3D COMPRESSION

TOPICS TO BE COVERED

3D Compression

- **Introduction and motivation**
- **3D Compression in relation to other media compression**
- **Types of 3D Compression methods**
- **Topological surgery**
- **Progressive Meshes**

WHY IS 3D COMPRESSION NECESSARY?

Detailed Geometric Models have become commonplace in most computer graphics related industries – CAD & Manufacturing, Entertainment, Visualization, Medical Imaging.

3D accuracy and realism is achieved is by fine geometric detail and this now possible given advances in data capturing technologies and rendering performances.

Here 3D models are often represented as complex triangle meshes, which present challenges in

- **Rendering performance – bounded by number of polygons to be rendered**
- **Transmission bandwidth**
- **Storage capacities**

All these motivate a need for compressing geometric data.

3D MESHES

Most graphic 3D models are stored as triangle meshes – there are other 3D representations like Voxels, Generalized Cylinders, Minkowski Sums, CSG representations etc. But triangles are widely used for efficiency in rendering.

A triangular mesh is defined by

- Location of its vertices (positions),
- Association between each triangle and its sustaining vertices (connectivity),
- Color, normal, and texture information

The last point defines properties that do not affect the 3-D geometry but influence the way it is shaded or its appearance

3D MESH COMPRESSION EXAMPLE

Each triangle has

- **(x, y, z) specifying a 3D location for each vertex**
- **(n_x, n_y, n_z) specifying a face normal**
- **(u, v) texture coordinates for each vertex**
- **(r, g, b) color per vertex (?)**

If you represent each coordinate, normal and texture coordinate with 4 bytes → 8x4 = 32 bytes per vertex

To download a simple model of 20K vertices over a 56K modem would require around 4 mins.

Need hours to download a 2M triangles model !!

3D COMPRESSION IN RELATION TO OTHER MEDIA COMPRESSION

Other media (images, video, audio) normally have lossless and lossy modes of compression.

What does it mean to have lossless modes in 3D compression – normally achieved by variable length coding of errors between actual and predicted values eg motion vectors in video, DPCM in audio - in 3D this is not straight forward !

What does it mean to have lossy modes in 3D compression –

- **Geometry thrown away cannot be completely recovered**
- **Should be used so that it least affects the 3D shape of the mesh**

TYPES OF 3D COMPRESSION METHODS

Recent methods in 3-D compression may be divided into three categories –

- **Compression of positions and properties**
- **Encoding of the connectivity information**
- **Polyhedral simplification**

Encoding connectivity information aims to primarily to minimize the information used in representing the edge connectivity

Polyhedral simplification aims at modifying the mesh geometry by removing/moving vertices so that the mesh is represented by lesser number of points.

CONNECTIVITY ENCODING TECHNIQUES

Connectivity encoding techniques attempt to reduce the redundancy inherent in the geometry and connectivity of the 3D mesh.

This redundancy lies in

- **Local neighborhood coordinates in 3D don't differ by much.**
- **Same edges used to represent different faces**

To exploit these, we need a way to traverse the mesh in 3d! Aim to get vertex runs – easy to compress because semantically represented as a run of vertices instead of 3 vertices per face.

TOPOLOGICAL SURGERY

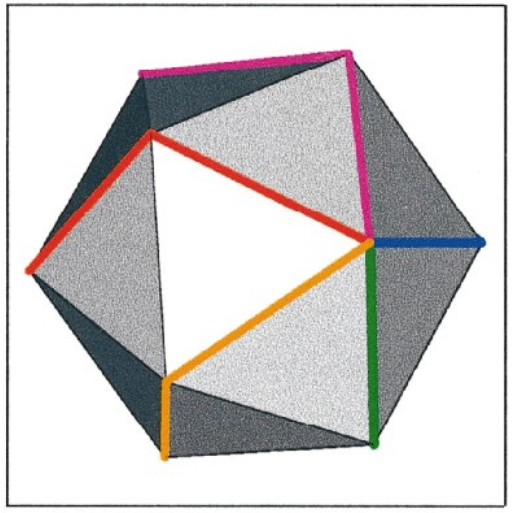
This is a connectivity encoding technique developed by Taubin and Rossignac and has been adopted as a standard in the MPEG4 format

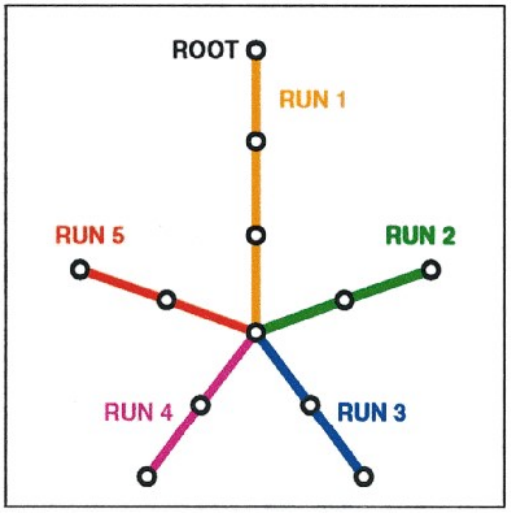
The algorithm –

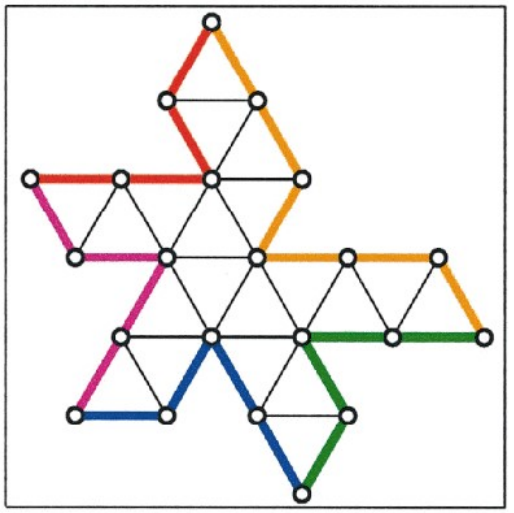
- **Construct the vertex spanning tree**
- **Encode the vertex tree**
- **Compress the vertex positions**
- **Encode the triangle tree, and**
- **Compute and compress the marching pattern**

EXAMPLE OF VERTEX SPANNING TREE

The 3D mesh represents a graph. Start with a vertex (node) and traverse the mesh from vertex to vertex using some heuristic like - length of edges, euclidean distance from a root node.



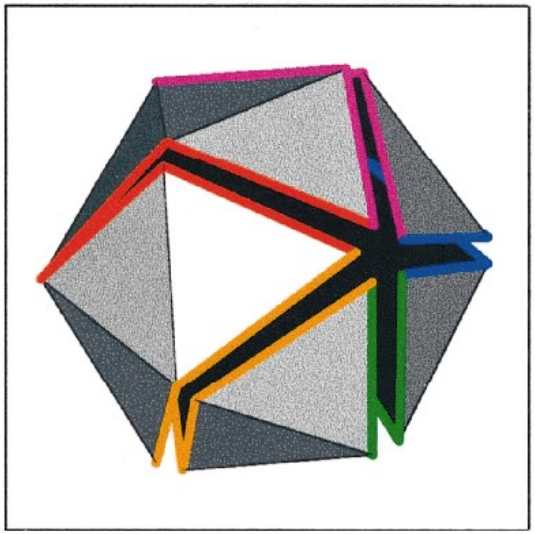


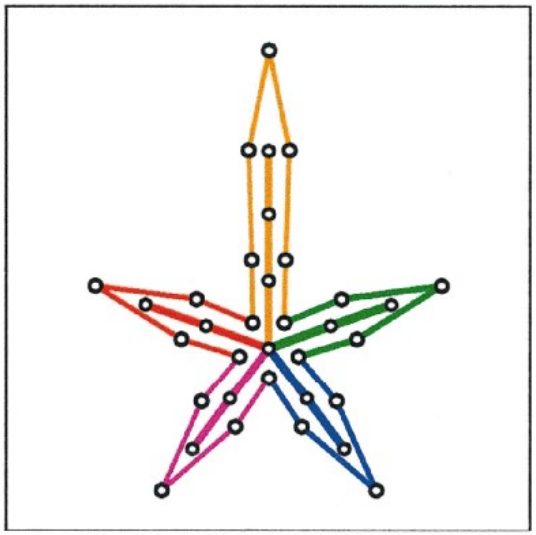


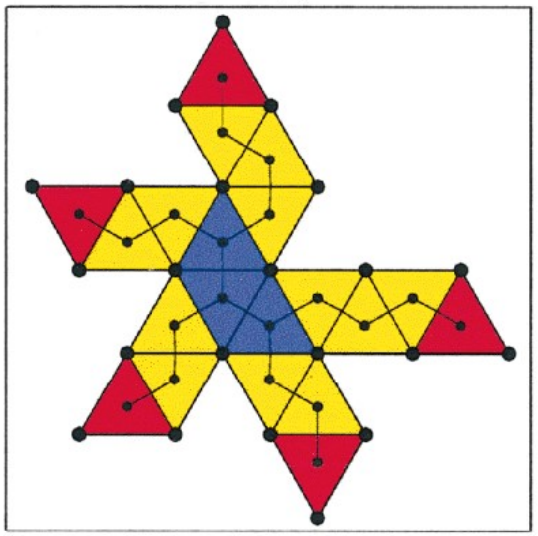
USING SPANNING TREE TO CUT OPEN MESH

Once you span the surface of the mesh using a tree, you can cut open (surgery!) the mesh along the spanning tree edges. The dual graph now gives you vertex runs!

Each run

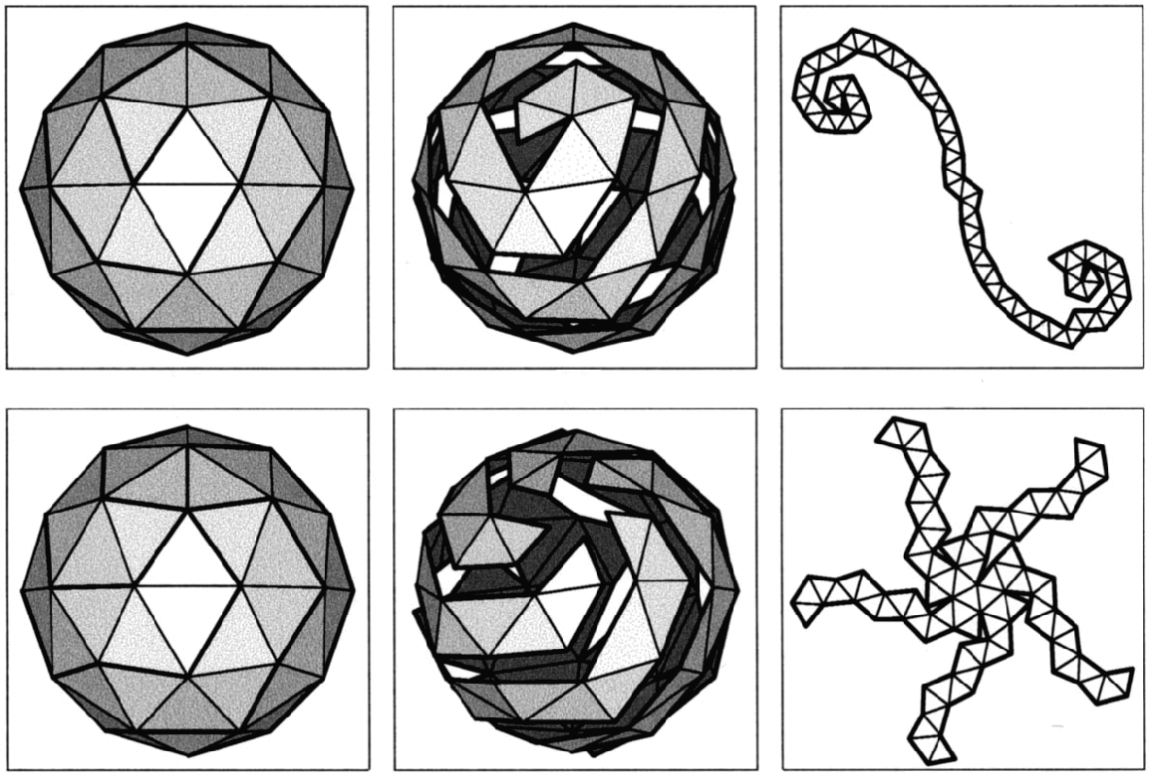




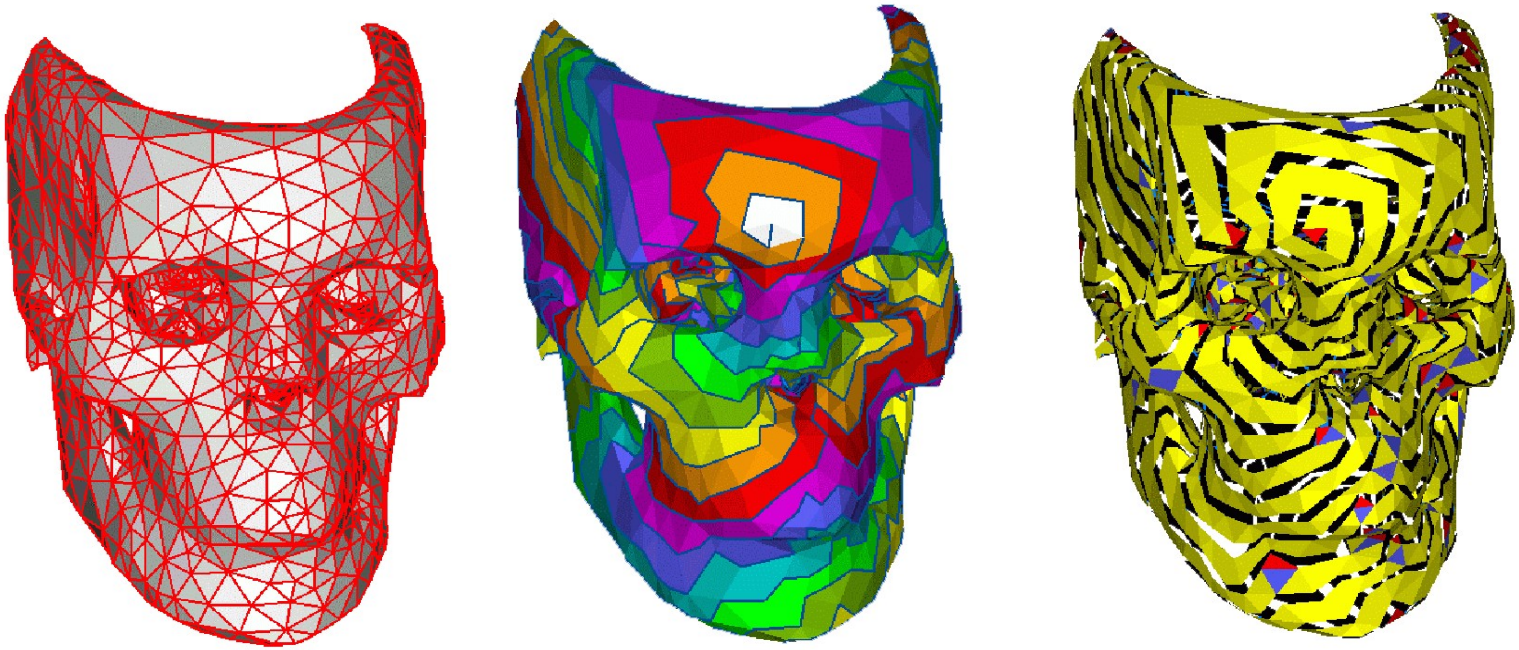


DIFFERENT CUTTING STRATEGIES

Different cutting strategies produce triangle trees with different numbers of runs. You want to minimize the number of runs and maximize the length of vertex runs



TOPOLOGICAL SURGERY EXAMPLE



POLYHEDRAL SIMPLIFICATION

Polyhedral simplification techniques [1] reduce the number of vertices in the mesh by

- **Altering the model's connectivity and**
- **Adjusting the position of the remaining vertices to minimize the error produced by the simplification.**

These techniques concentrate on the generation of multiple levels of detail (LOD) for accelerated graphics

A form Polyhedral Simplification has been described in Progressive Meshes (Hoppe – 1996)

PROGRESSIVE MESHES

Basic idea - Simplify an arbitrary mesh through a sequence of transformations. Record the simplified mesh and sequence of inverse transformations

In Progressive Meshes, an arbitrary mesh *M* is stored as a much coarser mesh M_0 together with a sequence of *n* detail records that indicate how to incrementally refine M_0 exactly back into the original mesh $M = M_n$.

This representation of *M* thus defines a continuous sequence of meshes $M_0, M_1, \ldots, M_n$ of increasing accuracy

COMPUTATION OF PROGRESSIVE MESHES

An *edge collapse* operation is sufficient for effectively simplifying meshes. An edge collapse transformation *ecol* (V_t, V_s) unifies 2 adjacent vertices V_t and V_s into a single vertex V_s. The vertex V_t and the two adjacent faces (V_s, V_t, V_l) and (V_t, V_s, V_r) vanish in the process.

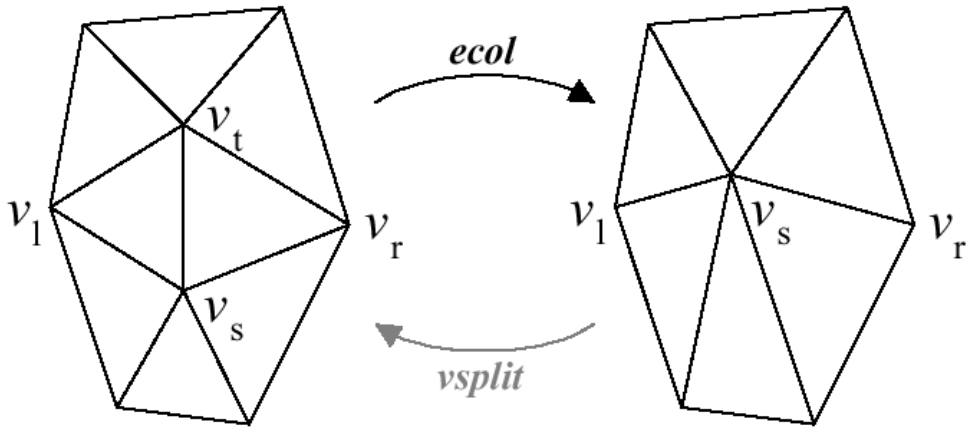


Illustration of the edge collapse transformation.

COMPUTATION OF PROGRESSIVE MESHES

Sequence of edge collapses on the above example would produce the following table

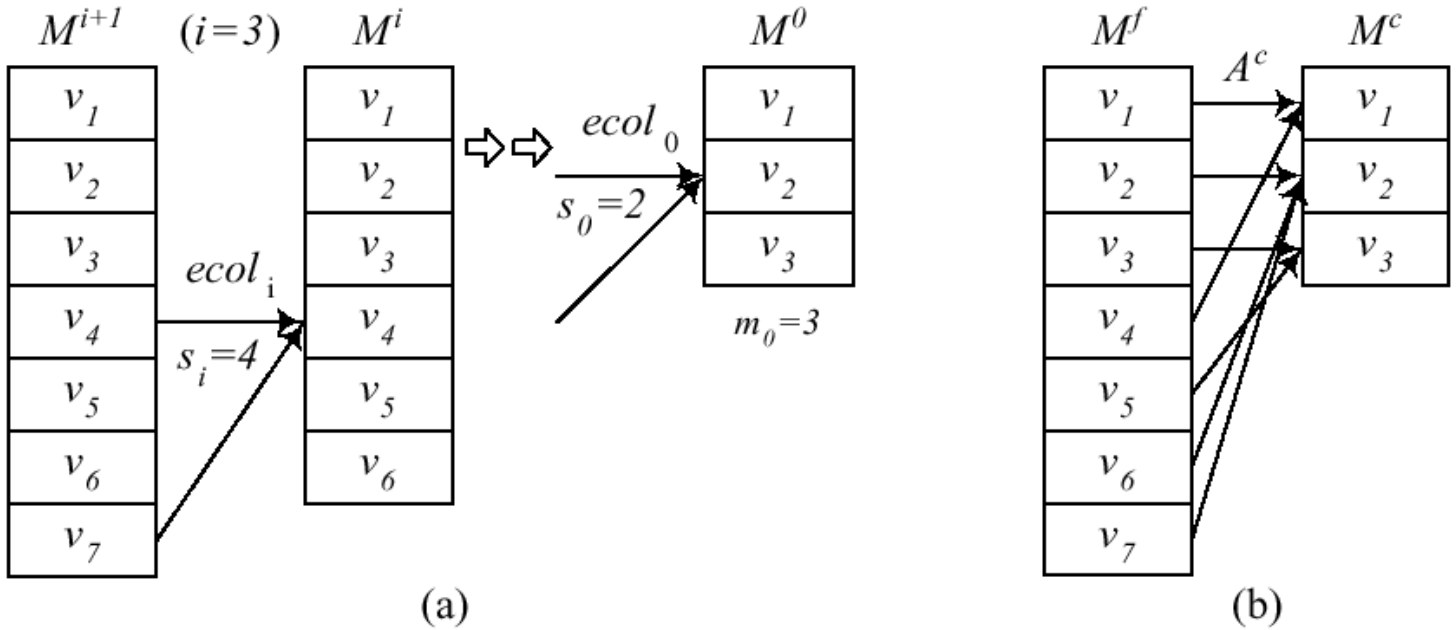
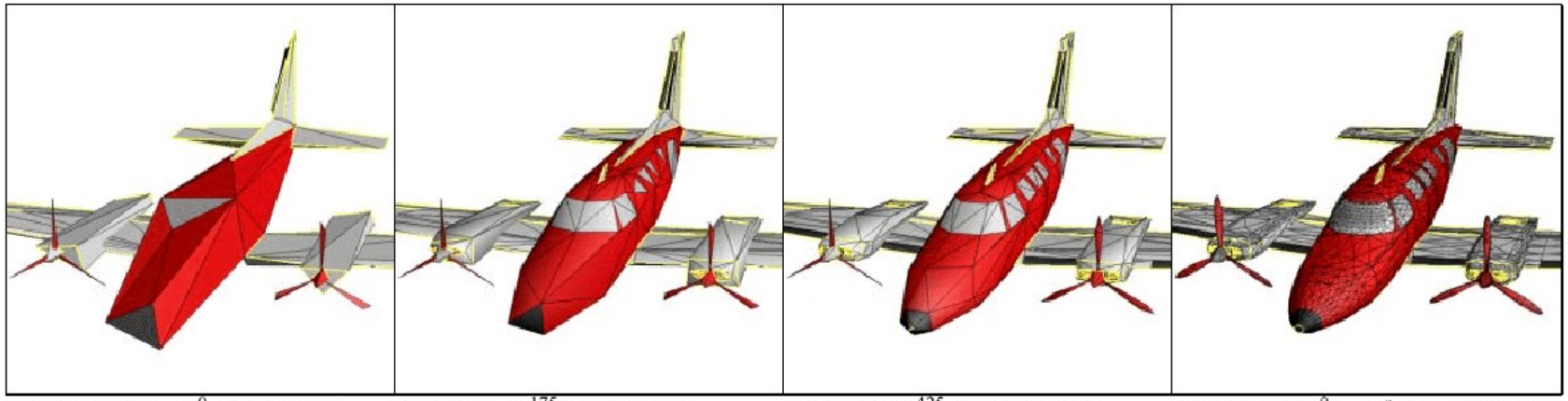


Figure 2: (a) Sequence of edge collapses; (b) Resulting vertex

PROGRESSIVE MESH – EXAMPLE



(a) Base mesh M^0 (150 faces) (b) Mesh M^{175} (500 faces) (c) Mesh M^{425} (1,000 faces) (d) Original $\hat{M}=M^n$ (13,546 faces)

Aside from mesh simplification and compression, Progressive Meshes are useful for

- **Multi resolution modeling and Level of Detail (LOD)**
- **Selective Refinement**