A Computational Framework for Calculating Discrete Bending Energies of Deformed Surfaces Represented by Point Clouds

R.S. Hutton and J. A. Hanna
Department of Mechanical Engineering, University of Nevada, 1664 N. Virginia St. (0312), Reno, NV 89557-0312, U.S.A.

The geometry of surfaces is intrinsically linked to their physical properties. Curvature, a fundamental geometric descriptor, plays a critical role in the mechanical behavior of surfaces, particularly in the context of elastic energies of plates and shells. We present a computational framework for the estimation of Gaussian and mean curvatures on discrete surfaces represented by point clouds. The method operates independently of pre-defined normal vectors and is designed to enable the extraction of elastic energies of post-buckling stable states of plates and shells. This framework facilitates the study of surface geometry and mechanical properties through 3D-scanning analysis.

I. INTRODUCTION

We introduce a computational method for curvature estimation and deformation energy calculation on surfaces represented as point clouds. This method is composed of: data acquisition, Voronoi processing and mesh generation, curvature calculation, and energy estimation. We begin with the collection of 3D point cloud data, utilizing 3D scanning technologies. This initial stage captures the geometric details of the object of interest, laying the groundwork for subsequent mesh processing and analysis. Upon acquisition, the point cloud data undergoes processing with PyMeshLab, a step crucial for mesh generation. The process employs a specific filter, which iterates to refine seed placement and achieve desired sampling density across the surface. This step is critical for creating a Voronoi mesh that serves as a base for curvature calculations, ensuring that the mesh accurately represents the underlying geometry of the scanned object. With the mesh prepared, both Gaussian and mean curvatures are calculated without relying on pre-defined normal vectors. It uses singular value decomposition (SVD) on neighborhoods of points to determine the principal directions and fits a quadratic surface to these neighborhoods. Curvature signs are adjusted for consistency, and explicit functions (F(x, y) = z) derived from classical differential geometry are used to calculate curvatures. The algorithm's independence from traditional normal vector calculations allows for enhanced accuracy in estimating curvature. We then calculate both the bending and stretching energies based on the estimated curvatures. Bending energy is linked to mean curvature, while stretching energy is derived from Gaussian curvature. The formulas provided for these calculations enable a detailed analysis of the material's response to deformation, which is essential for understanding the mechanical behavior of surfaces. This methodology presents an advancement in the field of structural analysis. By enabling precise curvature estimation and deformation energy calculations, it offers a valuable tool for researchers and engineers working on the geometric and mechanical analysis of materials.

II. PLACEHOLDER

III. METHODS

An Einscan-Pro 3D scanner is used to capture the geometric details of deformed sheets. The deformed sheets are scanned from multiple angles to ensure comprehensive coverage and to accurately capture their intricate geometries. The scanner operates in a manner that allows for the collection of high-resolution point

cloud data, effectively translating the physical characteristics of the sheets into digital representations. This process results in a dense aggregation of data points, each representing a specific location on the surface of the material in three-dimensional space. The output from the Einscan Pro scanner is converted to a text file that contains the coordinates of each point in the point cloud. These coordinates are used in the subsequent steps of our process, as they form the raw data from which we will generate the meshes and calculate curvatures. To ensure the integrity and usability of the data, we perform preliminary checks for completeness and accuracy, removing any outliers or anomalies that could potentially skew the analysis.

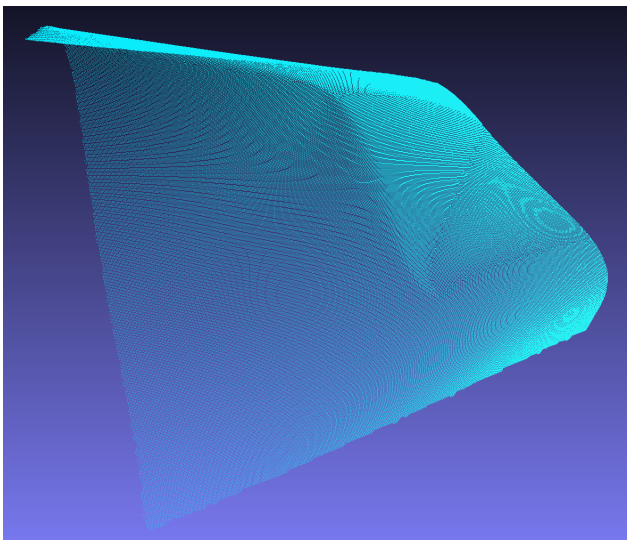


FIG. 1. Point cloud constructed by scanning the structure

Upon obtaining the point cloud data, we employ the Voronoi Sampling Meshlab plugin (or throught PyMeshLab) to generate and process the mesh:

```
ms.generate_sampling_voronoi(iternum = 10, samplenum = 2264, radiusvariance = 1.000000)
```

Where we set the samplenum to minimum 1% of the total points, higher for final results. This filter consists of the following steps:

1. It starts by preparing the mesh and auxiliary meshes for output and processing. It sets various mesh data masks necessary for the operations to follow.

2. Seed Generation and Distribution: Through Poisson sampling, it generates a set of initial points (seedVec) over the mesh. These seeds are used to create Voronoi cells in the subsequent steps.

3. Voronoi Relaxation: Performs Lloyd's relaxation algorithm to optimize the placement of seeds for a more uniform Voronoi diagram. It iteratively adjusts seed positions to minimize the distance to points within their Voronoi cells, potentially using different distance metrics (EuclideanDistance, IsotropicDistance, AnisotropicDistance).

4. Voronoi Diagram Conversion: Converts the Voronoi diagram into a mesh representation and a polyline representation, visualizing the Voronoi cells and their edges.

5. Scaffolding Construction: Builds a "scaffolding" mesh that represents the structural layout of the original mesh in a simplified form.

6. Solid Wireframe Construction: Depending on flags set (edgeCylFlag, vertCylFlag, vertSphFlag, faceExtFlag), it constructs a solid wireframe model by replacing mesh elements (edges, vertices, faces) with geometric primitives (cylinders, spheres, prisms) of specified dimensions. This is for aesthetics only.

This allows us to realize a "Voronoi shell" of the points cloud, which is a mesh that is composed of polygons, each with triangular sub-faces, where the area of the polygonal cell has been iteratively selected using Lloyd relaxation:

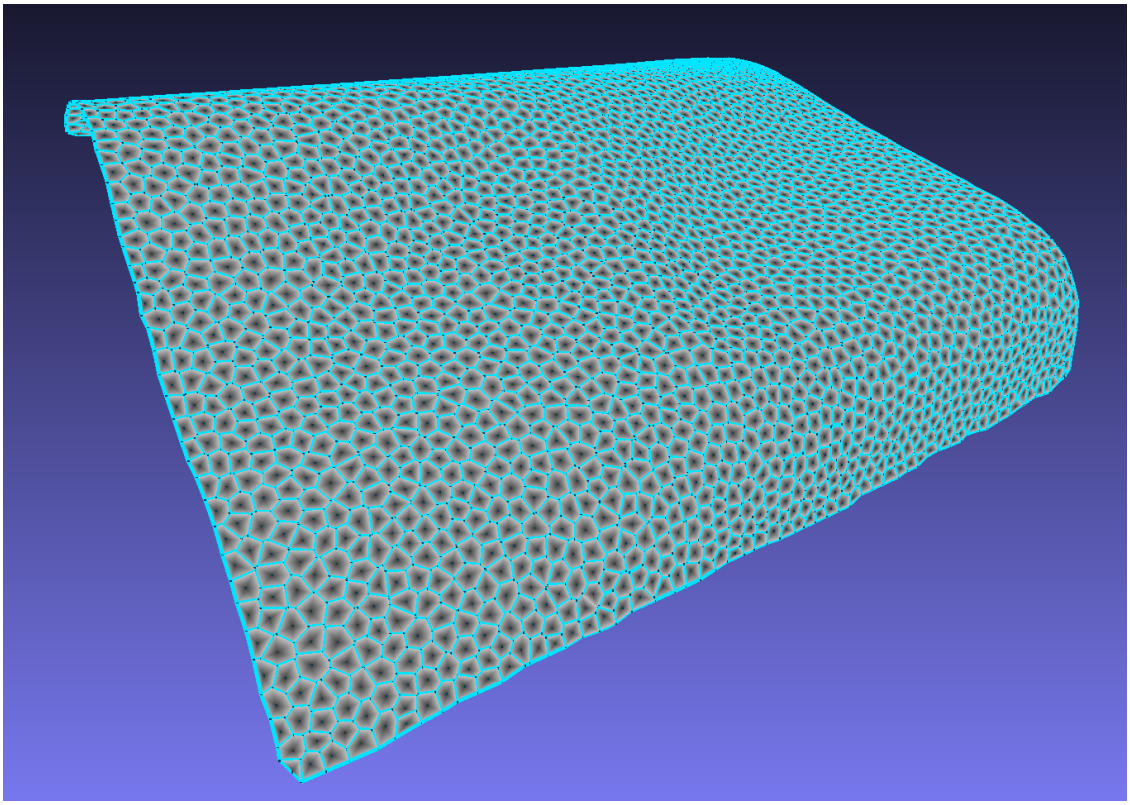


FIG. 2. Voronoi shell calculated from the point data

A. Curvature Calculation

With the mesh prepared, we proceed to calculate the discrete curvature at each point on the mesh using a custom program written in Python, and hosted on my Github. This allows us to estimate the discrete curvatures associated with each point, and provides better results compared to algorithms currently available within popular software such as MeshLab, CGAL, and Libigl.

The program is structured as follows:

1. A k-dimensional tree is constructed from the Voronoi points to organize the data spatially.

2. For each point, the k nearest neighbors are determined using the k-d tree, supporting both distance-based and epsilon-ball queries.

3. The singular value decomposition (SVD) is employed on each neighborhood to ascertain the characteristic plane, aligning with the first two vectors.

4. The neighborhood is rotated to align this plane with the xy-axis, repositioning the central point at the origin. The z-axis is approximated as the normal.

5. Consistency in curvature signs is ensured by adjusting the orientation based on the dot product between the z-axis and vectors in the neighborhood.

6. A quadratic surface is fitted to the neighborhood points using least-squares regression, yielding an explicit function $F(x, y) = z$.

7. Curvatures are computed using the following formulations, from classical differential geometry:

$$K = \frac{F_{xx} \cdot F_{yy} - F_{xy}^2}{(1 + F_x^2 + F_y^2)^2}$$

$$2H = \frac{(1 + F_x^2)F_{yy} - 2F_x F_y F_{xy} + (1 + F_y^2)F_{xx}}{(1 + {F_x}^2 + F_y^2)^{3/2}}$$

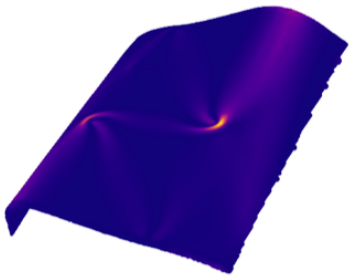
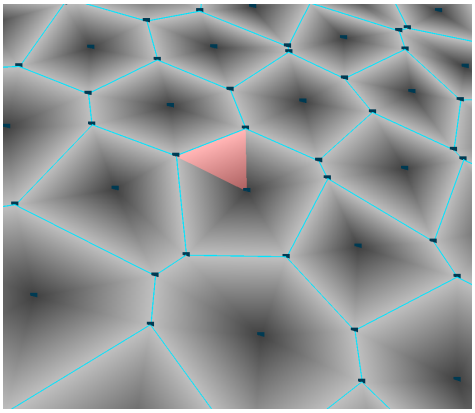


FIG. 3. Mean curvature squared plotted as a colormap on the point data

B. Face Area and Mapping Curvature

We then step through each face on the Voronoi mesh sequentially, and assign a face quality which is the average of the curvature values associates with the three points composing it. This can be done in an arithmetic or geometric fashion. In the arithmetic case, for each face in the Voronoi scaffolding we have $K_{face} = (1/3)(\sum_i K_{v_i})$, and $H_{face} = (1/3)(\sum_i H_{v_i})$.



SURFACE ENERGY

Under Development The bending energy is associated with the mean curvature of the surface. For a discrete surface, the bending energy W can be calculated as:

$$W = \frac{1}{2} \sum_{i=1}^{N} (H_i^2 \cdot A_i)$$

where H_i is the estimated mean curvature at vertex i, A_i is the area associated with vertex i, and N is the total number of vertices. The area A_i can be computed as one-third of the sum of the areas of the triangles adjacent to vertex i, for a triangular mesh.

STRETCHING ENERGY

The stretching energy functional based on Gaussian curvature, denoted as E, is calculated as:

$$E = \sum_{i=1}^{N} K_i A_i$$

where: E represents the stretching energy, S is the surface, K is the Gaussian curvature of the surface, and K_0 is the reference Gaussian curvature (zero for cylinder).

I would rather do this using the polygonal areas, since they are planar and the point in the center can provide the curvature value

The total energy of the deformed surface is a sum of the bending and stretching energies.