CGGS CW2: Discrete Analysis and Parameterization

s2693586

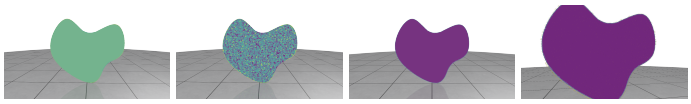
March 22, 2025

1 Section 1

1.1 Background

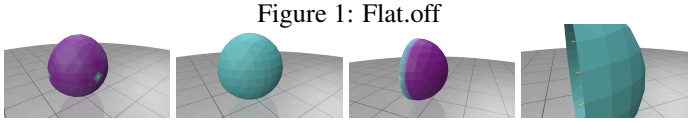
Curvature analysis is an important technique in computational geometry for examining the geometric properties of meshes. In this section, we will evaluate Gaussian curvature, mean curvature, Gaussian region and curvature normals to gain insights into the characteristics of complex meshes.

1.2 Results



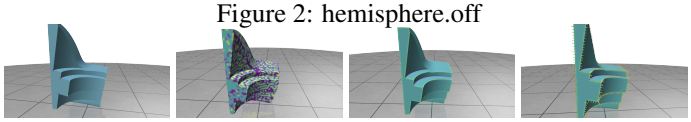
(a) Gaussian (b) G. Region (c) Mean (d) Mean Normal

Figure 1: Flat.off



(a) Gaussian (b) G. Region (c) Mean (d) Mean Normal

Figure 2: hemisphere.off

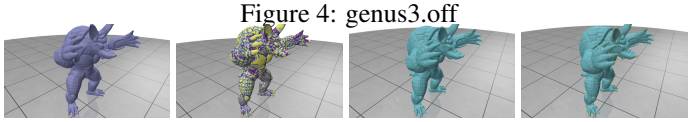


(a) Gaussian (b) G. Region (c) Mean (d) Mean Normal

Figure 3: fandisk.off

(a) Gaussian (b) G. Region (c) Mean (d) Mean Normal

Figure 4: genus3.off



(a) Gaussian (b) G. Region (c) Mean (d) Mean Normal

Figure 5: armadillo.off

1.3 Analysis & Insights

From Section 1.2, we can see the behaviour of different functions on different types of meshes. In this section, we will present a detailed analysis of the results we obtained to demonstrate the effectiveness and accuracy of curvature computation methods.

1. Flat (Planar Geometry, Figure 1)
   The Flat model matches theoretical expectations as both Gaussian and mean curvatures are near zero across the surface, and they are consistent since the model displays nearly no colour variation. Curvature normals are negligible; they appear only near the boundaries, which highlights boundary effects. The Gaussian region shows that there are possibly some small bumps or artefacts, as any bending in the surface will produce positive or negative curvature in different regions. The Gaussian region confirms that the curvature is uniform.

2. Hemisphere (Figure 2)
   This mesh displayed uniformly positive Gaussian and mean curvatures as expected. Gaussian curvature is always positive as it is a convex surface. Mean curvature is also positive; this indicates that it is a smooth, convex shape. The Normal vectors are uniformly distributed, which again indicates that the mesh is convex.

3. Fandisk (Figure 3)
   The fan disk shows Gaussian curvature peaks at its sharp corners; this can be easily differentiated from other parts of the mesh. Mean curvature displays transition at the edges; it shows all sharp features of the mesh. We can see that the normal vectors at the edges have very big magnitudes, which is expected to happen in these cases. Gaussian regions are similar to the Flat mesh, which means there are possibly small bumps or artefacts in the mesh.

4. Genus3 (Complex Saddle Structure, Figure 4)
   The Genus3 mesh shows the alternating regions of positive and negative Gaussian curvature, reflecting the saddle topology. Mean curvature is smooth across the model and only varies at the edges. Normal vectors are mostly around the saddle points and point in different directions; this clearly shows that the mesh is complicated. Gaussian region highlights clearly the positive and negative curvature areas of a saddle mesh.

5. Armadillo (Detailed Geometry, Figure 5)
   This mesh clearly shows how the algorithm handles detailed shapes. Gaussian and mean curvature both have a big range, but most of them lie inside a small interval, this indicates that the mesh is smooth overall, but there are still isolated spikes due to small geometric details. We can see that all the detailed parts are captured clearly, and the curvature normals also show the geometric details through directional changes. Gaussian region captures many small detailed areas; this indicates the complexity of the model.

1.4 Discussions

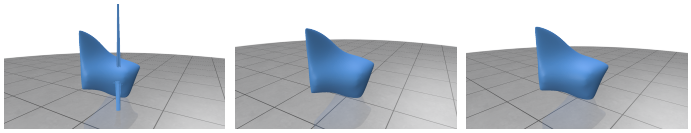
We can see that the properties of the meshes are accurately reflected by the Gaussian curvature, mean curvature, Gaussian region and mean normal. The plane-like model shows zero curvature everywhere except boundaries; the hemisphere shows uniform positive curvature, and the fan disk shows sharp curvature at the edges, the saddle-mesh genus3 has an alternating curvature between negative and positive, and the Armadillo with the most details demonstrates how well our algorithm can capture the curvature in a mesh. These findings align with the true nature of these functions and our expectations and also validate the reliability of the curvature estimation.

2 Section 2

2.1 Background

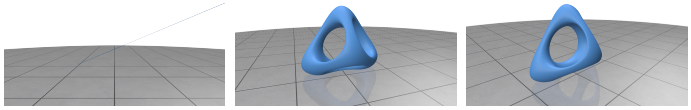
In this section, we will evaluate the behaviour and performance of our function. By adjusting the timestep of our function, we can control how rapidly or smoothly a mesh is transformed. We will compare explicit and implicit approaches as well as timestep selections of our function on various meshes.

2.2 Results



(a) ts=0.9, explicit (b) ts=0.9, implicit (c) ts=0.8, explicit

Figure 6: fandisk.off



(a) ts=0.2, explicit (b) ts=0.2, implicit (c) ts=0.1, explicit

Figure 7: genus3.off

2.3 Discussion

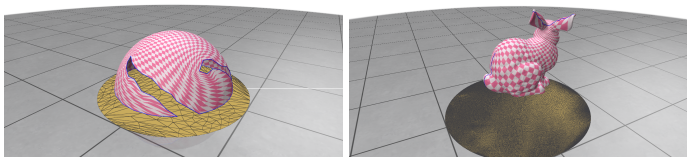
In our experiments, we have tested our function on a few different meshes and selected 2 representable examples as shown in Section 2.2. From Figure 3, we can see that when applying the explicit function to simple models, if the timestep exceeds 0.9 or gets close to 1, it will result in significant artefacts, including vertex explosions due to the rapidly varying geometry, whereas for complex model the artefacts will appear at a smaller time step as shown in Figure 7. On the other hand, we can see that the implicit method will not lead to any artefacts as it solves the linear system $(M + \delta t \cdot L)v(t + \Delta t) = M \cdot v(t)$, this maintains the stability and allows larger timesteps without affecting the flow. However, the implicit method will require more computation resources. Hence, while the explicit method can be computationally efficient for simpler, uniformly detailed meshes, the implicit approach is far more robust when dealing with complex geometries to ensure the mesh retains its overall structure.

3 Section 3

3.1 Backgound

Parameterisation is an important process in geometry processing. It maps a 3D surface onto a 2D domain, which will allow, for example, texture mapping onto the model. In this section, we will mainly examine the distortions created by the Tutte parameterization technique.

3.2 Results



(a) hemisphere_non_convex (b) bunny_fine

Figure 8: Distortion in a Tutte parameterisation.

3.3 Discussions

Figure 8 clearly shows the distortion that occurs in Tutte parameterization. From Figure 8a, we can see that the boundary of the hemisphere is not convex as the Tutte parameterization forces the boundary vertices onto a circle of fixed radius. A mismatch will occur between the actual boundary of the shape and the circular embedding. Figure 8b is the Standford bunny; we can see that around the ears, the curvature varies. The mesh has a single boundary loop cut. However, this introduces a discrepancy between high-curvature areas and flatter areas.
Building on these observations, since the Tutte parameterization is simple and computationally efficient, it cannot fully accommodate meshes with high curvature or non-convex boundaries. Forcing any boundary shape onto a loop with a fixed radius will introduce discrepancies between the geometry and the embedding. Additionally, the harmonic nature of Tutte parameterisation does not preserve angles or areas; this will lead to non-uniform stretching or compressing depending on the curvature.

4 Section 4

4.1 Introduction

In this section, we compute the principal curvature and principal curvature directions at each vertex of the mesh by fitting a local quadratic surface at the vertex's 1-ring neighbour. This method is based on algebraic quadric fitting, which has shown to be effective for robust curvature estimation even for irregular meshes [1].

4.2 Mathematical Backgroud

Given a vector v on a surface with normal n, we approximate the surface around v by rotating its neighbourhood until n aligns with the $\hat{z}$ axis. The surface is treated as the quadratic function

$$z_v(x, y) \approx ax^2 + by^2 + cxy + dz + ey + f. \quad (1)$$

The coefficients (a, b, c, d, e, f) are set by a least squares fit to the 1-ring neighbours; the Hessian of this function will become:

$$H = \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix}.$$

The matrix H includes the local curvature information. After diagonalising H, the eigenvalues will be the principal curvatures, and the eigenvectors will be the principal directions. These directions will then be rotated back into the global coordinate system. This approach is consistent with the framework of computing second-order differential quantities on meshes proposed in [2].

4.3 Implementation

1. Load Mesh and Compute Vertex Normals:
   We load the mesh from an OFF file into vertex positions and faces and compute the vertex normals by accumulating face normals and normalizing them.

2. Local Coordinate:
   For each vertex v, let n be its normal, we construct two tangent axises $\mathbf{x}_{\text{axis}}$, $\mathbf{y}_{\text{axis}}$ so that $R = \begin{bmatrix} \mathbf{x}_{\text{axis}} & \mathbf{y}_{\text{axis}} & \mathbf{n} \end{bmatrix}$, which aligns n with the z axis in local coordinates.

3. Quadratic Fitting:
   For each neighbour v_j, we transform it to the local frame (x_j, y_j, z_j). We then fit the polynomial equation 1 using least square. In matrix form, we obtain:

   $$\mathbf{A} \cdot \alpha = \mathbf{b}, \quad \alpha = (a, b, c, d, e, f,)^{\mathbf{T}}.$$

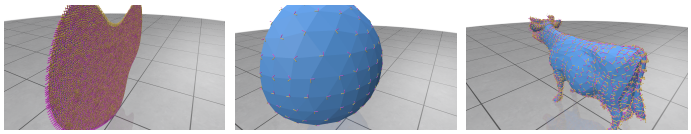
   This can then be solved with the SVD-based pseudo-inverse to ensure the minimum 2-norm solution.

4. Compute Hessian and Principal Curvature:
   After obtaining the coefficients by solving the SVD, we can then fit them into the Hessian in the local x,y-plane:

   $$H = \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix}.$$

   Then by diagonalising H we can get the eigenvalues k_1, k_2 and eigenvectors e_1, e_2. In the end, we rotate e_1, e_2 back into the global coordinates.

4.4 Results



(a) Flat (b) hemisphere (c) cow

Figure 9: Principal Directions visualise

4.5 Discussions

After running our script on the chosen benchmark mesh, we have observed that, for principal curvatures, both have larger values in regions with high curvature and are close to zero, such as in the Flat mesh. For principal directions, the vector field aligns with features on the mesh. In highly curved areas, the direction clearly followed the local bending lines. Computing principal curvatures via quadratic fitting is a straightforward implementation, but it is computationally expensive as it is recursively solving least squares for each vertex. Apart from this, the results demonstrated correct principal curvature, and this is suitable for many geometric processing tasks [1, 2].

5 Conclusion

In this coursework, we have explored a number of techniques in discrete differential geometry, from basic curvature estimation, explicit and implicit surface reconstruction, and Tutte parameterization to principal curvature computation.
Throughout our experiments, we have demonstrated the trade-offs of efficiency and stability. In general, whether the method is correct will depend on both the target mesh and the application.

References

[1] M. Makovník and P. Chalmoviansky, "Curvature estimation for meshes via algebraic quadric fitting," *arXiv preprint arXiv:2304.08909*, 2023.

[2] X. Jiao and H. Zha, "Consistent computation of first-and second-order differential quantities for surface meshes," in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, 2008, pp. 159–170.