Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features

Dinesh Shikhare, Sushil Bhakar and S. P. Mudur

National Centre for Software Technology
Gulmohar Cross Rd. 9, Juhu, Mumbai 400049, India
Email: {dinesh|sushil|mudur}@ncst.ernet.in

Abstract

In this paper, we present a new geometry compression technique particularly suitable for 3D mesh models of engineering class – architectural models, machine plants, factories, etc. We observe that such models have a number of repeating features at various levels of granularity. In most of the widely available models in this class, the geometric description of such features are also repeated.

A major distinctive aspect of our compression technique is that repeating geometric features in the geometric model are automatically discovered and then compactly encoded. The proposed method discovers repetition first at the connected component level and then at the subcomponent level across components and also at the aggregate component level. The results from a straight forward implementation tried on large mesh models downloaded from the net are extremely encouraging.

1 Introduction

Large 3D models like architectural designs, heritage monuments, chemical plants and mechanical CAD designs are increasingly being deployed in various applications involving interactive visualization and Internet-based access. To enable compact storage and fast transmission of such models, compression of geometric description of these models has been an area of considerable interest [6, 9, 17, 19, 20]. However, the earlier efforts do not specifically address engineering models.

We introduce a new geometry compression strategy that exploits some common characteristics of 3D mesh models in engineering class. These models consist of many small to medium sized connected components and have geometric features that repeat in various positions, scales and orientations. Although modeling tools (like 3DStudio MAX [8]) and file formats (like VRML) have facilities for specification and representation of repeating features, in practice almost all the models we see have these features repeatedly described.

Our research pioneers a new strategy for compression of such models through the following new contributions:

- a new algorithm for automatic discovery of repeating feature patterns in 3D polygon mesh models,
- compression of geometry of 3D polygon mesh models by removing the redundancy in the representation of repeating geometric feature patterns,
- a new scheme that can incorporate the best results achieved in the area of connectivity compression of polygon meshes, thereby always achieving compression ratios at least as good as those reported in the literature.

In fact for engineering models, our technique gives large benefits which cannot be realised by using the connectivity compression algorithms alone. We demonstrate the efficacy of this new strategy through examples of large engineering models – including some that are freely available on the net – with excellent compression ratios. Figure 1 illustrates some examples. To the best of our knowledge, ours is the first significant attempt to address compression of engineering models.

2 Previous Work

Polygon mesh models: Polygon mesh models are typically defined in terms of (a) *geometry* – the coordinate values of vertices of the meshes making up the model, (b) *connectivity* – the relationship among the vertices which defines the polygonal faces of the mesh (also called as *topology* of the mesh), and (c) *attributes* - such as color, normal and texture co-

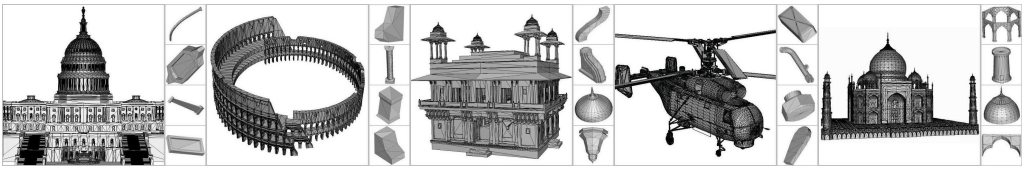


Figure 1: Architectural and engineering models with some examples of most frequently repeating features.

ordinates at the vertices. Other information such as texture images and material properties are also commonly present as a part of the model, usually associated with meshes or groups of meshes. Various compression algorithms have attempted to compress these different components of 3D models.

2.1 Geometry Compression Strategies

Most of the geometry compression strategies have two distinct components: (a) compression of topology or the connectivity information and (b) compression of geometric data.

Connectivity Compression: Typically, 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. Schemes that minimize repeated references to vertices result in compression. Almost all of the connectivity compression techniques [6, 9, 17, 19, 20] encode triangle/polygon connectivity in a lossless manner. Recent techniques [20, 6] 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 [17].

Geometric Data Compression: Deering [3] used quantization of coordinate values from 32 bits to 16 bits, a 9-bit index into a global list of values for vertex normals and a 15-bit representation of color values.

Use of Predictors: 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 [19, 20, 15].

Signal Processing Techniques: Signal processing based techniques which have been in use for surface fairing [18] and multiresolution analysis [7] are based on the construction of a set of basis functions for decomposition of a triangle mesh 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 [10] and wavelet based technique of Khodakovsky et al [11] are examples of this approach.

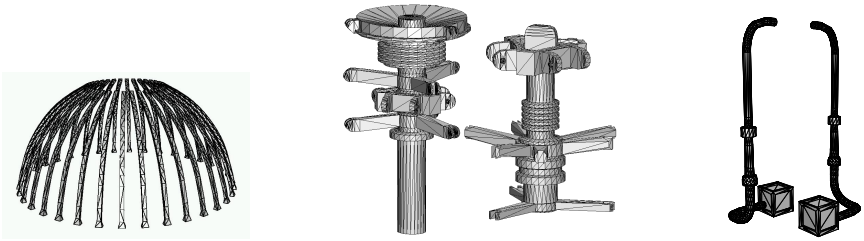
Large Engineering 3D Models: The graph traversal techniques are best suited when very long traversals are possible. Longer the traversal sequence, smaller is the average cost of representation. However, engineering models have small components leading to short traversals, hence somewhat higher average costs.

Note that both predictive encoding and the signal processing approach work best for smooth surfaces with dense meshes of small triangles [11]. Large 3D models of the engineering class usually have a large number of meshes, with small numbers of large triangles, often with arbitrary connectivity. The architectural and mechanical CAD models typically have many non-smooth meshes making these methods less suitable.

3 Discovery of Repeating Features for Compression

3.1 Some Preliminary Definitions

A *polygonal mesh model* O consists of a set S of polygon meshes and associated non-geometric properties. A *polygon mesh* $s \in S$ consists of a set V of vertices, a set E of edges and a set P of polygons. Each vertex corresponds to a point posi-



(a) (b) (c)

Figure 2: Features repeat at different granularities: (a) *component*: part of an heritage model has 36 instances of a feature having 68 vertices and 132 triangles, (b) *sub-component*: components of a mechanical CAD model, and (c) *aggregate*: each structure adds up to 18 meshes, 1112 triangles and 560 vertices.

tion from the set $X = \{x_i \in \mathbf{R}^3\}$. An edge e is represented as a pair (v_1, v_2) of references into the list of vertices. A polygon p is represented as a sequence $(v_1, v_2, ..., v_k)$ of references into the list of vertices. A *triangle mesh* is a special case having all triangular polygons.

In a mesh model O, we call two polygons as *neighbouring polygons* if they share an edge. There exists a *path* between polygons p_i and p_j if there is a sequence of neighbouring polygons $p_i, p_1, p_2, ..., p_j$. A 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 breadth-first or depth-first traversal based labelling algorithm of complexity $\Theta(n)$.

If $U(O)$ and $C(O)$ denote the number of bits for the uncompressed and the compressed versions of the model O respectively, then the *compression ratio* is defined as $CR = \frac{U(O)-C(O)}{U(O)}$.

3.2 Our Approach

The basic approach in our compression scheme is to look for redundancy in the form of repeating geometric features in a given model. This redundancy is then eliminated by suitably encoding the component shapes and their repeating instances. Our algorithm discovers repeating shape features at three different levels of granularity:

▶ *Connected components in polygon meshes*: For example, in a machine plant model, nuts, bolts, fasteners, etc. repeat many times; in architectural models it is common to see structures having many identical parts (see example in Figure 2(a)).

▶ *Sub-component level structures*: Often these models have features repeating within or across connected components. For example, in a mechanical CAD model, a component representing a gear has many teeth. Each tooth corresponds to a feature of this kind. (See example in Figure 2(b)).

▶ *Aggregates of repeating features*: Groups of disjoint features are also found to repeat in many 3D models. Our algorithm discovers such macro-level aggregate features which may be composed from features of the above two types. An example of such macro-level features is illustrated in Figure 2(c).

Our algorithm automatically discovers component sized and sub-component level features and their aggregates that repeat with a rigid body transformations within the model O. We also derive the transformation for each repeating instance of a feature to enable compact encoding in the form of "master geometry – instance transform" hierarchy and for later reconstruction. The master geometry itself can be compressed using the most suitable geometry compression algorithm developed earlier and thus our work is complementary to the earlier work in this field.

3.3 Repeating Component-level Features

In a given 3D model, one does not know which repeating feature patterns to expect. Hence, the traditional approach of matching 3D objects [23, 2, 4, 14, 13] by maintaining a dictionary of features/objects and then retrieving those features in the given model is not applicable for our work. Our goal is to automatically *discover* repeating feature patterns in a polygon mesh models, without using a knowledge base of known features.

In most engineering models there are a large number of small to medium sized connected components, each having upto a few hundred

polygons on an average. For the purpose of compression, it is best to detect redundancies at as large a granularity as possible. A good heuristic for engineering models is to first carry out discovery of repeated feature patterns at the level of connected components. For achieving this goal, we carry out the following steps:
(1) reorganize the total set of polygons in the source model into a set of connected components.
(2) carry out discovery of repeating feature patterns at the connected component level (described in 3.3.1), and build the "master geometry – instance transform" hierarchy.

3.3.1 Detecting Repeating Components

Partial matching of polygonal shapes has been an area of interest in the recent years [23, 2, 4, 14, 13]. Many sophisticated algorithms have been developed for robust matching and alignment of similar shapes. In our implementation we have used a simple and efficient technique based on principal component analysis (PCA) with suitable extensions to overcome its limitations. While this is a very simple technique, it has given us very good results as we will see later in this paper.

Normalized Orientation: We compute an orthonormal basis in 3-space that describes the eccentricities of the connected component using the Hotelling transformation [5]. This basis is used as a pure rotation matrix to bring a component to a normalized (or canonical) orientation.

We take the list of vertices defining the mesh as a cluster of points $X = \{x_1, ..., x_n\}$ to obtain the mean $m = \frac{1}{n}\sum_{i=1}^{n} x_i$ and the covariance matrix, $C = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T - mm^T$. We then find eigenvectors and corresponding eigenvalues of C. The eigenvectors are sorted in increasing order of eigenvalues. The three normalized eigenvectors are used to construct a pure rotation matrix R. If T_{-m} represents a translation by vector $-m$, the transformation $T_{-m} \circ R$ places the mesh in a normalized orientation at the origin. On obtaining an oriented basis for a component, the extremal vertices along each of the vectors in the basis give the size of the oriented bounding box (OBB).

Matching Components: Let us denote the components as O_{c1}, O_{c2}, their means as m_1, m_2 and the orthogonal bases representing the respective eccentricities as R_1, R_2. If the dimensions of the OBBs of the two meshes do not match then no further matching is attempted. Otherwise we carry out a fuzzy comparison of positions of vertices across the components. If the geometry so aligned matches in at least 99% of the vertices, then we declare O_{c2} to be an instance of O_{c1} and also record the transformation composited as $T_{-m_1} \circ Rot \circ T_{m_2}$ required to reconstruct O_{c2} from O_{c1}, where $Rot = R_2 \circ R_1^{-1}$. Matching vertices implies matching their positions, texture coordinates and vertex normals, if they are defined for the given model.

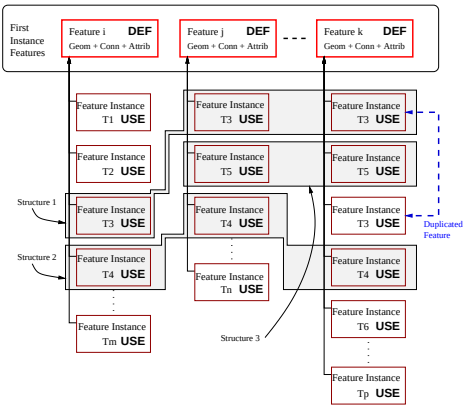


Figure 3: Equivalence classes: tagging of feature instances and identification of *iso-transform* instances as repeating structures. Also note the identification of duplicated components.

On detecting similarity among the components in the model, we partition the model into equivalence classes. All near-identical components belong to a single class. Figure 3 illustrates the construction of equivalence classes of components having similar geometries as USE-instances of the first occurrence of the geometry which is tagged as the DEF-instance. A USE-instance is represented as a reference to the DEF-instance and the transformation required to reconstruct the original component.

Overcoming limitations of PCA: While attempting to match two components using PCA, ambiguity can arise when the components have a completely symmetrical mass distribution because the eigenvectors will be similar. Examples of such a case are: a cylinder, which has a symmetrical mass distribution about an axis and a sphere, which has symmetrical mass distribution about the centre of mass. This limitation of PCA is overcome by a minimization procedure of Novotni and Klein [13] to

obtain the best rotation transformation for matching of components. We consider all possible rotations around the axis or centre point of symmetry and choose the rotation corresponding to the maximum match. For a single axis of symmetry, R is given by

$$R = \max_{R(\alpha)}(M(R(\alpha), O_{c1}, O_{c2})|\phi \in [0, 2\pi])$$

where α denotes the angle of rotation around the axis of symmetry, $R(\alpha)$ the rotation and $M(R(\alpha), O_{c1}, O_{c2})$ the measure of match between the *aligned* components O_{c1} and O_{c2}. In case of spherical symmetry, the best alignment is obtained as

$$R = \max_{R(\phi,\alpha,\psi)}(M(R(\phi, \alpha, \psi), O_{c1}, O_{c2}))$$

where the angles (ϕ, α, ψ) denote Euler angles for parameterization of three-dimensional rotations and $\phi \in [0, 2\pi], \theta \in [0, \pi], \psi \in [0, 2\pi]$.

The rotation space is discretized uniformly. For objects where the mass is symmetrical around one of the principal axes (say, a cylinder), this is an efficient approach. However in cases of spherical symmetry this approach is costly due to the large number of configurations.

3.3.2 Analysis and Acceleration

A naive implementation of the above scheme, involving exhaustive pair wise matching among components, for discovering identical connected component level features in the model has a complexity of $O(n^2)$, where n is the number of connected components in the given model. In most engineering models, n tends to be large. We have accelerated the algorithm by using geometric hashing [12, 22] of components. To achieve hashing of components such that candidates for detailed geometric matching are put into the same bucket, we need some invariant descriptors. In our implementation, we carry out hashing using the number of vertices and triangles as the key, and before performing detailed matching we also check if the dimensions of the OBBs match. While the worst case complexity even after hashing is $O(n^2)$, for most practical situations the benefit is substantial.

Matching two aligned components A and B, each having m vertices, is a procedure of $O(m)$ complexity. Finding the correspondence between the first pair of vertices across the components needs $O(m)$ effort. The correspondence between the rest of the vertices is established by carrying out identical spanning tree explorations in the connectivities of the mesh components while verifying geometric match – another $O(m)$ step.

3.4 Sub-component Feature Patterns

The problem of discovering repeating patterns within and across polygon meshes representing connected components is difficult. The difficulty lies in automatic partitioning of the meshes to cut out some parts that represent repeating patterns. Our technique uses the strategy of "growing" patterns bottom up from vertices. Bioinformatics community has been interested in automatic discovery of repeating structural motifs in molecular structures [21, 16]. However, they limit the discovery to a fixed class of structures.

To enable effective classification of vertices in a given model, we associate a *footprint*[1] with each vertex in the model. To form a footprint we must select invariant properties $I(f)$ over the features $f \in F$ in the model such that under a linear transformation $X' = TX$ of coordinates, $I(f) = I(T(f))$. In our implementation we have sought invariance to rigid-body transformation.

Examples of invariant properties that can become footprints for a vertex v in a polygon mesh are: (a) *Density*: $|N_v|$, the cardinality of N_v, the set of vertices connected to v. The elements of N_v are also called the *neighbourhood* of v, (b) *Size*: L_v, average of the lengths of the edges connected to v; and (c) *Curvature*[2]: D_v, the average of the dihedral angles of the faces meeting at the edges connected to vertex v. In our implementation we use a vector footprint consisting of $|N_v|$, L_v, D_v and a second order descriptor, D_v^2, given by $\frac{1}{|N_v|}\sum D_j$, $j \in N_v$. A simple inner product distance measure has been constructed for determining the similarity or disparity between two given features in the space defined by the footprints. For engineering models this composite footprint has performed very well.

We have chosen to associate the footprints with vertices and not with higher order elements like edges or polygons because in polygon meshes the

[1] We have borrowed the term *Footprint* from the work of Barequet and Sharir on Partial Surface and Volume Matching in Three Dimensions [2]. However our definition and use of footprints are significantly different.

[2] This is only an approximate descriptor of the curvature.

number of polygons and the number of edges are in multiples of the number of vertices.

We start discovering sub-component features across the DEF-instance components obtained in the earlier subsection (3.3). Below, we only give a broad outline of our algorithm:

1. Initialization: Compute the footprints for each vertex v in the model. Create equivalence classes of vertices having near-identical footprints. The vertices having identical footprints represent matching local features and are likely to form good starting points for the "growth" of identical patterns around them.

2. Growth Phase: This phase attempts a simultaneous growth of the patterns around the vertices in the equivalence classes (seeds). The connectivity in the polygon mesh is considered as a graph with vertices as the nodes of the graph and the sides of the polygons connecting vertices making up the edges in the graph. Considering one equivalence class at a time, a breadth-first growth is carried out simultaneously around each seed (see Figure 4), while geometrically verifying (step 3) if the patterns match after each step in traversal.

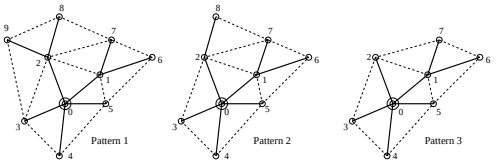


Figure 4: Simultaneous breadth-first growth of repeating features: *Patterns 2* and *3* partially match with *Pattern 1*. Seeds of growth across the pattern are highlighted. Vertices are numbered in order of traversal.

3. Verification Step: This step has two purposes, firstly verify geometric match between the two features identified as similar based on their footprints and connectivities, secondly determine the transformation. Given two patterns represented as two sequences of vertices having one to one correspondence, the patterns tested for a match we select some three non-collinear vertices from one pattern and the corresponding set of three vertices from the other pattern to determine the transformation to align these two sets of vertices. The two patterns then aligned using this transformation to verify whether the patterns match geometrically within a tolerance.

This algorithm returns a set of patterns (subgraphs) which correspond to features that repeat in the given model. Since the process of discovering the repeating patterns uses mesh connectivity for traversal, the matching patterns are identical in coordinates of the vertices (in the sense of rigid-body transformation) as well as the connectivity. This "growth" algorithm enables us to discover partially matching feature patterns within and across components.

The discovered repeating features are also organized into a data-structure represented by Figure 3. Thus the data structure now consists of repeating features of component and sub-component types.

Compression of Vertex List: On obtaining the patterns, we renumber the vertices and also the indices in the polygon lists such that the sequences of the vertices belonging to the repeating patterns are grouped by the patterns and have a contiguous numbering within the groups. The vertex list for each pattern forms a node that represents either the first occurrence of the pattern or its repeated occurrence. If the pattern node is the first occurrence then it consists of an integer k indicating the number of vertices in the pattern followed by $3k$ coordinates. If it is a repeated occurrence, then it consists of an integer reference to the first occurrence, a rigid-body transformation to reconstruct the original position and orientation and a bit-pattern indicating the elements of this occurrence that match with the first occurrence. The rotation component of the transformation is represented using a quaternion (4 floats). Translation vector requires 3 floats.

Thus a repeated instance of a pattern having k vertices requires three integers and seven floating point numbers as opposed to $3k$ floating point numbers needed in the original vertex list. We note that this scheme of encoding begins to yield compression for a repeating pattern of four or more vertices.

3.5 Aggregate Features

After carrying out feature instance detection, many USE-instances of different meshes are found to have identical rigid body transformations. This *iso-transformation* set enables us to infer *repeating macro-level structures*. Figure 3 shows aggregation of iso-transformation instances (drawn in shaded envelopes) to identify macro structures in the model. Note that these macro structures may

not always correspond to complete user identifiable component aggregates, there may be gaps. These gaps do not represent any shortcoming of this approach, since the goal is not to precisely reconstruct the complete structures.

As a side benefit of considerable value, our scheme also lets us automatically identify the common problem of erroneous replication of components in large models. In Figure 3, we see that there are two instances of *Feature-k* having the same rigid body transformation associated with them. This is an undesirable case of coinciding geometries in the model. It is almost impossible to identify such cases visually to heal the model to remove such artifacts. It must also be noted that a USE-instance having an identity transformation associated with it is also a duplicate component, and must be removed.

4 Results and Discussion

We carried out our tests with highly encouraging results on a large number of 3D models that are available for download on the net. It is important to note that this performance has been achieved without using any special techniques for connectivity encoding of the DEF-meshes in the model.

Discovery of repeating component shapes: The following table shows the results of our technique for discovery repeating component shapes in five models.

Model	# comp.s in original	# DEF comp.s	# USE comp.s
Capitol	2662	93	2569
Colosseum	1129	20	1109
D'-Khaas	3726	848	2878
Helicopter	976	480	496
Taj Mahal	375	45	330

The savings achieved in storing the data in terms of actual amount of geometry and connectivity are listed in the table below.

Model	# verts (orig)	# tris (orig)	# verts (DEF)	# tris (DEF)
Capitol	52606	87258	10347	19944
Coloss	69868	135159	18912	38103
DIK	295695	162590	44363	46165
Helicopt	105079	187929	76231	136372
Taj M	65323	126453	28427	55238

Discovery of repeating sub-component features: The sub-component level discovery of patterns leads to additional savings in the storage requirements for the list of vertices. The following table lists savings obtained by compressing the vertex list of the DEF-components.

Model	# verts in DEF comps	# verts in repeating patts	% savings
Capitol	10347	2145	20.73
Colosseum	18912	4203	22.22
D'-Khaas	44363	8945	20.18
Helicopter	76231	11334	14.86
Taj Mahal	28427	6532	22.97

Reduction in Storage Requirements: The following table shows reduction in file sizes (in bytes) from the original raw data to the compressed format.

Model	orig	orig (gzip)	cmpr	cmpr (gzip)	CR (%)
Capit	1.8M	875K	433K	211K	88
Colos	2.5M	2.1M	570K	402K	84
D'-Kh	10.1M	2.9M	1.9M	671K	93
Helic	4.9M	1.9M	1.7M	899K	82
Taj	2.3M	1.1M	757K	467K	80

The compression achieved here is by avoiding detailed multiple descriptions of repeating features. Only DEF-instances of features are written out in detail. For the DEF-instance geometry encoding, we have simply used `gzip`. This gives us an idea of the minimum compression performance we can expect. Clearly we can achieve even better performance by using better connectivity compression schemes for the DEF-components.

5 Conclusion

We have presented a new 3D compression scheme particularly suited to very large engineering models with automatic discovery of repeating features at its core and test results from a our implementation of this scheme on a number of large models including those that are available on the net have shown excellent compression performance.

The fundamental contribution of automatically discovering similar shape features in a large 3D model has ample scope to be applied to the other 3D geometry handling processes of healing, simplification and progressive transmission.

Acknowledgements: For the test results reported in this paper, the models of Capitol building, Colosseum and Taj Mahal were downloaded from `www.3dcafe.com`. Diwan-i-Khaas model [1] is

from Visions Multimedia Visualization Studio and the Helicopter model was downloaded from the Avalon 3D archive (`avalon.viewpoint.com`).

References

[1] Fatehpur Sikri: An Epic in Red Sandstone. http://www.visions-net.com, Visions Multimedia Visualization Studio, Mumbai, India., 1999. (First public demonstration as a showcase application during the launch of Intel PIII processor in San Jose, USA.).

[2] Gill Barequet and Micha Sharir. Partial Surface and Volume Matching in Three Dimensions. *IEEE PAMI*, 19(9):929–948, 1997.

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

[4] Chitra Dorai and Anil K. Jain. COSMOS - a representation scheme for 3d free-form objects. *Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, 1997.

[5] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[6] S. Gumhold and W. Strasser. Real-time Compression of Triangle Mesh Connectivity. In *SIGGRAPH 98*, pages 133–140, 1998.

[7] I. Guskov, W. Sweldens, and P. Schroeder. Multiresolution Signal Processing for Meshes. In *SIGGRAPH 99*, pages 325–334, 1999.

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

[9] M. Isenbueg and J. Snoeyink. Face Fixer: Compressing Polygon Meshes with Properties. In *SIGGRAPH 2000*, pages 263–270, 2000.

[10] Z. Karni and C. Gotsman. Spectral Compression of Mesh Geometry. In *SIGGRAPH 2000*, pages 279–286, 2000.

[11] A. Khodakovsky, P. Schroeder, and W. Sweldens. Progressive Geometry Compression. In *SIGGRAPH 2000*, pages 271–278, 2000.

[12] B. Lamiroy and P. Gros. Rapid object indexing and recognition using enhanced geometric hashing. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England, volume 1*, pages 59–70, April 1996.

[13] Marcin Novotni and Reinhard Klein. A Geometric Approach to 3D Object Comparison. In *Proceedings of International Conference on Shape Modelling and Applications (SMI2001)*, May 2001.

[14] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3D Models with Shape Distributions. In *Proceedings of International Conference on Shape Modelling and Applications (SMI2001)*, May 2001.

[15] R. Pajarola and J. Rossignac. Compressed Progressive Meshes. Technical Report GIT-GVU-99-05, GVU Center, Georgia Tech., Atlanta, USA, 1999.

[16] Xavier Pennec and Nicholas Ayache. A geometric algorithm to find small but highly similar 3D substructures in proteins. *Bioinformatics*, 14(6):516–522, 1998.

[17] J. Rossignac. Edgebreaker: Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, January-March 1998.

[18] G. Taubin. A Signal Processing Approach to Fair Surface Design. In *SIGGRAPH 95*, pages 351–358, 1995.

[19] G. Taubin and J. Rossignac. Geometry Compression through Topological Surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[20] C. Touma and C. Gotsman. Triangle Mesh Compression. In *Proceeding of Graphics Interface 98*, June 1998.

[21] Xiong Wang, Jason Wang, Dennis Shasha, Bruce Shapiro, Sitaram Dikshitulu, Isidore, and Kaizhong Zhang. Automated discovery of active motifs in three dimensional molecules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD97), California*, pages 89–95, August 1997.

[22] Haim J. Wolfson and Isidore Rigoutsos. Geometric Hashing: An Introduction. *IEEE Computational Science & Engineering*, pages 10–21, October-December 1997.

[23] Andrew Zisserman, David Forsyth, Joe Mundy, Charlie Rothwell, Jane Liu, and Nic Pillow. 3D Object Recognition using Invariants. Technical Report OUEL 2027/94, Robotics Research Group, University of Oxford, November 1994.