GLUING VERTEX UNFOLDINGS OF TRIANGULATED MESHES

BRITTANY TERESE FASY AND DAVID MILLMAN

Abstract. We present a system for creating an edge-unfolding of a triangulated simplicial manifold from a vertex-unfolding by gluing edges, while maintaining two rigid geometries. We implemented a known vertex-unfolding algorithm as well as several gluing techniques which we created. The entire project consisted of a total of $11,280$ lines of code.

1. Introduction

The polyhedral unfolding problem can be described as follows: Can a polyhedra in $\mathbb{R}^3$ be unfolded into a simple polygon in $\mathbb{R}^2$? The problem is easily stated for unfolding a polyhedra in 3-dimensional space to a polygon in 2-dimensional space, but the ideas can be extended to higher dimensions. In this paper, we present and contrast several options for edge *gluing* to re-connect the disconnected interiors created by a vertex unfolding.

2. Vertex Unfolding

We focus on unfolding a simplicial orientable 2-manifold M with n triangular facets. For example, we will unfold a polyhedra in 3-space. We will now briefly describe the topological vertex unfolding algorithm presented in [4], and the modifications that we made for our implementation.

The first step of the algorithm is to create a tree graph τ where the nodes store the faces of M, and two nodes are connected if the two corresponding faces share an edge. We constructed this tree by choosing a starting face and using a depth first search. Since we began with a 2-manifold M, we know that all faces of M will be in τ.

The next step in the algorithm is to create a scaffold s, a graph where the nodes are the faces and vertices of M, each face-node has degree two and all vertex-nodes (except potentially two) have an even degree. The algorithm for computing the scaffold is given in the proof of Lemmas 3 and 4 of [4]. We have added the condition that the edges of the scaffold are directed. In doing so, we must account for the orientation of the edges when creating and connecting scaffold paths and cycles. In the end, we have implicitly created a (non-crossing) Euler walk while constructing s. The resulting walk will start by going into the face at the beginning of the path (or the root of Υ if such a face does not exist), and will end by leaving the face at the end of the path (or the face previous to the root of Υ).

The vertex unfolding is equivalent to a path p on the surface of the the original mesh M such that: p only visits each face once, and p enters and leaves a face through a vertex (which can be visited any number of times). The output of this algorithm with be an alternating list of vertex and face ids:

$$L = \{v_0, f_0, v_1, \ldots, f_n, v_{n+1}\}$$

that specifies how the triangles will be laid out. We will say that two faces f, g are *adjacent* on L if there is only one vertex v_i between f and g in L. This path is a simple path if we consider parts of the path that meet at a vertex to be non-intersecting since they do not cross at the vertex.

3. Flattening the Unfolding

The above algorithm describes how to toplogically unfold a polyhedra. Next, we describe the geometry of this unfolding which projects the tree manifold Υ onto $\mathbb{R}^2$ using the triangle congruency preserving algorithm described below.

Let $\Gamma = \{\gamma_0, \ldots, \gamma_n\}$ be the result of laying out all n triangles as specified by L. In order to correctly construct the set of triangles in the plane after unfolding, we must preserve the geometry for each triangle, f_i of the original manifold M such that γ_i is congruent to f_i. Suppose we have laid out the first i facets: $f_0, \ldots, f_{i-1}$, and would like to add f_i to the triangle strip. We know that $f_i = t$ for some $t \in M$, and let p_i, q_i, r_i be the vertices of f_i. Without loss of generality, assume that $v_i = p_i$ and $v_{i+1} = q_i$ where v_i and v_{i+1} are the vertices directly before and after f_i in L. We seek to find the vertices p'_i, q'_i, r'_i of γ_i, the triangle corresponding to f_i in Γ. We first note that vertex v_i was placed when laying out triangle γ_{i-1} (or was predetermined if $i = 0$), thus $p'_i = q'_{i-1}$.

Now, we wish to lay the triangle γ_i in its own vertical slab. To do so, we let q'_i be the right most point of γ_i, and compute the new position of r'_i by orienting the triangle in the plane. We are now are able to rotate the triangle around the point p'_i until we find the layout desired. To remove this degree of freedom, we add the following constraint on the x-coordinates of the vertices: $2r'_i.x = q'_i.x - p'_i.x$. In other words, the x-coordinate of r'_i will be half way between the x-coordinates of q'_i and r'_i.

The following corollary follows from this restriction:

Corollary 3.1. *The geometric layout produced by this algorithm is unique for a given vertex unfolding.*

4. Edge Gluing Algorithms

In a vertex-unfolding, the resulting planar polygon is connected, but the interior may be disconnected. We now consider different methods for re-connecting the components with disconnected interiors.

As mentioned above, the output of the vertex unfolding algorithm is a list L that specifies how the triangles will be laid out. Here, we note that we may not assume that f_i and f_{i+1} share an edge in the original manifold M, however:

Lemma 4.1. *In the vertex unfolding L, there exists at least one vertex v_i such that f_i and f_{i+1} share an edge.*

Proof. Suppose, by contradiction, that all pairs of faces f, g that share an edge in M, f and g are not adjacent on L. Without loss of generality, suppose that f appears before g in L. We now define the tight path: $L_{fg} = \{f, v_k, ..., v_j, g\}$, which is a path contained in L that starts at f and ends at g. We will define C_{fg} to be L_{fg} with v added to the beginning and the, where v is a shared vertex of f, g. If the face immediately following g in L is inside C_{fg}, then create the cycle $C_{f'g'}$ by letting $g' = f_n$ and f' be any face that shares an edge with g'. See Figure 1 for an illustration.

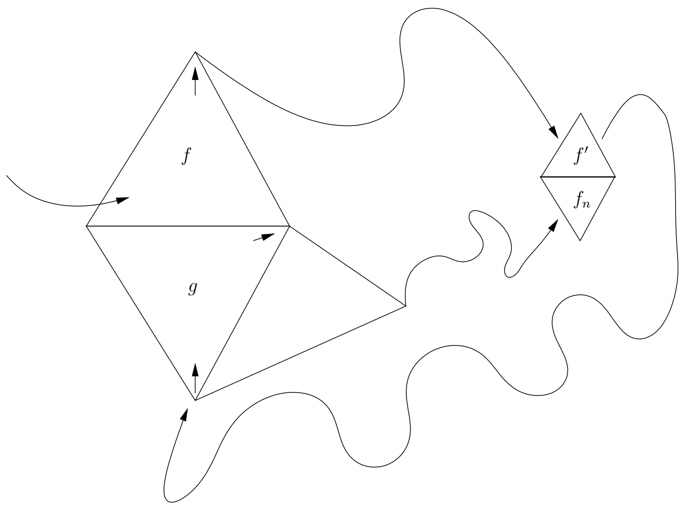


Figure 1: Creating the Cycle C_{fg}

Now, let h be a triangle on or inside the cycle C_{fg}, then we know that h is after f in L since L is a simple path through the faces of M. If there were no such triangles, then that would mean that $L_{fg} = \{f, v_k, g\}$. And so, f, g would be adjacent on L. Similarly, if there was only one such triangle h, then $L_{fg} = \{f, v_k, h, v_j, g\}$. If v_k is not a vertex of g, then the triangle defined by the vertices v_k, v_j, v, where v is a shared vertex of f, g must exist interior to the cycle. Otherwise, f, h and g, h are two pairs of adjacent faces in L.

And so, we can assume that there are at least two triangles on or inside the cycle C_{fg}. Thus, $L_{fg} = \{f, v_k, h_1, v_{k+1}, h_2, .., v_j, g\}$. Since $v_k \neq v_{k+1}$, we know that at least one of the faces that shares an edge with h_1 must be in L_{fg}. We will call this face h_i. Again, we have found: $L_{h_1,h_i} = \{h_1, v_{k+1}, \ldots, h_i\}$. From this, we can find C_{h_1,h_i}, which means that we have a set of faces contained in a simple cycle such that no faces are adjacent in C_{h_1,h_i} and hence in L. If we continue to find smaller cycles in this manner, we will eventually have one triangle other than f, g in the cycle, which would mean that triangle is adjacent to its neighbor, which is a contradiction. □

We call two faces that share an edge in M, and appear in L with only one vertex in between a potential gluing. In practice, we have seen that the number k of potential gluings of adjacent faces in p is actually much more than one. In one model, we have 54 of the 99 adjacent faces in p were potential gluings. In fact, at least half of all adjacent faces were potential gluings in all models that we used.

Below, we describe edge-gluing techniques for vertex unfolding. These algorithms try to connect faces f_i, f_j of L by re-connecting their shared edge (if such an edge exists). If this gluing causes another piece of the unfolded manifold to overlap, we will not re-attach this edge. Before describing three different algorithms for re-connecting edges we define some terminology.

At each vertex v of L, v separates the scaffolding into two arms. The *left arm* is defined to be all faces that are before v in L. Likewise, the *right arm* is all faces that appear after v in L. The length of an arm is the number of triangle faces that compose the arm.

4.1. **Validating a Glue-Step.** In the algorithms, we choose a vertex v_i in the scaffold L and attempt to re-attach the faces f_{i-1} and f_i. Lemma 4.1 shows that at least one such vertex exits. We will call a vertex v_i a potential gluing vertex if f_{i-1} and f_i share an edge in M.

Lemma 4.2. *If v_i is a potential gluing vertex, then the faces can be rotated about v_i until the two edges align without intersecting f_{i-1} with f_i.*

Proof. We first note that we started with an orientable 2-manifold. Thus, doing DFS on the exterior will find all exterior faces. And so, when the triangles are laid out, all orientations align. Second, we note that since all faces are triangles, the faces are convex.

These two facts combine to show that the gluing of these two triangles is feasible. □

The lemma above says that the two faces can be glued without f_{i-1} intersecting f_i. Although the two triangles can now be attached without intersection, this may cause collisions in the left and right arms of v_i. Thus, we will explore how this gluing would affect the remainder of the manifold.

Let Δ_l be the left arm at vertex v_i, and Δ_r be the right arm at v_i. For simplicity of explanation, we will assume that Δ_l will not change; therefore, we will glue the edge e_i by rotating Δ_r about v_i until the shared edge aligns. Then, we check to see if any triangle in Δ_l intersects any triangle in Δ_r. We have explored several options for this collision detection, including testing to see if the convex hull of Δ_l intersects the convex hull of Δ_r. We have decided that checking all $O(k^2)$ possible triangle collisions was the best option as it is correct and complete. We note that at best, this algorithm has $O(k^2)$ triangle collisions at each step for each vertex in L. Since k is rather small (usually less than 500), the total time is not significant.

We find the new vertices that would form based on rotating about v_i as described and test to see if there exists a vertex in Δ_r that lies within any triangle of Δ_1. If this check fails, then these edges cannot be glued back together with the present geometries on Δ_l and Δ_l. Otherwise, we validate this potential gluing, and update the unfolding Γ.

4.2. **Algorithms.** The vertices of L are the locations in the unfolding where two components of the unfolding are adjacent, but have disconnected interiors. We have explored several options for re-connecting these components while preserving the vertex connections specified by L.

- *Roll-up* We glue two faces of Γ by connecting the faces in order from v_0 to v_n, where v_i is the i^{th} vertex of L. For each v_i, the faces f_i and f_{i-1} are validated for gluing as specified in Section 4. If a gluing exists and does not cause a collision, $\gamma_i\gamma_{i+1}$ is glued, and the shorter arm is rotated accordingly.

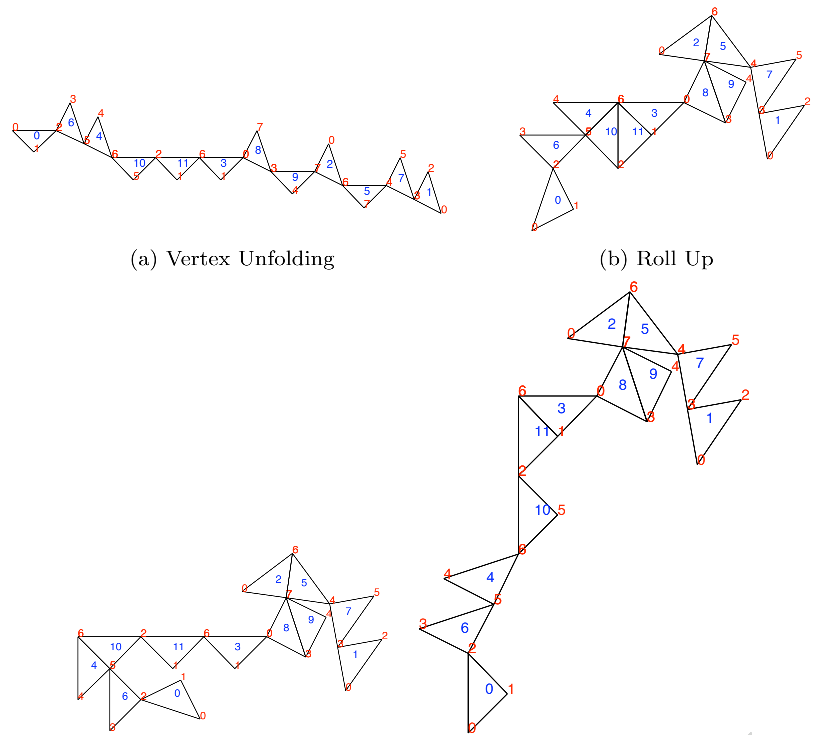
- *Binary Splitting* We will attempt reconnecting the unfolding at vertex $v_{n/2}$ and then recurse on the two subsets of vertices : $v_1 \dots v_{n/2}$ and $v_{n/2} \dots v_n$. Alternatively, recursing on the two subsets could precede the reconnection attempt at vertex $v_{n/2}$.
- *Randomized Algorithm* We will choose a random ordering on the vertices v_1 through v_n. Then, we will attempt connecting the unfolding in the order specified by this ordering.

Above, we have described the three basic algorithms with one-pass. For each algorithm, we could make a second (or third, etc.) pass to attempt reconnecting at vertices that separate two components in Γ.

Options for re-connecting the components with disconnected components are discussed in Section 7.

5. Analysis

Of the three techniques described above, we have seen that the Roll-Up performed the best, while Randomized performed the least number of gluings. Figure 2 shows the unfoldings resulting from the three methods on a triangulated cube.



(a) Vertex Unfolding

(b) Roll Up

(c) Forward Binary

(d) Random

Figure 2: Unfoldings

Table 1 and Table 4 show the statistical results of vertex-unfolding and edge-gluing for three manifolds using the Roll-up and the backward binary splits techniques. Tables 2 and 3 show the results for the other two techniques. The numbers for the randomized algorithm are averaged over several trials. In these tables,

Table 1: Roll-Up Experimental Results

	$n-1$	k	m	g	c
Tetrahedra	3	3	1	3	100
Cube	11	5	1	5	50
Double Torus	99	57	1	17	41
Duck	499	260	1	155	31

Table 2: Forward Binary Splitting Experimental Results

	n	k	m	g	c
Tetrahedra	3	3	1	2	50
Cube	11	5	1	3	33
Double Torus	99	57	1	33	34
Duck	499	260	1	131	26

Table 3: Backward Binary Splitting Experimental Results

	n	k	m	g	c
Tetrahedra	3	3	1	3	100
Cube	11	5	1	3	33
Double Torus	99	57	1	35	36
Duck	499	260	1	137	27

Table 4: Randomized Experimental Results

	n	k	m	g	c
Tetrahedra	3	3	1	3	100
Cube	11	5	1	3	33
Double Torus	99	57	1	31.5	32
Duck	499	260	1	139	28

let n, k be as previously defined: n is the number of triangular facets and k is the number of potential gluings that exist in the vertex-unfolding. We use $n-1$ in the tables since there are $n-1$ pairs of adjacent faces in L. In addition, let m be the number of iterations of gluing attempts, and g be the number of gluings that were actually made. Finally, c is the percentage of an edge unfolding that was made, assuming one exists.

While we expected randomized to perform better, the structure of the result of gluing the vertex unfolding is highly affected by the ordering of gluing attempts. Thus, random gluing is not a reliable way to perform these operations. The backward binary splits consistently performed more poorly than the forward binary splits, which is also the opposite than the expected results.

In all of the above mentioned algorithms, the geometries were stiff and we visited each vertex once to see if an edge gluing were possible. We now evaluate the experimental results of a multipass algorithm. Table 5 shows the results. Multipass

Table 5: Multipass Experimental Results

	n	k	m	g	c
Tetrahedra	3	3	1	3	100
Cube	11	5	3	9	83
Double Torus	99	57	2	45	46
Duck	499	260	4	198	39

did improve the percentage of the edge unfolding that was glued, but we see that more improvement can still be made. One such option could be *plucking* a face off the end and placing it in an appropriate position. In addition, we could allow for more than one degree of freedom.

6. Conclusion

Although we limited ourselves by a stiff geometry on the arms, we did see that the algorithms performed rather well, with connecting more than twenty-five percent of an edge-unfolding with one pass, and the multi-pass algorithm improved upon those percentages.

7. Future Direction

The algorithms described above are limited in two ways. First, we only allow for gluing of edges of currently adjacent faces. Also, we would like to explore options that break a decision that was previously made. For example, we could pluck the first or last face off and attempt gluing it to an edge that is not on an adjacent face in Γ.

Second, we reject a potential gluing at vertex i if the current geometries to the left of and to the right of vertex i would overlap. Perhaps a slight adjustment of the geometries of the arms would allow for more gluings.

There are also two theoretical results that we are currently investigating. Lemma 4.1 lends itself to the possibility that we can construct a vertex unfolding for an arbitrary manifold that would guarantee $\log(n)$ potential gluings at vertices, we would like to investigate the existence of such an algorithm. Also, Lemma 4.1 does not guarantee that the potential gluing that we have will be a valid gluing, but we would like find necessary and sufficient conditions for a gluing to exist.

Finally, we would like to explore how adding tabs to the faces would affect the resulting unfolding. Since we would have to leave room near an edge for a reasonable-sized tab, adding this feature could invalidate certain gluings.

References

[1] T. Biedi, E. Demaine, M. Demaine, A. Lubiw, M. Overmars, J. O'Rourke, S. Robbins, and S. Whitesides. Unfolding Some Classes of Orthogonal Polyhedra. in Proc. 10th Canadian Conf. on Computational Geometry, Aug. 1998, pp. 7071.

[2] T. A. Brown, Hamiltonian Paths on Convex Polyhedra. unpublished note, the RAND Corporation, August, 1960.

[3] E. D. Demaine, Folding and Unfolding Linkages, Paper, and Polyhedra. In Revised Papers From the Japanese Conference on Discrete and Computational Geometry (November 22 - 25, 2000). J. Akiyama, M. Kano, and M. Urabe, Eds. Lecture Notes In Computer Science, vol. 2098. Springer-Verlag, London, 2001, pp. 113-124.

[4] E. Demaine, D. Eppstein, J. Erickson, G. Hart, and J. O'Rourke, Vertex Unfoldings of Simplicial Manifolds. SoCG 2002.