Patient-Specific Artificial Intestine: Biomechanics of Auxetic Structures

Daniel Wei, Mingruifu Lin

February 2025

Abstract

Colorectal cancer (CRC) is the third most commonly diagnosed cancer globally, with over 1.9 million new cases and approximately 930,000 deaths estimated in 2020. PubMed Despite advancements in tissue engineering, current research predominantly focuses on replicating the intestinal absorption layer, with limited progress in recreating the muscular layer essential for bowel motility. In effect, the organicity of biological macrostructures often cannot be broken down effectively into manufacturable units. Current solutions like colostomy bags are impractical and fail to fully replicate natural intestinal function, highlighting the need for more functional alternatives. PMC This research aims to develop a bio-inspired, patient-specific artificial intestine that can be folded from a flat auxetic lattice generated upon the input of a desired 3D shape (such as a colon), which approximates smooth peristaltic movements through expansion and contraction from each auxetic unit. The algorithm can be generalized for other applications beyond prostheses. A combination of computational testing and visualization, and 3D printed prototypes are used to verify the method's effectiveness. The result of this research is an algorithm that processes data from the scanned model of a colon to generate a 3D printable flat mesh that morphs under the propagation of tension or compression along multiple control units located on its boundary. By the 2D nature of the printed surface which folds into 3D, the research impacts material engineering by preventing problems related to supports in 3D printing involving more vertical structures, allowing for more complex shapes to occur. Also, it gives options for the development of new types of prostheses that better reproduce organic movement. When combined with advanced absorption layer technologies, this approach provides an efficient solution to the production of artificial intestines and other similar organs.

Keywords: Artificial intestine, patient-specific design, biomechanics, auxetic materials, material engineering, 3D printing, 3D models, parametric surfaces

Contents

1 Outline of Algorithm

The algorithm takes in a triangulated mesh with a simple-enough, well-behaved topology, and outputs the information necessary to generate a 3D printable auxetic mesh that folds into the desired original mesh. We were inspired by Tachi's presentation. [2]

1.1 Boundary

Topology: Choose a topology: disk, cylinder, sphere.
Identification:

1. For disk: Take all vertices with 4 edges.
2. For cylinder: Take all vertices with 4 edges, and user defined cut, becomes disk. (If have time, pathfinding.)
3. For sphere: Cut into disk with random walk.

Mapping: Compute distance. Put edge-vertices until reach 1/4 of distance, then change direction. Keep going. Connect the last vertex to the first.

1.2 Parametrization

If we take 0 as the equilibrium length of a spring, then the spring energy between two vertices with spring constant D is given as follow:

$$E_{ij} = \frac{1}{2} D \|\Delta \vec{x}\|^2 = \frac{1}{2} D \|\vec{x}_i - \vec{x}_j\|^2$$

The total spring energy is thus:

$$E = \frac{1}{2} \sum_i \sum_{j \in N_i} \frac{1}{2} D_{ij} \|\vec{u}_i - \vec{u}_j\|^2$$

where each $\vec{u}_i$ lies on the 2D plane, and N_i are the neighboring vertices of i. The fraction $\frac{1}{2}$ at the beginning is added because each edge is counted twice in this formula. Notice that E is a scalar field depending on the position of each vertex. The idea is to minimize the spring energy, so we want the derivative to be zero for each parameter:

$$\frac{\partial E}{\partial \vec{u}_i} = \vec{0}$$

which is a fancy notation for gradient with respect to $\vec{u}_i$. Doing the computation, we get:

$$\frac{\partial E}{\partial \vec{u}_i} = \sum_{j \in N_i} D_{ij} (\vec{u}_i - \vec{u}_j) = \vec{0}$$

Notice that the fraction $\frac{1}{2}$ is not required anymore, because we are not counting edges twice. Anyways, we see that:

$$\sum_{j \in N_i} D_{ij}\vec{u}_i = \sum_{j \in N_i} D_{ij}\vec{u}_j$$

Hence,

$$\vec{u}_i = \frac{\sum_{j \in N_i} D_{ij}\vec{u}_j}{\sum_{j \in N_i} D_{ij}}$$

Rewritten:

$$\vec{u}_i = \sum_{j \in N_i} \lambda_{ij}\vec{u}_j$$

where $\lambda_{ij} = \frac{D_{ij}}{\sum_{k \in N_i} D_{ik}}$, we then isolate the constant terms, which are the things that involve the boundary vertices:

$$\vec{u}_i - \sum_{j \in N_i, j \leq n} \lambda_{ij}\vec{u}_j = \sum_{j \in N_i, j > n} \lambda_{ij}\vec{u}_j$$

where the boundary vertices are counted with indices after n. We can go component-wise. Let u_k and v_k be the coordinates of the 2D vertex $\vec{u}_k$, then we can write:

$$u_i - \sum_{j \in N_i, j \leq n} \lambda_{ij}u_j = \sum_{j \in N_i, j > n} \lambda_{ij}u_j$$

$$v_i - \sum_{j \in N_i, j \leq n} \lambda_{ij}v_j = \sum_{j \in N_i, j > n} \lambda_{ij}v_j$$

Let us consider only the first coordinate of the vertices. I can rewrite like this:

$$\begin{pmatrix} 1 & -\lambda_{12} \text{ or } 0 & -\lambda_{13} \text{ or } 0 & \cdots & \lambda_{1n} \text{ or } 0 \\ -\lambda_{21} \text{ or } 0 & 1 & -\lambda_{23} \text{ or } 0 & \cdots & \lambda_{2n} \text{ or } 0 \\ -\lambda_{31} \text{ or } 0 & -\lambda_{32} \text{ or } 0 & 1 & \cdots & \lambda_{3n} \text{ or } 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\lambda_{n1} \text{ or } 0 & -\lambda_{n2} \text{ or } 0 & -\lambda_{n3} \text{ or } 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix}$$

where each $a_k = \sum_{j \in N_k, j > n} \lambda_{kj}v_j$ is just the RHS of our previous equation. As you see, some components of the matrix are either λ_{ij} or 0, depending on whether $j \in N_i$. We can do the same procedure for the vertical coordinates v_i. Notice that a_i are zero if $\vec{u}_i$ is not a neighbor of a boundary vertex.

In any case, we have found a way to parametrize any mesh. [1] Solving the linear systems can be done by a library.

1.3 Choice of metric

We will choose $D_{ij} = \frac{1}{\|\vec{u}_i - \vec{u}_j\|}$. This is reminescent of the Newton per meter.

1.4 Auxetics

We can now use the difference in length between the edge of the parametrized and the original mesh to compute the rotation and radius of our auxetic units.

To determine the maximum rotation of an auxetic unit centered at $\vec{u}_i$, we will cap it to the smallest angle of its adjacent triangles:

$$\phi_i = \min\{\theta_{j,i,j+1} \mid j \in N_i\}$$

where θ_{abc} means the angle formed by the points $\vec{u}_a, \vec{u}_b, \vec{u}_c$. We will always engineer the auxetic unit to make the biggest rotation.

To determine the maximum radius of the same auxetic unit, we take the minimum of half the distance to each neighbor:

$$R_i = \min\{\frac{\|\vec{u}_i - \vec{u}_j\|}{2} \mid j \in N_i\}$$

To determine the knob's placement radius for each of the 3 rods that will connect to the auxetic unit, we use the difference in length mentionned previously. Let r be the radius at which we want to place the knob, then we want:

$$(r \sin\phi_i)^2 + (r - r\cos\phi_i + D - \frac{1}{2}\Delta_{ij})^2 = D^2$$

where Δ_{ij} is the difference in length between the parametrized edge and the original edge, and $D = \|f(\vec{u}_i) - f(\vec{u}_j)\| - r_i - r_j$, where f denotes that they are the corresponding vertices in the original mesh. We only take half of the difference, because the neighboring auxetic unit is going to take care of the second half. Thus, solving for r, we get:

$$2(1 - \cos\phi_i)r^2 + 2G(1 - \cos\phi_i)r + G^2 - D^2 = 0$$

where $G = D - \frac{1}{2}\Delta_{ij}$. If r protrudes out of the unit, then we cap it to R_i. Capping shouldn't happen frequently, since our previously-discussed parametrization aims to be the most optimal. Also, we will actually cap it smaller, because we need to leave space for the rods between auxetic units.

Now, we have all the information needed to produce a printable mesh!

2 Software Details

Additional steps:

1. Convert OBJ to graph G(V, E).
2. Chain the boundary vertices into a loop and place them in a square shape.
3. Apply Cramer's rule to solve the linear system.

References

[1] Hormann Kai, Levy Bruno, and Sheffer Alla. *Mesh Parametrization: Theory and Practice*. URL: `https://www.inf.usi.ch/hormann/parameterization/CourseNotes.pdf`. (accessed: 02.09.2025).

[2] Tomohiro Tachi. *Architectural Origami*. URL: `https://courses.csail.mit.edu/6.849/fall10/lectures/L23_images.pdf`. (accessed: 02.09.2025).