Surface Reconstruction From Non-parallel Curve Networks

Group 13

ASA SINGH and HIMANSHU RAJ

1 INTRODUCTION

This project implements the algorithm and methods outlined in the referenced research paper to construct surface networks from cross-sectional curves, relevant to fields like biomedical modeling and computational geometry. While previous studies have focused on reconstructing surfaces from closed curves on parallel cross-sections, this project addresses the more complex issue of reconstructing surfaces from curve segments on non-parallel planes. The algorithm is designed to handle various object segments (e.g., muscle, bone, or tissue layers) on each plane, enabling a more detailed and segmented surface reconstruction. The goal is to achieve a closed surface network that accurately represents both the exterior boundaries and internal segmentation of the object in 3D.

2 LITERATURE REVIEW

Surface reconstruction from cross-sectional data is a critical area in geometry processing and medical imaging. Boissonnat and Memari laid the groundwork for reconstructing surfaces from closed curves, primarily focusing on parallel cross-sections. Their methods effectively address simple geometries but do not account for the complexities introduced by non-parallel, multi-labelled curve networks, which are essential for accurately modelling anatomical structures.

Reconstructing surfaces from non-parallel curves presents challenges in ensuring closedness and continuous interpolation between segments. The referenced paper proposes an advanced approach using polyhedral decomposition and a divide-and-conquer strategy, providing a robust solution to these challenges. This method divides the input space into manageable cells, allowing for the construction of closed surfaces within each cell, thus maintaining continuity across non-parallel planes and effectively managing multi-label structures.

Additionally, the work by Sherbrooke on the medial axis transform of 3D polyhedral solids offers valuable insights into capturing the geometric properties of shapes, which can complement the surface reconstruction techniques discussed. This understanding of the medial axis can enhance the accuracy and efficiency of reconstructing complex structures from non-parallel curves.

To enhance the reconstructed surfaces, the authors incorporate mesh fairing techniques, such as surface-diffusion flow and edge-swapping, to refine the mesh. This optimization improves the smoothness of surfaces, which is crucial for applications in biomedical modeling, enhancing both visual quality and functional applicability.

By implementing the methods described in this paper, our project aims to advance the field of surface reconstruction, particularly in contexts involving non-parallel and complex curve networks. This literature review provides a foundational understanding of the existing gaps in research and the potential of the algorithm to effectively address these challenges.

Authors' address: Asa Singh, asa22113@iiitd.ac.in; Himanshu Raj, himanshu22216@iiitd.ac.in.

3 MILESTONES

3.1 Mid Evaluation

S. No.	Milestone	Member
1	Computing partition of surface by boundaries (2D).	Asa Singh
2	Medial axes construction (2D).	Himanshu Raj
3	Compute partitioning of space by all planes (3D).	Asa Singh
4	Obtaining projection surface for all the cells (3D).	Himanshu Raj

3.2 Final Evaluation

S. No.	Milestone	Member
1	Projecting contour planes.	Asa Singh
2	Interpolating surfaces.	Himanshu Raj
3	Rendering surfaces.	Asa Singh
4	Basic UI to interact with the 3d model.	Himanshu Raj

4 MID EVALUATION DISCUSSION

4.1 Keeping up with milestones

Of the 4 milestones mentioned, we could completely implement three of them, and the remaining one was done in two iterations and yet not complete to be delivered. The medial axes calculation in 3D requires a bounded box which is very close to the external planes to clip of the medial axes and that's where we are facing issues.

4.2 Medial Axes Construction in 2D: Implementation Details

Cell division and medial axes construction in 2D
Medial axes are sets of points that are equidistant from two or more boundaries, in our case, vertices of the plane. In a plane in 2D, we use Voronoi's approach to find medial axes. We find the midpoint of all pairs of vertices of the plane, and these points form our medial axes. It's a simplified approximation, and we only consider points that are closer to the vertices of the plane.
Computing partitions of space bounded by planes
We will call these partitions of space as cells. The cells, according to the research paper, are regions bounded by 2 or more planes and are generally convex in nature.

4.3 Space Partitioning Algorithm: Implementation Details

The Space Partitioning Algorithm implemented in the provided C++ code divides a 3D space into convex cells based on a set of contour planes. The core functionality revolves around classifying the planes into parallel and non-parallel groups, computing their intersections, and then constructing and rendering the resulting convex cells. Here's a breakdown of the key implementation details:

4.3.1 1. Classifying Contour Planes. The first step in the algorithm is to classify the contour planes into two categories:

- **Parallel Planes:** These planes are grouped based on their normal vectors being nearly identical. Two planes are considered parallel if the normalized normal vectors of the planes are sufficiently close.
- **Non-Parallel Planes:** These planes do not share parallelism and are handled separately during the partitioning process.

Parallel planes are grouped together into `PlaneGroup` structures, while non-parallel planes are kept in a separate list for further processing.

4.3.2 2. Handling Parallel Planes. For the planes that are parallel, the algorithm performs the following steps:

- **Sorting:** The parallel planes are sorted based on their distance along their normal vector. This sorting is necessary because it helps in identifying the regions between consecutive parallel planes that will form the slab cells.
- **Slab Cells Creation:** The algorithm creates convex cells (slabs) by computing the regions between consecutive parallel planes. Each slab is bounded by two parallel planes, and its vertices are determined based on the intersection of these planes with a predefined bounding box. The resulting slab cell consists of a set of vertices, edges, and boundary planes (the parallel planes that define the slab).

4.3.3 3. Handling Non-Parallel Planes. Non-parallel planes are more complex to handle as they do not define simple regions between them. The algorithm calculates intersections between sets of three non-parallel planes:

- **Intersection of Planes:** For every combination of three non-parallel planes, the algorithm computes their intersection point. The intersection of three planes is a point in 3D space, provided the planes are not parallel or degenerate.
- **Validating Intersections:** Once an intersection point is computed, the algorithm checks if this point satisfies the conditions imposed by all the contour planes. If the point lies within the region defined by all the planes, it is considered a valid vertex for the cell.
- **Cell Creation:** After determining the valid intersection vertices, the algorithm constructs the convex cell formed by these vertices. Each cell's boundary is formed by the set of contour planes, and edges are created between vertices that share at least two common planes.

4.3.4 4. Cell Construction. Once the intersection vertices are found, the algorithm constructs the convex cells:

- **Vertices:** The vertices of each cell are derived from the valid intersection points calculated earlier.
- **Edges:** The edges of a cell are defined by pairs of vertices that share at least two common planes. This ensures that the edges are correctly associated with the boundary of the cell.
- **Boundary Planes:** Each cell is bounded by the set of contour planes that define it. These planes form the faces of the convex cells.

4.4 Medial Axis: Implementation Details

The Medial Axis algorithm, implemented in the `MedialAxis` class, computes a 3D approximation of the medial axis of a given convex cell. The key components of the implementation are based on CGAL's functionality for polyhedral geometry and alpha shapes, with the goal of determining the skeleton of a 3D object defined by its boundary planes. Below is a detailed explanation of the implementation steps:

4.4.1 1. Class Overview. The `MedialAxis` class computes and visualizes the medial axis of a convex cell. The class stores the convex cell and computes its medial axis using alpha shapes. The results are stored in the `MedialAxisResult` structure, which includes:

- **Vertices:** A list of 3D points representing the skeleton of the cell.
- **Edges:** A list of pairs of vertices that form the edges of the medial axis.
- **Sheets:** A list of sets of vertex indices that form the "sheets" or connected components of the medial axis.

4.4.2 2. Conversion of Convex Cell to Polyhedron. The first step in computing the medial axis is converting the convex cell into a polyhedron using CGAL. The conversion process involves:

- **Vertices:** Each vertex of the cell is converted into a CGAL 3D point using the `convertToPoint3` method.
- **Faces:** The algorithm extracts faces of the polyhedron by iterating over the boundary planes of the convex cell. It finds which vertices lie on a given plane, calculates the center of the face, and sorts the vertices around this center to define the face. The vertices are sorted based on their angle relative to a reference vector, ensuring consistent orientation.

4.4.3 3. Result Storage. The results of the medial axis computation are stored in the `MedialAxisResult` structure:

- **Vertices:** Each unique vertex in the alpha shape is added to the `vertices` list.
- **Edges:** The edges are stored as pairs of vertex indices, representing the connections between vertices in the skeleton.
- **Sheets:** The sheets of the medial axis, which correspond to connected components, are stored as lists of vertex indices, though their calculation is not fully implemented in the provided code.

4.5 Challenges Faced

Throughout this project, we faced an array of challenges that pushed us to our absolute limits. There were moments when the sheer complexity of the tasks and the pressure to deliver felt insurmountable. The weight of setbacks, coupled with the struggle to find solutions, left us feeling deeply frustrated and emotionally drained. This moment of vulnerability underscored the intensity of the situation and our deep investment in the success of this project.

The primary reason for such a dire situation was the lack of resources and references. Even the CGAL library, primarily used for computer graphics, had only a few tutorials, and the documentation was not really easy to comprehend. From finding medial axes to cell partitioning, everything required a deep and broad search of various research papers and tutorials. Even after finding algorithms to do the task, implementation is also quite unconventional.

4.6 Intermediate Results

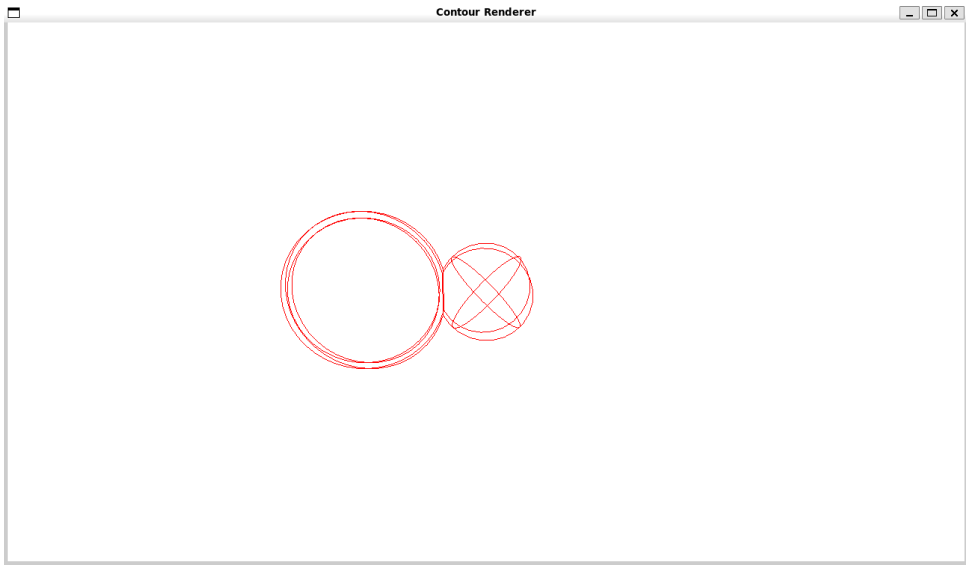


Fig. 1. Ring Contour File

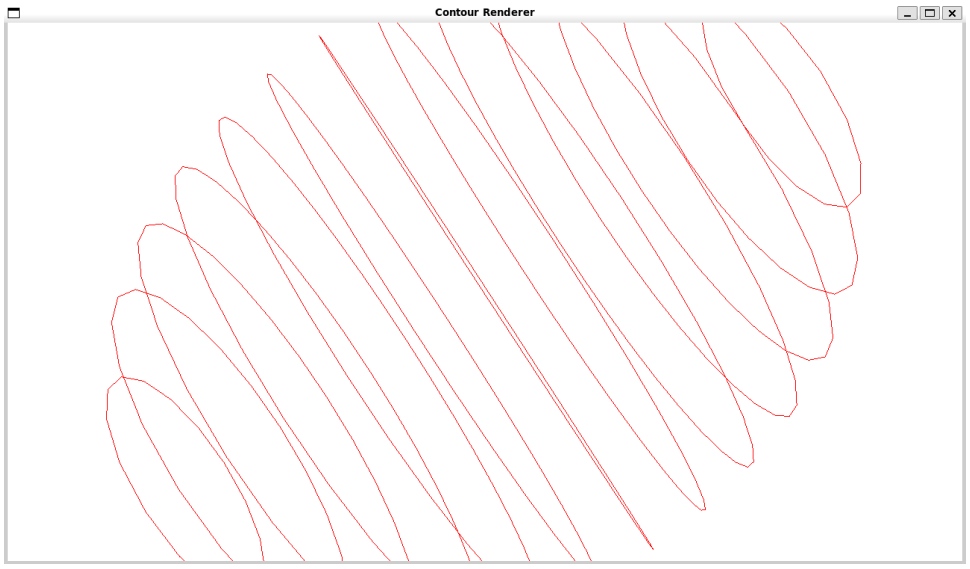


Fig. 2. Pellip Contour File

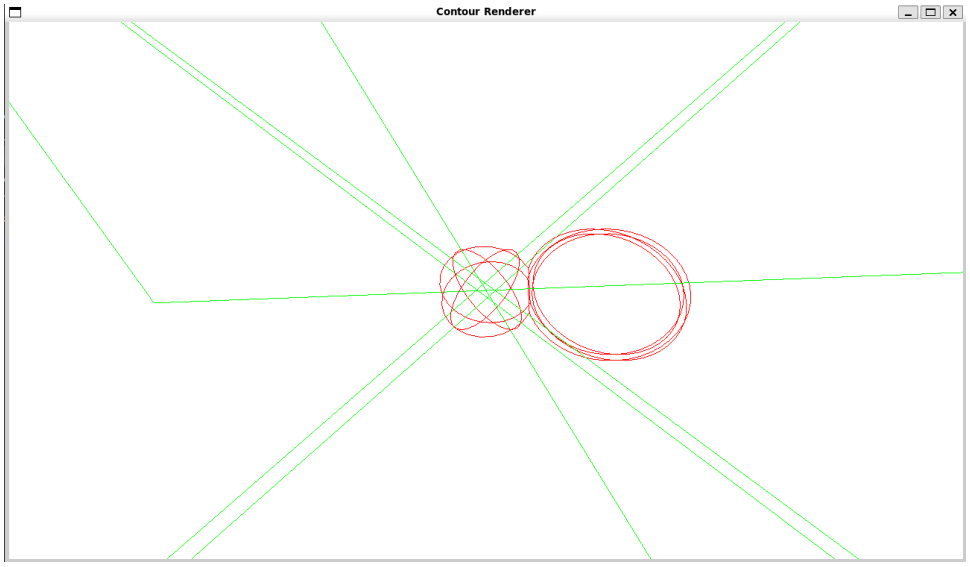


Fig. 3. Ring Curve Network divided into Convex Cells

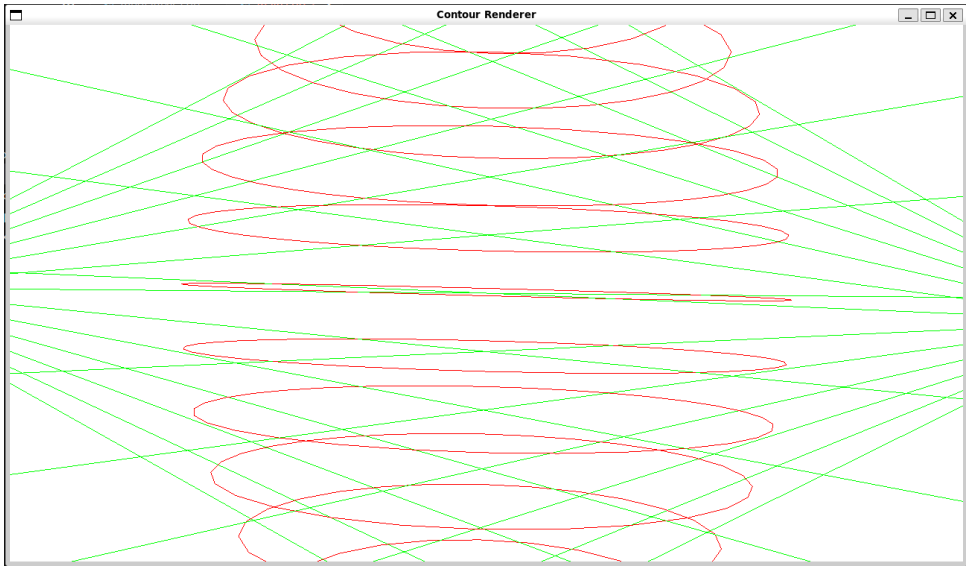


Fig. 4. Pellip Curve Network divided into Convex Cells

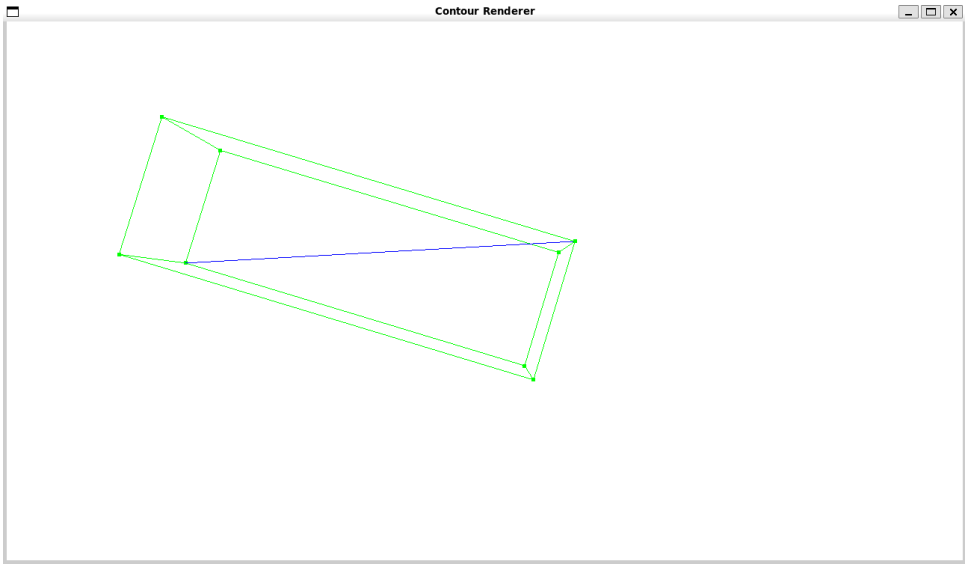


Fig. 5. Medial Axis Testing for a Cuboid

5 FINAL EVALUATION DISCUSSION

5.1 Keeping up with the milestones

Of the milestones decided for the final submission, we could achieve all of them except for mesh smoothening. Also, we decided to move forward with the axis-aligned planes approach instead of the medial axis approach because the medial axis was neither efficient nor sufficient to project the contours.

5.2 Axis aligned planes for 3D cells

The resultant polyhedrons generated during the processing phase are generally cuboidal or close to cuboidal in shape. To facilitate further analysis and processing, axis-aligned planes are constructed for these 3D cells. These planes are defined such that each one corresponds to a pair of parallel faces of the polyhedron. The selection of the appropriate axis-aligned plane is guided by the associated contour planes.

The chosen axis-aligned plane is either parallel to the contour plane or determined to be the most horizontal among all contour planes associated with the polyhedron. This ensures consistency and alignment with the geometric characteristics of the cell.

To compute the axis-aligned planes, we enclose each polyhedron within its bounding box. The bounding box defines the extent of the polyhedron along the x, y, and z axes. By averaging the coordinates across these axes, we establish the optimal positions for the planes. This method provides a robust and systematic way to define axis-aligned planes that capture the spatial properties of the polyhedron while maintaining alignment with the global coordinate system.

5.3 Contour projections on plane

Once the axis-aligned plane for a given cell is determined, the next step is to project the vertices of the associated contours orthogonally onto this plane. This process involves transforming the 3D contour geometry into a 2D representation on the axis-aligned plane.

The projection ensures that the contours are accurately represented on the selected plane while preserving their geometric fidelity. Each contour projection becomes a critical input for subsequent surface construction. By aligning these projections on a common plane, we simplify the process of creating connections between them.

The result of this projection step is a set of 2D contour vertices on the axis-aligned plane. These vertices provide the foundation for defining the geometry of surfaces that link the contours, enabling the reconstruction of the cell's internal structure.

5.4 Mesh construction

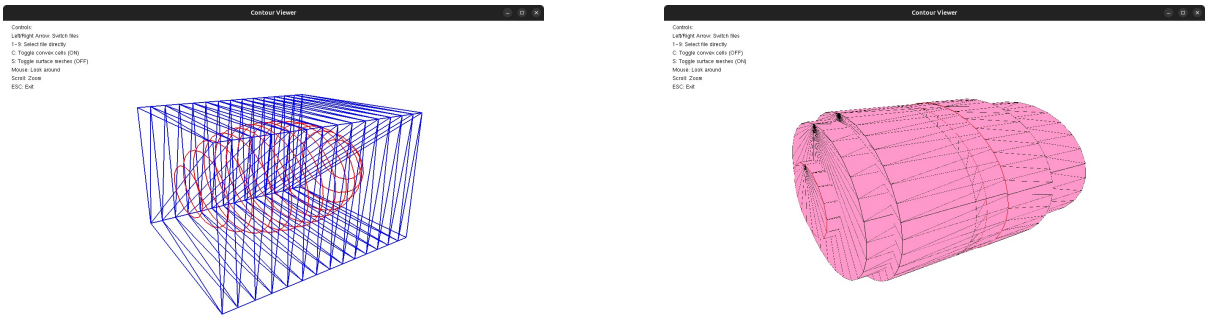
To create surfaces that connect the vertices of two or more contours within each cell, Delaunay Triangulation is employed. This triangulation technique is particularly suitable for this purpose due to its ability to create well-shaped and non-degenerate triangles. Delaunay Triangulation ensures that the resultant mesh is both geometrically robust and visually coherent.

The process begins by applying Delaunay Triangulation to the set of projected vertices on each axis-aligned plane. This generates a surface mesh that seamlessly connects the contours within the cell. If multiple contour levels are present, the triangulation process is repeated to create surfaces between adjacent levels.

Finally, all the generated surfaces are stitched together to form a complete mesh representation of the polyhedron. The resulting mesh not only captures the 3D structure of the cell but also provides a high-quality approximation that is suitable for visualization, analysis, and further processing.

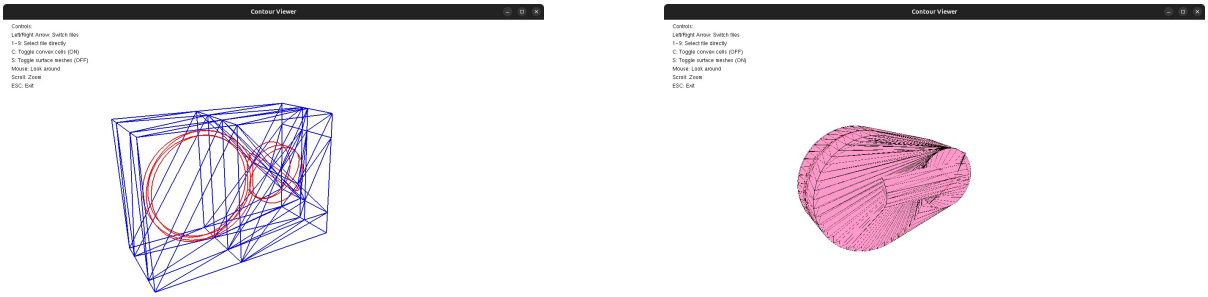
By leveraging Delaunay Triangulation and carefully projecting contours onto well-defined planes, we achieve a structured and accurate representation of 3D cells that is both computationally efficient and geometrically precise.

5.5 Final Results



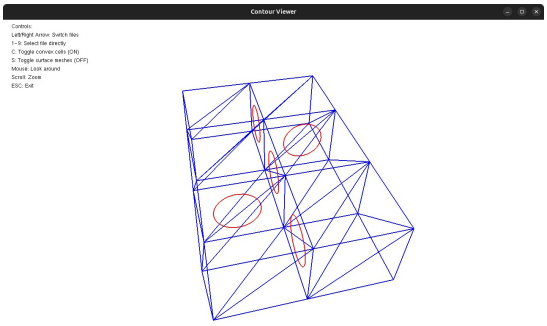
(a) convex cells (b) reconstructed surface

Fig. 6. pellip shape

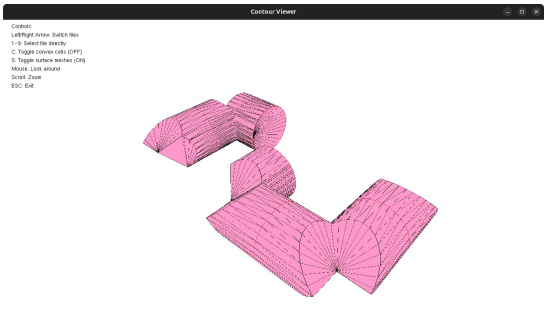


(a) convex cells (b) reconstructed surface

Fig. 7. ring shape

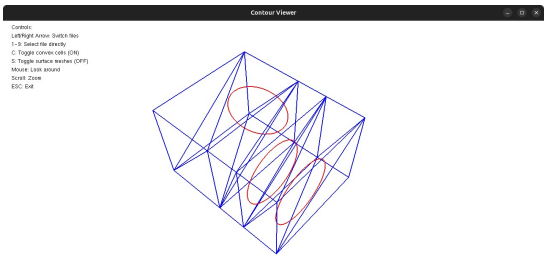


(a) convex cells



(b) reconstructed surface

Fig. 8. s shape



(a) convex cells

(b) reconstructed surface

Fig. 9. tube shape

6 REFERENCES

[1] L. Liu, C. Bajaj, J. O. Deasy, D. A. Low, and T. Ju. Surface reconstruction from non-parallel curve networks. In Computer Graphics Forum, volume 27, pages 155–163. Wiley Online Library, 2008.
[2] Sherbrooke E. C., Patrikalakis N. M., Brisson E.: An algorithm for the medial axis transform of 3d polyhedral solids. IEEE Transactions on Visualization and Computer Graphics 2, 1 (1996), 44–61.