Developing Piecewise Flat Manifolds

Joe Crouch, Mark Doss, Taylor Johnson, Kira Kiviat, Justin Lanier

July 30, 2010

Abstract

Given a source point on a piecewise flat manifold and a vector in its tangent space, the exponential map returns the point in the manifold determined by the geodesic corresponding to the vector. The inverse relation of this map, the development, is one-to-many and fills the entire tangent space with a panoramic view of the manifold from the source point. In this paper, we propose an algorithm for constructing a model of the development via source unfolding. We also describe the implementation of our algorithm in JAVA for two- and three-dimensional piecewise flat manifolds and discuss some resulting images.

1 Background

1.1 Piecewise Flat Manifolds

An n-dimensional manifold is a space that is locally homeomorphic to $\mathbb{R}^n$. In our work, we deal exclusively with **piecewise flat manifolds (PFM)** that, almost everywhere, have flat Euclidean geometry; these manifolds are analogous to polytopes. Restricting to this class significantly simplifies the task of writing visualization algorithms. Also, the resulting algorithms are computationally less intensive. Thus, by using such manifolds to "approximate" smooth manifolds, we are able to retain important global topological information while making the implementation of visualization algorithms more tractable.

In order to develop a more precise definition, we draw from the work of Miller and Pak[3]. We need to introduce some specific vocabulary before we are able to describe PFM's precisely.

Definition 1.1. Given $d \geq 2$, a d**-facet**, or d-dimensional facet, is a finite intersection of closed half-spaces in $\mathbb{R}^d$ with finite, non-zero volume. For example:

- a 0-*facet* will be a point
- a 1-*facet* will be a line segment

Remark 1.2. The astute reader may have realized that a d-facet can be defined inductively as the convex hull of a finite collection of $(d-1)$-facets. This is analogous to how simplices are constructed.

Now that we have an understanding of facets, we can discuss how they are pieced together to create a manifold.

Definition 1.3. An **n-dimensional piecewise flat manifold** or **PFM** is a finite collection of n-facets, glued together by isometries between pairs of codimension 1 facets, known as **ridges** such that:

- Each ridge neighbors exactly two facets
- $\forall\, k$ The link of a codimension k facet is "nice" (i.e. the link is homotopy equivalent to a $(k-1)$-sphere).

Remark 1.4. Through the course of this paper, when we use the term **facet** without specifying a dimension, it will refer to an n-facet (with respect to an n-dimensional PFM).

Remark 1.5. A ridge in an n-dimensional PFM is an $(n-1)$-facet.

Remark 1.6. We emphasize that this gluing is abstract in the sense that the resulting PFM need not be embeddable in $\mathbb{R}^{n+1}$.

1.2 Combinatorics and Coordinate Charts

We know that one PFM condition states that every ridge is incident to exactly two faces. We can extend this concept to all d-facets to get a clear picture of exactly how the elements of a PFM are connected.

Definition 1.7. The **combinatorics** of an n-dimensional PFM is a listing of local d-facets, $(d \leq n)$ $\forall$ d-facets.

The idea of local facets is intuitively defined in a case by case manner. For example, local vertices are ones connected by an edge. It is a simple, practical way of knowing everything that is connected to a given d-facet.

When creating an n-dimensional PFM M, we encode combinatorial data by describing how the facets in M are connected. In order to study geometric qualities of the PFM, we can assign lengths to each 1-facet in such a way that no facet becomes degenerate. This is a complete, combinatorial description of PFMs.

However, it is possible to describe a given n-dimensional PFM M by coordinate charts and transition maps just as one would a smooth manifold. Note that we are working with a manifold composed of **closed** facets, and, logically, we wish to apply coordinate charts to entire facets. So, it is essential to discuss how these charts are to be defined.

Definition 1.8. A **coordinate chart** is map $\phi : F \to F'$ Where F is an n-facet of an n-dimensional PFM and F' is a copy of $F \subset \mathbb{R}^n$.

Consider two n-facets F_1 and F_2, both neighboring a common ridge I. Because all n-facets were constructed as intersections of half spaces in $\mathbb{R}^n$, we can easily find coordinate charts $\phi_1 : F_1 \to \mathbb{R}^d$ and $\phi_2 : F_2 \to \mathbb{R}^d$. Then we introduce the transition map $\psi_{12} = \phi_1 \circ \phi_2^{-1} : I \to I$. This map can be represented by an affine transformation, enabling the F_2 to be connected to F_1 along the shared ridge, I.

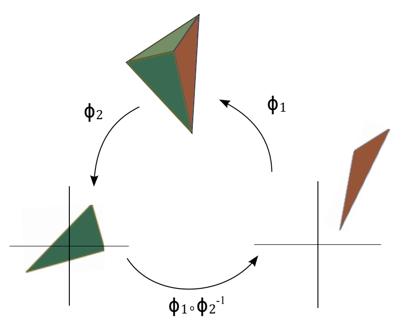


Figure 1: The connection between coordinate charts and a transition map.

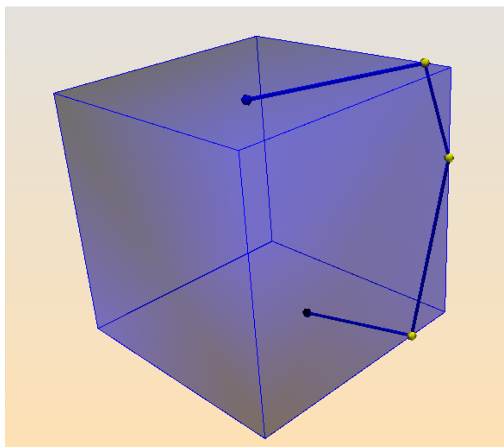
1.3 The Visual Field

Now that we have defined piecewise flat manifolds, we define our notion of a visual field in terms of geodesics, tangent spaces, the exponential map, the development, and frustums. We adopt much of our terminology from [3].

Definition 1.9. A **flat point** in an n-manifold is one that has a neighborhood isometric to $\mathbb{R}^n$. A **warped point** is any point that is not flat.

Remark 1.10. Warped points can only occur on facets of codimension 2 or more, but not every point on such a facet must be warped.

Definition 1.11. A **geodesic** is a locally shortest path between two points.



(a) A geodesic on a cube

The extension of a geodesic along a ray can be calculated by moving along a Euclidean straight line within a facet and within subsequent facets, while transitions between successive facets through ridges are handled by transition maps. Note that as the geodesics we consider here are locally shortest (rather than straightest) geodesics, any geodesic that intersects a warped point cannot be uniquely extended beyond it (see [5] for details). Each geodesic ray emanating from a point is a **line of sight** that determines what is seen in the manifold in a given direction.

For more information about geodesics on piecewise flat manifolds, see the work of Stone [6].

Definition 1.12. Given a flat point on an n-dimensional piecewise flat manifold, the point's **tangent space** is the n-dimensional vector space that contains the facet containing the given point and has the point at its origin.

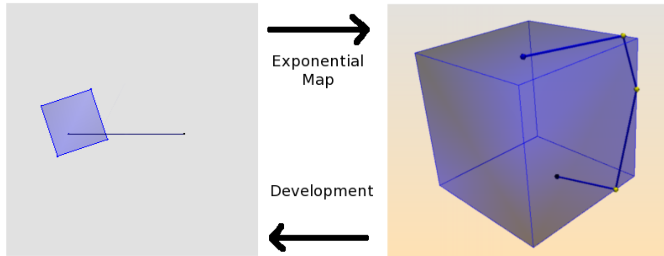
Remark 1.13. For a flat point that lies on a ridge, either adjoining facet may be taken for the purpose of determining the point's tangent space.

Given a point and its tangent space, we can consider the tangent space as the **visual field** of the point. In this context the point is referred to as the **source point**. The visual field is composed of all vectors emanating from the source point within the tangent space, as these correspond to to the lines of sight in the manifold. This correspondence is specified by the **exponential map**.

Definition 1.14. The map that takes each vector in the tangent space of a point to the geodesic in the manifold with the initial direction of the vector and its length is the **exponential map**.

Some vectors in the tangent will have no corresponding geodesic, since such a geodesic would pass through a warped point. Any such geodesic ray is called a **blocked line of sight**.

Definition 1.15. The inverse of this map, which is one-to-many in general, is a relation called the **development**.



The development populates the tangent space with points from the manifold, thus providing content to be seen in the visual field. Individual lines of sight and points along them can be bundled together into natural partitions called **frustums**. Frustums are used to divide up and manage the visual space at a point on a piecewise flat manifold. Imagine standing within a wire-frame box. You can look out and see all of space, but your field of vision is broken up into six pieces, one for each "window" you look through.

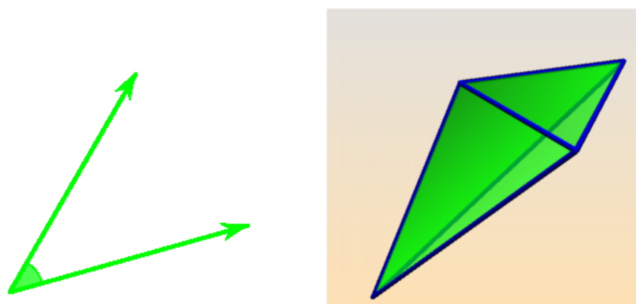


Figure 2: Examples of two- and three-dimensional frustums

We give two definitions of a frustum.

Definition 1.16. A **frustum** is a collection of vectors that constitute what can be seen over a series of ridges, i.e., $\mathcal{F}(R_0, R_1, \cdots, R_n) = \{\vec{v} \in T_{x_0}F_0 \mid exp(k\vec{v}),\ k > 0$, has a ridge crossing sequence that begins with $R_0, R_1, \cdots, R_n\}$

A second definition is useful in more computational settings.

Definition 1.17. A **frustum** is a region in n-space defined by the intersection half-spaces, all of whose boundaries pass through the source point.

The way in which these definitions differ is that the former has the feel of binding many vectors by a shared property, while the latter has the feel of taking a chunk of visual space along with all of the vectors that fall within it. That these two definitions are equivalent is key to showing that our **source unfolding algorithm** is equal to the development.

Remark 1.18. Note the following:

- A frustum defined over a sequence of convex facets is convex.
- The intersection of two frustums of the same dimension with the same source point is another frustum.
- The intersection of an n-dimensional frustum and a convex n-facet is another convex n-facet.

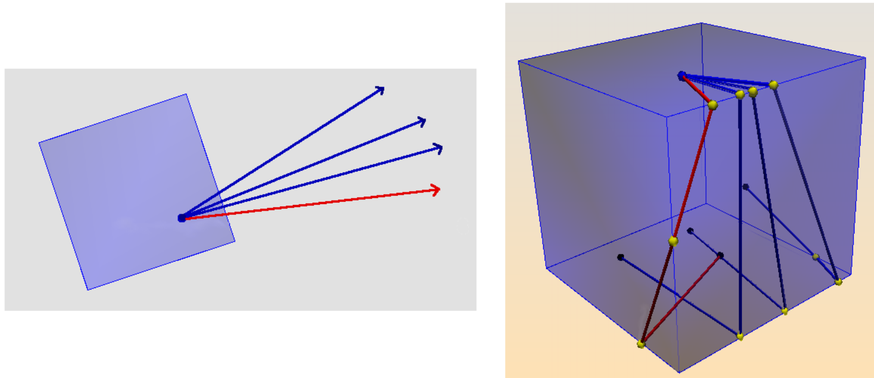


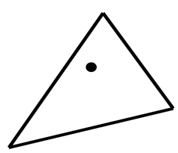
Figure 3: All of the vectors in the diagram are in the frustum defined by the first edge that they cross, but the red vector is in a different frustum from the blue ones when longer edge sequences are considered.

2 Algorithm

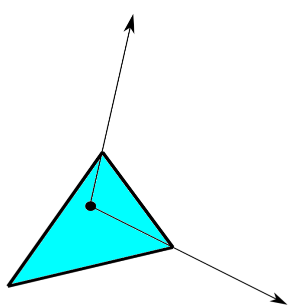
2.1 Overview with Example

The unfolding algorithm is a recursive method which takes in three parameters: a facet, a frustum, and an affine transformation. Initially a point contained in one of the facets is selected as the source point, and the recursive method is called on that facet along with the affine transformation taking the source point to the origin, and the "full" frustum, which is the whole plane.

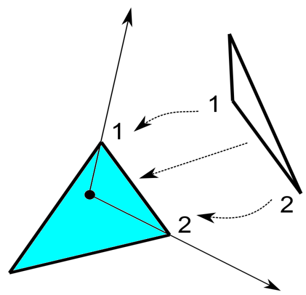
The following is an example in two dimensions to illustrate the source unfolding algorithm:



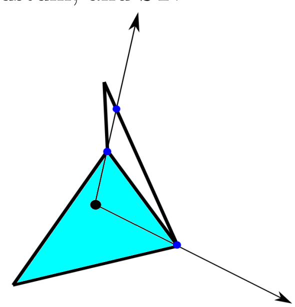
1. Choose a source point in one of the faces, F_0, and find the affine transformation S that takes this point to the origin. Call the recursive method with F_0, the "full" frustum, and S.



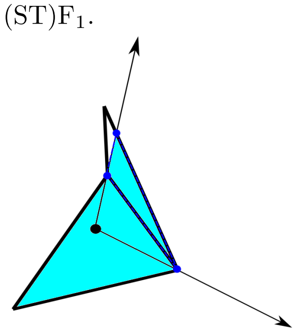
2. Intersect SF_0 with the "full" frustum, and add the result (all of SF_0) to the unfolding. Choose an edge e on F_0 and create a new frustum from vectors through its endpoints.



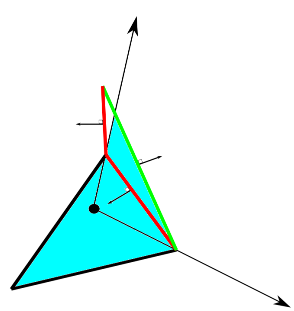
3. Look up the face F_1 connected to F_0 along e, and the affine transformation T to place it in the tangent space of the source point. Call the recursive method again with F_1, the new frustum, and ST.



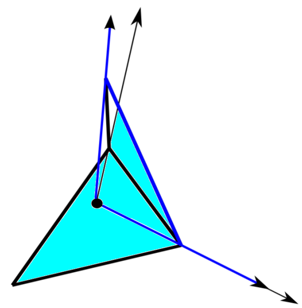
4. In order to clip F_1 with the frustum, first find the vertices of the intersection between the frustum and $(ST)F_1$.



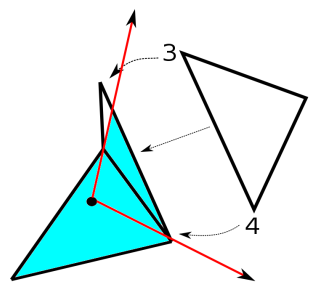
5. Form a new face from the convex hull of these points, and add this "clipped" face to the unfolding.



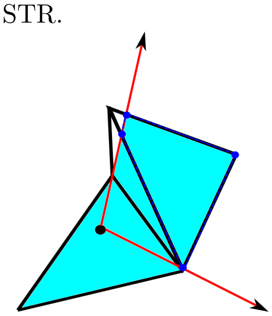
6. For each edge on F_1, take the dot product of its outward-pointing normal with the frustum ray that passes through an endpoint. If this product is negative, discard the edge.



7. Begin developing across a good edge by creating a new frustum through its endpoints.



8. Intersect this new frustum with the old, and find the face F_2 adjacent across this edge along with the affine transformation R placing it next to TF_1 in the tangent space of the source point. Call the recursive method with F_2, the intersected frustum, and STR.



9. Apply the given transformation, STR, to this face and clip it with the frustum. Continue as from step 6.

The method is called with specified bounds, and will continue developing off of facets until one is found such that one of its transformed vertices is outside of these bounds. At that point, the recursive method returns.

2.2 Subroutines

2.2.1 Affine Transformation

Affine transformations are needed to start the source unfolding process and subsequently attach facets along ridges. An affine transformation is simply a linear transformation followed by a translation, i.e., we can represent an affine transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ by the expression $T\mathbf{x} = A\mathbf{x} + \mathbf{b}$, where A is a linear transformation and $\mathbf{b}$ is a vector (for our purposes A will be an orthogonal transformation). Here is a summary of the procedure we use to find an affine transform that matches two n-dimensional simplices along corresponding ridges.

1. Begin with an n-simplex P with vertices $p_0, \ldots, p_{n+1}$, and an n-simplex Q with vertices $q_0, \ldots, q_{n+1}$ that are to be joined along the ridges specified by $p_0, \ldots, p_n$ and $q_0, \ldots, q_n$ such that p_i is identified with q_i for $i = 0, 1, \ldots, n$.

2. On P create the ridge vectors $\mathbf{u_i} = \overline{p_0 p_i}$ for $i = 1, 2, \ldots, n-1$. Similarly create ridge vectors $\mathbf{v_i}$ on Q.

3. Calculate the outwardly pointing normal $\mathbf{n_1}$ of the facet of P that we are interested in joining. Similarly calculate the outward normal $\mathbf{n_2}$ to the facet of Q we want to join.

4. Construct the matrices U and V, noting the minus sign on $\mathbf{n_2}$:

$$U = \begin{bmatrix} \mathbf{u_1} & \mathbf{u_2} & \ldots & \mathbf{u_{n-1}} & \mathbf{n_1} \end{bmatrix}, V = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \ldots & \mathbf{v_{n-1}} & -\mathbf{n_2} \end{bmatrix}$$

5. The transformation $A_0 = VU^{-1}$ takes the ordered basis $\{\mathbf{u_1}, \ldots, \mathbf{u_{n-1}}, \mathbf{n_1}\}$ to the ordered basis $\{\mathbf{v_1}, \ldots, \mathbf{v_{n-1}}, -\mathbf{n_2}\}$.

6. The net transformation translates P so that p_0 coincides with the origin, applies A_0 to P, and then translates P once more so that p_0 coincides with q_0. The final transformation will be of the form $\mathbf{x} \to A_0\mathbf{x} + \mathbf{b}$ and affine transforms P to join Q along the appropriate ridge.

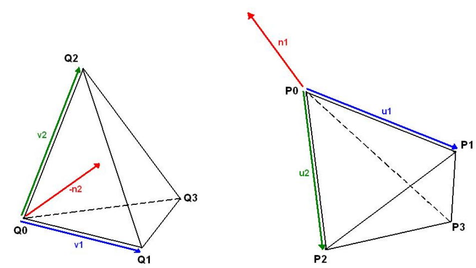


Figure 4: The ordered bases to be matched in the special case $n = 3$

2.2.2 Intersecting Frustums

In two dimensions, simply take the "right-most" left vector and the "left-most" right vector of the two frustums, and form a new frustum from these.

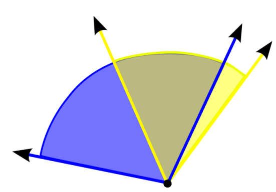


Figure 5: Intersection of a blue frustum and a yellow frustum.

In three dimensions, frustums are represented as a list of vectors representing rays in counter-clockwise order (looking at the origin through the frustums). If a single frustum is truncated with a sphere, the intersection points between the the sphere and the rays of the frustum form a convex spherical polygon. Intersecting two frustums can be thought of as intersecting the two spherical polygons formed when a sphere is placed through both frustums, where vectors correspond to vertices and edges correspond to sectors formed by adjacent vectors.

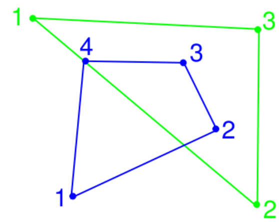


Figure 6: intersection of vectors of two frustums with a plane

1. If either frustum is contained in the other, return it.
2. Label the vectors of each in counter-clockwise order.
3. Begin on vector 1 of one frustum.
4. Add the current vector to a list if it is contained in the other frustum.
5. Check the sector between this vector and the next for intersections with the other frustum.
6. If any intersections are found:
   (a) add the intersections in order
   (b) move to the other frustum, beginning at the vector after the sector forming the last intersection
   (c) walk ccw around this frustum, adding vertices until one is found outside the first frustum
7. Move to the next vector on the first frustum and continue from step 4 until the first vertex is reached again.
8. Form a new frustum from the list of vectors.

2.2.3 Facet Clipping

In this algorithm facets are clipped by frustums, which are defined by vectors from the origin. This is done by finding (1) points of intersection between the rays of the frustum and the boundary of the facet, (2) points of intersection between the boundary of the frustum and edges of the facet, and (3) vertices of the facet that are contained inside the frustum. Then the convex hull of these points is taken, producing a new facet.

In two dimensions, this is done with a simplified version of the Graham Scan algorithm[1]. In our implementation all of the input points are guaranteed to be vertices of the hull, which reduces the complexity to $O(n)$.

For three dimensions (or above), we use an incremental algorithm, which is $O(n^2)$. Common algorithms like the gift wrapping algorithm would also be $O(n^2)$ since again all given points will be on the hull, which is the worst case. However, performance could be improved by using the divide and conquer algorithm, which is $O(n \log n)$.

3 Pseudocode

Below is pseudocode for our source unfolding algorithm. The initial section takes as input the facets of a triangulated PFM with incidence information, a source point, and the boundary of the area to develop within. It then calls Loop, the recursive subroutine.

3.1 Overall

INPUT:

facets, source, bounds

START:

Find facet F from combinatorics s.t. source $\in$ F

$\mathcal{V}$:= a full frustum

S := affine transformation taking source to origin

Loop(F, $\mathcal{V}$, S, bounds)

END

3.2 Loop Subroutine

INPUT:

facet F, frustum $\mathcal{V}_0$, affine transformation S, bounds

START:

C := convex hull of intersection points between S$\mathcal{F}$ and $\mathcal{V}_0$

add C to the unfolding

list := list of ridges of F to develop across

for each ridge r in list:

  form new frustum $\mathcal{V}_1$ from vectors to endpoints of r
  $\mathcal{V}$:= intersection of frustums $\mathcal{V}_0$ and $\mathcal{V}_1$
  F' := facet adjacent to F across r
  T := affine transformation taking F' to F
  $\mathcal{F}$:= (ST)F'
  v := vertex on F' not on r
  if (ST)v is inside bounds):
    Loop(F', $\mathcal{V}$, ST, bounds)

end for loop

END

4 Notes on Implementation

Our algorithm is based on an abstract collection of facets with combinatorial and incidence information. We create these structures using Geometric Evolutions on Computational Abstract Manifolds [GEOCAM][4]. We used jReality[2], an open source 3D package for Java, to visualize manifolds and unfoldings.

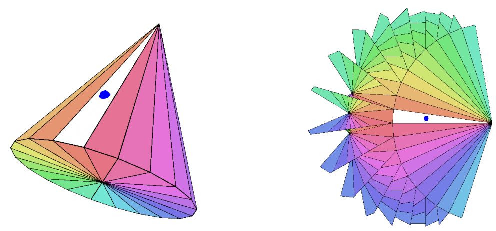


Figure 7: A 40-faced cone and its unfolding, visualized via jReality.

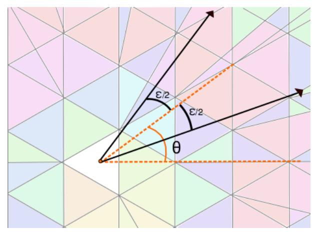
5 Results

Given abstract 2-dimensional PFM T, the source unfolding algorithmm creates a tiling of the plane. By *tiling* we mean a covering of the plane convex polygons that are disjoint except at common edges. A *vertex* of the tiling is a vertex of any one of these convex polygons.

Suppose that you select a particular point P on this tiling. Call an angle θ *blocked* if the ray emanating from P at an angle θ to the horizontal passes through a vertex of the tiling. We prove the following.

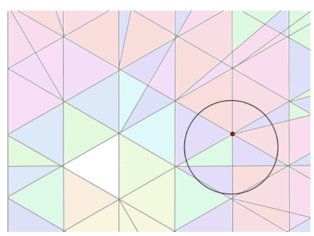
Theorem 5.1 (Blocked Angles are Dense)**.** *Starting at any point P of a tiling of the plane produced by the source unfolding algorithm, the set of blocked angles is a dense measure zero subset of S^1 (with the usual additive group structure).*

Proof. Showing that this subset is dense is equivalent to showing that given any angle θ, and $\epsilon > 0$, we can find a vertex of the tiling within the sector enclosed by the rays emanating from P at angles $\theta - \epsilon/2$ and $\theta + \epsilon/2$.

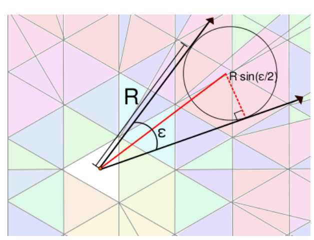


Consider a particular face F of the PFM T; recall that F is in general a convex polygon. We define diam(F) to be the supremum of the distance between any two points on F. Then set $s = \max_{F \text{ on } T} \operatorname{diam}(F)$. We note that any circle of radius $r > s$ in the tiling will contain a vertex. For the center of the circle, call it O, lies on some face F. By the definition of s, O lies within s of some vertex of F. This vertex lies within the circle of radius r centered at O.

The set of blocked angles must be countable, since there are a countable number of vertices within our tiling (since the source unfolding works inductively). Any countable subset of S^1 must have measure zero.



Now consider the region A bounded by the rays from P at angles $\theta - \epsilon/2$ and $\theta + \epsilon/2$ and the circle centered at P of radius R. Construct a circle C of some radius $r > s$ with center O a distance R from P along the ray at angle θ. Now pick R such that $R \geq r/\sin(\epsilon/2)$. We show that this circle will lie completely within A. This will complete the proof since the lemma then guarantees that a vertex lies within this circle.



The distance from O to a point on either of the rays is at least $R\sin(\frac{\epsilon}{2})$, since the perpendicular segment from O to the either one of the rays has length $R\sin(\frac{\epsilon}{2})$. But $r \leq R\sin(\frac{\epsilon}{2})$, so the circle lies entirely within A. Now since the circle has radius $r > s$, it contains a vertex, and thus so does A. $\square$

Notice that not every vertex in the tiling corresponds to a vertex of the original PFM. There are vertices of the tiling, which we will refer to as *ghost vertices*, that are introduced solely by the clipping procedure within the source unfolding algoritm. Any other vertex within the tiling will be called a *real vertex*.

Now imagine that we are at a point P on a PFM and we look in a given direction. Call this direction *blocked* if your line of vision traces out a geodesic along the PFM that eventually passes through a vertex *on the PFM*. When we think about the situation from the perspective of the tiling of the plane generated by the source unfolding, these blocked directions will correspond to rays emanating from P that pass through *real* vertices; that is, a ray passing through a ghost vertex need not correspond to a blocked direction. Therefore our above result does not imply that blocked directions are dense in S^1. However, the next theorem shows that the set of blocked directions is dense nevertheless.

Theorem 5.2 (Blocked Directions are Dense)**.** *The set of blocked directions are dense in* S^1*.*

Proof. Again we want to show that given any angle θ, and $\epsilon > 0$, we can find a vertex of the tiling within the frustum $\mathcal{F}$, enclosed by the rays emanating from P at angles $\theta - \epsilon/2$ and $\theta + \epsilon/2$. From the previous theorem, we can find some vertex within the interior of this frustum. If it is a real vertex, we are done. Otherwise it is a ghost vertex.

But ghost vertices are created as the intersection of a frustum and a ridge within the clip face procedure. But the boundary rays of any frustums must pass through a real vertex. This follows by induction after considering how frustums are produced within the source unfolding: each of the original frustums created by developing across ridges of the first facet have boundary rays that pass through a real vertex; and each boundary ray of any newly created frustum will either coincide with the boundary ray of some previous frustum or be created due to the presence of some real vertex.

Thus the presence of a ghost vertex within a frustum actually implies the presence of real vertex within the frustum. Thus we have a real vertex within our original frustum, completing the proof. □

6 Conjectures and Future Work

Our experience with applying our unfolding algorithm to various PFM's has led us to the following conjecture:

Conjecture 6.1. The source unfolding algorithm, thought of as a map from points on the PFM to the development plane, is identical to the abstract development map.

Remark 6.2. We have sketched an outline of a possible proof to this result, but have yet to write a formal proof.

Conjecture 6.3. Suppose you are given a facet F on a PFM T, and let P be a source point in the development plane of T produced by the source unfolding. Call a facet in the development *inherited* from F if the facet is the result of face clipping F at some point in the source unfolding. Then in a circle of radius R centered at P, let $\rho(R)$ denote the ratio of the sum of the areas of the all inherited faces of F to the total area of the circle. Then

$$\lim_{R\to\infty} \rho(R) = \frac{\text{volume of } F}{\text{surface area of } T}$$

Conjecture 6.4. A given facet has an infinite number of inherited facets in the development plane created by the source unfolding.

We close with some comments on possible directions for future work. A first step would be to finish the implementation of our algorithm for the visualization of 3-dimensional PFMs. From there we may look to implementing more sophisticated interactions with the manifold visualizations. We could simulate the placement of objects within our PFMs, model the propagation of sound waves, or model interactions between charges and force fields. Other groups interested in extending our research could look into methods for visualizing projections of higher dimensional PFMs, or improving upon the methods we have implemented for visualizing 2 and 3-dimensional PFMs. Such research, aside from being relevant to workers in the fields of discrete and computational geometry, would likely find exciting applications in coming years.

7 Acknowledgements

We would like to thank David Glickenstein and Danny Maienschein for their guidance and mentorship.

References

[1] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972.

[2] jReality. `http://www3.math.tu-berlin.de/jreality`.

[3] E. Miller and I. Pak. Metric combinatorics of convex polyhedra: cut loci and nonoverlapping unfoldings. *ArXiv Mathematics e-prints*, December 2003.

[4] Geometric Evolutions on Computational Abstract Manifolds [GEOCAM]. `http://code.google.com/p/geocam`.

[5] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 30–38, New York, NY, USA, 2006. ACM.

[6] D. Stone. Geodesics in Piecewise Linear Manifolds. *Transactions of the American Mathematical Society*, 1976.