Healing of 3D Geometry Databases

Dinesh Shikhare
National Centre for Software Technology, Juhu, Mumbai.

May 1999.

Abstract

This report discusses the problem of mismatch between the geometry modelers and the end-applications that need to use the geometry data. The mismatch is seen at various levels: data formats, geometric artifacts, quantization errors, combinatorial complexity and lack of desired structure. To make the geometry usable, it has to undergo extensive preprocessing which is also referred to as "healing" and "CAD data repair." We also describe the various approaches taken by researchers to get around these and similar problems for their specific applications.

1 Introduction

In recent years there has been a large growth in the development of geometric modeling packages, geometry visualization and walk-through applications. Enormous digital models of 3D geometry have been constructed using these modelers. Very often these digital models are required for multiple applications. For example, a digital model of an aircraft is required for digital visual mock up, ergonomic studies in virtual environments and computational fluid dynamic analysis. These different applications have their own specific needs and expectations from the digital models. Sometimes the modeling packages do not have the facilities to build such multi-purpose digital models. But more often many future applications are not known and hence their requirements can not be catered to while creating the digital models, thereby making it difficult to directly utilize the models. There are many such examples in various fields ranging from CAD to virtual walk-through to 3D computer games.

In such a scenario, rebuilding the entire model is a very expensive proposition. Also, tinkering with the 3D geometry using ad-hoc geometry manipulation tools is also quite time consuming and labour intensive. Having recognized this, many researchers have created geometry and topology "healing" software tools to solve their specific problems.

In this report we describe the major classes of mismatch between the two sides of geometry data manipulation – 3D geometry data acquisition, modeling on one hand and the end applications using the digital models on the other. In section 2, we briefly describe the different classes of healing requirements. In sections 3 through 6, we outline specific techniques used for healing problems in geometry.

2 Mismatch between Modelers and End-Applications

The mismatch between the digital models created by modeling packages and the requirements of the end-applications can be broadly classified as follows:

1. **Data formats:** The modeling packages often have their own proprietary data formats for storing the 3D models created. The format primarily used by a particular package usually captures all features of the construction tools and procedures with high fidelity. The end-applications, however, often require these models in a different file format as well as in terms of different data entities. For example, a versatile NURBS modeling package may be used to design surface geometry of a 3D object. For using the 3D object in a visualization package, one would have to tessellate these models into triangulated surfaces. This leads to approximations inherent to the discretization process. Various "neutral" file formats exist for CAD data exchange – IGES, STEP, etc. Modelers for architectural geometry like AutoCAD, 3D animation packages like 3DStudio Max, Alias Wavefront, Softimage have their own proprietary file formats. Translation of models across data formats is a significant mismatch problem.

2. **Lack of structure in the data:** In many cases the geometry data is organized as a "soup of polygons" [1]. Handling such data for geometry manipulation is extremely difficult because of lack of structure and grouping definition. A healing package must give a structure to such a geometry by looking for a coherent structure in the unstructured 3D data.

3. **Gaps, holes and T-junctions:** 3D models described using polygon meshes can have topological insanities like gaps between two meshes attempting to represent an object with closed surfaces. Such gaps/holes in the data can cause errors in simulation programs such global illumination computation by radiosity methods [3]. T-junctions create topological holes in models defined using multiple surface components. Topological surgery is required to correct such problems in the geometry.

4. **"Invisible" geometry:** There are many different situations when parts of the modeled geometry is not visible due to the following reasons:

   - Unused vertices
   - Dangling edges
   - Hidden polygons

   This invisible geometry does not contribute in useful manner to the end application, but poses combinatorial overhead to the computation.

5. **Excess detail:** Either due to the data acquisition process or content creator's preference, excessive details are captured for models which can be described using much less data. Excess level of detail affects the performance of data visualizers, walk-through programs etc. Many approaches exist for reducing the details of the meshes by way geometry simplification [10, 6] and topology simplification [5].

6. **Overlapping geometry:** Probably the most difficult geometry insanity to handle is that of overlapping geometry. Completely duplicated geometry can be trivially eliminated. But partial geometric overlap between two surfaces requires trimming

of surfaces are some arbitrary point of contact between the surfaces. handling this automatically is difficult, but interactive manipulation by expert user is feasible.

7. **Volume of data:** With increasing usage of 3D content over the Internet and increase in the complexity of the digital models, special techniques for compression of geometry data have become important. Various topological and geometric encoding methods [4, 6, 9] have been explored by researchers for achieving high compression ratios.

3 Data Exchange

Given the various different file formats and the use of different data structures and data formats used for represent almost similar kind of 3D digital models, it should be possible to obtain a neutral file format that will be the union of all features provided across the different archival formats. Such a neutral format must have the following modeling features:

1. Curves, Surfaces, Volumetric data should be representable in terms of simplices.
2. Smooth curve and surface representations should be possible for modeling NURBS entities.
3. Visual attributes such as colour, textures, vertex normals should be modeled in the package. Textures are usually references to image files.
4. Facility to define associations across geometric entities and across geometry and visual attributes should be provided.
5. User defined attributes must be available in a generic fashion.
6. Facility for grouping of entities is necessary for giving a structure to the 3D content.
7. It should be possible to supply information about hierarchical structure among entities in the database.
8. The file format must be extensible by adding user-defined entities. This way new features can be added to the structures. For example, animation paths, transformations and any other temporal changes may be added later on.

Geometry healing package must have a run time data structure which supports all these features and a mechanism to absorb any new feature that may need to be added. This requires a plug-in support to be built at the core level of the data-structure. The code entities for handling geometry must have a generic associative array that can hold any additional associated attributes in a “name-value” fashion.

Having obtained a neutral file format, it is possible to develop a translator from any file format X to another file format Y. Conversion of data across file formats often requires quantization of data and approximations. For example, if a NURBS model has to be exported to VRML world, the smooth surfaces must be discretized to obtain an "indexed face" representation. The user of the translator must be able specify the maximum permissible error in the conversion and then determine the combinatorial complexity of the mesh. On the other hand, the user should also be able to specify the combinatorial complexity of the required output and obtain an output data with the least possible error. A data exchange utility must be able to parameterize the conversion process with many such quantitative factors. Each pair of source and target formats can have different set of parameters.

4 Geometry and Topology Editing

In this section we list the geometric and topological editing features a typical geometry database healing package must have. Many of the topological modification facilities also modify the geometric definition of the objects. However, these modification often needed to remove otherwise insane geometric features. The following discussion has concentrated on the 3D models having surfaces represented as triangle meshes. A representative package that has some features discussed below is CADFix which is described on its web-site [7].

4.1 Removal of slivers, stray geometry

Slivers are triangles having a very high aspect ratio (AR). Aspect ratio is unity for an equilateral triangle.

$$AR = R/2r$$

where, R = radius of the circum-circle of the triangle and r = radius of the in-circle of the triangle. Such triangles are almost like line-segments. Such triangle almost never play any useful role in visualization, simulation of the meshes. Such triangles must be detected and removed.

Another common anomaly in geometry database is stray geometry. This geometry persists in the database either due to buggy algorithms in the modeling package or oversight of the content creator. Such geometry also does not play any useful role in any usage of the meshes. However, it is not always easy to detect such stray geometry. This stray geometry occurs in the form of:

- unused vertices (which are trivial to detect)
- duplicated triangles (easy to detect)
- disconnected pieces of "small" meshes (it is hard to determine how small a mesh should be eliminated by labeling it as stray)
- stray geometry pieces connected to the useful part of the geometry (this is hardest to detect in general)

The challenge is always to build a usable interface to a healing package to let the user specify what is a stray element.

4.2 Surface orientation

A triangulated surface is orientable if it has the following properties: (1) adjacency of triangles must be defined in terms of two triangles sharing two vertices which forms a shared edge between them, (2) each edge is shared by at most two triangles in the mesh, (3) the edges shared by two triangles are called as interior edges and an edge used by only a single triangle is called a boundary edge. The boundary edges can be connected to obtain boundaries of a mesh. A mesh that has no boundary is a closed mesh.

For a mesh to be oriented, the sequence of vertices in the definition of triangles is such that each shared edge in opposite direction in the respective triangles. That is, a shared edge defined by vertices v_1 and v_2 should occur in the triangles that share them as: $T_1 = (v_1, v_2, v_x)$ and $T_2 = (v_2, v_1, v_y)$. A simple observation of the above definition of orientability is that each orientable mesh can have two orientations.

Often the geometry database has meshes that are not orientable due to the following reasons:

1. Duplicated vertices: for the sake of texture mapping, geometrically congruent vertex in 3D is repeated because it has a different texture coordinate with respect to the different polygons sharing it.

2. Non-manifold geometry: edges shared by more that two triangles leads to a non-manifold geometry.

Incorrect orientation of geometry leads to erroneous visualization and errors in simulations. Orienting such meshes involves removal of duplicate geometry, keeping intact the topological structure, and recursive correction of orientation.

4.3 Merging along surface boundaries

Some surfaces are designed in terms of multiple patches. If these patches are tessellated independently, gaps are left between the boundaries of the surface. These gaps adversely affect visualization and other simulation processes. Hence there is a need to merge these surface patches along the boundaries. There are various difficulties in merging surfaces patches, which a geometry healing package must overcome:

1. The “adjacent” surfaces to be merged may be tessellated at different resolutions, thereby making it difficult to find candidate pairs of points to stitch the surfaces along. If point pairs can not be trivially found, either extra points must be generated artificially on the surface boundaries along with modified local tessellation, or positions of points must be modified. Both these options are non-trivial to implement.

2. The pair points is usually determined by proximity between points with some tolerance value. This make the process very sensitive to the selection of the tolerance value.

3. This operation must have a good user interface to be able to guide the process in case of ambiguous situations.

Work reported in [1] by Barequet, *et al* iillustrates many special cases where points need to be shifted in order to effectively “stitch” the surfaces.

4.4 Closing gaps and Local triangulation

Sometimes gaps between surface patches is too large to eliminate using boundary merging tool. In such cases, a small surface must be constructed for patching the hole. This is usually done by generating triangulation of the boundary of the gap/hole. The procedure for the local triangulation is varies from system to system. In a package called VolGrid [8] developed at NCST, the patching of holes is done using the following procedure:

1. Identify the closed loop of gap in terms of an ordered list of vertices.

2. Determine the best plane representing the loop.

3. Project all the points of the loop on this plane.

4. Align the plane and all the points on the loop along the XY plane.

5. Carry out extended Delaunay triangulation of the 2D patch.

6. Apply the inverse transformation to place the plane back to the original position.

7. Using the surface merging procedure described above, this new triangulated patch is merged with the existing surfaces to close the gap/hole.

The approach described in [2] is similar, but the difference is that they carry out local triangulation in place in 3D domain. However, the results are comparable.

5 Geometry Compression

A major type of mismatch between content creation process and content consumption process is the bandwidth of communication of data between the two. Increasingly, 3D data is being shared across networks and disconcerting latencies in communication make the 3D applications less usable that they could be. The most obvious way out is to compress the geometry data for faster transfer. The general purpose compression techniques like LZW can not take advantage of the information and relationship among entities in the geometry data.

Hence some researchers have attempted to develop special compression techniques for triangle meshes [4, 9]. The approach taken by both these relies on orientability of the triangle meshes. The common strategy is to convert the triangle meshes into maximal triangle strips and then encode the triangle strips optimally. Besides triangle strip coding, some lossy encoding of geometry is also attempted to reduce the space required to encode coordinates of points and vertex normals.

The work in progress at the author's organization is based on a dictionary based technique to compress large 3D scenes consisting of triangle meshes. This technique derives its advantages from the fact that many mesh are often duplicated in such scenes at different positions and in different orientations. Typically, these transformations are rigid body transformations between instances of basically the same objects. Hence geometry and topology matching algorithms have been developed to determine instances of a replicated mesh in the scene along with the rigid body transformation used.

Such a strategy gives a very efficient coding scheme for large 3D scenes. Each mesh that appears multiple times in the 3D scene is described in detail only once. Each of its instances is encoded along with the index of the first instance and the rigid body transformation.

6 Associated Attributes

While manipulating geometry/topology in a geometry healing package, one must pay attention to the associated attributes such as:

1. Colour definition at vertices
2. Vertex normals
3. Texture coordinates defined at vertices with respect to each of the polygons using it

These attributes are difficult to handle particularly in places where surfaces are merged along boundaries and gaps are closed by local triangulation. In case of merging of surfaces along boundaries, the algorithm must avoid visual discontinuity in the attributes.

When local triangulation is carried out to fill gaps there is no information about the attributes at the vertices of the newly created patch. Here, either user must specify the values interactively or the algorithm must automatically determine the values.

As there is very little definitive work reported in this area, it forms a very challenging subject for new research and development. The applications of this are seen in many areas such as modeling for 3D architectural databases, animation character development, pre-processors for surface and volume grid generation for CFD analysis, and so on.

7 Conclusion

This report is a result of study carried out in the area of healing of geometry databases. We discussed the various classes problems faced in geometry datasets and representative algorithmic techniques used to solve the problems. The scope of the work to be done to develop a usable general purpose software package is very large. This is the reason why various researchers are developing tools that cater to their own problems in a narrow domain of problems that they face.

References

[1] Gill Barequet, Christian Duncan, and Subodh Kumar. RSVP: A geometric toolkit for controlled repair of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), April-June 1998.

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

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

[4] Michael Deering. Geometry compression. In *SIGGRAPH '95 Proc.*, pages 13–22, 1995.

[5] Jihad El-Sana and Amitabh Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), April-June 1998.

[6] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996.

[7] FEGS Ltd. CADFix: A product for CAD data exchange, repair and upgrade. http://www.fegs.co.uk, 1997.

[8] Ashwini Patgawkar, Dinesh Shikhare, S. Gopalsamy, Satyashree Mahapatra, and S. P. Mudur. Tetrahedral discretization of complex volumetric spaces: Implementation, efficiency, robustness and interactive control. In *Proceedings of International Conference on Visual Computing*, February 1999.

[9] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. Report GIT-GVU-98-35, GVU Center, Georgia Tech., Atlanta, USA, 1998.

[10] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proc.)*, 26(2):65–70, July 1992.