

# Tracking Objects on a Deformable Surface using Displacement and Orientation Information

Zachary DeStefano<sup>\*1</sup>, Kyle Cutler<sup>†1</sup>, Gopi Meenakshisundaram<sup>‡1</sup>, Thomas O'Sullivan<sup>§1</sup>, Albert Cerussi<sup>¶1</sup>, and Bruce Tromberg<sup>||1</sup>

<sup>1</sup>University of California, Irvine

## 1 Mesh Traversal Algorithms

### 1.1 Following a Deformable mesh

Even after calibration, due to measurement error and deformability, when we render the path in the virtual world there will be segments above or below the virtual surface. The overall arc length of the path will still be preserved. The path was still recorded on the real surface and we thus want to have a visualization of where the path was located on that surface. We thus need a method of converting the inaccurate path in the virtual world into a path that follows the virtual surface and shows the locations on the real surface that the probe traversed.

The path being referenced is a sequence of connected segments that starts at a specific point on the mesh. The mesh is a triangular manifold mesh. We will repeat our procedure for each segment in the path thus we only need to consider a single segment as our input. This segment has a start point on the mesh as well as a direction and length. We need to make sure the projection preserves the length of the segment. We are restricted to rotating the segment along the plane that it makes with the triangle's normal.

We will let  $v$  be the vector for our input segment,  $N$  be the normal, and  $E$  be the resultant vector that is both on the plane formed by  $N$  and  $v$  and has an acute angle with  $v$ . As can be observed in figure 1, the following equation will then hold

$$E + \text{proj}_N(v) = v$$

Thus, the following equation will get us  $E$

$$E = v - \text{proj}_N(v)$$

This can be further simplified to say

$$E = v - N(v \cdot N)$$

We then find the intersection of the vector  $E$  with the edges of the current triangle. If the segment is entirely in the triangle, then we move onto the next segment starting from the endpoint of  $E$ . If the segment leaves the triangle, then we find which edge the segment intersects and repeat the projection procedure for the end part of the segment that is not in the triangle. This procedure of course relied on finding the intersection of the triangle with the segment which proved to be non-trivial.

When finding the triangle and segment intersection, we had a triangle in a 3D space as well as a segment that was supposed to be coplanar to the triangle but could be slightly off due to

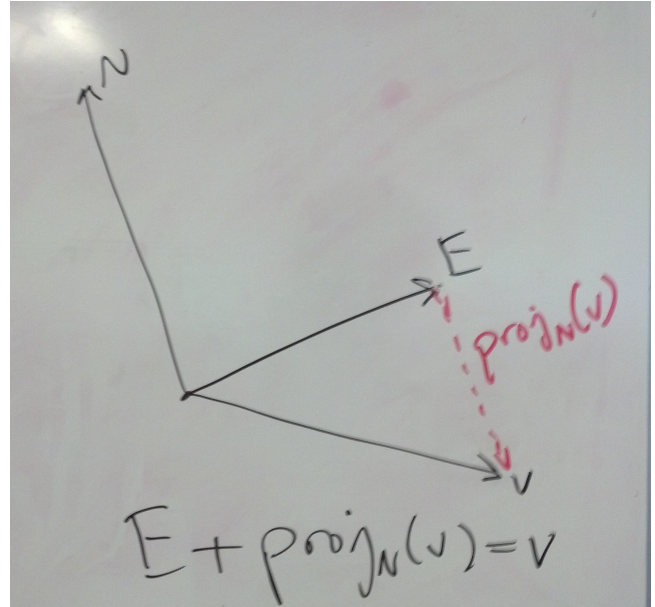


Figure 1: The segment vector, normal, and projected vector

floating-point errors with the coordinates. To find the intersection point, we converted the segment and triangle coordinates to the coordinate system where one of the vertices of the triangle is the origin and the basis vectors are the two vectors made by the segments coming off that vertex as well as the cross product of those vectors. In this coordinate system, the third coordinate should be zero. In practice, it was near zero due to floating point errors. Once in the new coordinate system, we just had to find the 2D intersection of the triangle and the segment.

The above procedure describes what we did with a single segment. With paths, we just iterated this procedure for each segment of the path. We assumed that the important aspect of each segment is its vector and not its origin point, since that is what we get from the probe. This meant that the end point of a projected segment was treated as the start point for the rest of the path when doing the loop to project the entire path onto the mesh.

### 1.2 Justification

We have a recorded path and an algorithm to transform that to the surface. Because a path is a series of short segments, we just need to justify that algorithm for a single segment. Each segment of our path represents the distance and direction traveled in a very small amount of time. We are given a start point on the surface and vector and need to calculate the path on the surface.

This tracking system is meant to be used with a system that rapidly samples the location of a probe. This means that an

<sup>\*</sup>zdestefa@uci.edu

<sup>†</sup>kbcutler58@gmail.com

<sup>‡</sup>gopi.meenakshisundaram@gmail.com

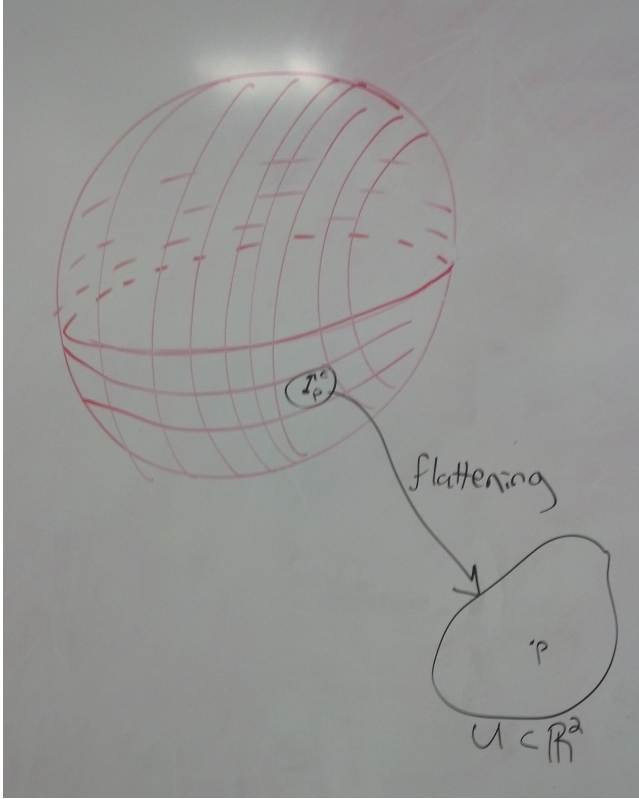
<sup>§</sup>tosullivan@uci.edu

<sup>¶</sup>acerussi@apple.com

<sup>||</sup>bjtrombe@uci.edu

individual segment is thus quite short. Additionally, the surface is generally assumed to be a low curvature. The surface we are tracking over must be a manifold. The combination of short segments and low curvature leads to the following assertion that will be used in my justification: *If  $\epsilon$  is the length of the segment, then the  $\epsilon$ -neighborhood around the start point of the segment is homeomorphic to  $R^2$ .* This is illustrated in figure 2.

Because the sampling rate was so rapid, we can assert that



**Figure 2:** The manifold with its  $\epsilon$  neighborhood at a point  $p$

between samples, a straight line on the surface was followed. We can thus assert the following: *The segment follows the geodesic path from start to end point along the surface.* We do not have the end point on the surface but when we assert an end point later, the path that was calculated will be the geodesic path from start to end point along the surface.

At a given point, the probe is either on the surface or pressing into the surface in the direction of the normal to the surface. This is because when someone is guiding the probe, they are pressing *down* on the surface meaning that the pressing is in the direction of the normal. This allows us to assert the following: *The transformed segment vector must be perpendicular to the normal.*

Because the rotation calibration step takes care of the orientation along the surface, we must transform our vector in such a way that preserves that. We thus assert that we can only rotate the vector in the plane formed by the segment vector and the normal. Rotating it in any other plane would change the orientation along the surface. The transformed vector still needs to be perpendicular to the normal, thus we will have two options for it. We must pick the option that leaves us with an acute angle between the transformed vector and original vector as picking the other one would

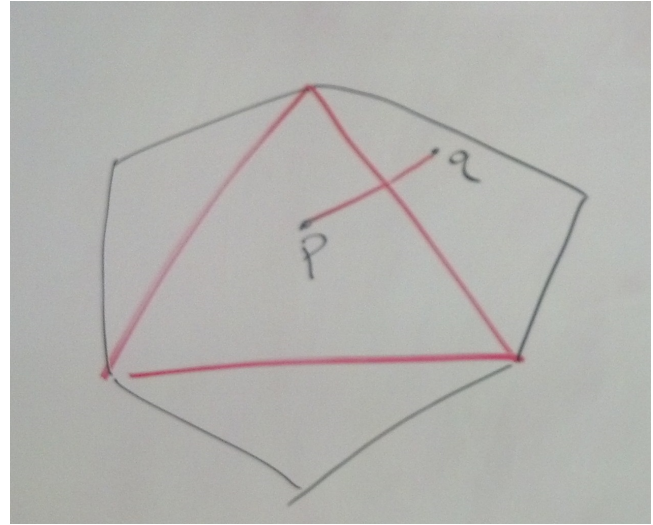
also change the orientation. We have now justified the following facts that were used to make the formula for the transformed segment vector:

*The transformed vector must be rotated along the same plane as the normal and the original vector*

*The transformed vector and original vector must form an acute angle*

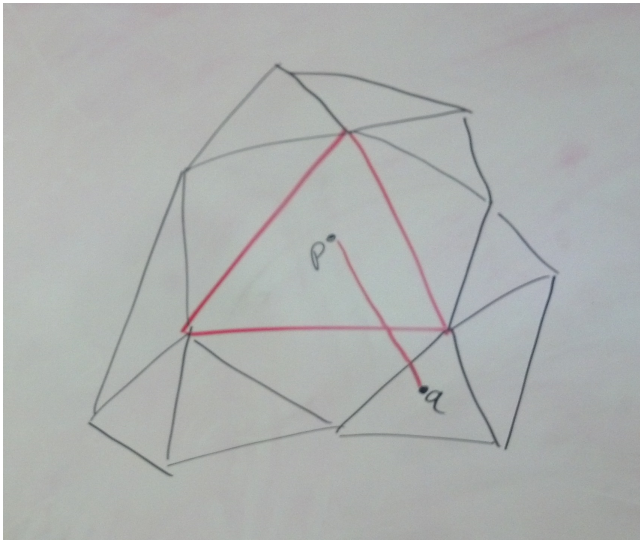
We have now justified the direction of our transformed vector. We need to preserve arc-length of the path hence the length of our transformed vector will be the same as the length of the segment vector. This could have the implication that the transformed vector goes past the triangle it is located on. If that is the case, then we cut off the segment at the edge and transform the remaining vector. We repeat this procedure until a segment lies entirely in a triangle.

We finally need to justify that this produces the geodesic in our desired direction on the surface. In the case where the entire segment can be put into its current triangle, this is trivial as we are following the straight line along that part of the surface. We thus need to justify the case where we traverse multiple triangles. Due to the homeomorphism mentioned above, we can locally flatten the triangles and avoid the gaps and overlaps that sometimes happen when there is mesh flattening. At each point in the algorithm, we are putting the path on the plane in the direction of the segment. Thus it is in the straight line direction. This means that when you flatten the triangles, the path will be a straight line in the direction we wanted. It will thus be the shortest path between the start and end points. This is illustrated in figure 3 and figure 4.



**Figure 3:** The triangle at  $p$  and its neighbors

## References



**Figure 4:** *The triangles in the neighborhood around  $p$*