Error Resilient Streaming of Compressed VRML*

C. Bajaj, S. Cutchin, V. Pascucci, G. Zhuang
Center for Computational Visualization
Dept of Computer Sciences and TICAM
University of Texas, Austin TX, 78712

1 Introduction

We present a Java[1]/VRML[2] visualization tool that supports the dynamic downloading of compressed VRML 3D models. The visualization tool runs in any Java enabled web client with a VRML browser plug-in[2], as well as a standalone Java application using Java3D. The visualizer supports client/server interaction with remote 3D object servers that can stream encoded VRML objects. The encoding server can stream a compressed representation of the model whose granularity and locality allows for robust incremental recontruction even using an error prone transmission layer. Retrieval from remote Web servers and JDBC compliant databases is also supported. The visualizer, using a Java decoder, can display decompressed VRML objects in a VRML browser. The decoding of the objects occur in independent threads providing visualization of incremental results as soon as they become available without interfering with the users ability to navigate the object. The visualizer is built using the component of a distributed Web based collaboratory toolkit[1]. The collaboratory toolkit uses open standards such as Java Beans, and CORBA, as well as a modular design in order to provide system wide components based interoperability. The Java decoder and display environment support error recovery, error resiliency, and error correction of received compressed VRML data. Our main contributions are:

- A new error resilient, incremental compression encoder, in C++, provided as a server process, that is particularly suitable for streaming compressed VRML over unreliable networks or for broadcasting on one-way connections (with no back channel).
- A interactive VRML client that encapsulates the pure java decoder of compressed VRML.
- A dynamically downloadable applet that enables internet access from any java enabled web browser.

Wide area access to three-dimensional models has many applications in CAD, CAAD, Education, Entertainment, and Advertising. The ability to dynamicaly, manipulate, and view a full scale model collaboratively can greatly enhance users ability to understand, and take advantage of this data. The current system has been used by the authors to actively demonstrate various geometric compression algorithms via laptop computers over the web at remote conferences at interactive rates.

2 Streaming of Arbitrary Triangle Meshes

In this section we detail the layered encoding of the geometric models being transmitted. At this end we assume two possible streaming setting: (i) duplex channel transmission like internet connection, (ii) simplex (one way) channel transmission like TV broadcasting.

*This research is supported in part by grants from NSF-CCR-9732306, NSF-KDI-DMS-9873326, DOE-ASCI-BD-485, and NASA-NCC 2-5276.
[1]Trademark of Sun Microsystems, Inc.
[2]Virtual Reality Modeling Language

We partition the input geometry into a sequence of small blocks where dependency of the data is confined to adjacent blocks. This aims to minimize the consequences of faulty transmission. In case (i) the number of blocks eventually being retransmitted is keept small, while in case (ii), where retransmition is not allowed, the geometric model can still be viewed even with the occurrence of small holes corresponding to lost blocks. In some cases such holes can be patched to reduce the visual impact of the data loss. In both cases we also use the locality of the block partitioning to achieve incemental encoding/transmission/decoding so that the client can start to decode and display the received portion of data when the sender is still encoding and transmitting the rest of the model.

The class of triangle meshes that can be encoded and transmitted is broader than the class considered by several single resolution and progressive compression methods [11, 4, 6, 3, 12, 10, 8, 7, 9]. This is particularly important since in the design of an encoding scheme for data transmission we cannot afford to reject commonly available models. As Gueziec et al. [5] have observed real life data encompass frequently non-manifold structures or other "bad" features that most encoding schemes like Topological Surgery [11] can deal with only indirectly by splitting the object at non-manifold features. Our definition of encodable mesh is simply "any set of triangles" withour any particular assumption about the class of connectivity among them.

Definition 1
A **triangular mesh** *is a set of vertices, edges and triangles such that (i) each vertex is the boundary of some edge and (ii) each edge is the boundary of some triangle.*

This is a very weak definition that encompasses any manifold, non-mainfold, non-orientable traingular mesh. It actually encompasses even meshes that are not usually considered triangulations and other "bad cases" that may not be included in any of the previous categories (e.g. pairs of coincident triangles). In this way we can deal with a general class of input data with no need to filter the input data to fix the "bad cases" cases or even discard or reject the input triangulation. Moreover the method guarantees that the output can faithfully reproduce the given input.

2.1 Data Partitioning: Triangle and Vertex Layering

The method we use for partitioning the data is inspired by the layering scheme introduced in [11] that has been shown to well capture the redundacy present in geometric data. Here we provide a modified definition of layering that allows to deal with non-manifold and other king of triangular meshes. The key difference is that in [11] the construction of the layering is made using sequences of edges su that one contour cannot degenerate to a vertex. Here we instead assume to perform a traversal of the graph formed by the mesh edges.

Consider a mesh M with a set $\mathcal{V}$ of vertices, $\mathcal{E}$ of edges and $\mathcal{T}$ of triangles. We consider the graph $G = (\mathcal{V}, \mathcal{E})$ formed by the edges and vertices of M. We partition the vertices in $\mathcal{V}$ into layers on the basis of a breadth-first traversal of the graph G.

Definition 2 (VERTEX-LAYER)

- *The 0^{th} VERTEX-LAYER includes a single (arbitrary) vertex of the mesh (actually the initial vertex-layer could be any arbitrary set of vertices).*
- *The k^{th} VERTEX-LAYER (with $k > 0$) includes a vertex V if V is not included in any previous VERTEX-LAYER and there exist an edge $E = (V, V^*)$ where V^* is included in the $(k-1)^{th}$ VERTEX-LAYER.*

Property 1
By construction we have that any edge in the mesh can have its extreme vertices included in two consecutive vertex-layers or have its extreme vertices included in the same vertex-layer.

Property 2
By construction we have that any triangle in the mesh can have its vertices included in two consecutive vertex-layers or have its three vertices included in the same vertex-layer.

Definition 3 (TRIANGLE-LAYER)
The k^{th} TRIANGLE-LAYER (with $k \geq 0$) includes a triangle T if T has one vertex in k^{th} VERTEX-LAYER and T is not included in any previous TRIANGLE-LAYER.

Figure 1 shows how the layering structure is constructed.

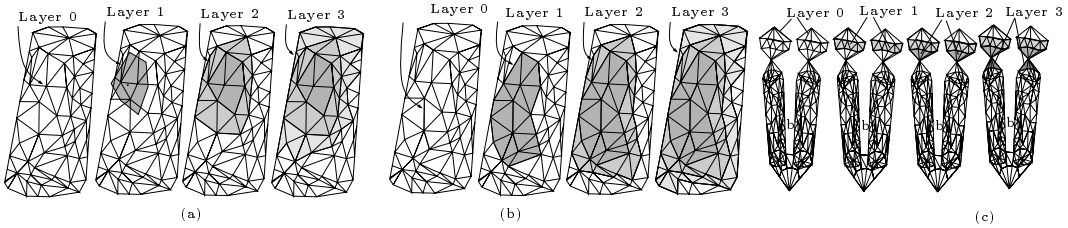

FIGURE 1: Layering construction (a) starting from a single point, (b) starting from a set of vertices forming a path and (c) starting from two unconnected points and passing through non-manifold vertices.

2.2 Data Encoding: Strips, Fans & Contours

Our connectivity encoding represents explicitly the layering structure of the input surface mesh. On the basis of the layering structure we characterize some edges/vertices of the mesh:

Definition 4 *A* CHORD *is an edge that connects two vertex layers.*

Definition 5 *A* TRANSVERSAL *is an edge that connects two vertices in the same vertex layer.*

Definition 6 *A BRANCHING POINT is a vertex of the mesh incident to more than two transversals.*

The geometric primitives we use for encoding are three:

Definition 7 *A* CONTOUR *is an ordered sequence $\{V_1, V_2, \ldots, V_n\}$ of vertices where each pair of consecutive vertices $\{V_i, V_{i+1}\}$ is connected by a transversal edge and which intermediate vertices $\{V_2, \ldots, V_{n-1}\}$ are incident to exactly two transversal edges. Note that a* CONTOUR *can be a single vertex in which case it is also called isolated point.*

Definition 8 *A* STRIP *is an ordered sequence of triangles in a triangle-layer where each pair of consecutive triangles share a common edge. The set of vertices of the triangles in the strip must belong to two COUNTOURS laying in two different vertex-layers.*

Definition 9 *A* FAN *is an ordered sequence of triangles in a triangle-layer where all the triangles have a common vertex and each pair of consecutive triangles share a common edge. Possibly the first and last edges are also coincident but no edge is shared by more than two triangles in the FAN. Note that a FAN can degenerate into a single triangle in which case is also called special triangle and may be encoded in a special way for efficiency reasons.*

In our encoding scheme we first partition the mesh into layers. Then each vertex-layer is partitioned into contours while each triangle-layer is partitioned into strips and fans. In the reconstruction stage we take the set of strips and fans and glue them together to recover the original shape. The structure of the decomposition into layers allows to perform this process quickly because the search for common edges between triangles is performed locally in a triangle layer or in a vertex layer.

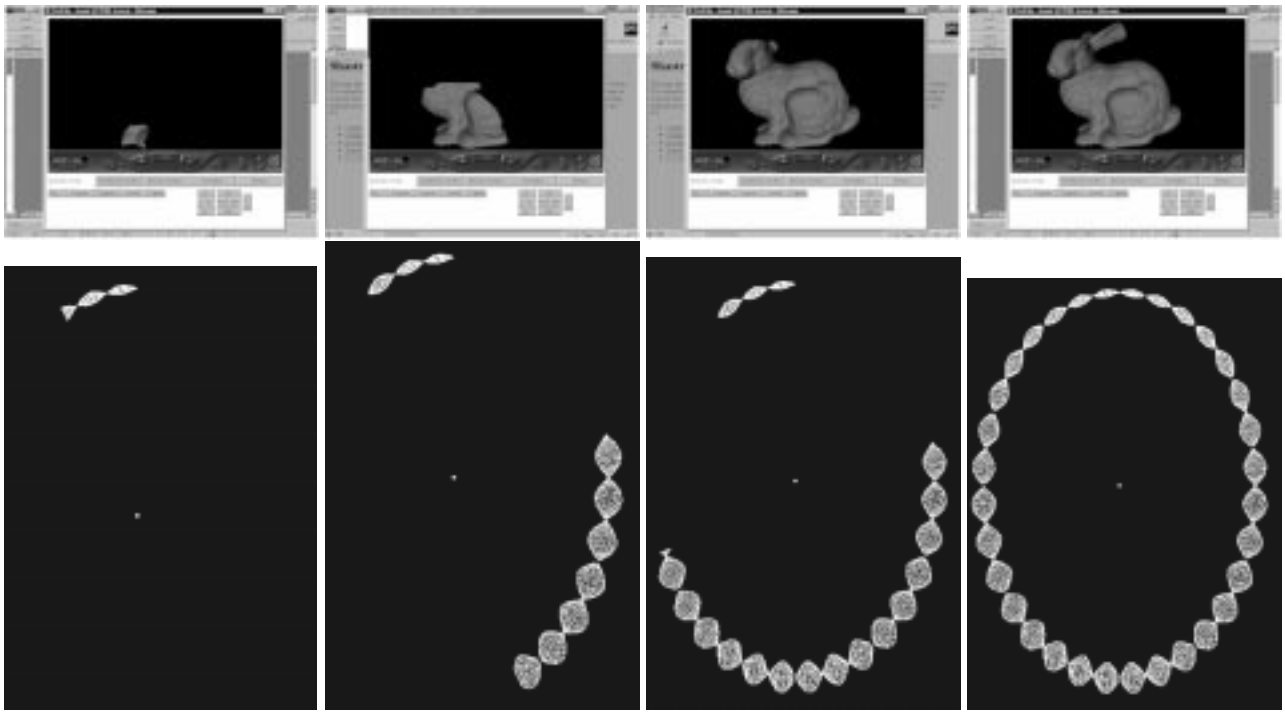

FIGURE 2: Top: Four snapshots showing the order from left to right in which the bunny model is incrementaly reconstructed on the client side. Bottom: the incremental reconstruction of the non-manifold necklace model.

COMPLETENESS To show that any mesh (under definition 1) can be encoded and recovered in this scheme we just need to observe that:

- The set of edges of any mesh can always be partitioned into layers by a breadth first traversal of the edges graph.
- Each triangle in the mesh will belong to some triangle-layer.
- One can always encode all the triangles in the mesh just representing each triangle as a fan of one element.

In the above proof we are using a worst case scenario were the encoding is very inefficient. In practice we tend to use long strips to achieve efficiency in the encoding without losing the generality of the approach. Figure 3-left shows the comparison between compression obtained with our method and the compression obtained using gzip that is currently the most used compression method for transmitting geometric models. In the diagrams, where each geometric model is a single point, the orizzontal axis is the size of the model while the vertical axis is the ratio of the size of the gizipped file over the size of our encoding. Using lossy compression on the vertices geometry one can achieve even better compression ratios. Using the Topological Surgery higher compression ratios may be achieved on average but at the cost of not being able to encode some of the models. In the following section we discuss some features that are not available in any other compression scheme and that make the present approach particularly suitable for streaming.

2.3 Error Resilient Incremental Transmission

In our scheme we encode each triangle layer and each vertex layer independently and transmit them in sequence. This decomposition has a very strong locality property:

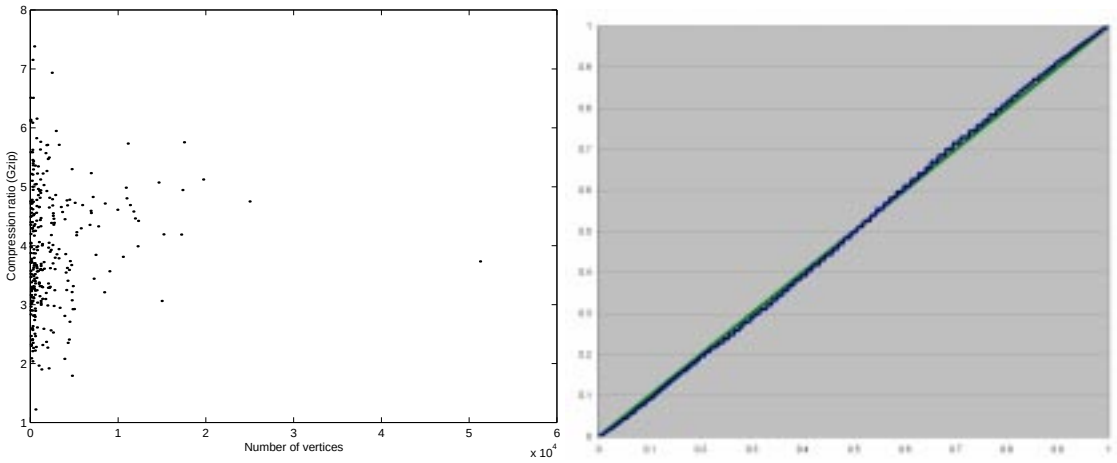

FIGURE 3: The left figure shows compression ratio between the gzipped version of the input model over our lossless encoding (left) and over our lossy encoding (right). The right the incremental decompression of the Bunny model plotted as number of triangles received versus number of triangles decoded and rendered. The stright line is the ideal incremental decoding line.

- every triangle layer is dependent only from the two adjacent vertex layers.
- every vertex layer is necessary only to decode the two adjcaent vertex layers.

As a consequence the encoder can start sending the data layer by layer as soon they are ready and symmetrically the cliend can decode and display the data while it is being received. Figure 3-right show the portion of triangles decoded (vertical axis) as a function of the portion of bits received. Note that actual line approximates very well the ideal stright line. Figure 2 shows the incremental reconstruction of a manifold and a non-manifold model.

Another important consequence deriving from the locality property stated above is the possibility to implement error resilient transmission. In particular the consequence of an error that may occurr in the transmission of one layer remains confined to the local region so that the decoder can still make progress in the triangle display. The retransmission required to recover the entire model is confined to the single layer. In a one-way transmission like Television broadcasting since the client cannot ask for retransmission it is imporant that it is able to make the best use of the portion of correct data received. Figure 4 shows the comparision of the actual geometry being sent and the model being displayed when received with different errors in the comunication.

Comparison of Compressed Model Sizes				
Model	RAW	GZIP	Our Scheme	No. Layers
Bunny	2,268,236	803,297	139,693	141
Horse	1,087,962	273,344	49,989	93
Eight	50,600	15,594	3,466	17
Cow	29,003	11,245	2,983	15
Fandisk	627,975	142,600	21,956	56
Mannequin	39,967	17,285	3,522	21
Golfclub	13,426	5,609	1,377	12
Necklace	761,809	179,661	37,312	468

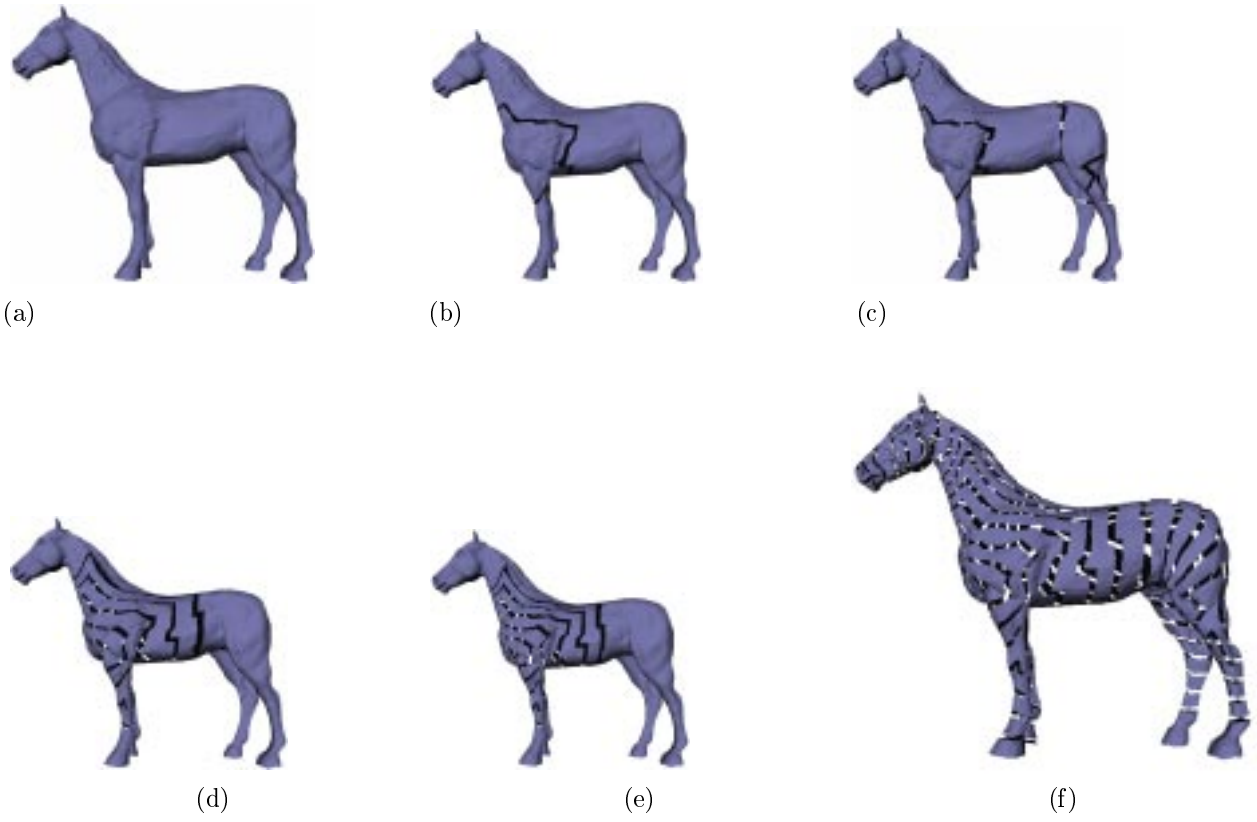

FIGURE 4: Example of error resiliance in horse dataset. (a) Original model, (b-f) models received from a noisy connection with 1,3,9,10, and 19 layers lost respectively.

3 Visualizer and Encoder Server

The visualization environment is composed of two interoperating applications. A java based visualization application, and a special purpose geometry compression server. These two applications interoperate to provide visualization of streaming compressed VRML objects and dynamic compression of VRML scenes.

3.1 Visualizer

The Visualizer is a pure Java client capable of displaying VRML scenes, including compressed geometry VRML scenes and binary format VRML scenes. It supports collaborative navigation and interogation of shared scene graphs. The Visualizer can be used independently or collaboratively to view objects and scenes retrieved from Web servers, JDBC compliant databases, and the specialized Encoding Server. The Visualizer is available as a Web based Applet using the VRML External Access Interface. This allows it to execute in any Java Capable Web browser with an VRML plugin. This ensures the widest possible distribution and access of the visualizer. Currently versions running in Netscape Communicator and Microsoft Internet Explorer are available. The visualizer is also available as a standalone Java application using the Java3D VRML browser and supporting CORBA interaction. This application is available for high performance situations in which

users with specialized high performance needs wish to engage in small group collaborative visualization of large scenes.

FIGURE 5: Architecture of the visualiztion environment

3.2 Encoding Server

The Encoding Server acts a back end server for the Visualizer. It supports multiple protocols for inter-application communication including CORBA IIOP. The server responds to requests for stored objects and can deliver these to the requester (visualizer) in several formats, Original format, Compressed VRML, Zipped VRML, and Ascii VRML. The server can be used to act as a compression back end server for database objects. For example several objects within a database can be transferred to the server over the network and maintained in a compressed format for deliver. When a visualizer requests an object in the database that has already been cached in the object server it can be retrieved in compressed format. Or the database itself could simply use the compression facilities to compress objects stored in the database.

3.3 Session Manager

The underlying collaboration substrate provides an session manager application that creates and manages collaboration sessions. It is responsible for controlling and coordinating the interaction of the other applications. The session manager maintains sets of Geometric entities shared amongst users in collaborative sessions.

Figure 1 shows a comparision of the time required to receive and display VRML models in uncompressed, gzipped and our compression scheme. As can be seen from the table our scheme provides improvements over the gzip compression, as well as being able to display the model as soon as some portion of the data is received.

Transmission Times in Seconds			
Model	RAW	GZIP	Our Scheme
Bunny	.71	.33	.22
Horse	.38	.22	.16
Eight	.16	.11	.06
Cow	.11	.11	.11
Fandisk	.28	.22	.16
Mannequin	.17	.16	.16
Golfclub	.11	.16	.008
Necklace	.27	.17	.11

TABLE 1: Transmission times for different model formats.

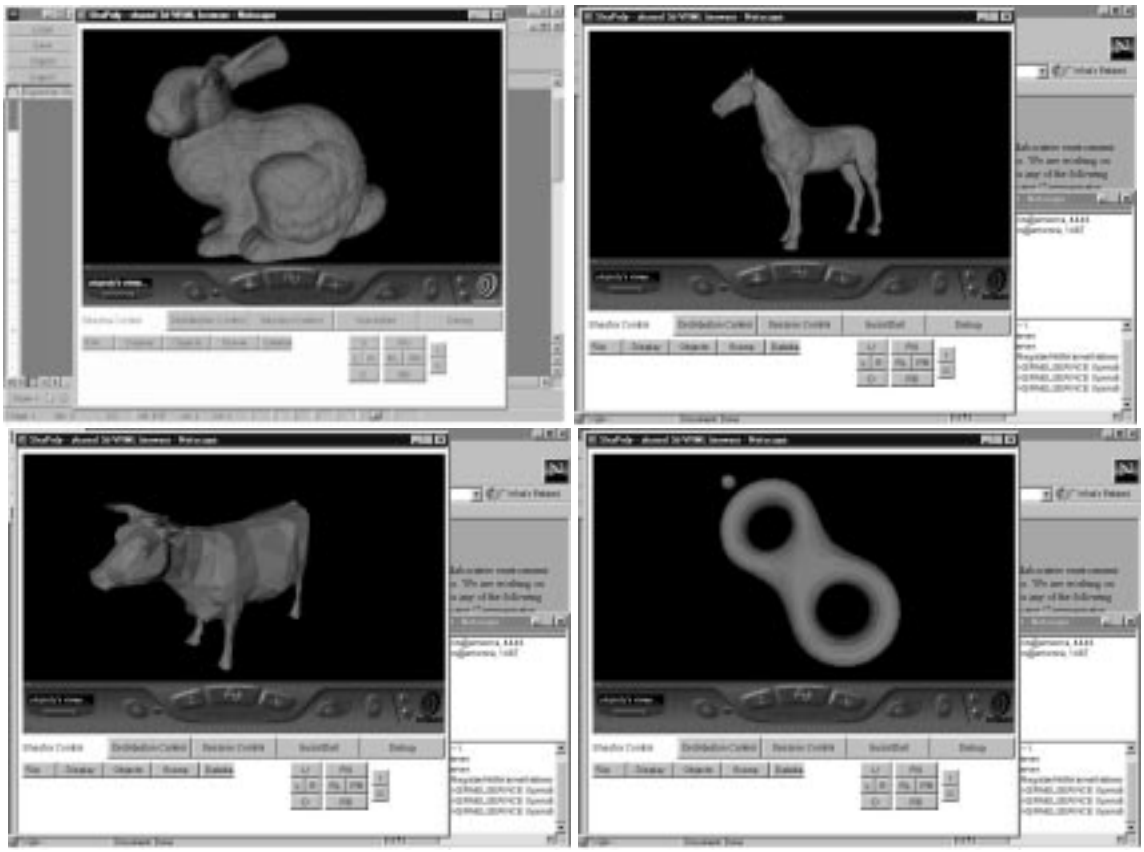

FIGURE 6: Four screenshots of completed incremental reconstructed models in the web based visualization browser.

References

[1] http://www.ticam.utexas.edu/ccv/collabweb.

[2] http://www.ticam.utexas.edu/cutchin/javabed.

[3] Mike M. Chow. Optimized geometry compression for real-time rendering. In *Proceedings of IEEE Visualization '97*, pages 347–354, Phoenix, AZ, 1997.

[4] M. Deering. Geometric compression. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 13–20, 1995.

[5] Andre Gueziec, Gabriel Taubin, Francis Lazarus, and William Horn. Cutting and stitching: Efficient conversion of a non-manifold polygonal surface to a manifold. Technical Report RC-20935, IBM T.J. Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, 1997.

[6] H. Hoppe. Progressive meshes. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[7] J. Li and C. Kuo. Embedded coding of mesh geometry. Technical report, ISO/IEC JTC1/SC29/WG11 MPEG98/M3325, March 1998, 1998.

[8] Jiankun Li, Jin Li, and C.-C. Jay Kuo. Progressive compression of 3d graphic models. In *IEEE Proceedings of Multimedia Computing and Systems*, Ottawa, Canada, 1997.

[9] Jiankun Li, Jin Li, and C.-C. Jay Kuo. Progressive coding of 3d graphic models. In *IEEE Multimedia and Systems*, 1998.

[10] Gabriel Taubin, Andre Geuziec, William Horn, and Fancis Lazarus. Progressive forest split compression. *Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 123–132, 1998.

[11] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2), 1996.

[12] Costa Touma and Craig Gotsman. Triangle mesh compression. In Wayne Davis, Kellogg Booth, and Alain Fourier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 26–34, San Francisco, June18–20 1998. Morgan Kaufmann Publishers.