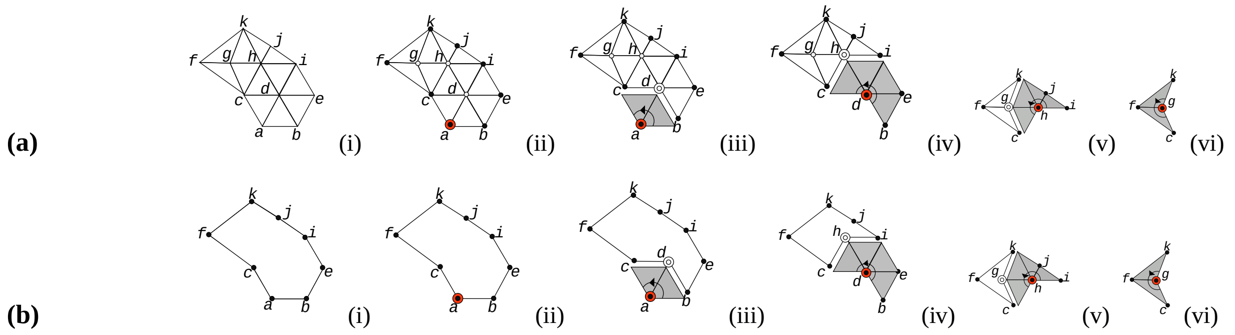


Figure 4. Advancing Fan-front algorithm – a simple example : (a) encoding and (b) reconstruction.

During this process AFF outputs the reordered vertex stream as (*a, b, e, i, j, k, f, c, d, h, g*), the number of mesh boundary vertices *8*, the initial seed *a* and the following sequence of fan descriptions:
<3><101> <010>,
<5><11101> <00010>,
<5><11101> <00010>,
<3><111> <000>.

Reconstruction: The reconstruction algorithm restores the initial boundary (*a, b, e, i, j, k, f, c*) of the mesh (see Figure 4(b)(i)), and initial seed *a* (see Figure 4(b)(ii)). Now from the first fan description <3><101> <010>, the degree of the fan is 3. So the first $3 - 2 = 1$ vertices are to be restored and assigned to the fan-front, which results in fetching of vertex *d* from the vertex stream. First and last of the fan-front vertices are the next and the previous to the seed on the mesh-boundary – so, the fan front is (*b,d,c*). Thus the first fan has been reconstructed. Construction of the fan results in a modified new boundary (*d, b, e, i, k, j, f, c*) (see Figure 4(b)(iii)). The *SeedBitPattern* in fan description indicates that the next seed is *d*. This successive reconstruction of fans continues till the entire mesh is reconstructed as shown in Figure 4(b).

In the simple example above, advancing the fan-front always resulted in the unconquered part of the mesh remaining as a single connected component. However in more complex situations, advancing of the fan-front into the mesh may result in the fan-front intersecting with the original mesh boundary in such a way as to split the mesh boundary into multiple loops, and as a consequence splitting the unconquered parts of the mesh into multiple connected components. Clearly we need one seed per connected component for further conquest of the mesh. If we observe carefully, we can see that the condition mentioned above for identifying a seed is guaranteed to give one seed per connected component for continuing the traversal. When more than one seed is identified on the fan-front, the seeds are pushed on a stack and processed recursively. The algorithm carries on its conquest from each of the seeds until all the connected components are conquered.

COMPLEXITY: Each triangle is visited exactly once during the conquest of fans in AFF's encoding as well as reconstruction algorithms. The queries to determine adjacency information for vertices, edges and faces can be satisfied in constant time by using a suitable data structure such as the half-edge data-structure[34]. Thus the time complexity of both the algorithms is linear in the number of triangles in the input mesh.

IMPLEMENTATION RESULTS OF AFF: The conquest of fans in the mesh is encoded using a vertex stream, the number of vertices in the boundary, the starting seed, a sequence of fan descriptions and the list of vertex re-references. The structure of a fan description seems to suggest that even for a fixed degree of fan, a large number of fan configurations are possible. However, our experiments show that only a very few of the fan description patterns dominate the distribution, thus enabling very high compression using techniques such as range encoding[45] of these patterns.

In order to get a practical feel as to why this compression scheme works so well, consider the performance of AFF for model called Dinosaur in Table 2, which has the largest number of distinct fan types in the table, i.e., 133. At most 8 bits would be required to represent each fan type without any kind of special variable length encoding. The total number of fans is half the number of vertices, hence the cost per vertex would be 4 bits and hence 2 bits per triangle. Clearly, for all the other models listed in Table 2, the cost would be less than this even without any special variable length encoding scheme.

The AFF algorithm has a number of desirable properties that make it highly suitable for distributed use of 3D models. Firstly average compression ratios are superior to earlier algorithms. Secondly, decompression is fast, as it is linear order complexity in terms of the triangles. Thirdly, unlike some of the other techniques, which decompress in reverse order of compression, decompression in AFF is in the same order as traversal. Thus decompression latency is minimal, just about the first fan description. Hence, 3D data can be streamed using this technique.

Model	#vertices	#fans	#fan types	#rerefs	bpv
Bunny	34839	17884	83	472	1.621
Dinosaur	56194	29680	133	1488	2.205
Horse	48485	25080	109	813	1.9
Igea	134245	68905	90	1691	1.629
Isis	187644	95562	105	1688	1.496
Knee	37890	18945	7	0	0.047
Sphere	1026	515	9	2	1.144
Vase	68098	33795	7	0	0.026

Table 2. Connectivity compression results.

Finally it is worth noting here that decomposing of triangle meshes in terms of fans has the added advantage that it is also well suited for improved performance using graphics accelerator hardware and standard graphics software APIs, which usually have special primitives to handle fans. Use of fans as the unit of rendering reduces the number of vertices transmitted, transformed and tested for culling in the graphics pipeline.

3.3 Compression using Automatic Discovery of Repeating Features

BACKGROUND AND MOTIVATION: The AFF algorithm described above and almost all of the previous research has concentrated on the compression of large models having a few connected components – often a single mesh of a complex surface with continuously varying curvature – formed by a dense collection of a large number of small polygons (mostly triangles). As we have observed earlier, engineering models are often constructed as a complex assembly of many small connected components, many of which repeatedly describe a few shape features. Application of earlier connectivity compression techniques to the components in such models may not yield high compression if the repetitive description of features is not discovered, recognized and suitably encoded.

Very large 3D geometric models of the engineering class usually have a large number of meshes, with small numbers of large triangles, often with arbitrary connectivity (see examples in Figure 5). The architectural and mechanical CAD models typically have many non-smooth meshes. For reasons cited earlier, the previously reported compression methods relying on smoothness are not the best suited.

The previously published algorithms do not exploit the fundamental property of repeating shape features that is almost always present in polygon meshes of large architectural and engineering models. These repeating features are repeatedly encoded even when using the earlier connectivity compression algorithms. Such redundancy must be automatically discovered in the given model and specifically attacked. We present our technique [46] for automatic discovery of repeating geometric features and a compression scheme that uses this very effectively for such polygon mesh models.

DISCOVERY OF REPEATING GEOMETRIC FEATURES: Based on our observation of characteristics of models of engineering class (see Subsection 2.2.4), we attempt to discover repeating features at the following levels of granularity: (a) component level, (b) sub-component level and (c) their aggregates that repeat with a rigid body transformations within the model. We also derive the transformation for each repeating instance of a feature. The repeating features at the three different granularities are discovered in different phases using different algorithms.

In the first phase, component-level repetition of shapes are discovered. This is done by by aligning components and matching their vertices for geometric congruence. The result of a successful match between two components also gives us a transformation that aligns the two models. These components are then organized in a “master geometry – instance transform” hierarchy for processing in a later phase. The hierarchy of components classifies them into first instances and repeating instance of shapes. We call this procedure a *top-down* approach since we match features at a level of maximally connected sub-graphs of the polygon mesh model. In our implementation we have used a simple technique based on principal component analysis (PCA) with suitable extensions suggested in the literature to overcome its limitations. Test results from the implementation have shown that this simple technique is quite efficient for the class of models under consideration.

Many engineering models consist of single connected components constructed using boolean operation of multiple components. Such components have many features repeating at a sub-component level. A top-down discovery procedure cannot

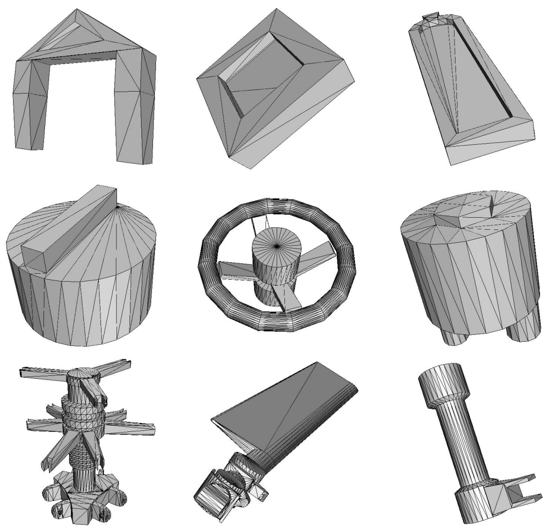


Figure 5. Mesh components that are parts of polygon mesh models of engineering designs. Note the sharp edges, corners, discontinuities, large triangles. These components have been extracted from models illustrated in Figure 2.

discover repeating features of such type. The aim of the second phase is to discover sub-component level features within a component and then amongst all connected components. For the discovery of sub-component level features, we take a *bottom-up* approach. Starting from an initial subset of vertices with matching characteristics, potentially matching features are grown using simultaneous, identical, breadth-first traversals in the model. This procedure results in a set of repeating features, which are again organized in the “master geometry – instance transform” hierarchy.

Finally, the third phase of the repeating feature discovery process builds aggregates of component and sub-component level features that repeat in the model. This phase uses the hierarchy of features constructed in the first two phases to build sets of higher level repeating features.

OVERVIEW OF THE COMPRESSION ALGORITHM: The geometry compression technique proceeds in the following steps:

1. *Pre-processing*: In this step, we eliminate replicated vertices that have identical triples formed by xyz-coordinates, vertex normals and texture coordinates. We then decompose the input model into its connected components using the simple recursive labeling algorithm.

2. *Discovery of Repeating Features*: To reduce the redundant description of repeating geometric features, we discover repeating features at three levels of granularity.

   (a) Discovery of repeating features at the level of connected components.

   (b) Discovery of sub-component level features within and across connected components.

   (c) Identification of aggregate level features.

   While identifying aggregate level features, multiple USE instance features referring to the same DEF instance with the same transformation or an identity transformation are marked as erroneous replications of the feature in the model. Such USE instances are not included in the compressed representation.

3. *Compact encoding*: After discovering the repeating features the hierarchy of DEF and USE instances and the aggregate features in the model are encoded. The USE instances and aggregates are compactly encoded using references to the DEF instances and transformations required to reconstruct the original features. The DEF instances that represent the first instances of the geometric features may be compressed using geometry and connectivity compression algorithms.

For our experiments, we have created a simple binary file format for the representation of the models. The *nodes* corresponding to DEF and USE features, material properties, etc. are encoded using simple name-value pairs. The file sizes of the original and compressed representations reported later in the section are measured for data in this format.

RECONSTRUCTION PROCEDURE: The reconstruction procedure to obtain the original model from the compressed representation is simple and efficient. The DEF instance features are obtained from the file by straight forward parsing of the name-value pairs. The vertex normals of these features are obtained from the global list of vertex normals.

While decoding the aggregate features, for the reconstruction of each USE-instance, the following steps are taken: (i) obtain the centroid m_1 of the DEF instance, (ii) obtain the rotation matrix R and the position vector m_2, denoting the mean position of the USE instance, (iii) transform the vertex positions and normals of the DEF instance by the composite transformation $\mathbb{T} = T_{-m_1} \circ R \circ T_{m_2}$.

Each vertex in the model is being reconstructed in a single pass and the complexity of reconstruction procedure is a linear function of the number of vertices – much faster than the compression algorithm. This algorithm too has low decompression latency. Compression ratios are much higher than for any of the techniques presented earlier. For individual components, any of the techniques described earlier may be used. If the model is a triangle mesh, then AFF is ideal.

RESULTS OF COMPRESSION TECHNIQUE BASED ON FEATURE DISCOVERY: We carried out our tests on a large number of 3D models of architectural and engineering class, some of which are available for download on the Internet.

Table 3 shows the results of discovery of repeating component shapes in seven large models. It is evident that a small number of component shapes are repeated many times in different positions and orientations. Note that in this experiment we have used only rigid body transformations between DEF component features and their corresponding USE components. This transformation is recorded in the instance transform hierarchy computed during the process of discovery.

Model	# comp.s in model	# DEF comp.s	# USE comp.s	Time (sec.)
Capitol bldg.	2662	288	2374	6.13
Colosseum	1129	301	828	48.24
Diwan-i-Khaas	3726	833	2893	243.95
Helicopter	976	455	521	12.62
Heritage	1124	72	1052	242.26
Piping	1051	118	933	0.60
Taj Mahal	375	70	305	42.91

Table 3. Results of discovery of repeating component features.

Our heuristic of looking for repeating features at the level of connected components results in discovery of a lot of redundancy and yields the largest savings in storage. Table 4 shows the results of component level discovery on some representative models that we experimented with. The table is partitioned into three sections – (a) name of the model, (b) the count of components, vertices and triangles of the original model, and (c) the count of USE and DEF instance feature components discovered by the top-down procedure. In the storage scheme, only the vertices belonging to the DEF instance features are stored in full details with their Cartesian coordinates, indices to vertex normals and texture coordinates. Note that in almost all the models we experimented with, the number of vertices and triangles belonging to DEF instance features is much smaller than that in the original model.

Model	# comp. (orig)	# verts (orig)	# tris (orig)	# comp. (USE)	# comp. (DEF)	# verts (DEF)	# tris (DEF)
Capitol building	2662	52606	87258	2374	288	12261	22437
Colosseum	1129	69868	135159	828	301	24624	48403
Diwan-i-Khaas	3726	295695	162590	2893	833	104914	49224
Helicopter	976	105079	187929	521	455	73969	132406
Heritage	1124	124740	244557	1052	72	30452	60602
Piping	1051	13103	20794	933	118	3721	6318
Taj Mahal	375	65323	126453	305	70	33929	66118

Table 4. Large number of repeating components are detected thereby yielding savings in the representation of the model.

We now estimate how much savings we actually achieve in terms of storage space in comparison with the state of the art in 3D compression. A 3D compression scheme that requires B bits/vertex for the storage of vertex coordinates and

C bits/vertex for representing the connectivity information would require at least $(B + C)v$ bits of storage space for a 3D model. The compression scheme based our discovery technique sharply reduces the number of vertex coordinates to be stored and the explicit encoding of the connectivity. We can see from Table 4 that this reduction factor ranges between 2 and 4.

While the discovery of component level features that repeat leads to large reduction in the storage space, the discovery of aggregate features yields only a small incremental improvement. This is expected since the savings achieved in encoding of aggregate features is obtained by only avoiding the repeated representation of a common rigid-body transformation.
NOTE: A compression algorithm that effectively uses knowledge of specific characteristics of a given model compresses it better than a general purpose algorithm. We have also discussed why existing 3D data compression techniques are not well suited to models of the engineering class. This compression technique not only uses special *a priori* knowledge of properties of the models of engineering class, but also "learns" more through the discovery of repeating features at different levels of granularity. This special capability enables the method to achieve high compression and also makes it more widely applicable across many 3D models where features are likely to repeat.

4 Conclusion

3D data representing geometric shapes of complex real world entities is emerging as a new media type. Computer supported collaborative work environments and remote access requirements, make high speed exchange of 3D data over the Internet essential in a large number of application domains. However, in most realistic applications, the volume of 3D data is very large and compression of this data will go a long way in ensuring that 3D applications proliferate on the Internet. However, 3D data has totally different structure in comparison with other media types such as text, images, sound or video. Hence special compression techniques have been devised. In this paper, we have described two new techniques for compression. The first technique, named as the Advancing Fan-front technique works by spanning a 3D mesh representing the shape into fans. The number of fans in a mesh is about half the number of vertices. The algorithm is efficient and fast. It is applicable to large and complex individual components. The second technique is applicable to large assemblies of components and is ideally suited for compressing of engineering type of models. Its innovativeness lies in the fact that it is based on automatic detection of repeating features. Both techniques have some of the best properties needed by compression techniques for distributed use on the Internet.

Acknowledgments: For the test results reported in this paper, the models of Capitol building, Colosseum, Piping and Taj Mahal were downloaded from *http://www.3dcafe.com*. Diwan-i-Khaas model is from Visions Multimedia Visualization Studio, Mumbai, India and the Helicopter model was downloaded from the Avalon 3D archive (*http://avalon.viewpoint.com*). The other triangle mesh models used for our experiments were obtained from the following sources: (a) Dinosaur, Igea, Isis, Knee and Vase were downloaded from *http://www.cyberware.com/samples/*, and (b) Bunny and Horse were downloaded from *http://www.cc.gatech.edu/projects/large_models/*.

References

[1] P. Alliez and M. Desbrun. Valence-driven connectivity encoding of 3D meshes. In *Eurographics 2001 Conference*, 2001.

[2] S. V. Babji. Advancing fan-front : An efficient compression technique for large 3d triangle meshes. Master's thesis, Department of Computer Science, Pondicherry University, India, February 2002.

[3] G. Barequet, C. Duncan, and S. Kumar. RSVP: A geometric toolkit for controlled repair of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), April-June 1999.

[4] G. Barequet and M. Sharir. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12:207–229, 1995.

[5] D. Baum, S. Mann, K. Smith, and J. Winget. Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. In *SIGGRAPH 91*, pages 51–60, 1991.

[6] M. De Berg and K. T. G. Dobrindt. On levels of detail in terrains. In *11th ACM Symposium on Computational Geometry*, June 1995.

[7] G. Butlin and C. Stops. CAD data repair. In *Proceedings of the 5th International Meshing Roundtable*, pages 7–12. Sandia National Laboratory, October 1996.

[8] Gotsman C., Gumhold S., and Kobbelt L. Simplification and compression of 3d meshes. In *Proceedings of the European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS), Munich*, August 2001.

[9] L. Chiariglione. MPEG home page. `http://www.cselt.it/mpeg`.

[10] CoCreate. OneSpace: Virtual conference room for collaborative CAD. http://www.CoCreate.com, 2001.

[11] The Web3D Consortium. The Virtual Reality Modeling Language. http://www.web3d.org, ISO/IEC 14772-1, September 1997.

[12] B. Curless and M. Levoy. A volumentric method for building complex models from range images. In *SIGGRAPH 96*, pages 303–312, August 1996.

[13] M. Deering. Geometry compression. In *SIGGRAPH 95*, pages 13–22, 1995.

[14] V. J. DeLeon and H. R. Berry. Vrnd: Notre-Dame cathedral – globally accessible multi-user real-time virtual reconstruction. In *Proceedings of Virtual Systems & MultiMedia (VSMM98), Ogaki, Gifu, Japan*, November 1998.

[15] Discreet (Autodesk Inc.). 3DStudio MAX R4. http://www.discreet.com, 2000.

[16] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, April-June 1998.

[17] G. Farin. *Curves and surfaces for computer-aided geometric design – a practical guide*. Academic Press, London, 1993.

[18] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice (Second Edition in C)*. Addison-Wesley Publishing Company, 1996.

[19] IBM Research (Visualization Group). VRML compressed binary format. http://www.research.ibm.com/vrml/binary/.

[20] A. Guiziec, F. Bossen, G. Taubin, and C. Silva. Efficient compression of non-manifold polygonal meshes. In *IEEE Visualization 1999*, pages 73–80, 1999.

[21] S. Gumhold and W. Strasser. Real-time compression of triangle mesh connectivity. In *SIGGRAPH 98*, pages 133–140, 1998.

[22] I. Guskov, W. Sweldens, and P. Schroeder. Multiresolution signal processing for meshes. In *SIGGRAPH 99*, pages 325–334, 1999.

[23] N. Haval. Three-dimensional documentation of complex heritage structures. *IEEE Multimedia*, 7(2):52–56, 2000.

[24] M. Isenbueg and J. Snoeyink. Face Fixer: Compressing polygon meshes with properties. In *SIGGRAPH 2000*, pages 263–270, 2000.

[25] M. Isenburg and J. Snoeyink. Compressing the property mapping of polygon meshes. In *Proceedings of Pacific Graphics 2001*, pages 4–11, 2001.

[26] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH 2000*, pages 279–286, 2000.

[27] A. Khodakovsky, P. Schroeder, and W. Sweldens. Progressive geometry compression. In *SIGGRAPH 2000*, pages 271–278, 2000.

[28] D. King and J. Rossignac. Connectivity compression irregular quadrilateral meshes. Technical Report GIT-GVU-99-36, GVU Center, Georgia Tech., Atlanta, USA, 1999.

[29] R. Koenen. MPEG-4: Multimedia for our time. *IEEE Spectrum*, 36(2):26–33, February 1999.

[30] M. Levoy, K. Pulli, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, B. Curless, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D scanning of large statues. In *SIGGRAPH 2000*, pages 131–144, 2000.

[31] P. Lindtrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, and G. Turner. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH 96*, pages 109–118, 1996.

[32] W. E. Lorenson and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH 87*, pages 163–169, 1987.

[33] FEGS Ltd. CADFix: A Product for CAD Data Exchange, Repair and Upgrade. http://www.fegs.co.uk, 1997.

[34] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, 1988.

[35] Sun Microsystems. Java3d API specification. http://java.sun.com/products/java-media/3D, June 1999.

[36] M. E. Mortensen. *Geometric Modeling*. Wiley & Sons, New York, 1985.

[37] M. Nelson. *Data Compression Handbook*. M & T Publishing, Inc., 1991.

[38] R. Pajarola and J. Rossignac. Compressed progressive meshes. Technical Report GIT-GVU-99-05, GVU Center, Georgia Tech., Atlanta, USA, 1999.

[39] L. Piegl and W. Tiller. *The NURBS Book*. Springer Verlag, Berlin, 1996.

[40] G. Roelofs. *PNG: The Definitive Guide*. O'Reilley & Associates, Inc., 1999.

[41] M. A. Rosenman and J. S. Gero. Modelling multiple views of design objects in a collaborative CAD environment. *Computer-aided Design*, 28(3):193–205, 1996.

[42] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, January-March 1998.

[43] J. Rossignac and D. Cardoze. Matchmaker: Manifold B-reps for non-manifold r-sets. In *Proceedings of the ACM Symposium on Solid Modeling*, pages 31–41, June 1999.

[44] Jarek Rossignac. The 3d revolution: CAD access for all! In *Invited key-note paper at the International Conference on Shape Modeling and Applications, Aizu-Wakamatsu, Japan, IEEE Computer Society Press*, pages 64–70, March 1997.

[45] M. Schindler. A fast renormalization for arithmetic coding. In *Proceedings of IEEE Data Compression Conference, Snowbird, UT*, 1998.

[46] D. Shikhare, S. Bhakar, and S. P. Mudur. Compression of large 3D engineering models using discovery of repeating geometric features. In *Proceedings of 6th International Fall Workshop on Vision, Modeling and Visualization (VMV2001)*, November 2001.

[47] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 95*, pages 351–358, 1995.

[48] G. Taubin, W. Horn, F. Lazaurus, and J. Rossignac. Geometry coding and VRML. *Proceedings of the IEEE*, 86(6):1228–1243, 1998.

[49] G. Taubin and J. Rossignac. Geometry compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[50] C. Touma and C. Gotsman. Triangle mesh compression. In *Proceeding of Graphics Interface 98*, June 1998.

[51] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *SIGGRAPH 94*, pages 311–318, 1994.

[52] W. T. Tutte. A census of planar triangulations. *Canad. J. Math.*, 14:21–38, 1962.

[53] G. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[54] P. Wayner. *Compression Algorithms for Real Programmers*. Morgan Kaufman, Academic Press, 2000.

[55] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, 1996.