Towards Robust Broadcasting of Geometry Data

Stephan Bischoff Leif Kobbelt
Computer Graphics Group, RWTH–Aachen, Germany

Abstract

We present new algorithms for the robust transmission of geometric data sets, i.e. transmission which allows the receiver to recover (an approximation of) the original geometric object even if parts of the data get lost on the way. These algorithms can be considered as hinted point cloud triangulation schemes since the general manifold reconstruction problem is simplified by adding tags to the vertices and by providing a coarse base–mesh which determines the global surface topology. Robust transmission techniques exploit the geometric coherence of the data and do not require redundant transmission protocols on lower software layers. As an example application scenario we describe the teletext–like broadcasting of 3D models.

Keywords: multimedia databases, digital libraries, videotext, interactive broadcasting, polygonal meshes

1 Introduction

During the last years, multimedia applications have conquered the internet. While one decade ago plain text documents were the standard type of data that was transferred over the global network, today most digital documents include at least images if not audio clips or even video footage. The next challenge is to extend this set of standard media by the new data type of *3D geometry*. The inclusion of 3D data into multimedia documents enables intuitive and realistic interaction with the displayed objects which goes far beyond effects one can achieve with pre–computed video animations.

In the context of digital library applications, we can think of extensive archives of 3D models that can be accessed and searched by remote users for various purposes. Large scale design and engineering projects, e.g., can employ such central data bases for configuration management — especially if some components are designed by distributed teams. Educational material, presented online, can be enhanced significantly by 3D models because they enable truly interactive learning experiences. Moreover, broadcasting 3D animations instead of plain video will open the door to new qualities of service in digital television.

Consequently, quite substantial research has been done in this area which is mainly focussed on two major problem complexes: *compression* and *progressive transmission* of geometric data sets. Both complexes are motivated by the fact that 3D datasets, e.g. polygonal meshes, can be huge (several megabytes) and hence the available transmission bandwidth severely limits the complexity of 3D models that can be used in web documents.

Geometry compression schemes [12, 14, 16] investigate highly compact encodings for geometric data sets. We distinguish between loss–less and lossy compression depending on whether the original geometry can be reproduced exactly or not. While powerful general–purpose compression algorithms are known, specialized geometry compression schemes can usually achieve higher compression rates since they exploit the particular coherence and regularity which is inherent to geometric data sets.

The most effective compression schemes encode the connectivity of a polygonal mesh as a set of instructions which can be interpreted by an abstract automaton that reconstructs the mesh. The vertex positions in a mesh can usually be reconstructed from a set of small correction vectors that are added to predicted vertex positions which are derived from a linear combination of neighboring vertices.

Progressive transmission [1, 6, 8] of geometric data means that the server starts by sending coarse shape information that can be reconstructed and displayed by the client immediately. Subsequently, more and more detail information is transmitted which allows the client to refine the coarse initial model. By this technique, the amount of data is not reduced but the ordering of the information chunks is optimized such that crucial shape information is sent first and less important detail information is then transmitted in the order of decreasing relevance. Since the client user immediately can see and interact with an approximation of the final 3D object the waiting time is not perceived as disturbing. Longer waiting time yields better visual quality and if the geometric details are not required, the transmission can be aborted at anytime.

One issue that has not been addressed so far is the *robustness* of the transmission of geometric data, i.e. how is the object reconstruction on the client side affected by partial loss of data during transmission. The reason for this is that the standard internet transmission protocol (TCP [7]) which is implemented on some lower software layer, is usually taken responsible for the error–free data transmission[1].

In principle this is true, however, as we already saw for the geometry compression schemes, specialized algorithms can perform much better if they exploit the special coherence in the particular type of data. This means that robust transmission protocols for geometry data are likely to require much less redundancy in the transmission code compared to general–purpose protocols.

Shifting the responsibility for robust reconstruction under partial loss of geometry data to some higher software layer also makes sense from the system design point of view since in today's desktop computers the resource of transmission bandwidth is much stronger restricted than the computing power of the main CPU. Hence more complicated geometric reconstruction algorithms are acceptable as long as they are able to process the incoming data faster than it is received over the internet.

In *broadcasting scenarios* where a central server sends the same data out to a multitude of clients, individual back–channels are usually difficult to implement. Hence, even if the clients could detect the loss of data during transmission, they could not report errors to the broadcasting server to trigger re–transmission. As a consequence, robust transmission becomes mandatory for 3D broadcasting — especially in heterogenous networks with widely varying bandwidth. In Section 4 we will describe a 3D broadcasting protocol that allows the receiver to tune into a running program with minimum latency.

The general principle for the robust transmission of geometric data is to find independent pieces of information that allow to reconstruct a certain part of the object even if other pieces get lost.

[1] The UDP protocol [7] enables faster data transmission than TCP since no explicit connection between the sender and the receiver is established. Packages of data are sent by the server without having the client acknowledge their proper receipt. If a back–channel is nevertheless required for a given application, it has to be established by the application software.

Notice that for compressed and/or progressive meshes very complicated local and global dependencies between all pieces of information exist. These dependencies include vertex neighborhood relations (*what happens if a neighbor is missing?*) as well as strict orderings of the individual information packets (*instruction sequence for the reconstruction automaton*). As a consequence, the whole reconstruction fails as soon as one single bit is lost.

The smallest independent unit in the context of polygonal mesh datasets is one single vertex. Hence our goal is to find an efficient algorithm that is able to reconstruct a polygonal mesh from its vertices alone. Obviously this is not possible in general since a given set of sample points allows a large variety of possible mesh connectivities. This is why we focus on meshes in *normal form*, i.e. among all possible triangulations of a progressively transmitted point cloud we want to reconstruct the (one) mesh that has a certain normalizing property.

If exact reconstruction on the client side is not possible due to loss of packets, the robustness of the transmission is measured by how good the result approximates the original mesh for a given percentage of lost data. Ideally we expect a *gradual degradation* such that losing a few packets will not have a strong impact on the resulting shape.

From the literature on point cloud triangulation [2, 3, 4, 5] we know that manifold reconstruction from a cloud of sample points is a difficult problem in general. In our case however, we do not have to reconstruct the surface topology from scratch. Since the sender (server) already knows the topology before it sends the independent vertices, it can add special attributes to the vertices that make the topology reconstruction easier.

In this paper we propose a set of algorithms for the robust transmission of geometry. In our case we simplify the general manifold reconstruction problem by first sending a very coarse base mesh in the conventional way. Here we have to rely on correct transmission guaranteed by some lower level communication protocol. Once the base mesh is received, the remaining vertices can be transmitted independently via an non–ideal (unsecure) channel. Since the base mesh complexity is usually much smaller than the complexity of the complete data set, the special treatment of the base mesh does not affect the overall performance significantly.

In the context of 3D broadcasting (Section 4), we achieve this redundancy for the base mesh by simply sending it more frequently than the other parts of the geometric data. As a side effect we can thereby reduce the latency for the reconstruction if the receiver misses a prefix of the data stream.

2 Normal forms for meshes

Given a set of vertices, a mesh is said to be in normal form if its connectivity has a specific structure or satisfies prescribed conditions. In the 2–dimensional setting, *Delaunay* triangulations are the most prominent example of normal forms since they are uniquely defined, optimal with respect to some minimum inner angle criterion, and there are efficient algorithms to compute them for a given set of points [21].

For 2–dimensional manifolds in 3–dimensional space, the situation is more complicated and we necessarily lose some of the ideal properties of planar Delaunay triangulations. The solution is to either be more restrictive with respect to the distribution of vertices or to be less restrictive with respect to the uniqueness of the normal form.

The first approach leads to meshes with special connectivity structures. The most common example for such a special connectivity are *semi–regular* meshes. These meshes consist of regular submeshes (with all vertices having valence 6) and isolated extraordinary vertices (with valences $\neq 6$). A semi–regular mesh can be thought of as being generated by taking a coarse base–mesh with arbitrary connectivity and applying a uniform refinement operator to it [15, 18].

Due to the regular structure of the subdivided patches, every vertex can be indexed by a reference b to the base triangle from which it was generated and barycentric indices (i, j, k) with $i+j+k = 2^r$ which indicate the relative position within that regular patch. Associating every vertex $\mathbf{v}$ with an index (b, i, j, k) enables a simple reconstruction scheme where the client can insert the incoming vertices at the correct position in the uniformly refined base mesh [10].

Since geometric datasets usually do not come with a semi–regular structure, they have to be converted into this normal form — a process called *remeshing* [11, 9, 17]. This requires a resampling of the original geometry with samples being aligned to a piecewise regular grid. Although the remeshed approximation of the original mesh can be tuned to meet any prescribed tolerance, the resampling always brings in alias errors, especially if the original shape has sharp features.

The other strategy to define normal forms for triangle meshes is to be less restrictive with respect to the uniqueness: instead of requiring global uniqueness, we might be satisfied with *local uniqueness*. Typical algorithms to generate such meshes are based on *edge flipping* (see e.g. [24]).

Usually some local criterion is used to measure the quality of an edge in the mesh. This criterion can use the length of the edge, the minimum inner angle of the adjacent triangles, or the normal jump across that edge to rate its quality. In addition we measure the quality of the same edge if it was flipped (Fig. 1). An edge is then considered locally optimal if its quality is better than the quality of its flipped version.

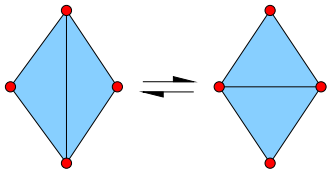

Figure 1: The edge flip operation switches to the other diagonal in the surrounding quadrilateral.

By using a greedy algorithm that runs over all edges and flips them if they are not locally optimal, we can generate meshes that are in normal form with respect to the flipping criterion. Obviously such flipping strategies tend to get stuck in local minima which prevents the normal form from being globally unique. Notice that there are important exceptional cases. For example there are simple edge flipping algorithms for planar triangulations that are guaranteed to converge to the unique Delaunay triangulation [21]. Hence, in this case local uniqueness implies global uniqueness. For 2–dimensional manifolds in 3–space, however, this convergence is not guaranteed.

2.1 Sharp features

When we defined the normal forms we implicitly assumed that vertices represent the geometric component of a given mesh while the connectivity represents the topology. Consequently we could change the connectivity to convert a given mesh into normal form without significantly changing its geometric shape.

In the presence of sharp features, however, this assumption does not hold. Here triangles and edges represent actual geometry and cannot be flipped (Fig. 2). In the context of robust shape reconstruction we therefore treat sharp features like curves on the mesh sur-

face and locally disallow the edge flipping to preserve their topology. This can be implemented by attaching a flag to each vertex indicating if it is part of a feature curve or not.

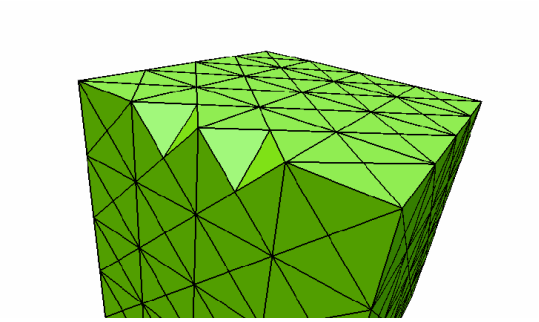

Figure 2: Edge flipping at sharp features leads to significant and disturbing changes in the surface geometry.

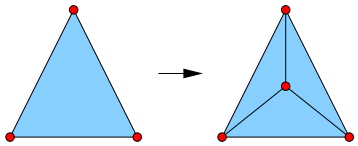

Figure 3: 1–to–3 split: Inserting a vertex into a triangle results in 3 new triangles.

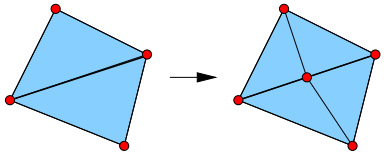

Figure 4: Feature edge split: The two edges arising from splitting the feature edge are also tagged as feature edges.

3 Robust transmission

Robust transmission is implemented by a geometric reconstruction algorithm on the client side that is able to recover the original shape from a coarse base mesh and a sequence of surface sample points. No explicit connectivity information is attached to these points and the algorithm must be able to reconstruct an approximation of the original shape even if some of the points get lost during transmission. The final result is a mesh in normal form based on the received portion of the vertices.

3.1 Semi–regular meshes

For semi–regular meshes every vertex $\mathbf{v}$ has a unique index (b, i, j, k). Here b indicates the base triangle from which this vertex was generated by refinement. From

$$l \;=\; \log_2(i+j+k) \;-\; \log_2(q)$$

with q being the greatest common divisor of i, j, and k we can derive the refinement level on which $\mathbf{v}$ was generated.

For every incoming vertex, the reconstruction algorithm has to refine the current mesh adaptively until the vertex with index (b, i, j, k) becomes existent. Then the position of that vertex is set to $\mathbf{v}$. The positions of all the vertices that have been generated in the vicinity of $\mathbf{v}$ to keep the adaptively refined mesh balanced, can be predicted by some stationary subdivision scheme (see e.g. [10]).

In the semi–regular setting, every point can be considered as a coefficient to some subdivision basis function (see e.g. [18]). The support of these basis–functions depends on the refinement level l which implies that coefficients from coarse levels can encode large deformations of the mesh. To propagate the insertion of coarse level coefficients to the neighboring mesh vertices one has to run a push–pull algorithm that re–evaluates the predicted vertex positions [10].

3.2 Irregular meshes

For irregular meshes no global indexing scheme is available in general. Hence the only global relation between surface points is geometric proximity. In this case the reconstruction algorithm works as follows:

If the incoming vertex is no feature vertex, we locate that triangle of the current mesh with the smallest Euclidean distance and insert the vertex into this triangle by a 1–to–3 split operation (see Figure 3). The three edges of this triangle are pushed on a FIFO queue. To re–establish the local normal form we use a greedy approach for flipping the edges in the queue. Each edge is tested against an edge flipping criterion (see below). If the edge is flipped, the edges in its vicinity are also inserted into the queue. The algorithm runs until the queue is empty since this indicates that all edges in the local vicinity satisfy the normal form criterion. Note however, that feature edges must never be flipped.

Feature points are treated differently. In this case we have to find the nearest feature edge in the current mesh. These feature edges have to be tagged in the base mesh and the tags are inherited during refinement: splitting a feature edge yields two new ones. The feature vertex is then inserted into the feature edge as shown in Figure 4. As before, non–feature edges in the vicinity are pushed on a queue.

3.3 Implementation details

The algorithmic tasks of our reconstruction scheme can roughly be divided into two parts: The first is to create and maintain a spatial data structure for the fast location of triangles and feature edges in the mesh. The second is to evaluate the flipping criterion.

3.3.1 Triangle and feature edge location

In order to quickly locate triangles and feature edges we use a simple space partitioning technique. An appropriate bounding box of the original model is subdivided into $n_0 \times n_1 \times n_2$ voxels. For each voxel we store a list of the triangles that intersect that voxel. When a new triangle is generated during vertex insertion, we "voxelize" it into the space partition in the following way: For each voxel in the bounding box of the triangle we test whether it intersects the triangle or not (see e.g. [22]). This can be done very fast by exploiting the separating axis theorem as described for example in [20]. If there is an intersection we add the triangle to the voxel's triangle list. This might seem inefficient at the first glance, but especially in later stages of the transmission process most triangles will be very small and therefore they will occupy only few voxels (or even just one voxel) which can be detected quickly. When we remove a triangle from the mesh we remove the triangle from the space partition accordingly.

In order to locate the nearest triangle to a point p, we check all the triangles that are within a ball of radius d around p. These triangles

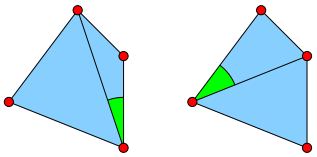

Figure 5: Minimum angle criterion: An edge should be flipped if the minimum angle of the adjacent triangles is maximized. In this case we would decide for the right configuration.

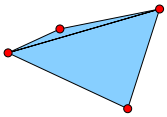

Figure 6: Area criterion: If the ratio of triangle areas is very small or very large, the smaller triangle is probably not relevant for the geometry.

can be located quickly by examining the voxels that intersect that ball. If we do not find a triangle within the ball, we iteratively try the radii $2d, 3d, \ldots$ and so on.

For locating the nearest feature edge to a point, we note that if an edge intersects a voxel, the triangles adjacent to the edge will also intersect that voxel. Hence, we can use the same data structure for locating the nearest feature edge to a point. The computational overhead compared to a dedicated data structure for locating edges is only minimal. Furthermore, most 3D models have only few feature edges compared to the overall number of edges, so this operation will only be used occasionally.

3.3.2 Edge flipping criterion

When re–establishing the normal form after inserting a new vertex into the mesh, we use an edge flipping criterion to decide if an edge is locally optimal or not. We have tested different geometric criteria and eventually decided to use a combination of them.

Minimum angle criterion
Our main intention was to generate visually appealing meshes. For two dimensional meshes this can be achieved by using the so-called Delaunay triangulation. Briefly, a triangulation of a point set P is called Delaunay if the circumcircle of each triangle a, b, c contains only the vertices a, b, c. Unfortunately, this criterion cannot be readily generalized to two dimensional manifolds embedded in the three dimensional space (see e.g. [21]).

However, for the planar case there is an equivalent Delaunay criterion which better matches our normal form definition. First create an arbitrary triangulation of the point set. Second, flip edges as long as the minimum angle of the adjacent triangles gets bigger. This is illustrated in Figure 5. The criterion can easily be generalized to three dimensional space. However, its use is strictly justified only in "almost" planar regions of the mesh.

Ratio of areas
If the ratio of the areas of two adjacent triangles is very small or very large (see Figure 6), the smaller triangle obviously has almost no geometric relevance. Consequently, the edge flipping strategy should avoid the generation of such "slivers", i.e. small triangles with large neighbors.

Crease angle
The crease angle corresponding to an edge is defined to be the angle

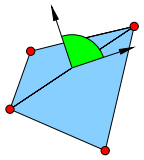

Figure 7: Crease angle criterion: The crease angle associated with an edge is the angle between the normals of the adjacent triangles.

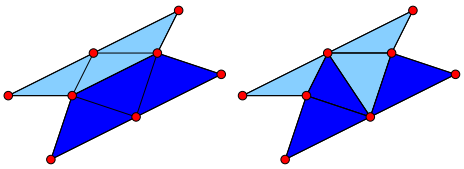

Figure 8: Normal variation criterion: The normal variation of the left configuration is zero while the normal variation of the right configuration is positive. This is perceivable as "ripples" in the surface.

between the normals of the two adjacent triangles (see Figure 7). If the points are sampled from a smooth surface the crease angles should converge to zero with increasing sampling density. Hence an edge should not be flipped if the resulting crease angle exceeds some threshold. Note that this observation is not true for feature edges where large crease angles might be intended.

Normal variation
In order to avoid "ripples" we try to minimize the normal variation around the edge, i.e. the sum of crease angles to the adjacent triangles (see Figure 8). Note however, that we don't measure normal variation across feature edges.

Putting it all together
We primarily would prefer to apply the minimum angle criterion alone, as this leads to visually pleasing meshes in the planar and almost planar case. However, this is justified only in cases where a) the local configuration is approximately flat and b) there are no exterior constraints to be fulfilled.

As a consequence we first apply the area criterion, the crease angle criterion and the normal variation criterion (in this order) with certain thresholds. If any of the thresholds is exceeded we initiate the corresponding edge flip right away. If none of the thresholds would be exceeded even after an edge flip we assume a locally flat configuration and decide about the flipping according to the minimum angle criterion.

3.4 Geometry decomposition on the server side

For creating the base mesh on the server side we use a custom designed mesh decimation algorithm. We tested various algorithms based on quadric error metrics and on Hausdorff distances. Our specific modifications to the standard algorithms were threefold:

- We trim the decimation process to prefer "round triangles", i.e. triangles which are almost equilateral. This is done because the minimal angle criterion explained above also prefers round triangles. Hence, if the base mesh had lots of bad triangles, the reconstruction might be prone to failure.

- In general, a vertex removal during decimation leads to a hole in the mesh which subsequently is retriangulated. We modified the decimation algorithm such that a vertex removal is

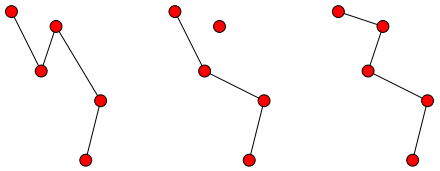

Figure 9: Decomposition. Left: Mesh before vertex removal. Middle: Mesh after vertex removal. Right: Immediate reinsertion of the vertex leads to a wrong triangulation. Hence the vertex removal is prohibited altogether.

only executed, if the closest triangle to the vertex after retriangulation is one of these new triangles. This guarantees that vertices will be inserted into the topologically right part of the mesh during reconstruction. Figure 9 illustrates this in a univariate example. The decimation stops when no more vertices can be removed due to this test.

- Univariate features, i.e. sequences of feature edges, are promoted from the original mesh to the base mesh. This is can be done analogously to the MAPS algorithm (cf. [11]): The endvertices of the paths are marked as unremovable and whenever a vertex v on a feature path is removed we ensure that the subsequent retriangulation does not break this path, i.e. if v_0 and v_1 are v's neighbors on the path we retriangulate such that v_0, v_1 becomes an edge and tag this edge as feature edge.

4 3D broadcasting

As a typical application scenario where robust transmission of geometry data matters, we consider the broadcasting of (maybe animated) 3D models. Due to the complexity of 3D datasets, we certainly need some kind of progressive transmission to quickly show an approximate model while waiting for the geometric detail information to be received.

The very nature of broadcasting communication implies that the sender cannot react to specific requests of a single receiver. The sender continuously transmits its data to a multitude of clients and the only interaction for clients is to switch channels.

One of the most important requirements in this scenario is to guarantee that the clients sees at least an approximation of the 3D models without significant delay. Standard progressive transmission does not perform very well in this context since the sender does not know when the receiver switches to a particular channel. Hence, if the receiver switches to a particular channel shortly after the base mesh has been broadcasted, it has to wait a complete cycle before the next base mesh is sent.

As we will show in Section 5, our robust transmission techniques allow to send the detail information in an almost arbitrary order. This can be exploited by sending the base mesh more frequently and interleaved with the stream of surface points. Then, when a receiver switches to a new channel, it only has to wait for the next repetition of the base mesh and can insert all subsequently transmitted points. In analogy to the teletext system (see e.g. [23]) where text data is broadcasted in parallel to a tv-signal, we can consider the base mesh as some type of index-page which is sent more often than other pages.

Suppose we are given a mesh model with n vertices that we decimate down to a coarse base mesh with m vertices such that $n = k\,m$. Grouping sequences of m successively removed vertices together, we obtain a set of k "chunks" C_i each representing a part of the geometric information.

Let s be the time that we need to transmit all k chunks. In the worst case of asynchronous progressive transmission, we would have to wait the whole cycle s if we just missed the base mesh. In average we have to wait for $s/2$. With our robust transmission technique, we can easily reduce the expected latency by sending the base mesh more often.

Let t be the maximum acceptable latency with $s = j\,t$ then we re-group the chunks C_i into sub-sequences such that C_i and $C_{\hat{i}}$ belong to the same sub-sequence if $i = \hat{i} \bmod j$. Obviously we obtain j different sub-sequences where the individual chunks within each sub-sequence preserve their original ordering.

Now we alternate the transmission of the base mesh and the transmission of those sub-sequences in a way that we start with the base mesh then send the first sub-sequence. Then the base mesh again followed by the second sub-sequence and so on.

It is obvious that this reduces the expected latency from s to $t = s/j$. Since the robust transmission can deal with partial loss of data, there are no difficulties in inserting any sub-sequence of vertices since this simply corresponds to a loss of the other $j - 1$ sequences. As the transmission continues, the receiver can insert more and more sub-sequences while ignoring the redundantly transmitted base meshes.

The overall redundancy of this progressive transmission procedure can be bounded by $(j-1)/k$ since this is the number of additional base meshes that have to be sent in one complete cycle. Figure 10 shows an example of the asynchronous progressive transmission.

5 Results and Conclusions

In Fig. 11 we show some results obtained with our robust transmission technique for irregular meshes. The vertices of the horse model have been transmitted without connectivity information. The client inserted these vertices into a base mesh that initially had 1000 triangles. Reconstruction results are shown for no data loss and 50% loss of data.

In all our experiments, the current implementation proved sufficiently robust. Starting with a moderately decimated mesh, it turned out that the vertices can be re–inserted in practically any order and the edge–flipping after every insertion will always generate meshes in local normal form. Moreover, the hierarchical nature of progressive refinement typically seems to guide the greedy edge–flipping to the global optimum. Besides robustness against loss of data, this allows on the one hand to generate vertex transmission orders which are suited for progressive broadcasting, as described in section 4. On the other hand we can exploit this flexibility to refine the mesh only *locally* in regions of interest (see Figure 11), i.e. we might intentionally discard vertices which are not relevant from the current viewpoint.

Using a space–partitioning technique on the client side to quickly identify the closest triangle for a given point in space, we can insert up to several thousand vertices per second on a standard PC. Non–robust vertex insertion based on global indexing is obviously more efficient but the difference is mainly due to the point location step.

In order to guarantee the preservation of feature edges in the model, edges in the base mesh as well as the subsequently transmitted vertices are tagged accordingly. Fig. 12 shows an example of this technique.

6 Acknowledgement

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

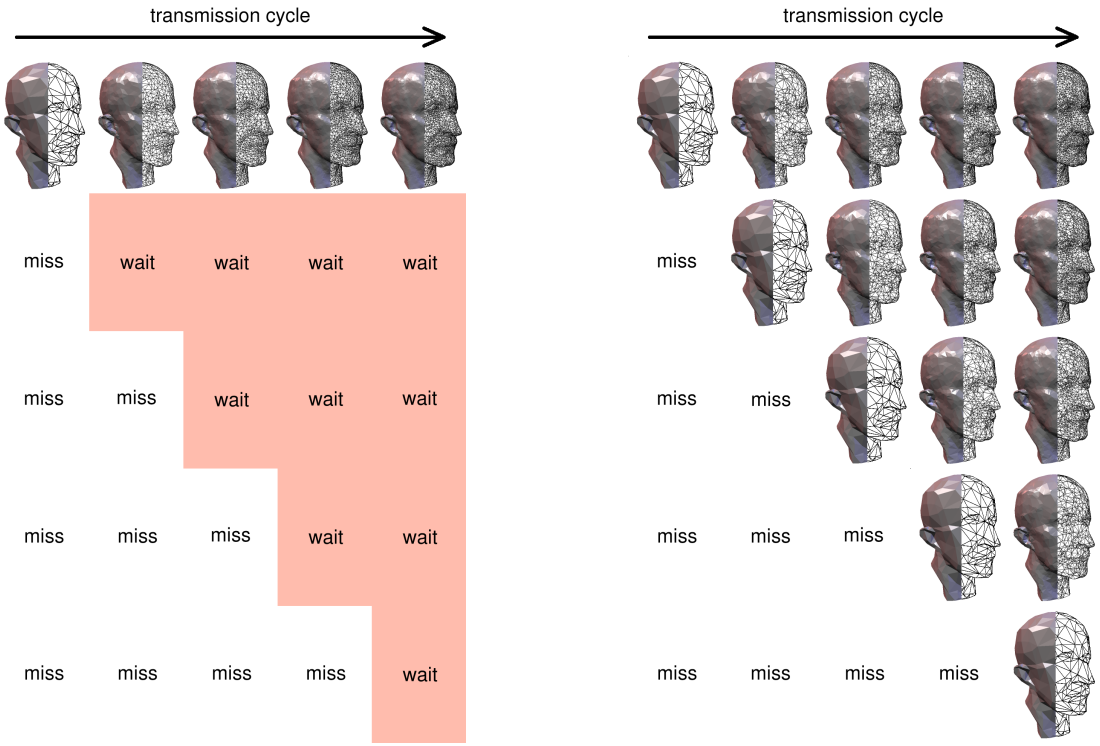

Figure 10: Broadcasting 3D geometry: On the left the situation for standard progressive transmission is shown. If the receiver listens from the beginning of the current cycle, it can display a sequence of meshes with increasing quality. However, if it misses even the smallest prefix of the stream, it has to wait until the next cycle. On the right we depict the teletext type transmission where we repeat the base mesh several times and interleave its transmission with the rest of the geometry data. Hence the client can start the reconstruction several times during one cycle. This significantly reduces the latency while not introducing much redundancy.

References

[1] Alliez, Desbrun, *Progressive encoding for lossless transmission of 3D meshes*, Computer Graphics (SIGGRAPH 01 Proceedings), 2001

[2] N. Amenta, M. Bern, M. Kamvysselis, *A New Voronoi–Based Surface Reconstruction Algorithm*, Computer Graphics (SIGGRAPH 98 Proceedings), 1998, 415 – 422

[3] Bernardini, Mittleman, Rushmeier, Silva, Taubin, *The Ball–Pivoting Algorithm for Surface Reconstruction*, IEEE Transactions on Visualization and Computer Graphics, 5(4):349 – 359, 1999

[4] Edelsbrunner, Mücke, *Three–Dimensional Alpha Shapes*, ACM Trans. on Graphics, 13(1):43 – 72, 1994

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, *Surface Reconstruction from Unorganized Points*, Computer Graphics (SIGGRAPH 92 Proceedings), 1992, 71 – 78

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

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

[8] Khodakovsky, Schröder, Sweldens, *Progressive geometry compression*, Computer Graphics (SIGGRAPH 00 Proceedings), 2000

[9] Kobbelt, Vorsatz, Labsik, Seidel, *A Shrink Wrapping Approach to Remeshing Polygonal Surfaces*, Computer Graphics Forum, 18(3):199 – 130, 1999

[10] U. Labsik, L. Kobbelt, R. Schneider, H-P. Seidel, *Progressive Transmission of Subdivision Surfaces* Computational Geometry Journal: Theory and Applications 15, 2000, pp. 25 – 40

[11] Lee, Sweldens, Schröder, Cowsar, Dobkin, *MAPS: Multiresolution Adaptive Parameterization of Surfaces*, Computer Graphics (SIGGRAPH 98 Proceedings), 1998, 95 – 104

[12] Rossignac, *Edgebreaker: Connectivity compression for triangle meshes*, IEEE Transactions on Visualization and Computer Graphics, 1999

[13] J. R. Shewchuk, *Delaunay Refinement Mesh Generation*, PhD–Thesis, Carnegie Mellon University, Pittsburg, 1997

[14] Taubin, Rossignac, *Geometric Compression Through Topological Surgery*, Computer Graphics (SIGGRAPH 98 Proceedings), 1998

[15] G. Taubin, *Is this a quadrisect mesh?*, Proceedings of ACM Solid Modeling Symposium 2001

[16] Touma, Gotsman, *Triangle mesh compression*, Graphics Interface '98 Proceedings, 1998

[17] Turk, *Re–tiling polygonal surfaces*, Computer Graphics (SIGGRAPH 92 Proceedings), 1992, 55 – 64

[18] Zorin et al., *Subdivision for Modeling and Animation*, SIGGRAPH Course Notes, 2000

[19] N. Dyn, K. Hormann, S.-J. Kim, D. Levin, *Optimizing 3D Triangulations Using Discrete Curvature Analysis*, in Mathematical Methods for Curves and Surfaces, Oslo 2000, 135–146, Vanderbilt University Press, 2001

[20] D. Eberly, *3D Game Engine Design*, Morgan Kaufmann, 2001

[21] Jonathan R. Shewchuk, *Delaunay refinement mesh generation*, PhD-Thesis, Carnegie Mellon University, Pittsburg, 1997

[22] J. Huang, R. Yagel, V. Filippov, Y. Kurzion, *An Accurate Method for Voxelizing Polygon Meshes*, IEEE 1998 Symposium on Volume Visualization, 1998, 119 – 126

[23] *Broadcast teletext specification*, published jointly by British Broadcasting Corporation, Independent Broadcasting Authority, British Radio Equipment Manufacturers Association, 1976

[24] R. van Damme, L. Alboul, *Tight Triangulations*, Mathematical Methods for Curves and Surfaces III, Vanderbilt Press, 1995, 517 – 526

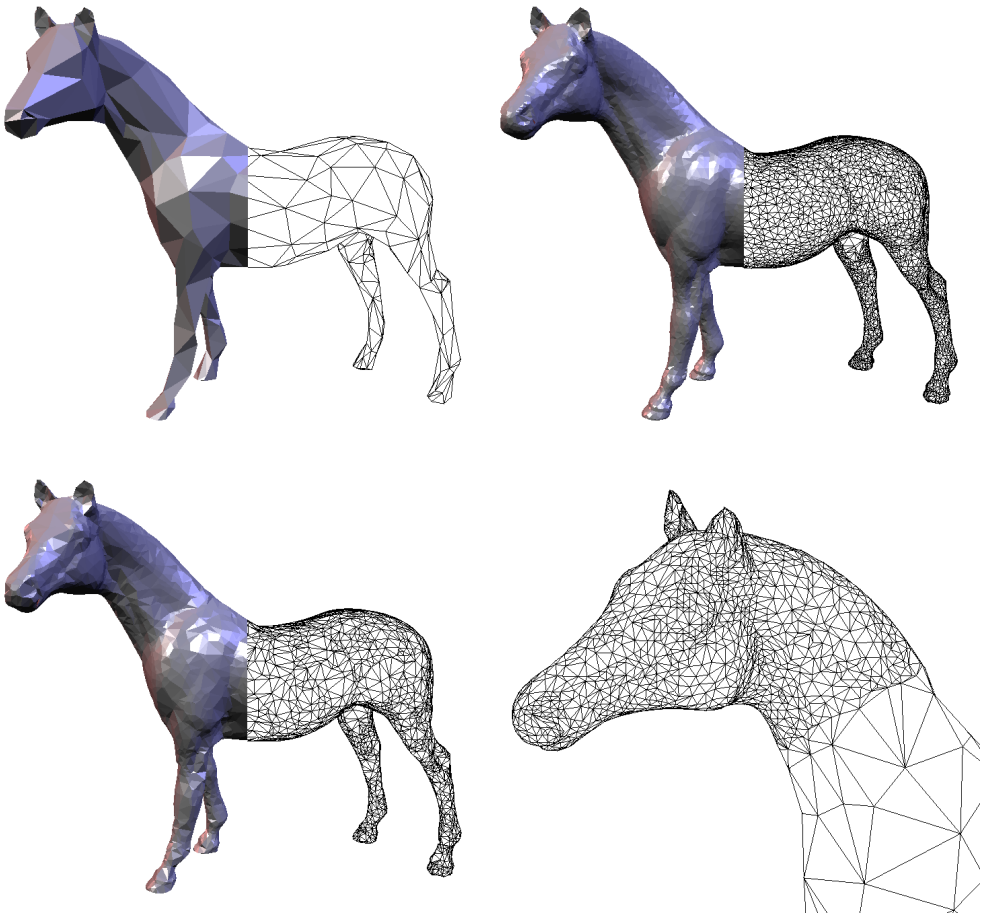

Figure 11: Robust geometry reconstruction. Top left: The base mesh that is used by the client to insert the incoming vertices (1000 triangles). Top right: Refined mesh after the insertion of 10000 points. Lower left: Refined mesh after the insertion of 5000 points (= 50% data loss). Lower right: Local refinement of the horse head after insertion of 2000 points.

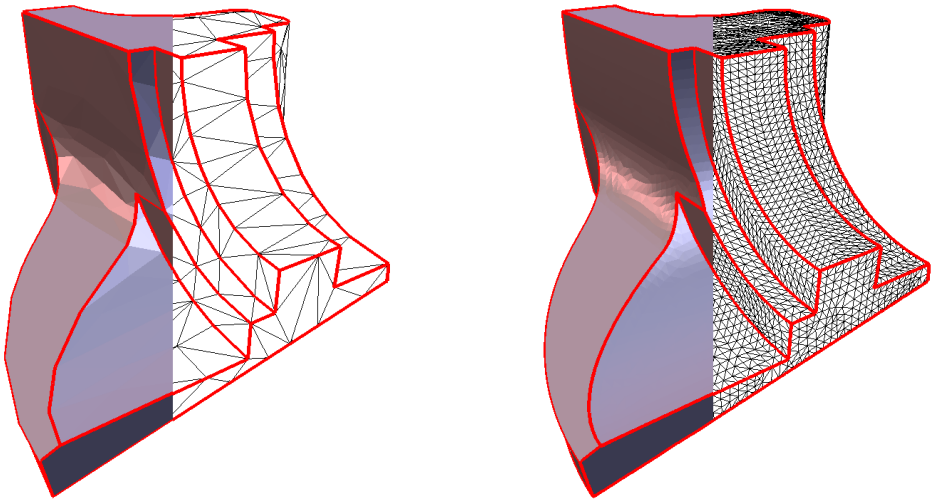

Figure 12: Feature preservation during robust transmission is achieved by vertex tagging and disallowing to flip feature edges. The base mesh is shown on the left while the reconstruction is shown on the right.