
From Digitized Data to NURB Surface Meshes

Authors: Dr. Anshuman Razdan¹, Ben Steinberg² and Dr. Gerald Farin³

I. Abstract

A common reverse engineering problem is to convert several hundred thousand points collected from the surface of an object via a digitizing process, into a coherent geometric model that is easily transferred to a CAD software such as a solid modeler for either design improvement or manufacturing and analysis. These data are very dense and makes data-set manipulation difficult and tedious. Many commercial solutions exist but involve time consuming interaction to go from points to surface meshes such as BSplines or NURBS (Non Uniform Rational BSplines). Our approach differs from current industry practice in that we produce a mesh with little or no interaction from the user. The user can produce bi-quadratic or bi-cubic BSpline surfaces and can choose the number of patches, tolerance etc. as parameters to the system. The BSpline surface is both compact and continuous. The former property reduces the large storage overhead, and the later eliminates the rippling effect that a scanner often produces. In addition, the nature of the BSpline allows one to easily and smoothly alter the surface, making re-engineering extremely feasible. The BSpline surface is created using the principle of higher orders least squares with smoothing functions at the edges. Both linear and cylindrical data sets are handled using a different set of parameterization schemes, a key factor in the correct solution of the resulting BSplines. Also, because of the BSpline's continuous nature, a multiresolutional-triangulated mesh can quickly be produced. This last fact means that an STL file is simple to generate. STL files can also be easily used as input to the system.

II. Introduction

Many reasons motivate the field of reverse engineering. As more of the engineering world goes digital i.e. vendors, suppliers and manufactures work from the CAD model, it becomes even more important to convert the paper drawings into actual CAD model. Many times a part undergoes several design changes and after a few iterations it is difficult to predict how much the actual part has veered off from the original design intent. Another reason is that for certain disciplines such as toy manufacturing the original character or the model is created in clay by the artist. Its easier (more so because of the tradition) to sculpt the original rather than crate in a CAD modeling software. In other cases even if the original part (such as a consumer part) is created in CAD, it may get modified (sanding etc.) as it undergoes prototype feedback process. The only way to get those changes into the CAD model is to reverse engineer them. In addition, the need for automated inspection and verification of rapid prototyped and manufactured parts is also driving the field.

¹ Technical Director PRISM, Arizona State University, Tempe AZ 85287-5106, USA.

² Graduate Research Assistant, PRISM, Arizona State University, Tempe AZ 85287-5106, USA.

³ Professor, Dept. of Computer Science and Engineering, Arizona State University, Tempe AZ 85287-5106, USA

Digitizing and Reverse Engineering

Recently many accurate data acquisition systems have come to market. There are three main types of systems, the touch probes or CMMs, optic or laser based systems and volumetric or CAT and MRI systems. At present the PRISM lab has two Cyberware laser digitizers. Each has different resolution and accuracy of measurement. The problem of reverse engineering (from laser scanned data) is as follows:

The part to be reverse-engineered is scanned with the laser digitizer. Based on the complexity of the part, it may take several scans to get all the geometry. The many scans after cleaning (removal of supports, props, etc.) is transformed to a single coordinate system. Different digitizing systems do it differently but usually the end result is the same. The scanned data constitutes of hundreds of thousands of points with or without a valid triangulation with it. In the case of Cyberware a triangulation is available. We use the triangulation to detect bounding edges only. See Figure 1 for an example of digitized data. The captured data can be directly input into the CAD model but its just points and the triangulation. Its tedious to work with such a large number of points and requires tremendous amount of memory (RAM) on the computer. Mathematical representation of the same data with NURBS ((Non Uniform Rational BSplines) or BSpline surfaces (see Section IIIc for a brief explanation of NURBS) is much more efficient both in memory requirement and ease of editing. This process is also called the surface fitting. Once the equivalent surface is successfully created, it can be then input into the CAD model. Two issues dictate the way the surfaces are created. These are accuracy and ease of surface creation.

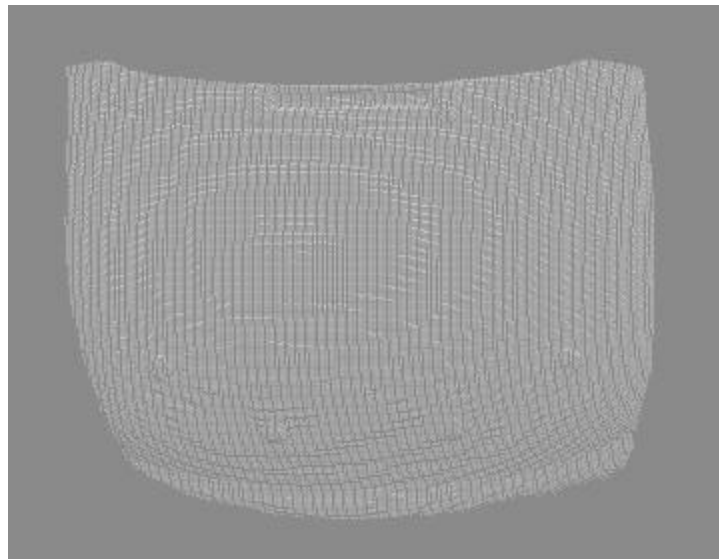


Figure 1 Digitized points from the hood of a toy car.

Previous Work

Many commercial software are now available that provide the user to interactively create surfaces from the digitized data. Of note are Imageware's Surfacer, Maya (Alias),

Geomagic's Wrap, Nvision, etc. See [Brocha and Young, 1995] for a discussion of the hardware and software vendors. Hoppe et al.[1992, 1993, 1994], Lounsbery [1992] – add more here – give a background for research in this area. Their work is motivated simply by the need for data reduction for faster display rather than geometric modeling requirements. The multiresolution wavelet modeling is excellent for reducing the data to be displayed quickly, however, it does not serve well in a CAD setting since currently no solid modeler supports wavelet based models.

Current solutions (at the time of writing this paper) involve time consuming interaction to go from points to surface meshes. Typically the user is required to interactively select a region from the set of digitized points, or select a small region to create a network of curves. The curves then are used as the basis for skinning or lofting operation. Even for simple regions it requires a great deal of interactive input and several hours of work to create the network of curves. The quality of the surface is also dependent on the skill level of the user or the operator since the key points, such as regions of high curvature need to be captured in the network of curve for the resulting surface to be accurate.

Our approach differs from current industry practice in that we produce a mesh with little or no interaction from the user. The user can produce bi-quadratic or bi-cubic BSpline surfaces and can choose the number of patches, tolerance etc. as parameters to the system.

The key concept here is the increase in automation and decrease in needed user input. The combination of the least squares method, smoothing functions and automated data parameterization conversion, produces extremely accurate BSpline surface approximation. These surfaces are smooth, continuous and have well behaved boundaries. In addition we use trimming curves to preserve holes that part of the scanned object as well as define non-rectangular boundaries.

III. Methods

Brief definition/explanation of NURBS

A point on a NURB surface can be mathematically represented as:

$$X_{ij}(u, v) = \frac{\sum_i w_i d_{ij} N_i(u) N_j(v)}{\sum_i w_i N_i(u) N_j(v)}$$

Where d_{ij} are the control points of the surface, w_{ij} are the weights attached to control points. N_i and N_j are the BSpline basis function. For more detail on basis functions and NURB Surfaces the reader is referred to (Hoschek and Farin). There are two important components called the parameters or knot sequences that also govern the shape of the NURB surface. The basis function evaluation depends on how the surface is parameterized.

Brief description of the Least Squares Methods (LSM)

Given a set of points P , the goal is to find a surface $X(u,v)$ such that sum of Euclidean distances between points P_i and $X(u_i, v_i)$ are minimized:

$$\text{Min } \sum_{i=1..n} |P_i - X(u_i, v_i)| \quad n \text{ is the number of data points}$$

This guarantees the best-fit surface with minimum error. The idea behind the least squares solution is to create the standard equation $Ax=b$

Where x represents the unknown d_{ij} , b represents the known data points we want to fit to, and A is a matrix of basis function contribution values for the data points. The equation $Ax=b$ becomes $AA^T x = bA^T$ which gives us the normal equations, a common form for the least squares method (Hoschek, Björck).

Description of my software and how “automated” it is

This program reads in the data points, and the user selects the resolution of the surface in the u and v directions. Surfaces which have a large number of convolutions or which change directions dramatically, generally need to have large amount of “patches” in both the u and v directions. Each patch represents a low degree polynomial surface which is guaranteed have a strong tangent continuity with its neighboring patches. The more patches that a NURBS has, the more “flexible” it is for fitting to a data set.

Once the user has chosen the number of patches in the u and v directions, the program automatically converts each data point in \mathbb{R}^3 into appropriate u, v parameter values in \mathbb{R}^2 . It is this conversion process that is a key to the surface generation. Knowing a point's parameter values allow the program to calculate the point's basis function contribution to the surface, which in turn allows for solving the unknown d_{ij} 's. After converting the point's u and v parameters, the program calculates each point's contribution to the surface. These calculations result in the A matrix which is then solved using standard linear solver methods. One of the problems with the least squares method is that “holes” or missing data can potentially cause the matrix to become singular. Each patch has a set of associated parameter values in u and v . If none of the data converted into u and v parameter space falls within a patch parameter “window” the matrix becomes singular. If only a few points are associated with the patch's parameter window then the patch becomes unstable and the surface will be poorly behaved in that region.

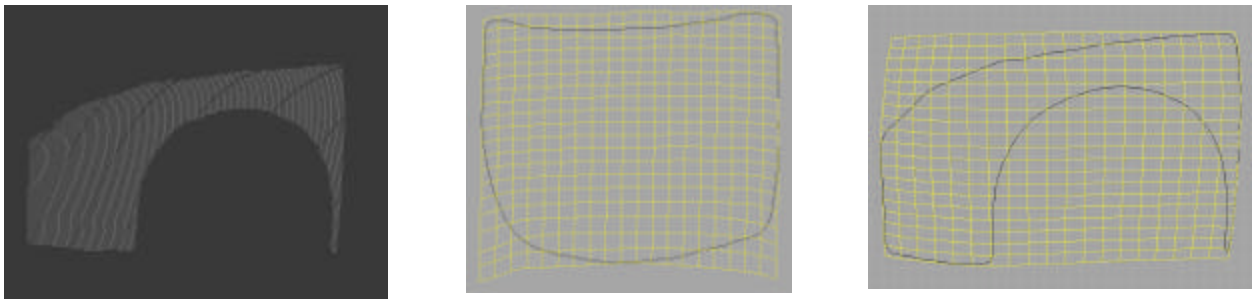
To prevent poorly behaved regions from being generated smoothing functions are used. The smoothing functions add information to the matrix which prevents the matrix from becoming singular. The smoothing functions guarantee that the behavior of a patch in a region of little to no data is stable by forcing the patch to conform to the shape of neighboring patches. Since the splines are defined over a rectangular domain, any “rounded” shape data leaves patches unsupported around the border of the spline. The

smoothing functions force these boarder patches to continue in the direction of patches, which did have data. The surface therefore continues smoothly through the areas of no data.

Another key feature is that the “extra” region of the spline, that is regions where there is no data because there original object had a hole, or a rounded boarder, can be removed using a trimming curve. The trimming curve is automatically fit from the boarder points of the data set. These points are gathered from the triangulation which is provided with the scan since triangles with free edges represents a boarder or hole in the object. This is the only use of the triangulation. Once the trimming curve is generated, the excess portion of the surface is cut away, leaving a smooth edge which follows the shape of the actual object.

IV. Results

We present some examples to show our results. The first example is of the hood and side fender of a toy car. The two pieces were individually scanned. See figureXX showing the



raw digitized points. Figures X shows the BSpline network with trimming cuve. Figures XX and YY show the trimmed surfaces in Maya (Alias software) with a blend. Details of the surface here.

The second example is of a flashlight. The top and the side of the flashlight were again digitized separately. Figures XX and YY show the raw digitized points.

Figure 2: BSpline network with the outer trimming curve for the digitized points in Figure 1.

V. Conclusion



