Meshalyzer Documentation

Justin Kinney

15June 2007

1 Summary

By reverse-engineering the output mesh of the contour tiler and the source code of Filtermesh, we discovered the set of four mesh attributes that are necessary and sufficient to insure the highest degree of mesh integrity and compatibility with mesh analysis/manipulation software. (1) Is the mesh closed? Are there any holes in the mesh lattice or is it water tight? (2) Is the mesh a 2D manifold in R3? Does the mesh neighborhood around every vertex resemble a disk as a 2D manifold must? (3) If the mesh is a manifold, then do the mesh faces have consistent orientation? Do all the faces have normals oriented the same (inward or outward) or are they of mixed orientation? (4) If the mesh has consistent orientation, then do all the faces have normals pointing outward? In most cases, the correction of meshes that violate these attributes is a manual process involving the removal of nonmanifold mesh elements, the reorientation of flipped meshes, and the addition of faces in mesh holes.

The purpose of the program is to quickly analyze and report on both general characteristics and four topological attributes of meshes. Running meshalyzer on a mesh file will provide enough information in a single report to determine if the mesh will fail as input to any of the manipulation processes described in this project.

2 Assumptions

- The surface mesh has triangular faces.
- The mesh file format is used. Vertices are specified by a vertex index, i, and three coordinate positions, xyz: $[Vertex \quad i \quad x \quad y \quad z]$ e.g. [Vertex 45 3.634 1.004 2.925]. Faces are specified by a face index, j, and three vertex indices, v1v2v3: $[Face \quad j \quad v1 \quad v2 \quad v3]$ e.g. [Face 1204 74 10 29]. All indices are positive integers larger than zero.
- The face normal vector, $\vec{n} = (\vec{v_2} - \vec{v_1}) \times (\vec{v_3} - \vec{v_1})$where $\vec{v_i}$ is the vector location of vertex index vi, points outward, away from the mesh object as illustrated by vector $\vec{m}$in Figure 1. This same convention is used by DReAMM, VTK, and Filtermesh. Notably, IRIT uses the opposite convetion.

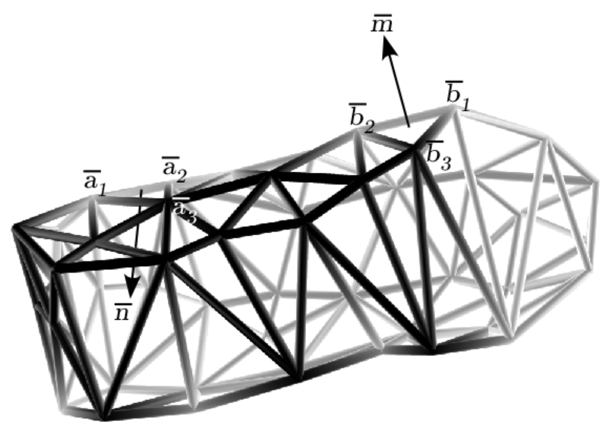

Figure 1: **Computing face normal vector.** Two faces are shown, one with vertices {a1,a2,a3} and normal vector $\vec{n} = (\vec{a_2} - \vec{a_1}) \times (\vec{a_3} - \vec{a_1})$and another with vertices {b1,b2,b3}. and normal vector $\vec{m} = \left(\vec{b_2} - \vec{b_1}\right) \times \left(\vec{b_3} - \vec{b_1}\right)$. Note how normal vector n points inward, while normal vector m points outward.

3 Mesh Characteristics

The following mesh characteristics are always measurable.

1. number of vertices, faces, edges
2. face area, individual and cumulative

3. face aspect ratio
4. edge length (distinguishability of vertices also checked)
5. bounding box of data set
6. number of components in file (number of separate objects in file)
7. number of boundaries
8. number intersecting faces (within one separate object, between separate objects in single file)♢
9. number of orphan vertices (vertices not referenced by any face)
10. missing vertices (face references vertex index that does not exist)
11. degenerate face (face references same vertex more than once)
12. contiguous numbering (vertex and face indices are sequentially numbered starting with 1)

♢Note that no differentiation is made between intersections of faces in a single mesh component and intersections of faces from different components in the same object as illustrated in Figure 2. Both cases are considered self-intersections.

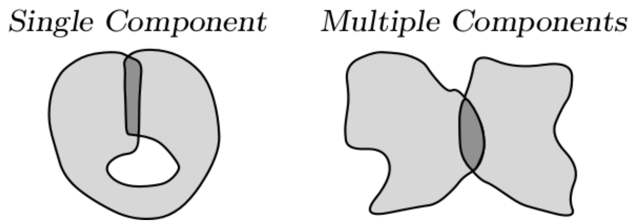

Figure 2: Self-intersecting meshes. The faces of a single mesh object may intersect under two scenarios. A single component may intersect with itself. Also a mesh object may consist of more than one component, and these components may intersect. Distinguishing between these two scenarios in the event a self-intersection is detected is left to the user.

The following mesh characteristics are only measurable for sets of mesh files.

1. (#13) number of intersecting faces between separate objects in different files
2. (#14) intermesh spacing (aka 'ECS width' and 'separation distance'. Distance between each vertex and its closest point, as defined in Section **??**)

The following mesh characteristics are only measurable if the mesh has specific attributes. The table entry indicates whether the attribute is required for determining mesh characteristic.

	Attribute			
Characteristic	Closed	Manifold	Consistent	Outward
15. Edge angle♠	no	yes	yes	no
16. Volume♡	yes	yes	yes	no
17. Genus♣	yes	yes	yes	no

♠Border edges are excluded from edge angle computation. If normals point outward, then exterior edge angles are measured. If normals point inward, then interior edge angles are measured.
♡If face normals point inwards, then the volume will be negative, but the magnitude will be correct.
♣Compute genus using Euler-Poincare formula: verts-edges+faces=2-2*genus. Actally, formula only assumes orientable mesh; future versions can check for orientable, not oriented. Formula also only applies to single components of mesh. The genus of a mesh is not computed if orphaned vertices are found.

Table 1: **Characteristic-attribute dependency.** The measureability of three mesh characteristcs (edge angle, volume, genus) is dependent on the mesh having four attributes (closed, 2D manifold in R3, consistently oriented face normals, outward-pointing face normals). For example, if a mesh is manifold and consistent, then its edge angles can be measured, regardless of whether or not the mesh is closed or has outwardly-oriented face normals.

4 Space Partitioning

The mesh data set occupies a volume of space, and by paritioning this space the search for intersecting faces (characteristics 8 and 13) and separation distances (characteristic 14) is accelerated.

5 Mesh Attribute Tests

The following tests will be used to determine the state of the attribute for each mesh file.

- Closed -each edge is shared by two or more faces; nonmanifold edges are allowed.
- Manifold - All adjacent faces of vertex are reachable by adjacent edge hops; non-closed meshes are ok.
- Consistent orientation - Assuming mesh is manifold, if no edge is crossed twice in the same direction , then face normals are consistently oriented. Mesh is allowed to be non-closed.
- Outward normal - Assuming mesh is closed, manifold, and consistent, ray trace from any face to world bound and count the number of intersected faces. If count is even, then normals faces outward. If count is odd, then normals face inward.

6 Command line behavior options

Output will be written to stdout and will include min, max, mean, median, variance and histogram.

-a

evaluate just the attributes of the meshes, report the results, and exit.

-g=aspect_ratio_threshold,edge_angle_threshold,edge_length_threshold

explicitly report whether or not mesh is good (i.e. mesh is closed manifold with outward normals, no intersecting faces, no orphaned or missing vertices, no degenerate faces, contiguous vertex and face numbering, maximum face aspect ratio is less than user-defined threshold, minimum edge angle is greater than user-defined threshold, and minimum edge length is larger than user-defined threshold). Returns 'mesh is good' or '1' and 'mesh is NOT good' or '0' if used with -p option or not, respectively. Mesh goodness is not affected by separation distances.

-p

print offending mesh elements (i.e. bad edges[nonmanifold,border,flipped], nonmanifold vertices, intersecting faces)

-i

If input folder is provided, check for intersections between faces of different objects, i.e. mesh characteristic 13.

-s

If input folder is provided, compute separation distance for each vertex in every object, i.e. mesh characteristic 14. Prerequisite: all meshes in set must be good.

7 Command line examples

Note options a and g are mutually incompatible.

meshalyzer mesh_file_name

Evaluate mesh characteristics 1-12 and 15-17 and five mesh attributes for the single mesh file.

meshalyzer folder_name

Evaluate mesh characteristics 1-12 and 15-17 and five mesh attributes for each single mesh file in folder. Also evaluate mesh characteristics 13 and 14 for the set of mesh files in folder.

meshalyzer mesh_file_name -a -p

Just evaluate the five mesh attributes for the single mesh file, print the state of each attribute, and print the mesh elements preventing the mesh from being good with regards to the attributes, if any.

meshalyzer mesh_file_name -p

Evaluate mesh characteristics 1-12 and 15-17 for the single mesh file, print the results, and print the mesh elements preventing the mesh from being good with regards to the mesh characteristics and the attributes, if any.

8 All possible meshalyzer commands

1	folder -a -p -i -s		11	folder -p -g=t1,t2,t3 -s		21	folder -i -s
2	folder -a -p -i		12	folder -p -g=t1,t2,t3		22	folder -i
3	folder -a -p -s		13	folder -p -i -s		23	folder -s
4	folder -a -p		14	folder -p -i		24	folder
5	folder -a -i -s		15	folder -p -s		25	file -a -p
6	folder -a -i		16	folder -p		26	file -a
7	folder -a -s		17	folder -g=t1,t2,t3 -i -s		27	file -p -g=t1,t2,t3
8	folder -a		18	folder -g=t1,t2,t3 -i		28	file -p
9	folder -p -g=t1,t2,t3 -i -s		19	folder -g=t1,t2,t3 -s		29	file -g=t1,t2,t3
10	folder -p -g=t1,t2,t3 -i		20	folder -g=t1,t2,t3		30	file

Table 2: **Summary of meshalyzer commands.** The command options determine which mesh characteristics are measured, whether offending mesh elements are printed, and whether the goodness of the mesh is printed. Note the order of the options in the command is insignificant. 'meshalyzer' precedes all commands and was ommited for brevity.

9 Edge Angles

An edge is defined by two mesh vertices $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$, and two adjacent faces f_1 and f_2 (Figure 3). The faces f_1 and f_2 each have a third vertex in addition to $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$, designated $\overrightarrow{o_1}$ and $\overrightarrow{o_2}$, respectively. Each face also has an outward normal vector computed using the face vertices. The edge angle, θ, is defined as shown in Figure 4 where $0 \leq \theta \leq 2\pi$.

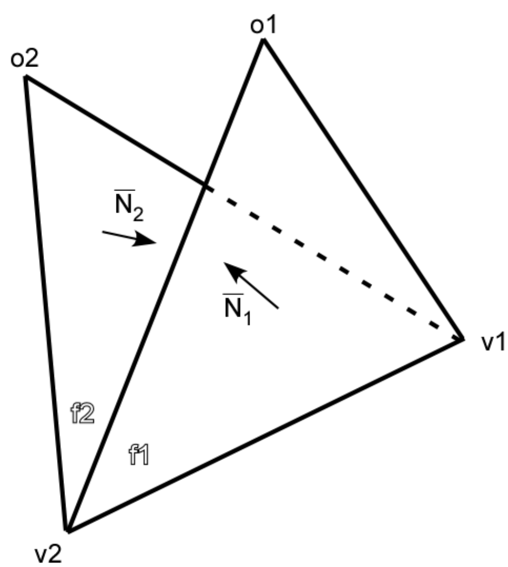

Figure 3: Edge structure.

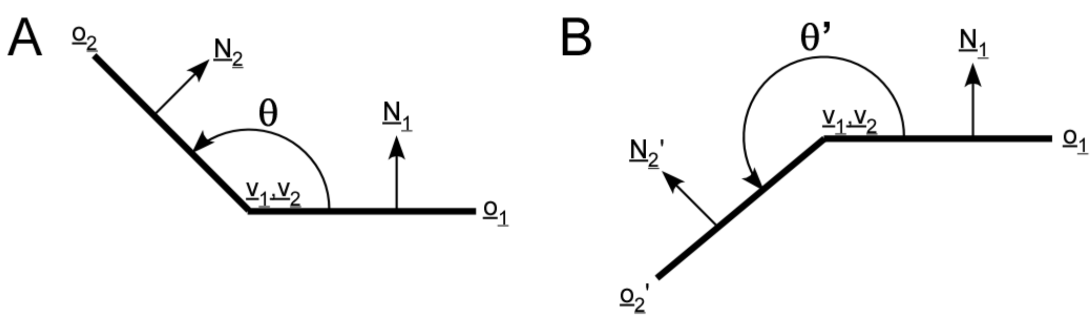

Figure 4: