**NURBS Surface Reconstruction of Point Cloud for Three-Dimensional Dynamic Droplet Modeling**

**Submitted by:**

**Ashesh Chattopadhyay**

**Department Of Mechanical Engineering**

**IIT Patna**

**Under the Guidance of:**

**Dr. Atul Thakur**

**Dr. Rishi Raj**

**images.tif**

**Acknowledgment**

I would like to express my deepest gratitude to Dr. Atul Thakur and Dr. Rishi Raj for their guidance, encouragement, patience and especially for their time and moral support. Their able guidance and constant inspiration was a major source of motivation that helped me carry out this project. It was a knowledgeable experience to work under them.

**Certificate**

This is to certify that the project report entitled **“Two-dimensional Modelling of Droplets on Rough and Heterogeneous Surfaces ”,** which is being submitted by **Ashesh Chattopadhyay** (Roll No:- 1101ME07) in partial fulfillment for the requirements of the degree of Bachelor of Technology in Mechanical Engineering is a bonafide record of investigations carried out by him under my supervision and guidance.

**-Dr. Atul Thakur**

**-Dr. Rishi Raj**

1. **Abstract:**

Wettability has always played an important role in surface engineering and its applications. Controlling the wettability of the surface enables us to control various physical process related to solid-liquid interactions. In our previous work we had investigated how structured surfaces affect (*1)* the evolution of a cylindrical two dimensional droplet and the results obtained were matching very accurately with the experimental results obtained previously(*2)*.In that particular study(*1)* we had successfully developed and implemented a geometric algorithm that would enable us to simulate the evolution of the two dimensional droplet on structured surfaces. Compared to full scale Navier-Stokes solution the algorithm promised to be computationally efficient and gave us a simpler geometric representation. However a proper scaling could not be performed since it had only been tested for a two dimensional case. Here we have extended our study to building a suitable data structure for a three-dimensional droplet model and understanding how our algorithm(*1)* performs in that case. The primary focus in this study would be to look at the various computational geometric algorithms that help us in 3-D point cloud reconstructions and determining which of the data structures may be used or in that matter modified to mimic our physical problem.

1. **Introduction:**

Various computational geometric algorithms have been studied over time for reconstruction of 3-D point cloud. Each of them have their specific field of application. Our physical problem needs to be compatible with the data structure used in each case. Moreover the algorithm that we intend to implement has certain operators which would perturb points resting on the surface in order to create dynamically evolving surfaces. One has to pay heed whether the reconstruction algorithm chosen is compatible with such operators. Hence an extensive mapping of the operators from two-dimensional space to three-dimensional space needs to be done. The thing of utmost importance in our simulation is capturing subtle geometric intricacies of the droplet surface(concavities and points of inflexion). Previously, in two dimensions the intricacies were captured efficiently using a vector parameterized cubic spline model(*1).* However in a three-dimensional model capturing such intricacies is algorithmically a more challenging job. Hence the algorithm reconstructing the three-dimensional model should be competent to capture these features. In this section we will go through the various algorithms explored in order to find a 3-D representation of the evolving droplet surface.

* **Delaunay triangulation:**

This (*3*) is amongst the most popular tessellation algorithms used to reconstruct three dimensional models from point-clouds. It involves constructing triangles with the point clouds such that none of the circum-circles formed would contain any other point. The beauty of this algorithm lies in the fact that it always gives us distinct triangles that never overlap each other.The triangles formed gives us the boundary of the surface. Meshing functions in most CAD kernels use this tessellation algorithm. Boissonnat and Teillaud(*4)* recently had been able to modify this algorithm by using tangent plane method, but the crux of the algorithm remains similar to date with a time complexity of *O(nlog(n))*

2000px-Delaunay_Triangulation_(100_Points).tif

Figure1. Delaunay triangulation of point cloud

However this tessellation process would always give me the convex hull set of all the points given. In other words it would fail to capture a point of inflexion on the surface. Hence any of the algorithms whose crux lies in Delaunay triangulation process would not be compatible with our representation and so the idea discarded.

2000px-Delaunay_Triangulation_(100_Points).tif

Figure2.Concave portion eaten away by the convex hull

* **Concave hull**

Although there is no formal definition or algorithm that implements a concave hull (U-{convex-hull}),yet coming up with something ad-hoc to cultivate the idea of concave hull was explored. The algorithm that was developed was very simple. It involved doing a graph search over all the triangles in a Delaunay process and eliminating the triangle whose centroid was located at an unnaturally large distance from any one of the vertex. However, deciding this “unnaturally large number “ would become a tuning exercise for different surfaces. Moreover since we are dealing with dynamically evolving surfaces,hence this parameter would change dynamically and for very small intricacies this thresholding will cease to hold correct.

* **Alpha-hull**

The alpha-hull(*5)* is a similar algorithm that has alpha-parameter as the radius of the convex hull which can be considered for tessellation. A smart selection of the parameter would enable us to roughly calculate the concave hull of the surface.with a time complexity of *O(n2).*The α value can be range from 0 to . When the α value is zero, the shape of Alpha shape is the point. Alpha shape will become convex hull of sample point when α value goes to infinity . The collection of all possible α shape of sample point **P** is called the family of alpha shapes of the **P.**

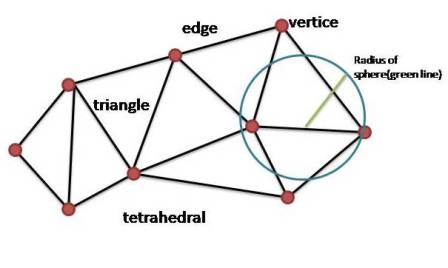


Figure3.Alpha parameter highlighted in green for the tessellation

However the issue remained that tuning this alpha value for dynamically evolving surfaces would be an estimate and counting on a perfect estimate is physically impossible.

* **Ball Pivoting Algorithm**

This algorithm(*6)* is a subset of the alpha hull which does not involve tessellation of the surface. Here a ball **P** b will be made to roll over a very dense point cloud on the surface. The radius of the ball would be basically analogous to the value of alpha in alpha hull. Hence . the points in the cloud which does not come within or on the surface of the ball will be considered to be outside the convex hull.

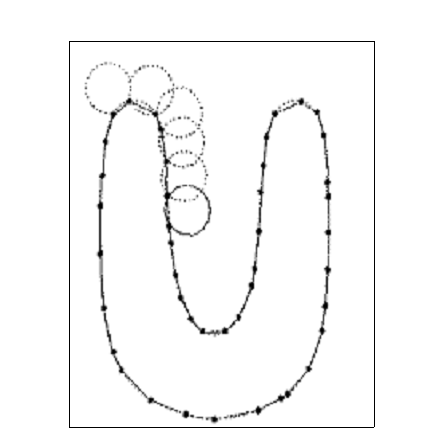


Fig4. BPA for a 2-D point cloud.

Here also the radius value of the ball is a tuning parameter and can only give a rough estimate of the concave hull.

Many such algorithms *(7,8,9,10,11,12)*were tried and tested which were mostly based on tessellation processes, in order to have an insight into their feasibility with regards to our problem, however with not much luck. The search was then directed to a different approach and tensor product surfaces were explored. The idea elucidated in tensor product surfaces enabled closed form piece wise analytic representation of surface patches and hence their inherent capabilities to change curvature, thus capturing concavities.

1. **Tensor Product Bi-Parametric Surfaces:**

Various surface modeling schemes have been discussed and explored over the years*(13)*, however in this study we would be closely looking at tensor product surfaces. In such surface representation, biparametric surfaces would be of concern. In general the surface vector is a function of two parameters which are spaced along any two directions. Hence .In general for a collection of coordinates ,the tensor product surface,.Here and are known as basis functions ,, are known as the control grid points and are itself represented as matrices. Hence the form shown above is known as tensor product form of a surface. The parameters both lie between 0 and 1.

**

Fig.5. The control grid and the parameters s and t.

In general, lots of representations of tensor product surfaces have been studied, however in our study we will concentrate solely on rational B-Spline surfaces. In the following sections , B-spline curves, surfaces and the various computational geometric algorithms required to evaluate these surfaces are discussed.

1. **B-Splines:**

A B-Spline basis curve is defined for a degree, *p* as ,where are the basis functions defined for a knot vector 0<<1.In order to understand this representation the definitions and properties of the basis functions and knot vectors are explained in this section.

* **Basis functions and Knot Vector:**

The basis functions are defined as a recursive algorithm*(14)* and is given as,

<

Here *u*  is defined as the knot vector. Large amount of studies*(15)*have already be done to design appropriate knot vectors.But in this study we will only be concerned with an open knot vector with the multiplicity being equal to the degree of the curve. Hence for n control points,the normalized knot vector sequence is,

The total number of knots must be equal (*n+p+1*). Considering this pattern of knot vectors we can make sure that the curve interpolates through the end points while passing through the convex hull of the rest of the control points. Different pattern of knot vectors can be designed and without going into the mathematical jargons of their effects, the figures below gives us an intuitive idea of the different knot vector designs

asdgfhhjiokpl.tif

Fig.6.Different basis functions and curve shapes for different knot vectors *x.*

* **Non-Uniform Rational B-Splines:**

A special form of B-Splines are NURBS, which are often termed as the ultimate solution to the surface problem*(16).* The flexibility of NURBS surface lies in the fact that the basis function is expressed as the weighted average of a rational polynomial function. The control of the weight functions allows fine control of the curves. Moreover the rational function polynomial gives more flexibility over the curves.The general structure of the NURBS curve is of the form, . is the weight vector while the other conventions remain similar to that of the B-Splines. By controlling the weights we can maneuver the curve within the convex hull of the control points. Fig.7 shows the effect of the curve through the manipulation of the weight vector.



Fig.7.Curve shows a stronger convex hull property with increase in weight.

Thus we can seethat by smartly choosing the weights and knot vector any curve can be designed using NURBS. Figure 8 shows the efficiency of NURBS to represent a circle compared to B-Splines.



Fig.8.NURBS representation,*Paul Mach,NURBS for Nerds(16)*

* **NURBS Surfaces:**

From the above discussion it is clear that NURBS surfaces is simply an extension of the NURBS curves with two different parameters along two different directions. A tensor product of the basis matrices would provide a NURBS representation of the bi- parametric surface. In this study this form of the surface is used extensively to develop all the computational algorithms. For similar parameters as described above ,,and degrees along their directions respectively, the NURBS surface would be of the form .The NURBS surface representation has a constraint where the parametric space must be mathematically a rectangular grid, hence in order to build the NURBS data structure the control points derived out of the parametric space must be in the form of a rectangular grid.Such was achieved through a mathematical technique called surface lofting*(12)* which is discussed in brief in section 5.

Untitled.tif

Fig.9.Example NURBS surface with changing curvatures

The figure shows a surface which has both positive and negative curvatures at different grid locations. This gives us enough confidence that the representation would be able to capture various surface intricacies if needed.

Eqipped with a proper representation technique we are now ready to build our data structure and design the computational algorithms required to evaluate the various functions and parameters needed to implement our algorithm(*1).*The following section exhaustively discusses how individual functions that were required in the two-dimensional modeling scheme is mimicked in three-dimensions.

1. **Mapping from 2-D space to 3-D space:**

In order to implement our final algorithm the derivatives, surface normals and curvature needs to be evaluated in for the surface in question. Although such evaluations are intuitive for analytically defined surfaces, for NURBS the ideas involving such calculations are more rigorous and separate computational algorithms were implemented to evaluate these parameters. This section would discuss some of the intuitive and some of the more involved strategies to evaluate these parameters.

* **Derivatives:**

The calculation of the derivative of the surface is simply an exercise that iteratively calculates the derivative of the tensor product of the basis functions.Fortunately, a little introspection shows us that calculating an order derivative of this tensor product is as trivial as calculating the partial derivatives of the individual basis functions with respect to each of the parameters. Although the approach is very trivial, yet the computation is very involved hence for the sake of brevity only the final result is provided.

The derivative of the basis function is given recursively as,

Given this form of the partial derivative of the basis function the derivative of the surface vector can be derived as,

* **Normal:**

Since the surface is a mesh of several isoparametric lines, so the normal vector to the surface at a point is simply given as the vector cross product of the partial derivatives of any point with respect to the parameters. Hence the normal is given in the matrix form hhhh.tif

Fig.10.Direction of normal at a point for example surface.

The direction of normal at point for the NURBS surface in figure 9 has been shown in figure 10.

* **Curvature:**

Calculation of curvature in case of a bi-paramteric surface is not a trivial computation, infact the concept of curvature in 3-D is in itself mathematically more intriguing than in 2-D. Usually curvature for a surface is given as gaussian and mean curvature in general, however in our case we will deal with gaussian curvature without any-loss in generality since, our intention is to capture points which vary inversely as radius. A detailed discussion on what curvature is defined as for a surface is given in *18.*In this study,for the sake of brevity only the final result of the curvature calculation is given. Gaussian curvature at a point on a surface is given as, where A,B,C,D are defined

B=

* **Perturbation:**

Whilst modeling the two dimensional case perturbation of points involved moving ponts along their normals and reconstructing a cubic spline to maintain continuity at each point.In case of a normal, however this is not as trivial since the the points have a chance to move out of the plane. So in order to maintain continuity, the NURBS data structure design should be able to reconstruct the NURBS through the perturbed point in order to maintain continuity.This is achieved through a technique called NURBS surface lofting which is described in section5.

* **Volume Calculation:**

The volume calculation is done by integrating the area of the iso-parametric curves along one of the parameters.Since this technique is a function of the control net hence the calculation is approximate with the error from the analytical volume decreasing with increasing the density of the control net. Figure11 shows the variation of volume with control net density in comparison to the analytical volume of a hemisphere of radius 1 unit.volume.tif

Figure.11. Variation of NURBS volume with analytical volume

We can see that the maximum error in the volume is a maximum of .09% which is well within the kind of bounds that we will be working with.

This section has thus encompassed all the operations that were internal to our surface optimization algorithm. In the following sections we will discuss the aspects of our reconstruction algorithm, the designed data structure and the computational schemes employed to recreate surfaces from a point cloud.

1. **Data structure and computational schemes for the reconstruction algorithm:**

This section will primarily describe the technique of surface lofting and the data structure used to achieve this technique. Moreover it will also describe the computational schemes (pseudo code or flow charts) that were employed to achieve a NURBS reconstruction of a data set.

* **Data structure for surface lofting:**

In order to maintain continuity, one can intuitively think that, if a point is perturbed, my data structure should be able to remember its neighboring points in order to connect to them locally to maintain continuity. In three dimensions this intuition does not remain as trivial as it sounds. It would involve a depth-first-search in a local neighborhood of all the perturbed points, to recollect the points which are closest to the original point. This would be algorithmically a complex structure to implement, so instead the surface which is inherently two iso-parametric set of curves, is reconstructed as these sets of curves instead of a 3-D structure as a whole. This would mean that each of the iso-parametric NURBS curves are reconstructed from the set of data points. So if a point is perturbed it would necessarily lie on one of these NURBS curves and would always share the neighbors which are points on the same NURBS curve. That would also necessarily imply that once perturbed, the curve does not remain planar, but it would ensure that continuity is maintained at the perturbed point by transforming the planar curve into a non-planar one. A set of such non-planarities would form the three-dimensional perturbation on the surface.

defr.tif

Fig.12.Two neighboring iso parametric curves with a perturbed point.

In Fig 12 we the red curve is a slice of a NURBS hemisphere while the blue curve is its neighboring iso- parametric slice with a point perturbed. However continuity has been ensured by considering that despite the perturbation the point belongs to same curve and hence shall share the same neighbors. However in this process, with an increased perturbation the distortion of the surface increases and the direction of normal evaluated comes out to be erroneous. Hence ideally a re-sampling of the parameters based on the new set of control points should be performed before reconstructing the lofted NURBS surface. In this study this re-sampling has not yet been performed. In order to implement surface lofting technique the data structure of points has been chosen as a three-dimensional array **P[i][j][k]**  where the parameter *u* has been chosen to vary along varying **j**  while the parameter *v* varies along varying k, while i represents the dimension of my coordinate space(3,for *xyz* and 4 for homogeneous ).Hence for each **k** value we have a an array of coordinates defined by a varying *u* value and constant *v* value. The computational scheme to store the point cloud in our data structure is as follows.

*Cloud\_storage(controlgrid)*

*{*

*do k=1 to N*

*{*

*do j=1 to M*

*{*

*do i=1to 4*

*{*

*P[i][j[k]=f(u[j],v[k])g(u[j]v[k])*

*}*

*}*

*}*

*}*

*return(P)*

What is critical to understand at this stage here is that once this storage scheme is implemented our points are no more recognized through their coordinates and rather by their indexes in the 3-D array. From this stage onward, now that we have a systematic data structure to store the control grid, we can use surface lofting technique to build NURBS curve at each slice of this 3-D array and hence reconstruct the NURBS surface. The following part of this section deals with the computation scheme of each aspect of the NURBS surface that was discussed in section4.

* **Computational Schemes:**

After storing the data cloud, we need to find the points on the NURBS surface, which would mean searching for the knotspan in which our parameter value lies and generate the basis functions required for that parameter value. This is performed by implementing a binary search tree on the knotspan and then implementing the recursive algorithm discussed in section 3.

*Findspan(u,sizeofknots(L),N-1,degree)*

*{*

*do i=1 to knotvector(U)*

*{*

*if u==L(N+1)*

*return u*

*else*

*value= binary search(degree,N+1)*

*return value*

*}*

*}*

The Basis functions are evaluated only for the required degree valule, however since the recursive algorithm is a recursion on both the index and the degree, a little introspection shows the pattern that the basis functions follow.

Fig.13.Schematic of the recursion for basis function

Figure 13 shows how the basis functions propagate with increase in degree and knot span. For a larger number of knot vector values and larger degree, this tree will become a lot bigger in depth and breadth . The computational scheme is used to evaluate this tree is given below. Here the convention follows that 0/0 is 1.

*Basis functions(u(parameter),U(knot),span(knotspan))*

*{*

*initialize left[M+1] and right[N+1] arrays as zero(first and 2nd term of the recursion relation)*

*do j=1 to degree*

*{*

*left[j+1]=u-U[span+1-j]*

*right[j+1]=U[span+j]-u*

*saved=0;*

*{ do r=0 to j-1*

*temp = N[r+1]/(right[r+2] + left[j-r+1])*

*N[r+1] = saved + right[r+2]\*temp*

*saved = left[j-r+1]\*temp*

*}*

*N[j+1] = saved*

*}*

*}*

*Return(N)*

The final part of the reconstruction algorithm is to evaluate the value of the NURBS curve at various parametric locations. This evalution is largely governed by the degree and we would see in the next section how the degree changes the fairness of the lofted surface.The computational scheme is as follows.

*Curve\_evaluation(P[i][j][k],u,degree)*

*{ do r=1 to M*

*{*

*span=Finspan(u[i]..)*

*N=basisfunction(u[i]..)*

*do i= 1to M*

*{*

*temp\_1=span-degree+1*

*initialize val[r]=0;*

*do{ j=1 to degree*

*value[r] = value[r] + N[j+1]\*P[row,temp1+j]*

*}*

*}*

*}*

*} return(value)*

The above scheme gives us the set of values on the NURBS iso-parametric curves which are required to generate the lofted surface. Figure.14. shows us a flow chart of the entire reconstruction process through NURBS data structure.

Fig.14.Schematic flowchart for NURBS reconstruction

This brings to the end of the reconstruction algorithm. The rest of the parameters evaluated for our final algorithm such as the normals and derivatives are evaluated following this reconstruction scheme. Those evaluations although not algorithmically complex, involve a lot of programming lines and for the sake of brevity are avoided in this study. The following section will deal with the results obtained when the droplet surface is reconstructed, perturbed and optimized using the algorithms described in this section.

1. **Results and Discussions:**

At the final stage each of these algorithms were implemented to mimic the 3-D droplet’s shape. In general it is assumed to be a hemisphere, hence the X,Y,Z point clouds for the surface is chosen to be of the form of quartic polynomials and is represented as ,,.Here *u* and *v* are the parameters along two mutually perpendicular directions along *X* and *Y*. The droplet shape was then distorted by moving a set of points along their normals and the new unstable shape was passed through the surface optimization algorithm that we had developed in our previous work*(1)*.various insights were gained in this new approach, and by changing the degree of the NURBS surface, the optimized shape varied.Figures below show, how at each perturbation continuity is maintained and how the perturbed surface’s fairness would drastically change with the change in degree of the curve. Moreover,the direction of the normal for a perturbed lofted surface comes slightly erroneous at each iterative step of the algorithm*(1)* hence the choice of points having maximum and minimum curvature required for our algorithm also has a certain amount of error. Through an accumulation of such errors, the kind of shapes that we were hoping to have has not yet been achieved.

pic2.tifpic1.tif

Fig.15.100 X100 point control grid with 99 as degree. Smooth perturbed surface.Continuity maintained at each point location.

pic3.tif pic4.tif

Fig.16.Same control grid with same perturbation, Fig17.degree 4, different perturbation

degree is kept at 4.Fairness of the surfaces reduces.

Errors for normals and curvature

increases.

pic5.tif

Fig.18.Equilibrium shape obtained for degree 99 after implementing optimization algorithms.

Unnatural curvatures near the contact line.

pic 7.tif

Fig.19.Equilibrium shape obtained for degree 4,large distortions in volume resulting in high deviations from circular shape.

pic6.tif

Fig.20.Two slices of the equilibrium shape obtained for figure18 ,showing

Large fluctuations of curvature and normals near the base,with fluctuating volume.Though continuity is maintained at each reconstructive step the movement of points along the surface causes large non-planarities owing to which the volume calculation,curvature and normal directions suffer, as a result the final shape in fig.18 although shows proper trend optimization(spherical cap with lesser contact angle) shows high distortions near the fixed contact line. A resampling of the parameters at each iterative step in order to restore continuity seems to be the intuitive solution along with a parameterization scheme that captures the dynamic movement of points. Several aspects of the three dimensional modeling strategy were obtained in this study,yet few remains unexplored and we believe that the resampling technique that restores planarity of the iso-parametric curves might be the solution to curbing down the fluctuations of the curvature near the base. Those aspects are currently being worked on.

1. **References:**
2. Ashesh Chattopadhyay,Atul Thakur,Rishi Raj, *International Conference of Fluid Mechanics and Fluid Power,2014*
3. R.Raj,Evelyn Wang,Ryan Enright,*Langmuir 2012*
4. Amenta, N. and Bern .M *Discrete Compuational Geometry,1998*
5. Boissonnat, J.D. and Teillaud, *Springer Berlin Heidelberg New York,2004*
6. Edelsbrunner, H. and Mucke, *ACM* *Transcations on Graphics 1994*
7. Amenta, N., Bern,M. and Eppstein, *Graphical models and Image Processing*,1998
8. Boissonnat, J.D. *ACM Trans.Graph.* 1984.
9. Boissonnat, J.D. and Cazals, F.. *In Proc, 16th Ann Symposium on computational geometry*, 2000
10. Boissonnat, J.D. and Cazals, F. *In A.chalmers and T-M.Rhyne ,editor* ,Eurographics, 2001
11. Dey, T.K. 2007 Curve and surface reconstruction :algorithms with mathematical analysis *.Cambridge University Press*
12. Kolluri , R. , Shewchuk, J.R. and Brien, J.F.. *EuroGraphics Symposium on Geometry Processing,* 2004
13. Newman, T.S. and Yi, H 2006 A survey of the marching cubes algorithm*. Computer & Graphics*
14. *The NURBS Book,Springer*
15. Carl-de Boor, *Computational Geometry*
16. Piegl,*On Nurbs:A survey,University of South Florida*
17. Paul Mach,*Technical report:Nurbs for Nerds*
18. Chaine, R. *In Symp, Geometry Processing*, 2003
19. Rogers,Adams,*Mathematical Elements of Computer Graphics*