Surface Reconstruction:
A Connect the Dots Approach.

Brent M. Dingle
(the person 'we' refers to in this paper)
(We all are royalty)

Texas A&M University
CPSC 645
Fall 2000

Abstract:
This paper presents a discussion and a proposed solution to the problem defined by: Given a set of arbitrary points in 3-space, known to have come from the surface of a 3-d object, reconstruct the surface of the object.
Specifically addressed are the problems in ordering the points in a consistent and useful manner.

"Why do they number these dots, they obviously connect to make a Christmas tree?"

- says the young child.

Today we begin yet another "long and boring technical paper." This paper's topic is all about surface reconstruction. The problem we will be dealing with is as follows: Assume we are given a set of points known to come from the surface of a 3-d object. We do not know how the points are connected, what order they are in, nor do we know what object they came from. Our goal is to somehow use them to reconstruct the surface of the object.

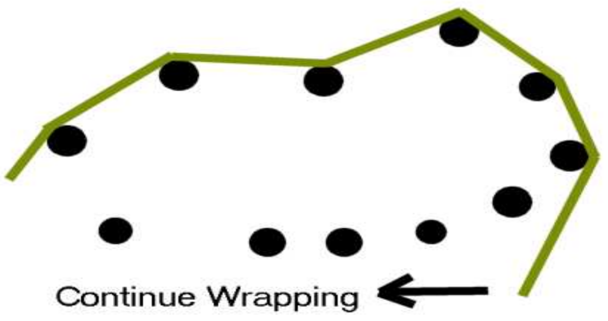
The first question that comes to mind is, "Who cares?" Well, as it happens there are two really good reasons. The first is that being able to scan a 3-d object with say a laser and immediately convert the scan into a CAD model would be a really nice thing for engineers. The second reason is that if we could put together an object based solely on points from it, then we would not need to store all the connectivity information AND we could store the points in a manner that would optimize compression techniques. These two uses alone make the world a much better place and the cause to solve the problem a noble endeavor.

So the next question of course is, "Hasn't it already been solved?" The answer is naturally, "ummmm, well, mostly." But we shall overlook that fact, and continue our not so noble goal of personal fame. However, it is worth mentioning that Nina Amenta at the University of Texas in Austin did some work on this problem using Voronoi techniques and Hugues Hoppe, currently working for Microsoft Research, applied various meshing and meta-ball techniques towards it.

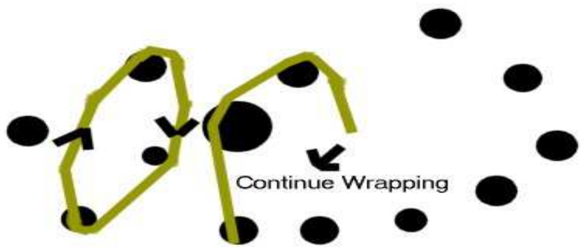
And the third and final question is, "What are we going to do?"

Well, first we need an idea. And that idea is: 3 dimensions are too difficult, and we want to make things simple. We know, through a rather large number of experimental trials and programs (located at: http://students.cs.tamu.edu/dingle/), that it is easy to run splines through ordered sets of points. And we can make closed curves using such splines (specifically a cubic spline).

So we take this idea of closed splines and try to apply it to our problem. Picture, if you will, a bunch of points floating in space. Grab some nearby duct tape and start wrapping rings of it around the points. Clearly this would give us some idea of what the object's surface originally looked like. Imagine wrapping points around parallel to the surface of this page.

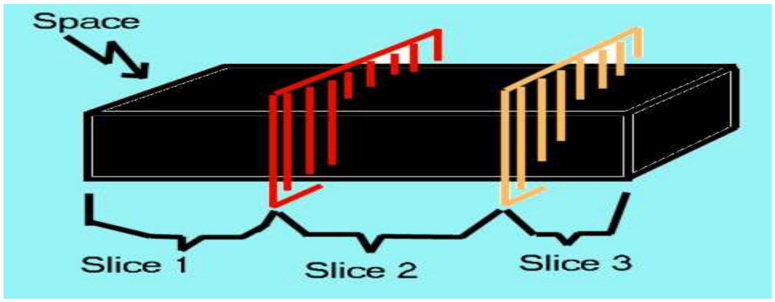


And then wrapping perpendicular to the surface of this page would give:

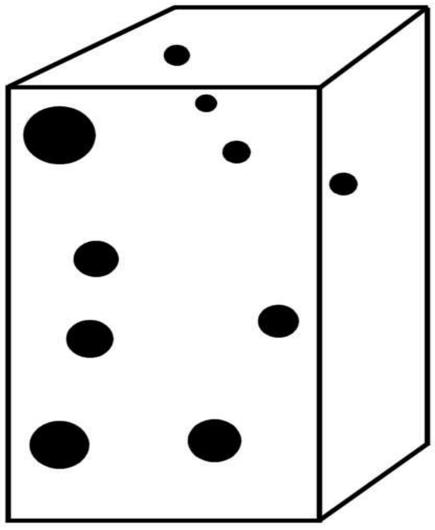


Combine both wrappings and we would have something like the surface of the object.

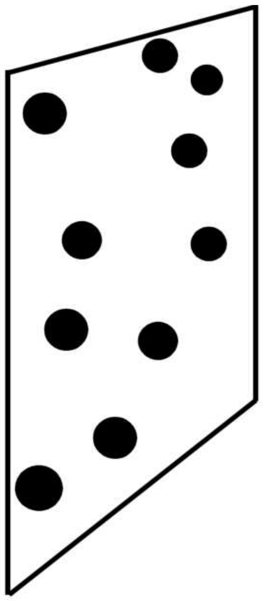
Now what does this have to do with closed splines?
Consider taking space and slicing it into narrow planes, like so:



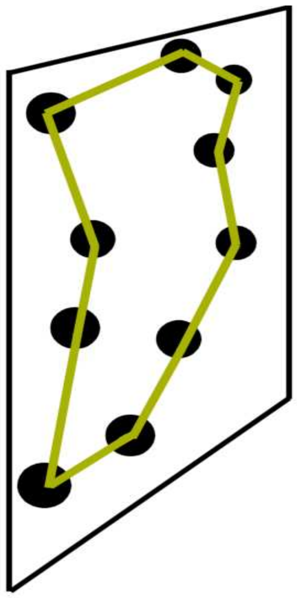
Assume slice 1 had points in it like so:



Then if we consider them coplanar (project them to a plane perpendicular to this page) we have:



And then apply our spline techniques, we would (hope to) get:

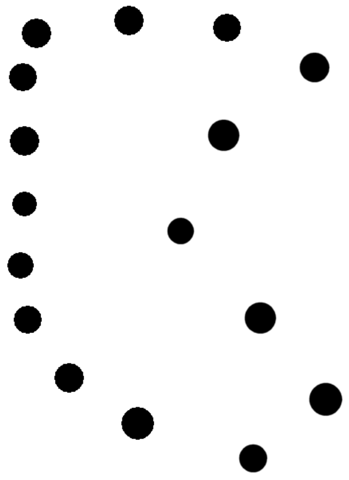


So, effectively we have wrapped duct tape around that slice using a spline. If we did this for every slice we would get nice rings around the object. If we then sliced in a direction perpendicular to the first slicing we would get nice rings perpendicular to the first set. Using these sets of rings it would be easy to establish a spline mesh for the surface.

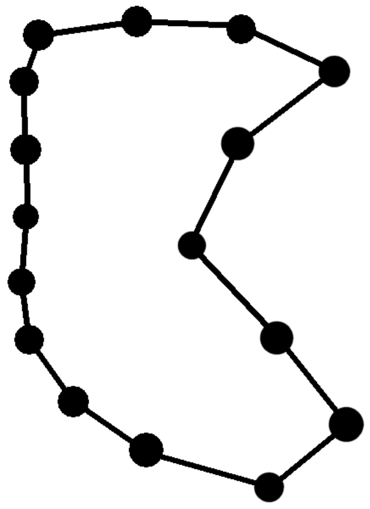
Unfortunately difficulties arise in how we determine the order of the points. All of the spline techniques require some order among the points.

This would not seem to be a difficult problem. Most convex hull methods establish some form of ordering of points, unfortunately cases such as the below knock that method out.

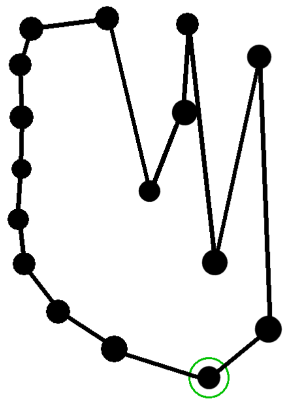
Consider:



Clearly we would connect the points as:



But if you use a convex hull method with the lower right point as anchor you get something as follows:

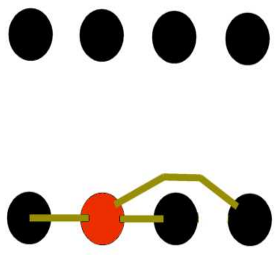


And that is no good at all. Even if there might be some form of consistency using the convex hull method the results clearly are not likely to be the ones desired.

Similar problems arise when using a center of mass idea (for example if the center of mass is outside the object then similar behavior to the above occurs). A very general medial axis method failed to work and using a random ordering of the points was right out. As an aside, the medial axis method may have merit, but we did not pursue it very far.

Instead we decided to pretend to be human and approach the problem as such. With that in mind we asked some randomly selected humans to connect non-numbered dot patterns. We observed the results and proposed questions to the humans. When asked what rules they were following to connect the dots, almost without exception, the reply was only one rule: connect the dots that are closest together.

It seemed like a good idea. So we arbitrarily attempted to connect a point to the three points that were closest to it within the entire world – we threw away the slicing idea. This had some nice results, but not exactly what we desired. A good example to consider is the below:

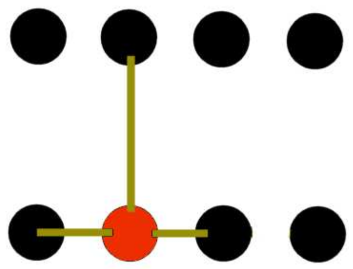


It would be nice if the three closest neighbors of the red dot included the one above it. As this would give us a nice grid effect. Alas, such is not the properties of Euclidean distance.

So something was missing. Through analysis of the human tests, and further discussion with the specimens we determined that continuity was also a consideration in connecting the dots. And here it actually proved profitable that we had attempted the slicing idea AND we threw it away.

The slicing idea was good in the sense that it gives a concept of direction from a point – i.e. the direction within the plane onto which the points are projected. However, it was too global of an idea, we needed something more local to each individual point. And thus came the idea of an angle tolerance.

Consider the previous picture, assume we still want the red dot to have 3 neighbors, but we only allow one neighbor in any given direction. Specifically we don't allow it to have 2 neighbors to the right of it. Then we would get:



Which if applied to each dot would give us the nice grid we desire. So the question became, how to get a computer to determine if two points are in the same direction from a given point? Which has an easy answer – the dot product:

$$\cos(\theta) = \frac{v \bullet w}{\|v\|\|w\|}$$

Thus let our point be P.
Let a known neighbor be Q and
let a point being checked to see if it can be a neighbor be R.
Let v = Q – P and w = R – P
From the above equation determine θ.
If it is outside some angle tolerance then add R as a neighbor.
If it is within some angle tolerance keep only Q or R, whichever is closer to P.

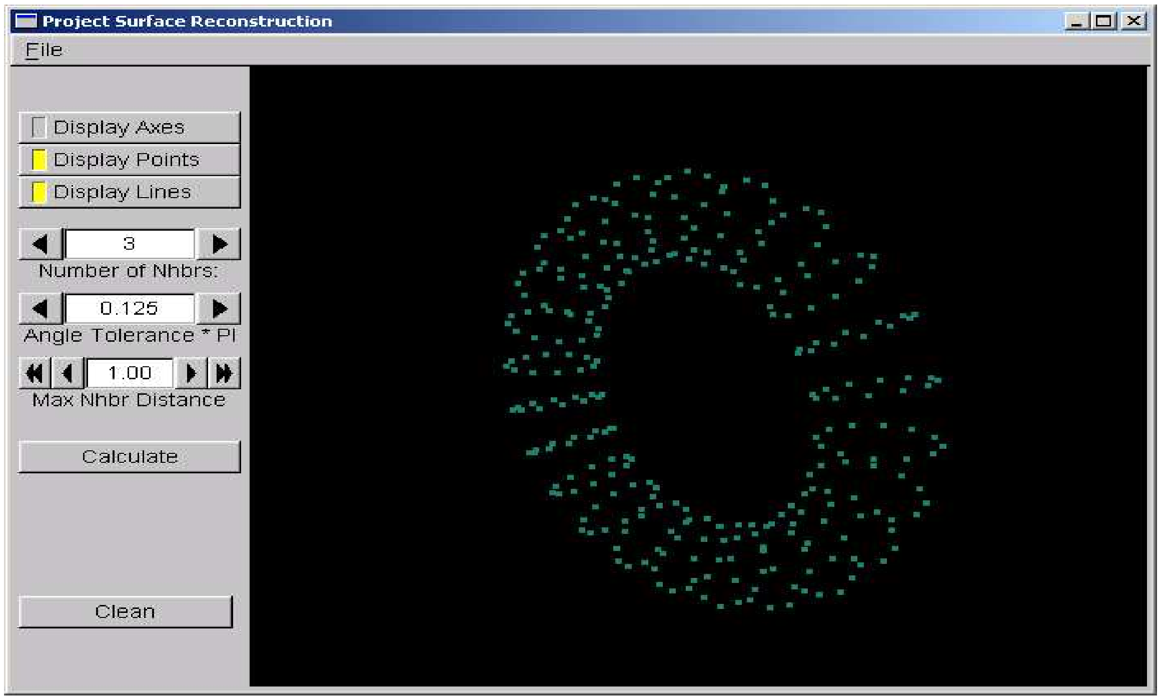
So now everything is happy, or at least we are, because when implemented the above gave us nice results. But why was getting rid of the slices a good idea? Well, we needed continuity not just in one or two directions, but in all directions. Or more specifically we wanted a point to be connected to the points that gave the 'best' continuity in as many directions as possible. Fortunately the angle techniques described above are not limited to just coplanar points. Since we only compare one neighbor against another we have a total of three points, which in and of themselves define a plane (for measuring the angle) so no projecting of points is required.

The program that implements the above idea can be found at:
http://students.cs.tamu.edu/dingle/
under the header Projects for CPSC 645.

It may seem that choosing a maximum of only three neighbors for connectivity is rather arbitrary, and it is. However, thinking in terms of the grid example above, it does have some intuitive basis. Further we did try experimenting with various other options (2 to 10 for the most part). From these experiments we decided the results we liked best were with three neighbors and a tolerance angle of 1/8 π. It should be understood the above program allows the user to choose the maximum number of neighbors as well as the tolerance angle used to determine 'in the same direction.'

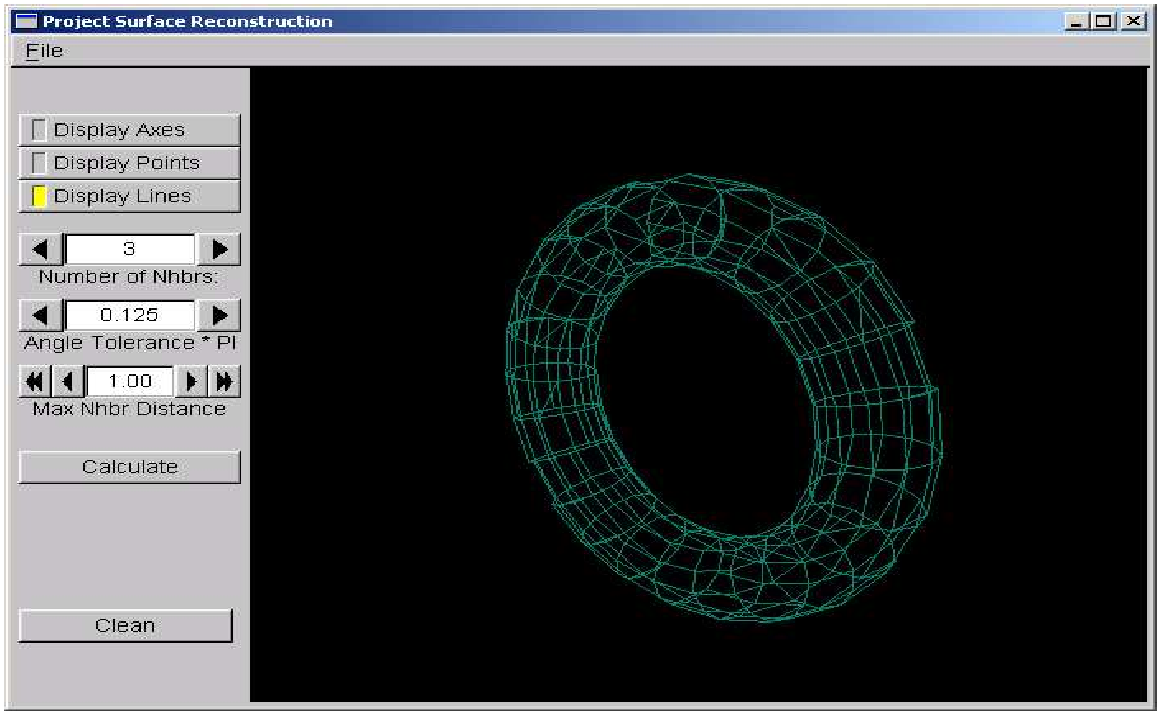
Some results of the program, i.e. the 'final' results of this project, are below:
All results were done in less than a minute for each individual case, running on a Pentium III 800 MHz, Windows 2000 machine. (Results look cooler when running the program):

A torus of points:



Number of Nhbrs: 3
Angle Tolerance * PI: 0.125
Max Nhbr Distance: 1.00

The torus of points connected:



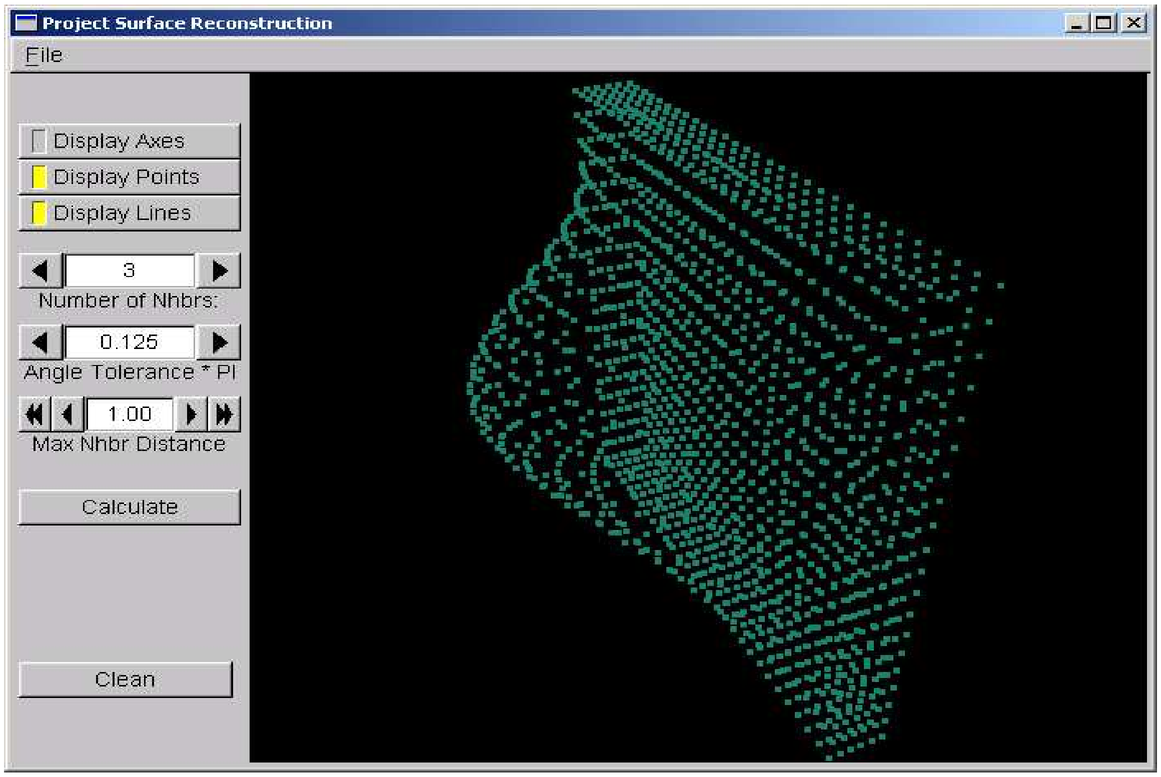
☐ Display Axes
☐ Display Points
☑ Display Lines

3
Number of Nhbrs:

0.125
Angle Tolerance * PI

1.00
Max Nhbr Distance

A collection of dots from a surface with a ‘bump’:



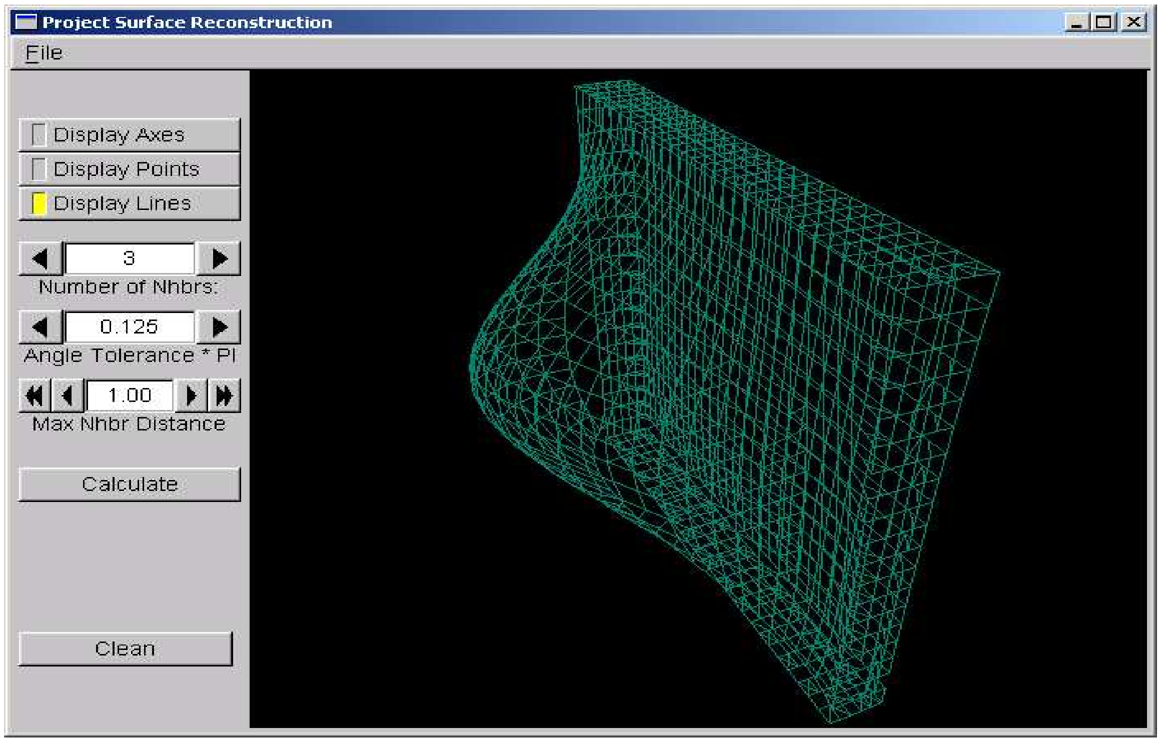
☐ Display Axes
☑ Display Points
☑ Display Lines

3
Number of Nhbrs:

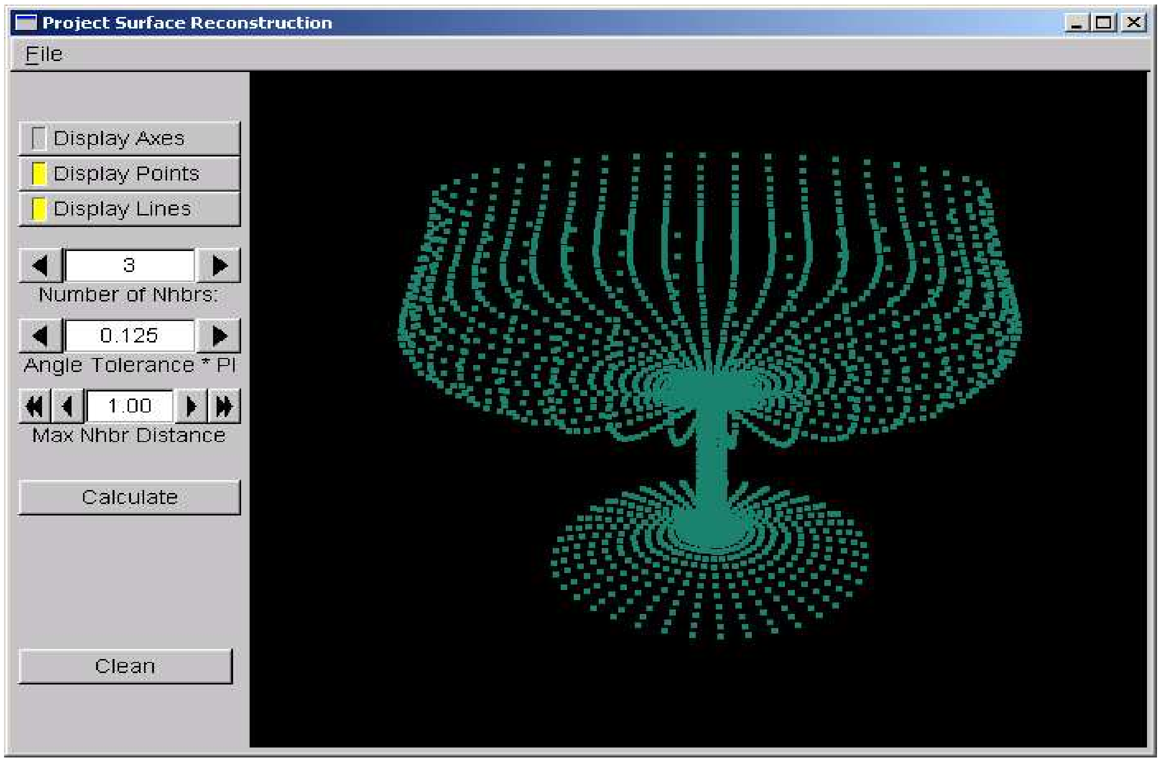
0.125
Angle Tolerance * PI

1.00
Max Nhbr Distance

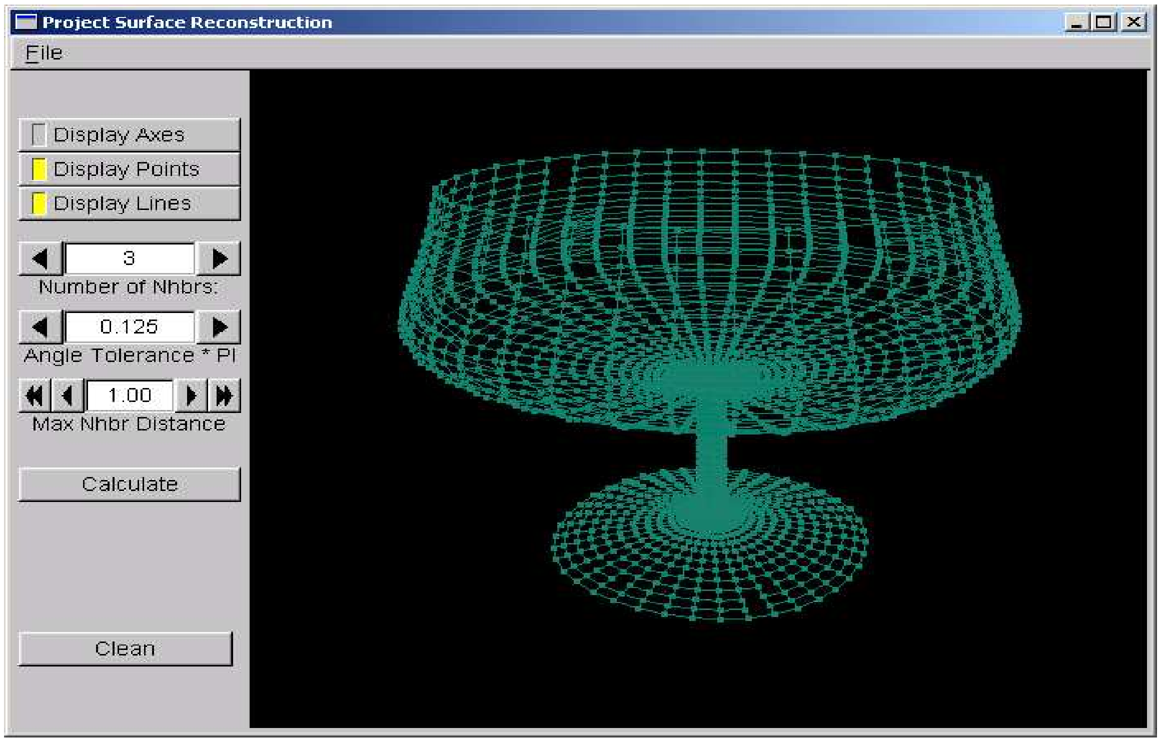
The ‘bump’ surface connected:



A chalice of points:



The connected chalice of points:



So did we achieve the goal we set? Did we successfully reconstruct a surface from an arbitrary set of points?

The answer is no, we did not. However we did solve a major problem in getting to such a result. From the above techniques we get a three dimensional graph of connected points and we know the connections yield some form of smooth continuity. Applying the angle tolerance idea above it should be possible to trace through the graph to obtain cycles of points. From these cycles we get an ordering of points to apply the spline techniques. Which should allow the construction of a spline mesh. And that in turn, gives us the surface of the object. Is it that simple? Probably not. But we are out of time.

It should be noted that to our knowledge the above technique of connecting points has not been used to solve this problem. Also of note, is that this method of creating a graph could be used for other problems – potentially it could be used as some form of local planner for motion planning. On other issues, this discussion and the problem itself points out some interesting differences between global and local properties of surface topology, and while we did not discuss them directly, they certainly should be obvious to those familiar. The intriguing idea of which would be: does the above idea really connect global and local properties, or rather can you recover (accurate) global behavior through examination of local criteria?

Another direction the above technique lends itself towards is not the use of splines to describe the surface but perhaps triangles or quadrilaterals. This would seem an obvious direction to go as the method relies heavily on limiting the number of neighbors a point may have.

In the end we learned quite a bit about this problem and the types of difficulties with this genre of problem. While we may not have achieved the desired goal in its entirety what we did learn is valuable and will undoubtedly aid us in the future.

And a final thanks goes to David Eberle for providing the point set data and to Dr. John Keyser for offering this opportunity to explore this ‘already solved’ problem.