STREAMING 3D GEOMETRY DATA OVER LOSSY COMMUNICATION CHANNELS

Stephan Bischoff *Leif Kobbelt*

Computer Graphics Group, RWTH–Aachen, Germany

ABSTRACT

In this paper we propose a progressive 3D geometry transmission technique that is robust with respect to data loss. In a preprocessing step we decompose a given polygon mesh model into a set of overlapping ellipsoids, representing the coarse shape of the model, and a stream of sample points, representing its fine detail. On the client-side, we derive a coarse approximation of the model from the ellipsoid decomposition and then re-insert the sample points to reconstruct the fine detail. The overlapping ellipsoids as well as the sample points represent independent pieces of geometric information, hence partial data loss can be tolerated by our reconstruction algorithm and will only lead to a gradual degradation of the reconstruction quality. We present a transmission scheme that is especially well-suited for geometry broadcasting where we exploit that the order of the sample points can be arbitrarily permuted.

1. INTRODUCTION

During the last decades the internet has evolved from a low-bandwidth, text-only medium into an ubiquitous resource of multimedia documents. Digital documents nowadays contain not only textual information but also images as well as audio and video footage. These multimedia extensions are not only mere document attachments but are often tightly integrated into the document structure, e.g. as (live) audio and video streams. User interaction, however, is hardly possible and mostly restricted to predefined actions at hot-spots [1]. Animated 3D models on the other hand enable intuitive and realistic interaction with the displayed objects and allow for effects that cannot be achieved with conventional video animations. Consequently, the current challenge is to integrate 3D geometry as a new data type into digital documents as is done e.g. in VRML and MPEG4 [2, 3].

Apart from the enrichment of individual digital documents, numerous other applications are emerging. Digital libraries could provide extensive archives of 3D models which can be accessed and searched for various purposes. Large scale design and engineering projects can employ such central data bases for configuration management — especially if some components are designed by distributed teams. In computer aided learning (CAL), the extension of educational material by 3D models enables the students to get a more thorough grasp of the subject-matter. Finally, broadcasting 3D animations instead of plain video will open the door to new qualities in immersive digital television.

Today, the most prevalent representation for 3D models are polygonal meshes in general and triangle meshes in particular. These representations allow to approximate models of arbitrary shape and topology within any desired precision and a wide range of data structures, algorithms and implementations for efficient generation, modification and storage of polygonal meshes is available.

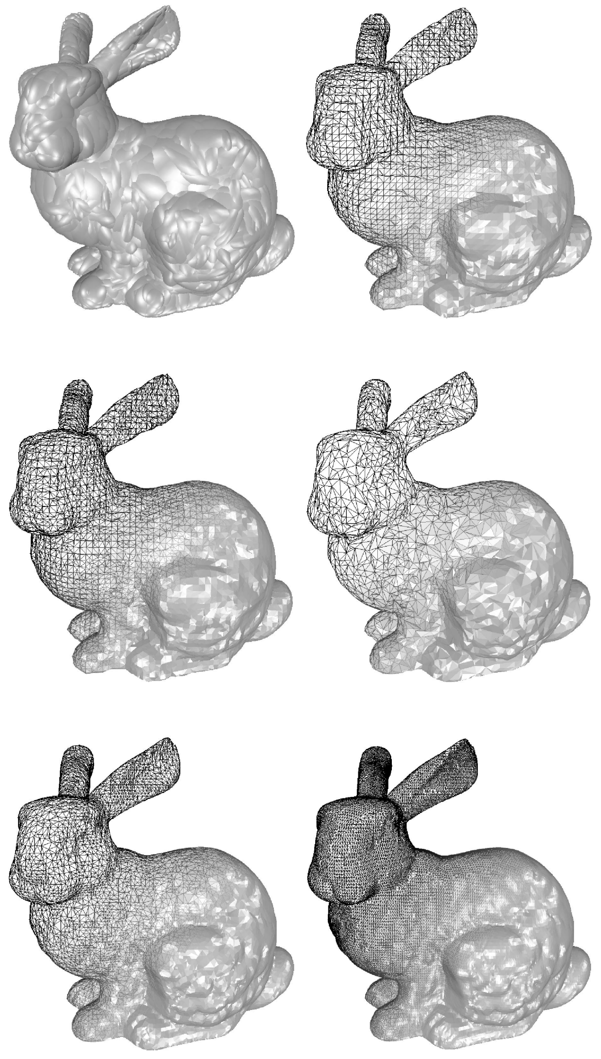
Figure 1: The figure above depicts a typical reconstruction sequence: From the ellipsoid model of the bunny (upper left, 300 ellipsoids) the reconstruction algorithm extracts a marching cubes mesh (upper right, 10000 vertices). After the first chunk of received vertices was inserted into the mesh (middle left, 5000 vertices inserted), the points of the marching cubes mesh are removed by a decimation algorithm (middle right, 5000 vertices). The remaining vertices are inserted into the resulting mesh (lower left, 10000 vertices inserted, lower right 30000 vertices inserted).

When transmitting polygonal meshes over possibly low-bandwidth network channels one has to carefully pay regard to the space- and time-efficiency of the employed algorithms. Geometry compression schemes provide highly compact encoding of polygonal meshes' geometry as well as of their connectivity [4, 5, 6]. Progressive transmission schemes reorder the data chunks such that crucial shape information is sent first while less important detail is transmitted later [7]. Recent development efforts even combine both strategies [8, 9].

An important issue that has to be addressed when transmitting 3D geometry is the robustness of the transmission, i.e. the way the reconstruction algorithm reacts to (partial) data loss. Note that compressed as well as progressive meshes exhibit complex local and global data inter-dependencies, like well-ordering of the data sequence and fixed vertex neighborhoods. Hence, the reconstruction process completely fails if only a single bit of data is lost or altered during transmission.

In principle, a standard communications protocol, like the internet transmission protocol (TCP/IP), which is implemented on some lower software layer, can taken to be responsible for the correct transmission of the data [10]. However, in broadcasting scenarios where a central server sends the same data to a multitude of clients, individual back-channels might be difficult or even impossible to implement. Since errors cannot be reported back to the server to trigger re-transmission, robust transmission schemes become mandatory for 3D broadcasting.

In the following we propose a transmission scheme that is both robust and progressive. In Section 2 we will give an overview of the scheme, Section 3 elaborates on a possible application scenario, experimental results are presented in Section 4.

2. ROBUST GEOMETRY TRANSMISSION

Robust transmission schemes should be able to handle data loss, data duplication and data disordering. In particular, partial data loss should only lead to a gradual degradation of the approximation quality. Hence, our general idea is to decompose an object into independent pieces of information that allow to reconstruct certain parts of the object even if other pieces get lost.

In our approach we chose ellipsoids as the basic independent piece of geometric information, i.e. we decompose the object into a set of overlapping ellipsoids that covers the object's interior. Note that as no data inter-dependencies between the ellipsoids exist, the transmission order of the ellipsoids is arbitrary. Furthermore, because each interior point of the object is in general covered by several ellipsoids, moderate data loss during transmission will not affect the reconstructed shape or its topology significantly.

As it turns out, already a few ellipsoids suffice to convey the basic shape and topology of an object. To save bandwidth, we chose to represent the fine geometric detail of an object by simple sample points, which can be considered as degenerate ellipsoids. Loss of sample points will result in a less detailed reconstruction but the overall shape of the object will not be affected.

Note that explicit connectivity information is attached neither to the ellipsoids nor to the sample points. Hence it is not our intention that the connectivity of the reconstruction is the same as the connectivity of the original model.

The general approach is as follows: On the sender side we first transmit the base geometry as a set of ellipsoids. Then the fine detail is transmitted as a stream of sample points. The client derives a coarse approximation of the geometry from the set of ellipsoids and then successively re-inserts the sample points into this approximation.

2.1. Encoding the base geometry

Our basic idea is to approximate a given 3D model by a set of overlapping ellipsoids that fill out the interior of the given object (Figure 3). As each of these ellipsoids represents an independent piece of geometric information, shape redundancy is introduced by the fact that the ellipsoids overlap each other.

The ellipsoid decomposition has to be done only once in a preprocessing step. For this we attach a small sphere to each of the mesh points and then grow it into the interior of the object until it cannot be stretched anymore.

The resulting set E of ellipsoids typically forms an extreme over-representation of the objects volume. For our purposes, however, it suffices to select a small but at the same time geometrically significant subset $E' \subset E$. For this we use a greedy algorithm: At the beginning, E' is initialized to the largest ellipsoid in E. Then we successively add those ellipsoids to E' that yield the largest volume gain until a prescribed error tolerance is reached. The set E' is then transmitted to the clients. For a detailed description of the whole ellipsoid fitting and decomposition process see [11].

Each ellipsoid Q can be represented implicitly by

$$Q(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} + 2\mathbf{b}^T\mathbf{x} + \gamma = 0$$

where $A \in I\!R^{3\times3}$ is symmetric, $\mathbf{b} \in I\!R^3$ and $\gamma \in I\!R$ (see [12]). Normalizing the above equation such that $\gamma = 1$, we see that each ellipsoid Q can be efficiently encoded by only 9 scalar values.

On the client side, we initialize a discrete spatial distance field to $+1$ (*outside*) everywhere. When an ellipsoid is received, the client first computes its bounding box by determining the main axes of the ellipsoid. By evaluating $Q(\mathbf{x})$ for each grid point $\mathbf{x}$ within the bounding box, every interior vertex is marked by -1 (*inside*). When all ellipsoids have been received, the client extracts the zero-surface from the spatial grid using e.g. the marching cubes algorithm [13]. The resulting surface is the desired base geometry.

Obviously, this method is invariant with respect to reordering and duplication of ellipsoids during transmission. Furthermore, because of the strong overlap of the ellipsoids, even in the reduced set E', each interior grid point is generally covered by several ellipsoids. Hence a small amount of ellipsoid loss will not change the resulting shape or topology significantly.

Notice that the vertices that were generated by the marching cubes algorithm are in general different from the original objects vertices. Hence they have to be removed as soon as enough original vertices are received to maintain the geometry (see below). This is easily done by collapsing them into their nearest original neighbor [14].

2.2. Encoding the detail

In our transmission scheme, detail information is represented by sample points that are distributed on the surface of the original model. In the simplest case these are the vertices of the original object. Other data, like materials and texture coordinates can also be associated to each point and are treated like additional point coordinates.

The client refines the initial base geometry by successively inserting the sample points into the current reconstruction. If points get lost during transmission the reconstruction will be less detailed but the basic shape of the object will not be affected.

Whenever the client receives a sample point, it is inserted into the current approximation. As connectivity information is not available, the point is inserted into the closest triangle by a 1-to-3 split. We use a space partitioning technique to make this nearest neighbor search reasonably fast. In general, however, this procedure creates triangles of bad aspect ratio, so-called caps and needles [15]. To remove these, we have to apply a sensible retriangulation strategy locally around the inserted point.

In a planar setting, the Delaunay-triangulation of a point set would be a good choice as it fulfills certain fairness criteria, e.g. it maximizes the triangles' roundness [16]. There is an easy algorithm to transform an arbitrary planar triangulation into a Delaunay triangulation, namely: Flip edges as long as the minimal angle of the triangulation increases. This algorithm can trivially be generalized to triangulations embedded in space and works well in areas that are almost flat. For highly curved regions, however, it produces counter-intuitive results, like flipped triangles.

We proceed as follows: If the surface around an edge is sufficiently flat, i.e. if the normal variation of the adjacent triangles is below some threshold, the edge is flipped so as to maximize the minimal angle of the triangulation. In the planar case this provably results in a Delaunay triangulation. However, when the normal variation is too strong, we use some custom-tailored heuristics to prevent the creation of e.g. flipped triangles.

Although the flipping process described above could in theory affect the complete mesh, in practice we observe only local edge flipping around the inserted point. A typical reconstruction sequence is shown in Figure 1. A detailed description of the complete reconstruction process is given in [17].

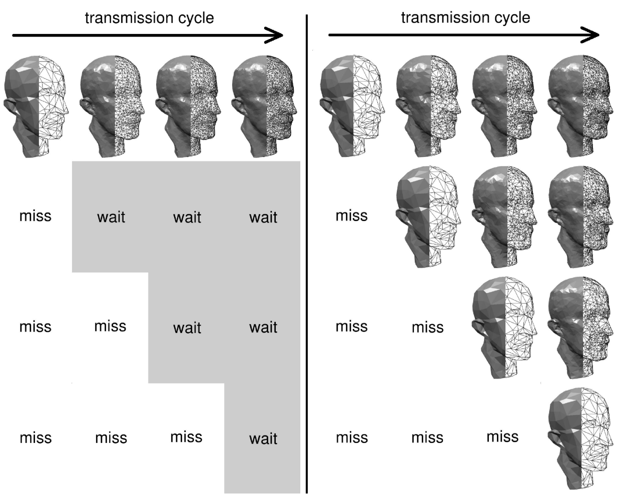

Figure 2: Broadcasting 3D geometry: On the left a conventional progressive transmission cycle is shown. Clients that listen from the beginning of the cycle, can display a sequence of meshes with increasing quality. However, if even the smallest prefix of the stream is missed, one has to wait until the next cycle starts. On the right we depict the transmission scheme proposed in Section 3 where we repeat the base mesh several times and interleave its transmission with the remaining geometry data. Reconstruction can therefore start several times during one transmission cycle. This significantly reduces the latency on client side while not introducing much redundancy.

3. PROGRESSIVE BROADCASTING

Broadcasting of 3D geometry is a scenario where robust transmission schemes are indispensable. Note that in general it is not necessary to transmit geometry information at the same rate as images or textures: it often suffices to update the geometry only once per scene. Even so, because of the sheer size of the models, progressive transmission is needed to promptly show an approximation to the client user while he is waiting for the detail information to be received.

In our setting we assume that in order to transmit geometry data over a *channel*, it is split into packets of fixed size that are sent from a central server over some (radio) network to a multitude of clients. Although packets may get lost or may be reordered during transmission we assume that the integrity of each *received* packet is guaranteed by the underlying channel. This can easily be achieved by appending a checksum to each packet and treating malformed packets as lost.

Interaction between clients and server is in general not feasible: A back-channel does most often not exist and even if it does, responding to individual clients' requests separately would most probably over-strain the network and/or the server capacity. Hence the only possibility for interaction on the client side is to switch channels.

Ideally, whenever a client tunes in, he should be served at least an approximation of the model without significant delay. Standard progressive transmission schemes do not perform very well for this task: If the client switches channels shortly after the base mesh has been transmitted he has to wait a complete transmission cycle until the next repetition of the base mesh takes place (Fig. 2).

Our robust transmission scheme allows for receiving the detail information in almost arbitrary order. We exploit this fact and send the base geometry more frequently interleaved with the stream of sample points. When the client switches to a new channel he only has to wait for the next repetition of the base geometry and then can insert all following sample points (Fig. 2).

Suppose, e.g., that the stream of sample points is divided into n equally-sized chunks $d_1, \ldots, d_n$ and let t be the time necessary to transmit a single chunk. Conventional progressive transmission schemes cyclically transmit the base geometry b, whose size is typically negligible, followed by the chunks in order:

$$\begin{tikzcd} b \arrow[r] & d_1 \arrow[r] & d_2 \arrow[r] & \cdots \arrow[r] & d_n \arrow[llll, bend left] \end{tikzcd}$$

Hence, in the worst case the client has to wait $n \cdot t$ seconds before he can reconstruct the base mesh. If, however, we alternate the base geometry and the chunks,

$$\begin{tikzcd} b \arrow[r] & d_1 \arrow[r] & b \arrow[r] & d_2 \arrow[r] & \cdots \arrow[r] & b \arrow[r] & d_n \arrow[llllll, bend left] \end{tikzcd}$$

the worst case waiting time for the base geometry is reduced to t seconds. Notice that any following chunk of sample points can immediately be inserted into the mesh.

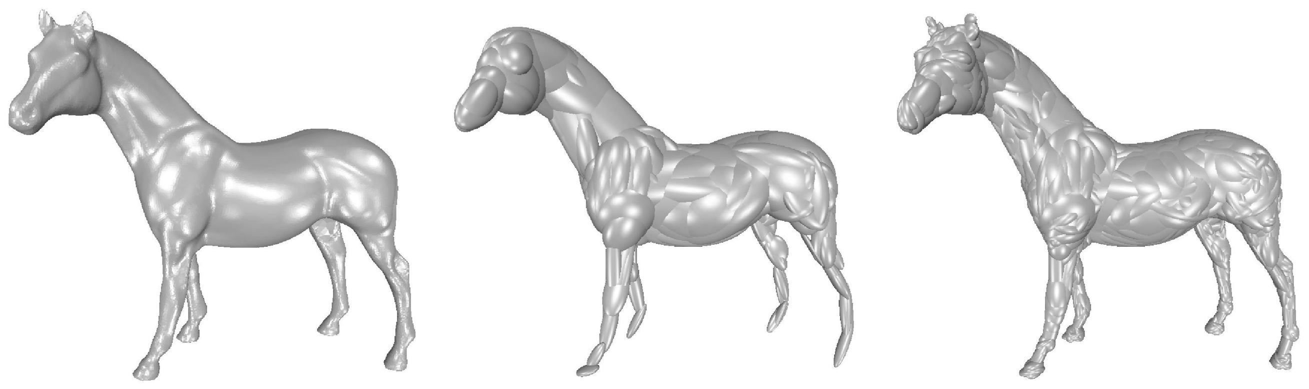

Figure 3: From left to right: Original horse model (49000 vertices), ellipsoid approximations consisting of 100 and 400 ellipsoids respectively (corresponding to 3600 and 14400 bytes).

4. RESULTS

Figure 3 depicts a typical ellipsoid decomposition of an object necessary as a preprocessing step for the actual transmission. We randomly selected 2500 of the original model's 49000 vertices and applied the ellipsoid growing algorithm to them. Then we computed a suitable subset E' as described in Section 2.1. The complete preprocessing step took about half an hour.

Figure 1 shows a typical reconstruction sequence as described in Section 2.2. The base geometry is represented by a 300 ellipsoid approximation (10800 bytes), the detail is divided into 6 chunks consisting of 5000 vertices (60000 bytes) each. The overhead for repeatedly transmitting the base geometry is only about 15 percent.

An alternative approach would be to approximate the base geometry by a decimated mesh instead of a collection of ellipsoids [17]. However, in this case the correct transmission of the whole base mesh would be critical while in our case the size of the critical blocks is only the size of the channel packets.

It turned out that the sample vertices can be re-inserted into the reconstruction in practically any order. In all our experiments the scheme has proven to be sufficiently robust even if more than half of the vertex data gets lost. This even allows us to refine a mesh only locally in regions of interest and deliberately discard vertices that are not relevant from the current viewpoint.

5. ACKNOWLEDGEMENT

This work was supported by the Deutsche Forschungsgemeinschaft under grant KO2064/1-1, "Distributed Processing and Delivery of Digital Documents".

6. REFERENCES

[1] S. E. Chen, "QuickTime VR — An image-based approach to virtual environment navigation," in *SIGGRAPH 95 Proceedings*, 1995, pp. 29–38.

[2] A. Ames, D. R. Nadeau, and J. L. Moreland, *The VRML 2.0 Sourcebook*, John Wiley & Sons Inc, 1996.

[3] R. Koenen, "Overview of the MPEG4 standard," in *http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm*, 2001.

[4] M. Deering, "Geometric compression," in *SIGGRAPH 95 Proceedings*, 1995, pp. 13–20.

[5] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proceedings of Graphics Interface*, 1998, pp. 26–34.

[6] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 47–61, 1999.

[7] H. Hoppe, "Progressive meshes," in *SIGGRAPH 96 Proceedings*, 1996, pp. 99–108.

[8] P. Alliez and M. Desbrun, "Progressive encoding for lossless transmission of 3D meshes," in *SIGGRAPH 01 Proceedings*, 2001, pp. 195–202.

[9] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *SIGGRAPH 00 Proceedings*, 2000, pp. 271–278.

[10] C. Hunt, *TCP/IP network administration*, O'Reilly, 1992.

[11] S. Bischoff and L. Kobbelt, "Ellipsoid decomposition of 3D models," to appear in 3D Data Processing, Transmission, Visualization Proceedings, 2002.

[12] W. Boehm and H. Prautzsch, *Geometric Concepts for Geometric Design*, AK Peters, 1994.

[13] W.E. Lorensen and H.E. Cline, "Marching Cubes: A high resolution 3D surface reconstruction algorithm," in *SIGGRAPH 87 Proceedings*, 1987, pp. 163–169.

[14] G. Turk, "Re-tiling polygonal surfaces," in *SIGGRAPH 92 Proceedings*, 1992, pp. 55–64.

[15] J. R. Shewchuk, *Delaunay Refinement Mesh Generation*, Ph.D. thesis, Carnegie Mellon University, Pittsburg, 1997.

[16] H. Edelsbrunner, *Geometry and Topology of Mesh Generation*, Cambridge Univ. Press, England, 2001.

[17] S. Bischoff and L. Kobbelt, "Towards robust broadcasting of geometry data," to appear in Computers and Graphics.