LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Alma Linder
almli825@student.liu.se

Thursday 10th April, 2025

Abstract

This report focuses on hands-on familiarization with the triangular half-edge mesh structure. The purpose is to understand the pointer connectivity mechanics of the mesh (which avoids redundant storage of vertices and edges and allows neighborhood consciousness and mesh traversal) and some of its computational operations (i.e. volume, area, genus and curvature calculations). The report details the theory behind these concepts, the practical implementation, and their numerical results. Area and volume were found to be slightly smaller than their real-world counterparts. Signs for Gaussian- and Mean curvature correctly approximates the predominant bulging of different mesh surfaces. Finally, the Genus computations exemplify complex cases where simply looking is not enough to determine the Genus.

Content wise and implementation wise, this laboratory work aims for `grade 4`.

1 Background

Firstly, the half-edge mesh data structure - sketched in Figure 1 - was completed. This involved creating and adding faces and their normals via their vertices. Edges of the faces were split into two - forming pairs - and connected counter clockwise to form inner loops in the faces (the counter clockwise direction ensures traversal over the mesh surface without getting stuck). This connection was in each triangle achieved by taking the first half-edge in each of the three pairs of the face and assign its `next` pointer to the coming half-edge and its `previous` pointer to the preceding half-edge. Half-edges pairs across adjacent faces effectively "glue" the mesh together.

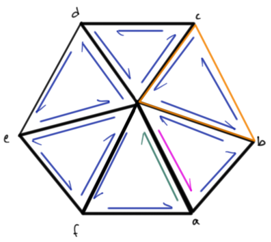


Figure 1: Sketch of the half-edge mesh structure. Arrows indicate edge orientation and the green and pink half-edges exemplifies a pair.

Vertex neighbors can be found by starting on any vertex. A 1-ring neighborhood (all triangles connected to the selected vertex) was created around it when leveraging on the knowledge that half-edges point away from their vertex. Navigation from vertex to vertex was thus achieved by utilizing the current half-edge's pointer to next to extract the current vertex and then went forward via the current half-edge's previous pair. For instance, starting on the `pink` half-edge in Figure 1 yields vertex `a`, as `a` is the vertex of `pink`'s `next`. The vertex corresponding to the previous half-edge pair of the `pink` is vertex `b`.

Collecting a neighborhood of faces included iterating from the current half-edge to the previous half-edge's pair, located in the adjacent face. Again, starting on `pink` would extract the face in which `pink` is located. The second half of `pink`'s previous half-edge pair is in the `orange` face, thus extracting this face as the next in the neighborhood.

The next task was to implement vertex normals. This normal can be computed as the normalized sum of all face normals of faces sharing the vertex in question, which Equation 1 explains. In Equation 1, $\vec{n_{v_i}}$ designates the vertex normal of vertex v_i, $\vec{n_{f_j}}$ is the face normal of face fj in the neighborhood, and the denominator is the normalization. Since the face structure stores face normals as an attribute (as mentioned earlier), computing vertex normals only require calling the face neighborhood of each vertex and accumulating the sum of the normals from the adjacent faces.

$$\vec{n_{v_i}} = \frac{\sum_{j \in N_1(i)} \vec{n_{f_j}}}{\left\| \sum_{j \in N_1(i)} \vec{n_{f_j}} \right\|} \tag{1}$$

The surface area of a triangular mesh is the sum of each individual triangle's area. Because the normal of a triangle can be computed from the cross product of its corners (perpendicularity) and the magnitude of that normal corresponds to the area of the triangle's parallelogram, Equation 2 for triangle mesh surface area holds, with v_{n_i} being the vertices of triangle i. From the position attribute of vertices and an iterator over edges, all vertices for all triangles were visited.

$$\text{Area} = \sum_i \frac{1}{2} \left\| \vec{v2_i} - \vec{v1_i} \times \vec{v3_i} - \vec{v1_i} \right\| \tag{2}$$

To compute the volume of a triangular mesh, the *divergence theorem* displayed in Equation 3 can be used.

$$\int_V (\nabla \cdot \vec{F})\, dV = \oint_A \vec{F} \cdot \vec{n}\, dA \tag{3}$$

, where the left side is the volume integral, and the right-hand side is the surface integral. By discretizing the surface on a triangle-by-triangle basis - see Figure 2 - the flow through each face can be computed by firstly finding the flow velocity at the face centroid, $\bar{F}$ (to adhere to triangle size influence on total flow), and secondly consider the direction of that velocity via the face normal $\vec{n}$ and the full triangle area A. Consequently, the mesh volume is approximated to Equation 4.

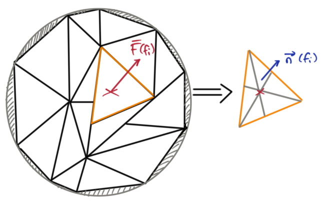


Figure 2: Sketch of volume computations, where the continuous surface integral of the Equation 3 is discretized down to each face in the mesh. $\bar{F}$ is the outflow of the triangle at its centroid point (red cross) and $\vec{n}$ is the face normal of the triangle.

$$\int_V (\nabla \cdot \vec{F})\, dV = 3V \approx \sum_{i \in S} \bar{F}(f_i) \cdot n(f_i) A(f_i) \tag{4}$$

By iterating all faces in the face vector and extracting their vertices' positions for the centroid, computing their areas, and retrieving their normals, the approximation sum in Equation 4 was aggregated.

Gaussian curvature is the geometric mean of the principal curvatures of the mesh surface, given by κ_1 and κ_2 in Equation 5. A designates the area of the surface, θ - illustrated in Figure 3 - is the angle created by the 1-ring of the current vertex v_j, and 2π is for the lap traversed around the 1-ring. The Gaussian curvature is positive for a convex surface, negative for a concave surface and zero if the surface is flat.

$$K = \kappa_1 \kappa_2 = \frac{1}{A}(2\pi - \sum_{j \in N_1(i)} \theta_j) \tag{5}$$

$$\theta_j = acos((a^2 + b^2 - c^2)/2ab) \tag{6}$$

Figure 3 shows that θ is given by Equation 6 using the notation of Figure 3, which is the standard law of cosine.

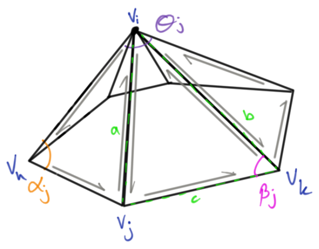


Figure 3: Sketch of Gaussian- and Mean curvature components.

The Mean curvature H is the arithmetic mean of the principal curvatures κ_1 and κ_2. Equation 7 shows that H can be approximated with the angles α and β - refer to Figure 3 - and the Voronoi face area A around the current vertex v_i.

$$H \approx ||(\cot\alpha_i + \cot\beta_i)(v_i - v_j)||/(4A) \quad (7)$$

To access the vertices v_h, v_j and v_k for the angles α and β, the face neighborhood for the starting vertex v_i was retrieved. Then each coming (v_k) and previous vertex (v_h) of each current vertex v_j was traversed until one lap in the ring had been completed. For each iteration the angles, Voronoi area and vertex positions were computed.

The Genus, G, is a measure of holes in a mesh, and was derived from the *Euler-Poincaré formula*. Since the mesh is closed, each triangle has one loop, and so the number of loops L = number of faces F. Thus, the *Eulter-Poincaré* formula can be simplified to 8.

$$G = \frac{E + 2 - V - F}{2} \quad (8)$$

, where E is the number of edges and V is the number of vertices. The number of shells S is assumed to be one. Since the mesh structure keeps edges, vertices and faces in vectors, simply calling their size before running Equation 8 was enough to fill Equation 8.

2 Results

Figure 4 shows that the vertex normals (green) and the face normals (red) are correctly computed and normalized as they point outwards, have equal lengths, and there is only one for each face or vertex. Since vertex normals are the sum of face normals of faces sharing a given vertex, some vertices in some shapes will get more contribution than other, hence the small angles on the vertex normals of the cube in Figure 4(a).

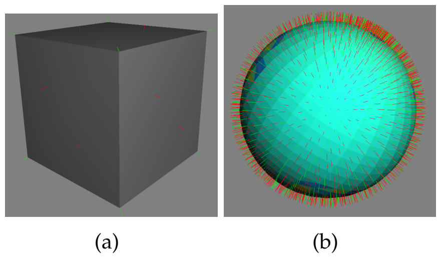


Figure 4: Resulting face normals (red) and vertex normals (green) on a cube and a sphere with radius $r = 1$. The *"jet"* color map was applied to the sphere faces for better visibility.

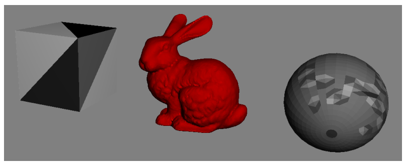
As Table 1 details, the area and volume results are slightly smaller than their ground truth counterparts for the sphere in Figure 4(b). This is to be expected since a continuous flat sphere surface was approximated with discrete triangle surfaces - leaving small pockets of air between the triangle mesh and the imagined full sphere, as is dashed in Figure 2 - and leverage on the *divergence theorem*, whose volume integral (left-hand side of equation 3) in reality is valid for force field flows through infinitesimally small cubes, not triangle shapes.

Table 1: Approximated results and ground truth values for area and volume on the sphere mesh in Figure 4(b).

Metric	**Appr. Values**	**Ground Truths**
Area	12.511	12.56637
Volume	4.15192	4.18879

Figures 5(a) and 5(b) and Table 2 show the Gaussian and Mean curvature results respectively, on a concave mesh (sphere), mixed mesh (bunny) and a flat mesh (cube).

Since Gaussian curvature finds curvature at a point by finding a sphere with radius r that tracks the surface at that specific point, the expected Gaussian result for the sphere (which has $r = 1.0$) would be to fit another sphere with $r = 1.0$ at any point on its surface. However, Table 2 shows that the calculation is quite off ($[0.28, 0.41]$). This is because the algorithm doesn't consider any neighborhood of a point, and that the sphere is large (1984 faces) and flat (zero curvature, which the cube accurately approximates to in Figure 5 and Table 2) at most of its surface points due its composition of triangles. Thus, every point Gaussian curvature is measured at, will contribute with more flatness than curvature. The inability to consider neighborhood also shows when coloring, as the cube in Figure 5 exemplifies. Only when considering a neighborhood with Mean curvature is it possible to get an average color across each side rather than one for each triangle face.


(a)

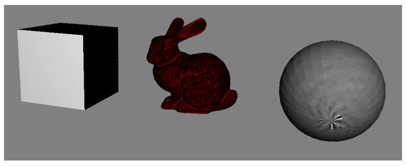

(b)

Figure 5: Results of Gaussian- (top) and Mean curvature (bottom) on a few meshes. Color map *"hot"* was applied to the bunny and all cases use color on face normals. Minimum and maximum curvature values are presented in Table 2.

Conversely, Mean curvature manages to capture the overall curvature of the sphere quite well - as Table 2 also displays - thanks to the algorithm considering the angles at two vertices in the neighborhood of the current vertex.

Table 2: Gaussian- and Mean curvature ranges for the three meshes in Figure 5.

Mesh	Gaussian Curvature	Mean Curvature
Sphere, $r = 1.0$	[0.28, 0.41]	[0.99, 1.00]
Bunny	[-42.5, 187.88]	[2.4e-7, 23.55]
Cube	[0.16, 0.19]	[0.58, 0.58]

The results for the bunny are also anticipated. Because it contains saddle points - especially around ears and legs - Gaussian angles θ around the current vertex can become negative, hence the lower range -42.5. Mean curvature on the other hand returns a length - i.e. the magnitude - of how much curvature there is, which takes on a brighter red in Figure 5(b) of the bunny. It is possible and more accurate to keep the direction as well of the Mean curvature vector. That way, curvature results for objects that are both concave and convex - e.g. the torus in Figure 6 - will correctly yield a range that cover both negative and positive values, rather than just a zero value and a larger positive value.

In Figure 6, the white numbers below each mesh indicate the result genus G of that mesh. Realizing that the cow mesh is without holes and that the torus has one hole is easy to use to verify the correctness of the genus algorithm. The cube on the other hand is not so simple. Its wireframe shows two cross-intersecting tunnels, indicating a genus of $G = 2$. The definition of a hole in a topological surface can be defined as the largest number of closed, non-intersecting lines that can be drawn on the surface without separating the surface [1]. Thus, the cube must have more, hidden disconnections, most probably located around the intersection of the two tunnels, since its genus value is $G = 5$. This case exemplifies that seemingly simple meshes are in fact rather complex.

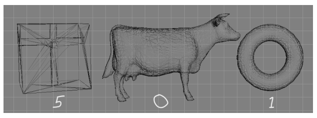


Figure 6: Genus computation results (white numbers) for three meshes with varying complexity and closedness.

References

[1] Weisstein, Eric W. Genus. [Online]. Available at: `https://mathworld.wolfram.com/Genus.html`, 2024. From MathWorld–A Wolfram Web Resource, Retrieved: 2025-04-06.